

DEVELOPMENT AND DEPLOYMENT OF A FIELD BASED SOIL MAPPING  
TOOL USING A COMPARATIVE EVALUATION OF GEOSTATISTICS AND  
MACHINE LEARNING

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Jeff A. Fiechter

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Agricultural and Biological Engineering

August 2019

Purdue University

West Lafayette, Indiana

**THE PURDUE UNIVERSITY GRADUATE SCHOOL**  
**STATEMENT OF DISSERTATION APPROVAL**

Dr. Dharmendra Saraswat, Chair

Department of Agricultural and Biological Engineering

Dr. Dennis Buckmaster

Department of Agricultural and Biological Engineering

Dr. James Krogmeier

Department of Electrical and Computer Engineering

**Approved by:**

Dr. Nathan Mosier

Head of the Departmental Graduate Program

## ACKNOWLEDGMENTS

I'd like to first acknowledge the Foundation of Food and Agricultural Research (FFAR), Indiana Soybean Alliance (ISA)/Indiana Corn Marketing Council (ICMC), and Purdue University for supporting this work financially. Without funding, works like this would never be able to be done, and I wouldn't have been able to receive my degree.

I'd like to also acknowledge my Advisor and Committee Chair, Dr. Dharmendra Saraswat, for the time he spent in mentoring me during the research process. His method-based convictions were firmly communicated to me, and forced me to slow down, read thoroughly, and conduct experiments carefully. It is because of this methods-based process that I feel confident in this work, and believe it to be a valid contribution to the field.

I would also like to acknowledge my Committee members, Dr. Dennis Buckmaster and Dr. James Krogmeier. They each took time to meet with me individually, gave insightful criticisms of my work, and suggested changes to my plan of study. Their breadth of expertise served to ensure my work had real depth.

Ben Hancock also played an indispensable role in getting the tool ready for the world. His unbounded patience, coupled with his vast technical knowledge, cut untold hours from my development process.

Along with these, I'd like to acknowledge my wife, Anna, for her continued support as a I worked through this degree. Her wisdom, discernment, and penchant for providing startlingly insightful advice and admonishments kept me balanced, focused, and healthy. Without her empathy, partnership, or gentle corrections, I would never have been able to do the work I've done in the time I've done it.

Lastly, I'd like to acknowledge Jeff Boyer, of the Davis-Purdue Agricultural Center, for providing the soil data used as the case study in this work. The unique density of

the dataset made it exceptionally rare, and not having to collect data for my research cut the time down by months. Without this contribution, the research would have been far more difficult and time consuming.

## TABLE OF CONTENTS

	Page
LIST OF TABLES . . . . .	vii
LIST OF FIGURES . . . . .	viii
ABSTRACT . . . . .	xi
1 INTRODUCTION . . . . .	1
2 THEORETICAL BACKGROUND AND LITERATURE REVIEW . . . . .	7
2.1 Current Interpolation Practices . . . . .	7
2.2 Machine Learning in Digital Soil Mapping . . . . .	11
2.2.1 Common Algorithms . . . . .	11
2.2.2 Explanatory Variables . . . . .	16
2.3 Conclusion . . . . .	19
3 METHODS . . . . .	20
3.1 Objectives 1 & 2 . . . . .	20
3.1.1 The Case Study . . . . .	20
3.1.2 Public Data . . . . .	24
3.1.3 Benchmarking Methods . . . . .	25
3.1.4 Developing the Prediction Framework . . . . .	27
3.1.5 Feature Data . . . . .	29
3.1.6 Validation and Testing . . . . .	34
3.2 Objective 3 . . . . .	37
4 RESULTS AND DISCUSSION . . . . .	42
4.1 Objective 1: Establishing a Baseline . . . . .	42
4.1.1 Applying A Field Average . . . . .	42
4.1.2 Applying A Grid Prediction . . . . .	44
4.1.3 Applying Ordinary Kriging . . . . .	47

	Page
4.2 Objective 2: Develop an Improved Method . . . . .	56
4.2.1 Experiment 1: Spatial Position . . . . .	57
4.2.2 Experiment 2: Relief . . . . .	66
4.2.3 Experiment 3: Spatial Position and Relief . . . . .	77
4.3 Objective 3: Make Accessible . . . . .	81
4.3.1 User Input . . . . .	81
4.3.2 Processing . . . . .	90
4.3.3 Output . . . . .	91
5 CONCLUSIONS AND RECOMMENDATIONS . . . . .	96
5.1 Objectives 1 & 2 . . . . .	96
5.2 Objective 3 . . . . .	98
LIST OF REFERENCES . . . . .	100

## LIST OF TABLES

Table	Page
4.1 MAE Scores, The Standard Deviations of The Absolute Residual Sets, MAPE Scores, and RMSE Scores . . . . .	42
4.2 A Key For Interpreting Table 4.3, Where Each Interpolation Is Assigned A Unique Number. . . . .	45
4.3 Two Sample T-test Results For Every Model Comparison; Created Using MEA And The Standard Deviation of The Absolute Errors. Any P-Values Significant At A 5% Level For A Comparison Are Displayed In Bold. The Null Hypothesis of The T-test Was That The Two Absolute Residual Sets Had The Same Mean. . . . .	46
4.4 The Random Forest Model Parameters. . . . .	57

## LIST OF FIGURES

Figure	Page
2.1 An Example Semi-Variogram with a Fitted Spherical Model and Nugget, Range, and Sill Indicated. . . . .	9
2.2 An Example of One-Dimensional Linear Classifier. Shown with Split (Solid Line) and All Possible Splits (Dashed Lines). . . . .	13
2.3 An Example of Recursive One-Dimensional Linear Classifier. Displays Three Nodes of Recursive Splits, Shown as Solid Lines. . . . .	14
2.4 Example Regression Tree, Using the Landform Attributes Observed at Percent OM Observation Locations to Predict Percent OM. . . . .	15
3.1 Site Boundary and Soil Sample Locations. Soil Percent Organic Matter Values are Color-Mapped onto the Point Locations. . . . .	21
3.2 A Histogram of the 110 Percent OM Values. . . . .	23
3.3 Points Selected For Training, Validation Points, and Site Boundary. Points Selected For Interpolation are Displayed in Red, While Points Kept For Validation are Displayed in Blue. . . . .	24
3.4 Flow Chart Representing the Creation of Euclidean Distance Features. . . . .	32
3.5 Visual Representation of Grid Based Features and How Feature Values for a Single Location Are Extracted. . . . .	33
3.6 An Example CSV File, Containing Columns For Latitude, Longitude, and Properties of Interest. . . . .	39
4.1 Predicted Percent OM, Using The Field Average. . . . .	43
4.2 A Grid Based Prediction of Percent OM. . . . .	47
4.3 Histogram of The Percent OM Values of The 28 Points Used For Interpolation. . . . .	48
4.4 Histogram of the Box-Cox Transformed Percent OM Values of The 28 Points Used For Interpolation. . . . .	49
4.5 Sampling Locations, Displaying Non-Clustered Layout. . . . .	50

Figure	Page
4.6 A Trend Analysis of The Percent OM Dataset, Revealing First and Second Order Trends. The Three Dimensional Polynomial Is Shown As Two-Dimensional Slices: Where It Intersects The Northern Boundary of The Field (Green), and Where It Intersects The Eastern Boundary of The Field (Blue). . . . .	51
4.7 A Voronoi Map of Entropy For The Percent OM Dataset. . . . .	52
4.8 Semi-Variogram and Accompanying Spherical Model of The 28 Percent OM Samples. . . . .	53
4.9 Ordinary Kriging Percent OM Prediction. . . . .	55
4.10 Euclidean Distance To A Sample Point, Measured In Raster Cells. . . . .	58
4.11 Euclidean Distance To A Sample Point, Measured In Raster Cells. . . . .	59
4.12 Euclidean Distance To A Sample Point, Measured In Raster Cells. . . . .	60
4.13 Random Forests Percent OM Prediction, Based On Spatial Position only. . . . .	64
4.14 Extra Trees Percent OM Prediction, Based On Spatial Position only. . . . .	65
4.15 The Elevation of The Site (in Feet). . . . .	67
4.16 The Slope of The Site In Percent, Calculated At 7.6 m Resolution. Note: While Slope Reaches 100% In Some Areas, Color Scale Has Been Set To 25% Max For Higher Contrast Viewing. . . . .	68
4.17 The Slope of The Site In Percent, Calculated At 174.8 m Resolution. Note: Here Color Scale Is Set To True Value Range, Much Smaller Than The Previous Figure. . . . .	69
4.18 The Profile Curvature, Calculated At A Scale of 83.6 m. . . . .	70
4.19 The Planform Curvature, Calculated At A Scale of 83.6 m. Note: Here The Color Scale Is Set To -0.1-0.1 For Viewing Contrast, But The True Value Range Is Much Larger. . . . .	71
4.20 The Profile Curvature, Calculated At a Scale of 174.8 m. . . . .	72
4.21 Random Forests Percent OM Prediction, Based On Relief Only. . . . .	75
4.22 Extra Trees Percent OM Prediction, Based On Relief Only. . . . .	76
4.23 Random Forests Percent OM Prediction, Based On Spatial Position And Relief. . . . .	79
4.24 Extra Trees Percent OM Prediction, Based On Spatial Position And Relief. . . . .	80
4.25 The Landing Page of The UI. . . . .	82

Figure	Page
4.26 The Shapefile Upload Page Of The UI. . . . .	84
4.27 The CSV Upload Page Of The UI. . . . .	86
4.28 The Field Boundary Creation Page Of The UI. . . . .	88
4.29 The Submission Page Of The UI. . . . .	89
4.30 The Status Display Page Of The UI. . . . .	93
4.31 The Output Display Page Of The UI. . . . .	95

## ABSTRACT

Fiechter, Jeff A. MS-ABE, Purdue University, August 2019. Development and Deployment of a Field Based Soil Mapping Tool Using A Comparative Evaluation of Geostatistics and Machine Learning. Major Professor: Dharmendra Saraswat.

Soil property variability is a large component of the overall environmental variability that Precision Agriculture practices seek to address. Thus, the creation of accurate field soil maps from field soil samples is of utmost importance to practitioners of Precision Agriculture, as understanding and characterizing variability is the first step in addressing it. Today, growers often interpolate their soil maps in a black-box fashion, and there is a need for an easy to use, accurate method of interpolation. In this study, current interpolation practices are examined as a benchmark, a Random Forest (RF) based prediction framework utilizes public data to aid predictions, and the RF framework is exposed via a webtool. A high density (0.20 ha/sample) field soil sample dataset provides 28 training points and 82 validation points to be used as a case study. In the prediction of soil percent organic matter (OM), the grid and ordinary kriging interpolations both had higher Mean Absolute Error (MAE) scores than a field average prediction, though the difference was not statistically significant at a 5% confidence level. A RF framework interpolation utilizing a high resolution (1.52 m) DEM and distances to known points as the featureset had a significantly lower MAE score than the field average, grid, and ordinary kriging interpolations. The results suggest that for the study site, RF framework performed better compared to a field average, a grid based, and an ordinary kriging interpolation methods.

## 1. INTRODUCTION

Field soil maps are one of the most frequently utilized spatial datasets in Precision Agriculture. A recent survey estimated that field soil maps have been privately created for over 50% of Midwestern cropland (Erickson and Widmar, 2015). Soil, as the growing medium, is often the source of variability that Precision Agriculture attempts to address (Atherton et al., 1999). For example, as soil organic matter (OM) provides a source of nitrogen via mineralization, many producers use maps of percentage OM while creating variable rate nitrogen prescriptions. In another case, as higher soil cation exchange capacity (CEC) dampens the effect of lime application, both soil pH and CEC maps must be used to create variable rate lime prescriptions. For a final example, as water holding capacity varies strongly with soil texture, most variable rate irrigation prescriptions attempt to tailor rates to maps of soil texture. As is the case in these and any applications, the precision and accuracy of any custom prescription is always limited by the base dataset. In the case of Precision Agriculture, access to high quality soil data is crucial to the creation of high quality prescriptions. While there are publicly available soil maps created by the NRCS, it is generally agreed that these maps are either too inaccurate or too coarse for the ideals of modern Precision Agriculture (Zhu et al., 2001). To compensate for this lack of high-quality public soil maps, farmer's have resorted to generating high quality soil sample datasets using personal resources.

As direct measurements of soil properties are costly, most field soil maps are based on a limited set of point samples (Wollenhaupt et al., 1997). While there are many sampling protocols and strategies that ensure quality point samples, it is through the process of interpolation that likely property values are assigned to unsampled locations. A common interpolative practice today is to simply assume that a point sample is representative of a certain zone: a block in an even grid or a

NRCS soil unit, for example (Erickson and Widmar, 2015). However, as the scale of soil property variation is often smaller than the scale of typical soil sampling layouts (Franzen and Peck, 1995), assuming a sample is representative of a zone has the effect of creating non-continuous maps with sharp boundaries not present in a soil catena. As an alternative to this practice, many interpolative methods attempt to model the relationship between spatial position and soil property values, which allows for the estimation of maps at a higher resolution than the soil sample grid. One example is geostatistical methods, a family of interpolation practices that predict using a weighted average of the known point values, with the weights assigned by the distances from the unknown point to the known points. Geostatistical methods have been used extensively in soil science since the 1980's (Burgess and Webster, 1980), and are well proven in the field. However, while geostatistical methods have a long pedigree and are still used today in soil mapping (Vaysse and Lagacherie, 2017; Ramcharan et al., 2018), there is still room for conceptually stronger models. Geostatistical methods use distance as an explanatory variable, but distance is not the only variable known to explain variations in soil properties (McBratney et al., 2003). For instance, it has long been shown that relief is an effective explanatory variable for soil properties (McBratney et al., 2003): hill-slopes typically have thinner soils than lowlands, river bottoms have higher silt content than the ridges above, etc. Thus, an interpolation method that utilizes both spatial position and relief as explanatory variables could theoretically explain more variation in soil properties than a spatial position only model. As a LIDAR generated, high-resolution (1.52 m) digital elevation map (DEM) is publicly available in the state of Indiana, any field soil mapping effort in the state could potentially utilize relief as an explanatory variable.

The creation of a multivariate model is bounded by the limited data nature of field soil map creation. To illustrate this, many Midwestern fields are sampled at a rate of 2.02 hectares, 1.01 hectares, or even 0.405 hectares per sample (Erickson and Widmar, 2015), but the typical field in the US corn belt is less than 50 ha (Capellades et al., 2009). If a 40 ha field was sampled at a density of 1.01 ha/sample, it would only

yield about 40 samples. Fitting any model to a dataset this small is quite difficult, as models based on limited datasets often are unable to capture the relationships present or become overfit to relationships only present in the sampled data (Schloeder et al., 2001). Despite this limitation, and considering that most growers are untrained in proper geostatistical modeling, geostatistically interpolated soil maps are typically created in a "Black Box" fashion. This implies that many private soil maps are currently generated using limited data and with model parameters left at default values. To improve on the current practice, any new multivariate model developed and provided to growers would need to address the low data dilemma and do so with limited user intervention.

A flexible alternative to geostatistical methods is the growing field of Machine Learning (ML) methods. In recent decades, ML methods have grown in popularity in the field of DSM (Heung et al., 2016). While there are many reasons for this, the ability of certain ML algorithms to model non-linear, non-stationary empirical relationships, with limited user intervention, has shown to be valuable in the field of DSM. Additionally, though geostatistical methods are still widely used in the field, a recent study found that a ML algorithm could match, and in some cases exceed, the performance of geostatistical methods using both distance and distance coupled with other variables (Hengl et al., 2018). While this makes ML a promising potential method for creating private field soil maps, few studies have attempted to apply a ML algorithm to this scope. Doubtless, there are many reasons for this, but chief among them is again the limited data available to fit a model. Many ML algorithms are best known for their uses in the field of Big Data, and as such are typically data hungry. The smallest dataset utilized in the aforementioned study (Hengl et al., 2018) had 155 samples for a 500 ha area; other DSM studies utilize up to 400,000 samples to estimate a model the size of the contiguous US (Ramcharan et al., 2018). In a review of the literature, the vast majority of ML DSM studies utilized more than 150 samples (most were far more); the only model that used less assumed strong priors but required site specific expert input to build the model (Zhu et al., 2015).

In each of the cases described above, the sampling density (3.2 ha/sample to 2000 ha/sample) of the observations was lower than typical private field surveys, but as the areas of interest were much larger than typical fields, there were far more observations to fit to a model. Though the use of ML for building DSM models with less than 150 sample points is challenging, there are algorithms that can be tailored for a limited data environment. One algorithm that has shown promise in a limited data environment is Random Forests (RF), as it is highly tunable for a variety of data conditions (Shaikhina et al., 2019; Gunduz and Fokoue, 2015). Additionally, RF is well proven in the field of DSM, (Ramcharan et al., 2018; Hengl et al., 2017, 2018; Heung et al., 2016; Vaysse and Lagacherie, 2017) and requires little user input once the initial parameters are tuned for the general application.

Bearing all the above in mind, it is clear that there is a need for a better way for growers to create field soil maps, but much work would have to be done before a new method would fill the need. First a baseline performance would have to be set. As there are currently methods of interpolation in use, there is an existing performance standard that any new interpolation method must exceed. The simplest benchmark would involve predicting the average value of the soil samples for the entire field, still a common practice for many producers (Erickson and Widmar, 2015). A second benchmark would mimic the current industry practice of assuming sample values are representative of the particular zone or grid they were sampled from. The last benchmark would need to mimic a more analytical interpolation, like ordinary kriging, where missing soil property values for the unsampled locations are estimated by the modeled relationship between spatial position and property values. This comparative evaluation of common interpolation methods would serve as a performance baseline against any proposed new interpolation method.

After the baseline performance was set, the next step would be to develop a new interpolation method that could utilize publicly available datasets for predicting soil property of interest. It would have to depend on either same or a lesser degree of user intervention than existing interpolation methods, to ensure user friendliness.

Additionally, the data required from the user would have to be the same, i.e. soil sample data and a field boundary. These conditions seem challenging, but could conceivably be fulfilled by an implementation of the RF algorithm, optimized for field soil interpolation. For validation, an evaluation of both the suitability of the RF algorithm, and the viability of including additional public datasets as features would need to be conducted. To be viable, any alternative interpolation method would need to match or exceed the baseline performances of current methods.

If all this was done, the model would need to be prepared for real-world deployment. First, the interpolation model would need to be integrated into an automated prediction framework. Again, to limit user intervention, the framework would have to automatically selectively download public datasets, prepare a feature set, train the model, make a prediction, and write the prediction, only requiring soil data and a field boundary from the user. Once the framework was automated, a User Interface (UI) would need to be created to collect and display user data and serve as a user friendly front-end to the framework. While this could be a stand-alone software, it would be simpler for users if the UI was web-based and the framework processing was in the cloud.

With the previous discussions serving as a motivation, the overall goal of this study was to create a better way to interpolate field soil maps using the process laid out above.

The actual study objectives were threefold:

- 1) Use a case study dataset and traditional interpolation methods to establish a performance baseline.
- 2) Develop a RF based interpolation framework, testing the viability of the method and the efficacy of including public datasets as features.
- 3) Automate the tested and finalized interpolation framework and deploy it through a web-based UI.

Once all the three objectives are complete, a baseline of current interpolation methods will have been established, an improved method will have been developed

and tested against baseline methods, and the improved method will have been exposed to the intended users, i.e. growers.

## 2. THEORETICAL BACKGROUND AND LITERATURE REVIEW

### 2.1 Current Interpolation Practices

For many years, most Midwestern growers did not estimate field soil maps, but instead practiced whole field sampling. As recently as 2005, only an estimated 17% of Midwestern acreage was either grid or zone sampled, with the remaining 83% either whole field sampled or not sampled at all (Erickson and Widmar, 2015). In the years since, the practice of tagging soil samples with a geolocation has grown steadily, with an estimated 45% of Midwestern acres either grid or zone sampled (Erickson et al., 2017). In the case of zone based sampling, the soil samples collected are used to estimate the value of a certain management zone. These zones are defined in many ways: classified yield maps, previous higher density grid sampling, grower knowledge, soil mapping units, and even soil electrical conductivity (Nawar et al., 2017; Erickson et al., 2017). This diverse collection of delineation methods, combined with the host of sampling protocols accompanying them, make zone based soil mapping a highly variable practice, and difficult to represent with a single example. Grid based soil sampling and mapping, on the other hand, is far more uniformly practiced, as suggested by the name. In fact, of agricultural service providers that offer soil sampling, 64% report offering sampling at a common density, 1.01 ha/sample (Erickson and Widmar, 2015). After sampling, a map can be created in two ways. The simplest is to assign the value of the sample to its grid block, which is effectively a nearest neighbor interpolation. The second way is analytically-based, and results in a continuous prediction: geostatistical interpolation.

The collection of methods known as geostatistics is unique in reference to other field soil mapping methods, in that an explanatory variable is used to make a predic-

tion: the spatial position of the point of interest. The core assumption of this suite of methods is the concept of spatial autocorrelation: observations of soil properties that are nearer to each other are more related than observations that are far apart (Goovaerts, 1999). Geostatistical methods apply this concept by comparing the location of the point to be predicted relative to the location of known points. Then, the point to be predicted is assigned a value based on the values of the known points near it. Exactly how the value for a point is determined varies from method to method. For example, one method (albeit not common in DSM), k - nearest neighbor (kNN), uses the average value of the nearest k known points to the unknown point as the predicted value, or, if k is set to 1, the predicted value is simply the value of the nearest point (Heung et al., 2016).

One of the geostatistical methods commonly used for soil mapping is called kriging (Minasny and McBratney, 2016). Kriging was developed in the 1950s and 60s in the South African gold industry. Mines used predicted ore grade, based on sample cores, to choose which areas to mine (Cressie, 1990), and kriging was developed as a more accurate way to generate these predicted grades. In kriging, the value assigned to each unknown point is a weighted average of the known values. The first step in calculating weights is an analysis of the level of spatial autocorrelation present in the data. This is accomplished by selecting sets of known points at a set distance apart, and analyzing how related they are. This is repeated across all possible distances in the dataset, and the results are plotted on a semi-variance graph: semi-variance on the y-axis, and separating distance (or lag) on the x (fig. 2.1) (Oliver and Webster, 1986). After the sample data has been plotted, a model is fitted to the data. The choice of model is determined by the user, but a common choice is the spherical model (fig. 2.1). The fitted model is then used to calculate the weights of the weighted average for every unknown point. Thus, the accuracy of the kriged result is dependent on the goodness of fit of the semi-variogram model Burgess and Webster (1980).

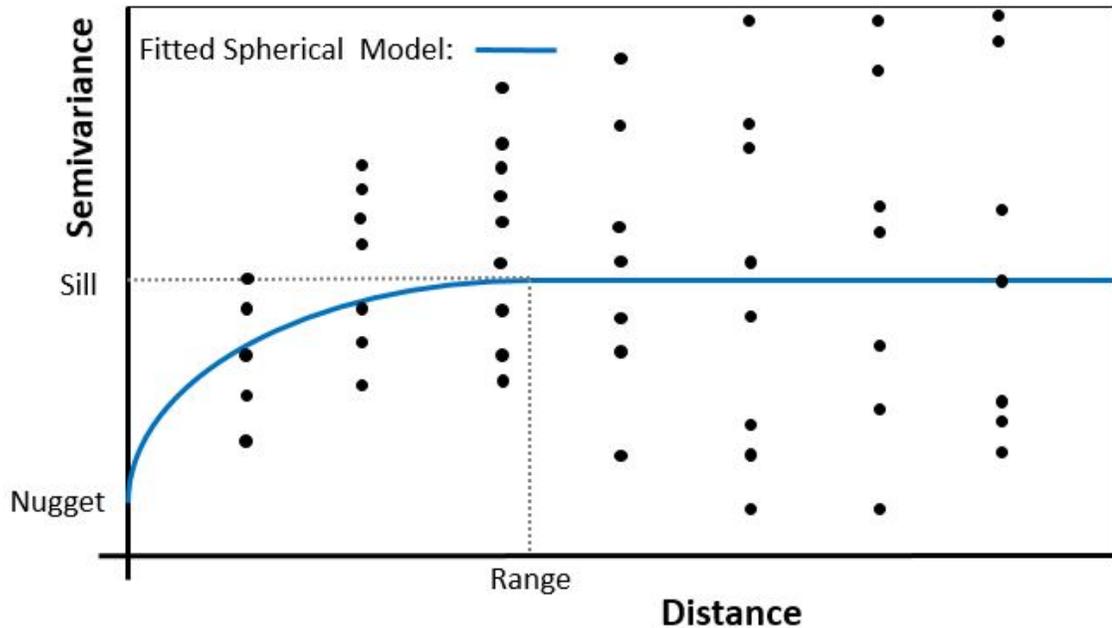


Figure 2.1.: An Example Semi-Variogram with a Fitted Spherical Model and Nugget, Range, and Sill Indicated.

In addition to setting the accuracy of the model, the semi-variogram also reveals several important attributes of dataset: the nugget, range, and sill (fig. 2.1). The nugget is the intercept of the fitted model, and it represents the amount of variation unexplained by distance. The nugget effect takes its name from the mining industry: it is possible for a gold nugget to be included in the sample core, and if there were, it would be very likely that a core taken immediately adjacent would not contain this super-high concentration of gold (Burgess and Webster, 1980). Variations in gold seams occur at a far smaller scale than sampling operations, thus, there is a certain amount of variation that cannot be explained by distance alone. Of course, in theory, a perfect model would have such a fine scale (infinitely small steps) that the nugget would always be zero, but this is of course impossible for physical sampling operations, which are confined to a finite scale (Oliver and Webster, 1986). In addition to the scale of the physical variation, the nugget is also affected by the sampling scheme and subsequent modeling. If the sampling scheme is not sufficiently dense, there will be

few semi-variance values to fit to at lower distances, and the model is likely to have a larger nugget effect, as more small scale variations will fall below the scale of the sampling. Moving past the nugget, the next dataset features are the range and sill. The range and sill, also indicated in figure 2.1 , both occur at the flattening inflection point. The range, indicated as the lag on the x-axis where the model flattens out, represents the point where distance no longer explains semi-variance between samples. For example, in a field sized survey area, two points with a 10 meter lag would be expected to be significantly correlated, but two points with a 500 meter lag would be expected to show little correlation, if any. The range is then the lag beyond which can be assumed that two points are essentially uncorrelated. The sill, on the other hand, is the corresponding point on the y-axis where the model flattens out. The sill is the maximum expected semi-variance between two data points beyond the range. If the range is shown to be 100 meters, the maximum expected semi-variance between points separated by 110 meters and 1000 meters is the same, the sill. Each feature, the nugget, range, and sill, is useful to the modeler as an assessment of both the physical structure of variation in the field and model performance.

As the entire product of kriging is based on the fitted model, a careful assessment of the semi-variogram and fitted model is indeed prudent for the modeler. While the features described above are useful metrics, the modeler must still have an understanding of the limitations of the model and data. As an example of model limitation, the idea of a range and a sill assumes that there is a clearly observed point where distance and semi-variance are no longer related, and that there is a well defined maximum semi-variance in the data. This may or may not be a valid assumption. For instance, a semi-variogram constructed for an agricultural field may not observe a clearly defined sill, if the physical variation is at a large enough scale. If a spherical model is fit to the data, there will be a range and sill, but this will not be representative of the observed data. In this case, another model type, like linear or exponential, may be able to fit the data more closely. In other times, the experimental data may not be dense enough to yield a strong representation of the area of interest, even

considering several types of models. In this case, the semi-variogram may still be a useful tool, as it could inform a secondary field sampling effort. For instance, if the nugget is very large, this could mean the sampling regime wasn't dense enough to capture variation. If resources allow, the dataset, and subsequently the model, could likely be improved by additional sampling (Mcbratney et al., 1981). While this is often impractical in an agricultural context, just understanding the limitations of the dataset, model and subsequent map is still very useful in any application. As with any algorithm or model, using kriging or other geostatistical methods in a Black-box mode is always precarious.

Since the soil science community first adopted kriging in the 1980's (Burgess and Webster, 1980), the method has enjoyed consistent usage (Hengl et al., 2017). Presumably, this is due to the fact that under ideal dataset conditions, kriging is a best linear unbiased predictor (BLUP) (Cressie, 1990). As the BLUP, kriging has been used in a multitude of applications ranging from geology to climatology to hydrology Hengl et al. (2017). Additionally, as it is the BLUP, kriging is often used as the benchmarking method in DSM studies developing new interpolation models (Ramcharan et al., 2018; Vaysse and Lagacherie, 2017; Hengl et al., 2018).

## **2.2 Machine Learning in Digital Soil Mapping**

### **2.2.1 Common Algorithms**

Many ML models have been utilized at some point in DSM studies: Logistic Regression Models (Kempen et al., 2009), k-Nearest Neighbor (Mansuy et al., 2014), and Artificial Neural Networks (Aitkenhead and Coull, 2016), to name a few. These studies have estimated soil class membership (Kempen et al., 2009), percent OM, bulk density, clay content, etc., (Mansuy et al., 2014), and even soil carbon stocks at a variety of depths in a variety of forms (Aitkenhead and Coull, 2016). Though many models have been used to estimate many things in DSM studies in the past, one algorithm family is favored: tree-based algorithms (Heung et al., 2016). First

formalized in 1984 (Breiman et al., 1984), tree-based algorithms were applied to soil mapping as early as 1999 (Bui et al., 1999), and have enjoyed continued popularity in the field (Guo et al., 2015; Subburayalu et al., 2014; Taghizadeh-Mehrjardi et al., 2014; Behrens et al., 2010). One particular variant, Random Forests (RF) (Breiman, 2001), has been shown to be among the best ML algorithms currently used by the DSM community (Heung et al., 2016), and as such has been the core of many recent works (Ramcharan et al., 2018; Hengl et al., 2017). Additionally, RF is a robust and flexible algorithm that can be tuned to minimize model overfitting.

Advanced though it is, the heart of the RF algorithm is the first tree-based algorithm: Classification and Regression Trees, or CART (Breiman et al., 1984). The CART algorithm is essentially a recursive 1D linear classifier (fig. 2.2 & fig. 2.3). The goal of the 1D classifier is to divide the dataset into two datasets that are as dissimilar as possible to each other and as internally homogeneous as possible. Once the dataset has been split along the explanatory variable (or feature) axis and value that meets this requirement, the process is recursively continued on both resulting datasets and their children (fig. 2.3). Once a child dataset reaches a desired purity or minimum number of values, the splitting process is stopped and that child dataset becomes a leaf on the tree. To make a prediction with the trained tree, the unlabeled data is fed in, and according to the values of its features, follows the path of the splits, or nodes, down to the terminal leaf. There, the values in the leaf are used to make a prediction of the likely value of the unlabeled data. This is illustrated by figure 2.4, where landform attributes at percent OM observation locations are used to predict percent OM.

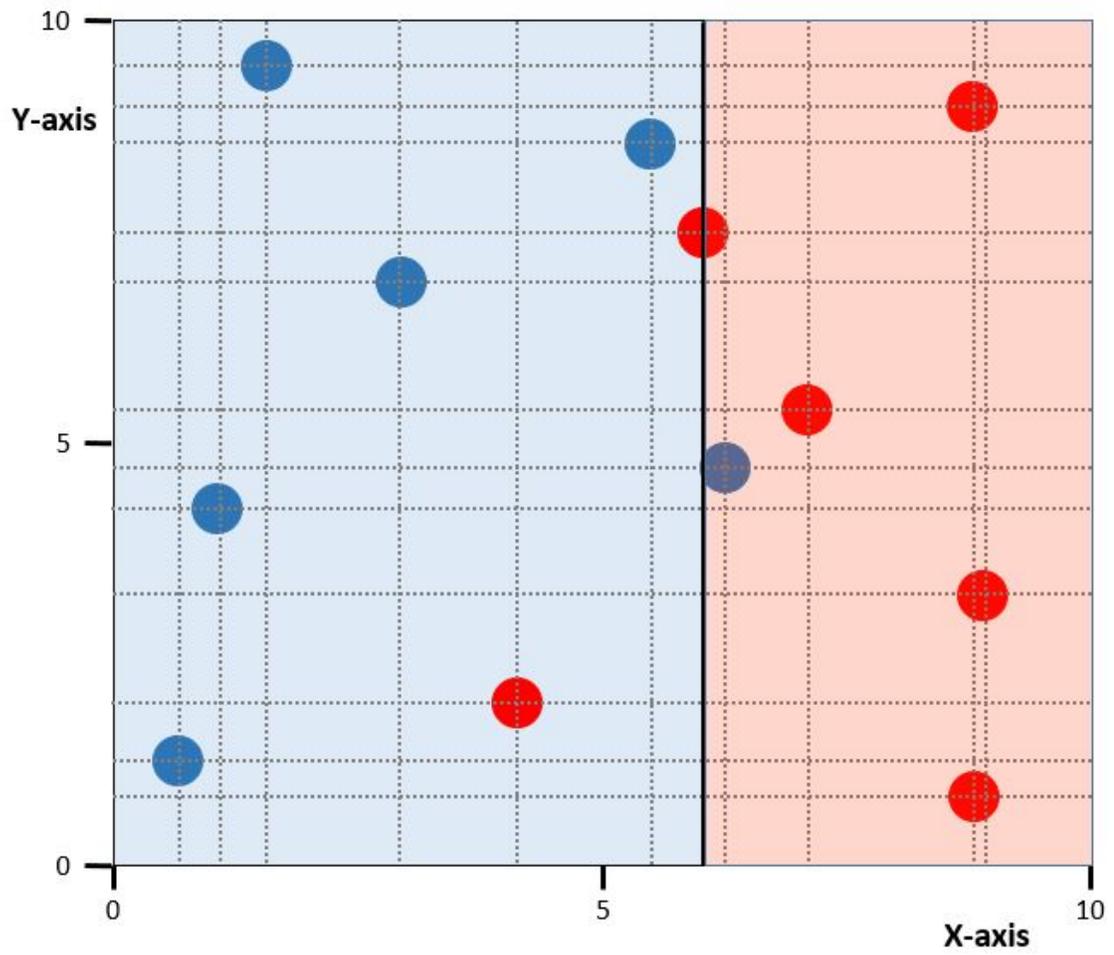


Figure 2.2.: An Example of One-Dimensional Linear Classifier. Shown with Split (Solid Line) and All Possible Splits (Dashed Lines).

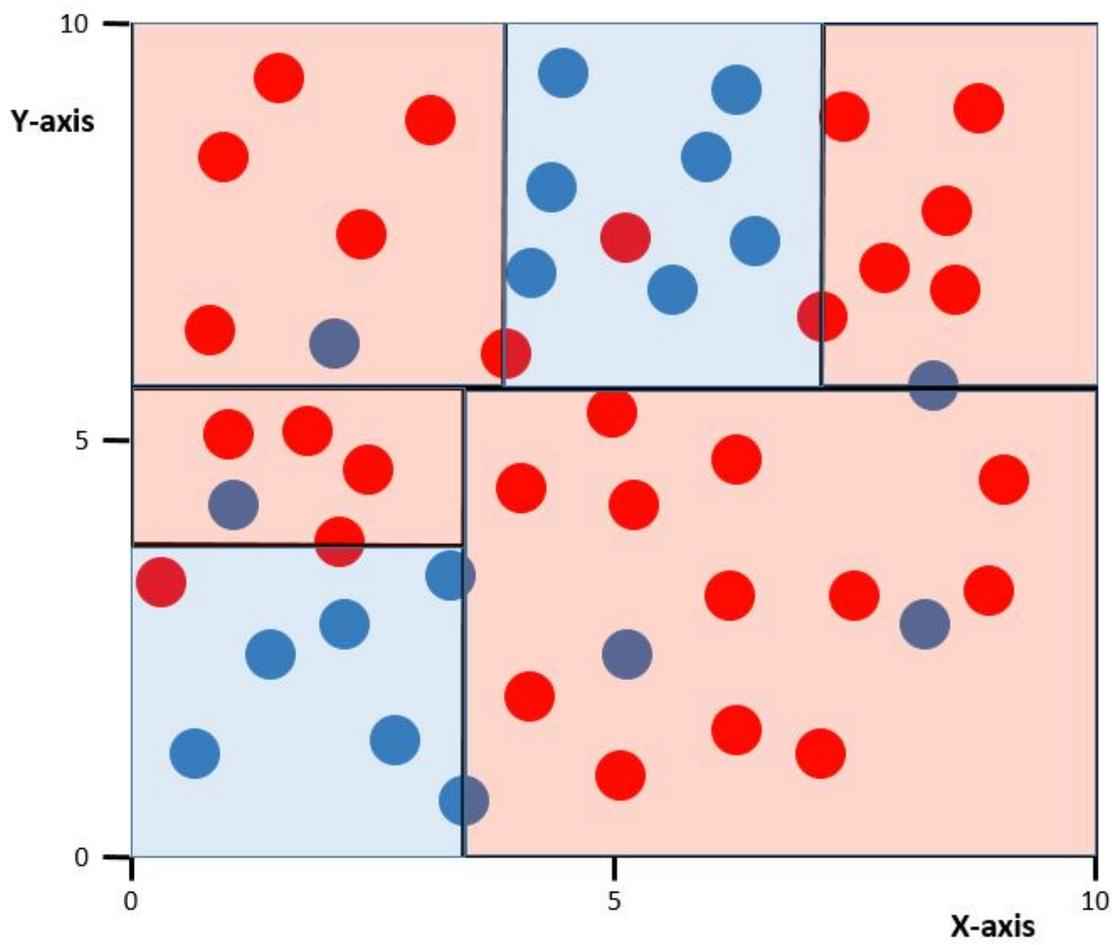


Figure 2.3.: An Example of Recursive One-Dimensional Linear Classifier. Displays Three Nodes of Recursive Splits, Shown as Solid Lines.

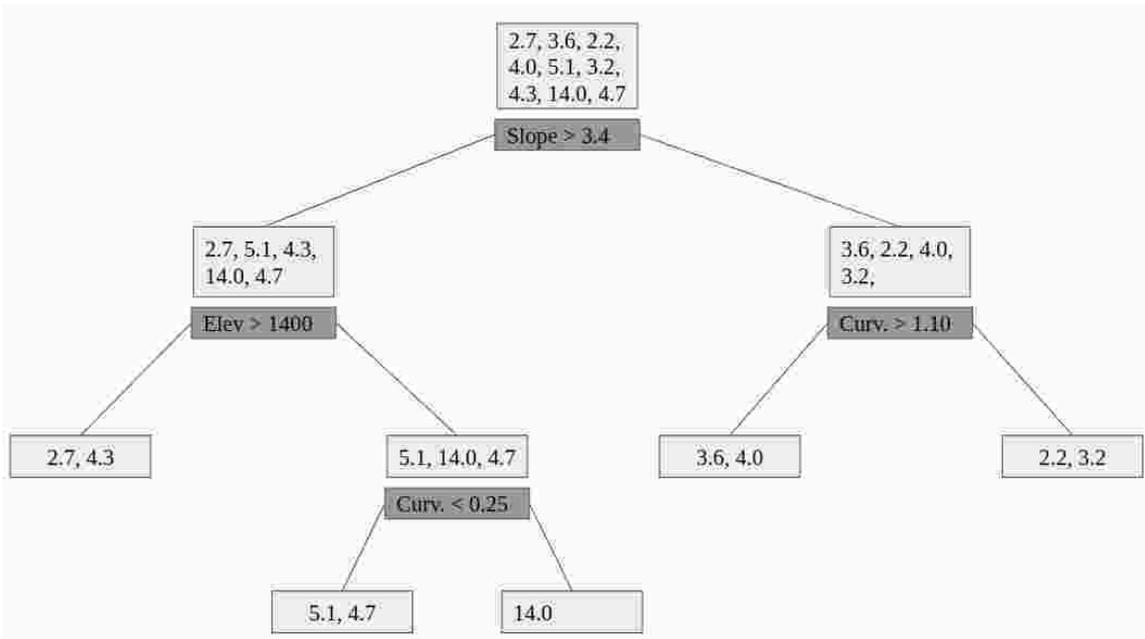


Figure 2.4.: Example Regression Tree, Using the Landform Attributes Observed at Percent OM Observation Locations to Predict Percent OM.

This process of recursive division gives the CART algorithm several useful properties. It is able to model non-linear, non-normal relationships, and additionally is suited to capture interactions between predictor variables, where the response of the predicted value to a change in a predictor is linked to the value of several other predictors (Heung et al., 2016). These properties are very useful in DSM, as the relationship of soil properties and environmental variables is often quite complex (McBratney et al., 2003). The main limitation of the CART algorithm is that very subtle relationships, present in only a subset of the data points or predictive variables, may be overlooked, as other, stronger relationships will give more clean divisions at the node. To combat this limitation, RF uses an ensemble, or Forest, of trees, all of which are trained on a random subset of the training data and features. By training each tree on a random subset of the data (drawn with replacement), and choosing each split point from a random subset of the features, each tree detects slightly different relationships in the data and outputs slightly different predictions for each data

entry. To make a prediction, RF simply averages all the predictions of all the trees, and returns that as the output.

RF has a selection of user defined parameters that could affect the efficacy of the model. The most obvious parameter the modeler can set is the number of trees in the forest; the more trees, the better the algorithm detects subtle relationships. The return does converge at some point, but in general, there is no harm (aside from computational speed) in training more trees. Two related parameters are the minimum number of observations that can be in a leaf, and the maximum depth of splits allowed. These two protect the model from overfitting by forcing a degree of heterogeneity into each terminal leaf. Lastly, there are two parameters that introduce randomness into the creation of each tree: the size of the sub-sampled set used to train each tree, and the number of features to consider when looking for a split. As with all models, to generate a quality output, the parameters must be tuned carefully, with a knowledge of the model limitations.

### **2.2.2 Explanatory Variables**

In any predictive effort (in any field), a set of known independent variables with a relationship to the variable of interest is required. In each case, the nature of the correlation between the variable of interest and these independent variables is somehow approximated, then used to make predictions of the variable of interest. As these independent variables (called features in the field of ML) form the base of all predictions, the selection of applicable features is of crucial to any predictive work. Features useful to soil prediction in DSM are summarized in seven categories: other soil attributes, climate, organisms, relief, parent material, age, and spatial position (McBratney et al., 2003). While each of these categories of features has been shown to have meaningful relationships with soil variation, not every category will have available datasets at the scale of the work. For instance, clay content and percent OM are highly related to CEC (Helling et al., 1964), but very seldom are these at-

tributes available at a higher density than CEC itself, rendering them unhelpful as explanatory variables. Thus, studies at smaller scales and higher resolutions often have fewer features available to them than studies at larger scales with coarser resolutions. Additionally, the scale of variation of many of the feature categories (climate, organisms, parent material and age) is often larger than the scale of smaller works; rainfall (a climate proxy) is unlikely to change much over a field. So, not only do smaller scale works have fewer features available, many of the features available may be more or less constant over the area of interest. The observation is illustrated well by the comparison of two recent studies. A 2017 study predicting soil properties at a 250 m resolution for the entire globe used over a hundred features, representing every category but age and spatial position (Hengl et al., 2017). Conversely, a 2015 study working at a watershed scale used only seven features, representing only relief and organisms (Zhu et al., 2015).

One feature category that is consistently used in every scale is relief, as a base dataset is typically available at higher resolutions and the scale of variation is typically quite small. In addition to the base dataset, i.e. elevation, many previous studies included other topographic derivatives (like slope, curvatures, and hydrologic features) to help represent the various processes by which soil properties are influenced by relief (Moore et al., 1993; Zhu et al., 1996; Hengl et al., 2004, 2017; Ramcharan et al., 2018). For example, soil on a steeper slope (the first derivative of elevation) is likely to have undergone more erosive processes during formation, and as such would be expected to have thinner topsoil and lower fertility. However, if that same slope coincides with concave curvatures (the second derivative of elevation, taken along various planes), it might be the toeslope of a hill, which could result in more soil deposition over time, thickening topsoil and increasing fertility despite the slope. Of course, these topographic processes occur over many scales, often with overlapping effects; a gully could be on hill which could be in a river valley. To compensate for these multi-scaled, layered phenomenon, some studies have used slope and curvature calculated at several different scales to increase the accuracy of DSM predictions (Smith et al.,

2006; Behrens et al., 2010; Zhu et al., 2008). These multi-scale DEM derivatives were created by calculating slope and curvature for several raster neighborhood sizes (3x3, 9x9, 27x27, etc.). Additionally, water flow and retention (governed by relief as well) also play an important role in local soil formation: a depressional area that retains water is more likely to have a build up of organic material over time, which would result in higher fertility and percent OM. Thus, many studies also use topographic wetness index (TWI), a DEM derivative representing the theoretical degree of perpetual wetness of a raster cell, to represent hydrologic processes, in the hope of explaining some observed variation in soil properties (Zhu et al., 2015; Hengl et al., 2017; Ramcharan et al., 2018).

By comparison with relief, fewer ML DSM studies use spatial position as a feature, though inherently any geostatistical DSM study relies heavily on spatial position. This is likely due to the fact that most ML DSM studies use algorithms that inherently have no spatial context: RF, a commonly used algorithm, considers each raster cell as a distinct individual, separate from its neighbors. Using spatial position as a feature presents a challenge then, as it requires either a new algorithm or some feature representing spatial position at each cell. One recent study attempted the former, replacing more typical algorithms with a convolutional neural network, which considers a neighborhood of raster cells when predicting the central cell (Padarian et al., 2019). Another study attempted the latter, including indices of spatial position as features for a RF implementation (Hengl et al., 2018). The second study experimented with two representations of spatial position: simply the latitude and longitude, and distances to known points. Interestingly, it was found that the distances to known points features performed better than the geographic coordinates features, as the coordinates features seemed to output strong linear artifacts in the prediction. However, details aside, ML studies that have included spatial position have seen noticeable decreases in prediction error.

### 2.3 Conclusion

Geostatistical methods, specifically Kriging, are numerically sound and well proven in the field of DSM. However, to ensure numerical accuracy, they must be applied carefully, and with an understanding of both the method and the data. ML algorithms, on the other hand, are noted for their flexibility and ability to easily include a host of features to aid in prediction. Despite the flexibility however, a ML model is typically less interpretable than a comparable geostatistical model, which means the output is less easily verified. That said, the results of both methods are of unknown accuracy when applied in a "black box fashion," and the results of both methods are exceptionally useful when applied knowledgeably. Bearing all this in mind, the potential of well done ML predictive efforts are still being explored, and there is much room for optimization and novel applications. No ML DSM studies were found in the review of the literature that attempted to create private field soil maps from private soil samples, as is done with geostatistical methods today. Thus, a novel scope exists for developing and evaluating an application of ML that uses publicly available datasets for developing field soil maps.

## 3. METHODS

### 3.1 Objectives 1 & 2

#### 3.1.1 The Case Study

The soil data used in this research was provided by the Davis-Purdue Agricultural Center (DPAC). The 22.26 ha sampled site (fig. 3.1) is located in Randolph County, in East Central Indiana. The elevation of the site ranges from 293.2 to 297.2 m, and the landforms are gentle; the slope of the site averages 1.1 degrees, with a maximum of 9.5 degrees. The parent material of the site is glacial till, and before the land was cleared for agricultural use, the native landscape was deciduous forest. The site is now entirely cultivated, aside from a constructed waterway on the southern side. The field boundary of the site was not provided, but was created in ArcMap using satellite imagery and the soil sample data to guide the selection of field boundaries (fig. 3.1).

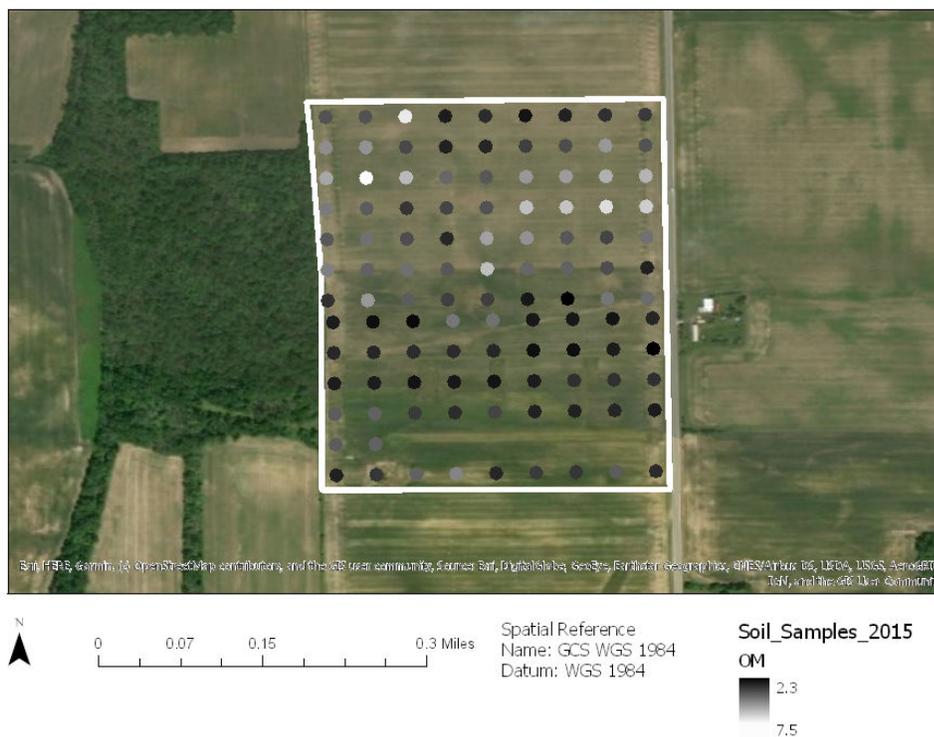


Figure 3.1.: Site Boundary and Soil Sample Locations. Soil Percent Organic Matter Values are Color-Mapped onto the Point Locations.

A total of 110 soil samples were collected with a density of about 0.20 ha per sample (fig. 3.1). At each sample location, 0.15-0.20 m deep sample cores were collected from several locations within a 3 m radius of the sample coordinates. These cores were aggregated into a single representative sample, analyzed for a collection of attributes of agricultural interest, and the sample values were attributed to the sample location.

Of these available attributes, percent OM was selected as the variable of interest for this study for three reasons. First, percent OM is of great agronomic importance. For one, a certain amount of plant available nitrogen is mineralized from OM in the soil every growing season, meaning some areas of the field require less fertilizer to

reach desirable nutrient levels (Ferguson et al., 2002). For another, percent OM has a strong influence on the CEC of soil, a fertility measure that impacts the amount of micro and macro nutrients that can be held in the soil (Helling et al., 1964). And as a final example, OM in the soil acts as a buffer against pH change; thus, when calculating lime application rates, both the soil pH and the percent OM must be considered (Keeney and Corey, 1963). The second reason was that percent OM is known to vary greatly within fields, often at a scale well below practical sampling limits (Wollenhaupt et al., 1997). This provides a modeling challenge, as much variation was likely not captured by the soil samples, and requires a strong model. The third reason is that while many ML DSM studies predict percent OM (Moore et al., 1993; Hengl et al., 2017; Ramcharan et al., 2018), no studies were found that attempted to predict percent OM using a higher resolution DEM than that available in the State of Indiana (1.52 m). In this field, the percent OM varied from 2.3 to 7.5 (fig. 3.1), with most points nearer the low range than the high (fig. 3.2). While the error of these samples is unknown, a study conducted in Idaho found that on average, the standard deviation of repeated soil percent OM samples was about 0.86% (percent being the unit, not 0.86% of the value range) (Hoskinson et al., 2004). While this seems large, considering the limited amount of variation visible in the point values (fig 3.1), it does point out that there is some amount of error in this dataset that any model must contend with.

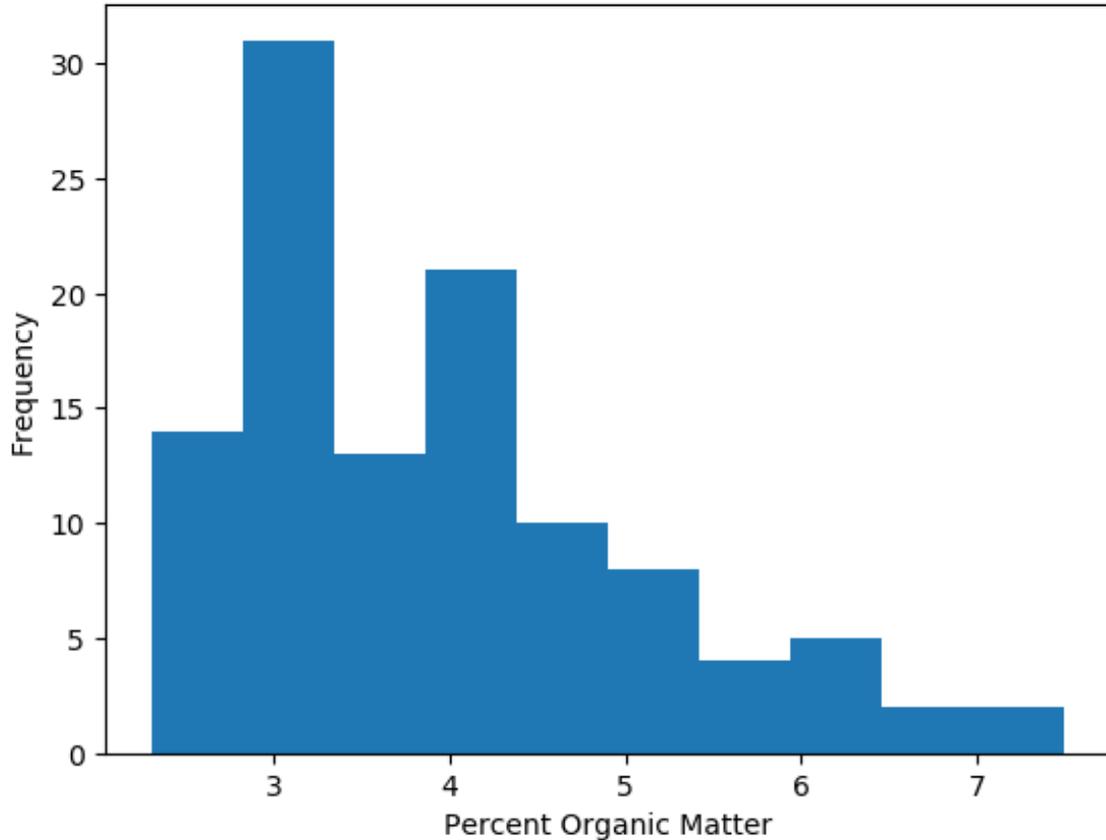


Figure 3.2.: A Histogram of the 110 Percent OM Values.

Typical sampling densities in industry are much less dense than the case study dataset: the most common sampling densities are 2.02 ha, 1.01 ha, and occasionally 0.40 ha per sample (Erickson and Widmar, 2015), whereas the sampling density of this soil data was 0.20 ha per sample. To more closely mimic a typical industry conditions, a smaller subgrid of 28 points was selected for interpolation from the original 110 samples dataset. This subset (fig. 3.3, in red), approximately a quarter of the overall data, reduced the sampling density to a more industry comparable 0.795 ha per sample while still preserving the grid sampling pattern. The remaining 82 sample points from the original set were withheld from the interpolation process and later used for validation (fig. 3.3, in blue).



Figure 3.3.: Points Selected For Training, Validation Points, and Site Boundary. Points Selected For Interpolation are Displayed in Red, While Points Kept For Validation are Displayed in Blue.

### 3.1.2 Public Data

The base dataset used to represent relief was a publicly available Digital Elevation Model (DEM). While there was a range of DEM datasets available, the 2011-2013 LIDAR survey of the State of Indiana was chosen, due to its combination of coverage and resolution (State of Indiana, 2011). As the State hosts the DEM tiles on OpenTopography (<https://opentopography.org/>), the DEM for the case study location was retrieved by manually selecting an area of interest in the selection User Interface (UI) on the OpenTopography website. The DEM had a resolution of 1.52 m, and was

in the Indiana State Plane East projection, which uses US Survey feet as its official horizontal measure. As it was desired to conduct analysis in meters, and as it was later planned to automate DEM downloading and processing for all of Indiana, it was decided to reproject the DEM into UTM zone 16, which encompasses all of Indiana. The reprojection was handled using a `gdalwarp` command line operation, which made for simple automation for objective three.

### 3.1.3 Benchmarking Methods

The first benchmarking method of the study was a simple whole field average. The first step in creating this baseline benchmark was to find the average percent OM value of the 28 sample points. To do this, the attribute table of the dataset was opened in ArcMap (Esri, 2018) and a window was opened with the summary statistics of the percent OM column. Next, in a Python script, the Rasterio library (Gillies and Perry, 2019) was used to open the elevation raster as a template. A Numpy (NumFOCUS Foundation, 2019) array of the same shape as the elevation array was created and filled with the average value. This average array was then written out as a GeoTIFF file with the same resolution, shape, projection, and grid orientation as the case study DEM. Finally, the `gdalwarp` command, part of the `gdal` library (GDAL/OGR contributors, 2019), was used to constrain the DEM to the field boundary.

The second benchmarking method was a substitute for grid interpolation, nearest-neighbor. To create this prediction soil surface, a Python script first used the Rasterio library to open every raster of euclidean distance to known points and load the values as Numpy arrays. Next, a new array of the same shape was created and filled with nodata values. Then, a function looped through each row and column of the arrays; examining which distance array had the smallest value. If a euclidean distance array had the smallest value, this meant that it was the closest sample point to that cell. Thus, the value of that sample point was assumed to be value of the cell at that

index, as it was closest. Once every point had been evaluated, the resulting array was written out as GeoTIFF file with the same metadata as the elevation raster, and the gdalwarp library was used to constrain the DEM to the field boundary.

Since the third method, Ordinary Kriging (OK), attempted to fit a model to the data, several assumptions about the dataset were examined. First, it was visually checked that spatial autocorrelation was present in the data using a semi-variogram; if this wasn't valid, the model would've been very weak. Second, clustering of data points was visually checked. Third, the dataset values were examined for a normal distribution. Fourth was check for trends in the data (as OK assumes a constant, unknown mean and estimates the residuals). Lastly, the data was examined for stationarity, or in other words, it was checked that the level of autocorrelation should be the same throughout the map; there should not be large areas of uniform values and small areas of highly variant values. As this list of assumptions was rather extensive, some of these assumptions were violated, and required tailored corrections to be applied before applying OK. To streamline all assumption checks and corrections, each was examined by using the explore data tab of the ArcMap geostatistical analyst extension, as these pre-made tools limited human error. Additionally, once all assumptions were complete, all corrections and model fitting were done in the ArcMap geostatistical wizard, as this limited human error and allows for automatic parameter optimization. After selecting the dataset and column value to kriging and the type of kriging, two dataset corrections were applied in the wizard: a transformation and a trend removal. Next, after selecting the semi-variogram model type, the wizard automatically optimized the model parameters (lag size, number of lags) to achieve the best fit. After the wizard output a prediction layer, the result was exported as a raster, with the same resolution, grid, and projection as the elevation dataset, and clipped to the field boundary.

### 3.1.4 Developing the Prediction Framework

#### Machine Learning Implementation

The particular implementation of RF used in this study was the `RandomForestRegressor` Python package published by `scikit-learn` (Pedregosa et al., 2011). This particular package was chosen for a variety of reasons (ease of use, well proven, etc.), but the most significant reason was that a metric of feature importance could be retrieved from the trained model (Pedregosa et al., 2011). After the model was trained, the Mean Decrease in Impurity (MDI) (Breiman et al., 1984) could be calculated for each explanatory variable (or feature) and used to generate a rough metric of how useful each feature was for dividing the dataset. This was very useful while prototyping which features would be included in the model featureset, as it provided a rapid metric for each feature, and was much more objective than training and testing the model with and without every possible feature. It is important to note that the size of sub-sampled set for each tree in the `scikit` implementation was hard coded to the size of the input set, but as this sub-sample was still generated by randomly drawing with replacement, it was considered random enough (Pedregosa et al., 2011).

As the dataset size was smaller in comparison to most DSM studies, and overfitting is a common issue with small datasets, it was decided to also implement another RF variant that was potentially more resistant to model overfitting. This variant was another Python package published by `scikit-learn`, `ExtraTreesRegressor`. It was chosen because the two algorithm implementations were very similar, requiring the same inputs and outputting data in the same way. This allowed for very simple tuning, as the `ExtraTrees` implementation was similar enough that both algorithms responded worked best with identical parameters. The difference between the two was that the `ExtraTrees` implementation added another layer of randomness when selecting split points. The first implementation, `RandomForests`, would check every value of every feature for a potential split, then choose the feature and value that returned the best split. `ExtraTrees` would randomly generate possible splitting values

for every feature, then choose the feature and random value that returned the best split. While this potentially reduced the ability of the algorithm to fit the data, it added another defense against fitting to exactly the values present in the limited dataset. As these algorithms were to be fitted to a very small dataset, it seemed worthwhile to include.

When tuning the models, the first objective in building the RF models was to make sure both RF algorithms were robust to the low data conditions. To achieve this, it was determined to tune the parameters (table 4.4) to keep the forests shallow and wide. By keeping the fit of each tree shallow, the likelihood of fitting the model too closely to relationships only found in the training data, or overfitting, is reduced. This is done by restricting the minimum number of observations in any particular leaf to a large (compared to the dataset) value, which limits the number of splits and forces heterogeneity in the prediction nodes of every tree. However, the shallowness of the individual trees limits the ability of any one tree to learn subtle relationships present; to offset this, a very large (or wide, keeping with the metaphor) ensemble of trees must be created. If each tree can detect a small portion of the subtle relationships, then the ensemble captures every relationship that can be captured. The larger the ensemble, the better the prediction.

As another check against overfitting, the picture of the data that each tree is built on was randomized. The more random the subset of data used to train each tree, the more likely each tree will detect a subtle relationship present in the data subset; this makes the ensemble much less likely to overfit to obvious relationships present in the training data (Breiman, 2001). Another parameter that introduces randomness during training is limiting which features could be evaluated at each splitting node. Randomly selecting a subset of features kept the trees from all splitting at identical split points, and forced the ensemble away from the properties only present in the sample observations.

### 3.1.5 Feature Data

#### Representing Relief

In addition to elevation, many studies found value in adding other DEM derivatives to the feautreset, like slope, curvatures, and topographic wetness index (TWI) (Moore et al., 1993; Zhu et al., 1996; Hengl et al., 2004, 2017; Ramcharan et al., 2018). With this body of works in mind, it was determined to include elevation, slope, curvatures, and TWI in the initial featureset. In the case of the first derivatives, slope and curvatures, some studies found value in calculating them over various neighborhood sizes (Smith et al., 2006; Behrens et al., 2010; Zhu et al., 2008). Typical geospatial softwares calculate these derivatives using the 3x3 neighborhood around each cell, which the slope length of interest is set by the raster resolution (Smith et al., 2006). Thus, if the resolution of the raster is 1.52 m, the slope is calculated at 4.56 m. This neglects the complex nature of topography however; as a small rise can be in a low area, which in turn could be in a valley, each contributing to the formation of the soil there. To represent these multi-scale processes, a Python script had to be written to calculate slope and curvatures over a varying neighborhoods. Before any the slopes and curvatures could be calculated however, the DEM first needed prepared for the sliding window operations needed to calculate the slopes and curvatures. In a sliding window operation, the cells in a square "neighborhood" around the cell of interest are used to generate the value of the cell of interest. If a 9x9 neighborhood is used, then the focal cell will be at the center of a neighborhood of cells nine wide and nine high. The challenge of this operation is that cells near the edge of the DEM may not have a full neighborhood: the cell on the edge won't have other values to fill the neighborhood, and therefore cannot be estimated. To be used as a feature, every cell needed for the prediction should have real values, and therefore cannot be an edge cell without a neighborhood. To make sure that every cell in the area of interest has real values, the DEM needs to have a buffer of cells outside the area of interest, large enough that every cell in the area of interest can have a full neighborhood.

To make sure the DEM had this buffer while still limiting the processing required to create slope and curvatures, the DEM was clipped a bounding box with a large buffer distance from the field boundary. To do this, the field boundary shapefile was loaded into the Python library Geopandas (McBride et al., 2019), and a new boundary object was created with a 150 m buffer from the original boundary, using the `geopandas.buffer()` function. Then, a new object was created to represent the bounding box of this buffered boundary, using the `geopandas.envelope` function. This rectangular bounding box, at minimum 150 m larger than the field boundary, was then written out as a shapefile. This new shapefile was then used in a `gdalwarp` command line operation to clip the DEM to an area at with edges at least 150 m away from any point of interest. This preprocessing kept the original resolution, projection, and grid orientation of the DEM, and made sure there was enough room on the edge for every point of interest to have a full neighborhood for sliding window operation. Additionally, all subsequent features were created at the resolution, projection, and grid orientation as the buffered DEM.

In the Python script for calculating slope and curvatures at custom neighborhood sizes, the DEM was first read in by the `rasterio` package as a 2-d numpy array, then, a sliding `X-x-X` neighborhood around each cell, with `X` set by the neighborhood size, was passed on for calculations. Then, the `scikit-learn` library, `linalg` (Pedregosa et al., 2011), was used to fit a 3D quadratic function to the neighborhood (using ordinary least squares (OLS) estimators). After the elevation surface was fitted, slope, profile curvature, and plan curvature were calculated as described by Smith et al. (2018, section 6.2) and attributed to the focal cell. Finally, after analysis was done, these arrays were written out as GeoTIFF files with the same projection and grid as the DEM. As a starting point, this was repeated at several neighborhood sizes, `5x5`, `25x25`, `75x75`, & `115x115`, and these were included as features in the initial feature set. In contrast to the multi-neighborhood curvatures, there were already established tools to calculate TWI, and ArcMap was used to calculate TWI rather than a custom Python

script. The TWI was calculated using the process flow described by Zimmerman and Shallenberger (2016) and the equation given below.

$$TWI = \ln \frac{\text{Flow Accumulation}}{\text{Slope}} \quad (3.1)$$

Along with TWI, several other hydrologic features were included in the initial feature set, as it was thought they could capture or better represent other hydrologic processes. The first additional feature was flow accumulation (calculated using the ArcMap toolbox), which was thought to represent perennial streams and gullies, rather than wetness like TWI. The second was Horizontal Distance to Stream Network, which was thought to represent possible erosive processes; the closer to concentrated flow, the more erosion, so it was thought. This was calculated by thresholding the flow accumulation raster in the ArcMap raster calculator to extract a stream network raster (1 if a stream, 0 otherwise). Then, the euclidean distance tool in the ArcMap toolbox was used to find the Euclidean distance to the stream network for each cell. Once created, TWI, flow accumulation, and horizontal distance to stream network were added to the initial featureset.

### Representing Spatial Position

It was determined to represent spatial position by using rasters of euclidean distance to each known point, as described by (Hengl et al., 2018). To create a raster for every point (fig. 3.4), the soil sample shapefile was read into a Python script with the geopandas library. Next, each soil sample was individually selected from the sample set and passed into a rasterizing function. Within the function, the Python rasterio library used the features sub-library to rasterize the point into an array with the same shape as the elevation feature. After the rasterized array was created, a boolean mask was created using numpy logical functions, showing the rasterized point as 1 and all nodata values as 0. This array was then written out to a GeoTIFF file (with meta matching the elevation feature) with rasterio i/o functions. Once all the

points had been rasterized individually, a second script used the files as the base for euclidean distance analysis. One by one, each file was passed as the input to the `gdal_proximity.py` command line script, which generated a second GeoTIFF file of euclidean distance (measured in cells, for simplicity) to the single point. Finally, once each of these files were created, all 28 distance rasters were added to the initial featureset.

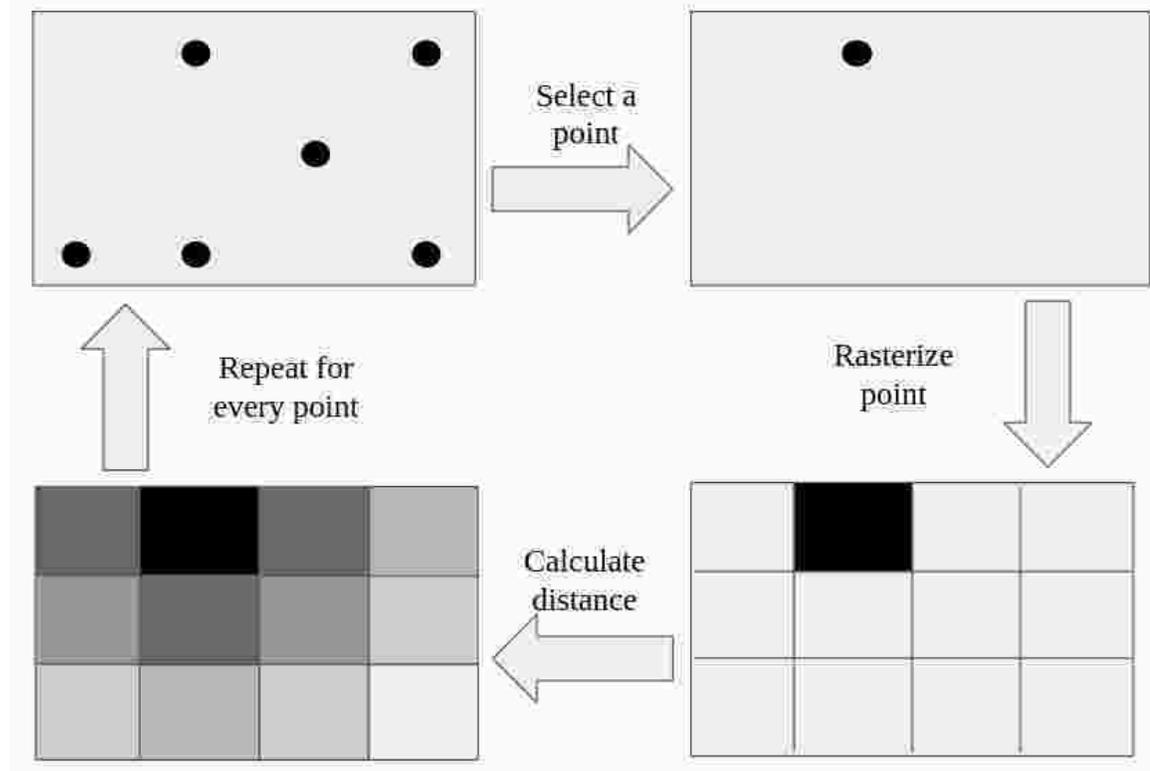


Figure 3.4.: Flow Chart Representing the Creation of Euclidean Distance Features.

Once the entire initial featureset had been created, the set of feature values corresponding to the location of each of the 28 training points was extracted (fig. 3.5). Every feature was a raster, and every raster was of identical resolution, shape, and snapped to the same grid, which allowed simple cell indexing. First, the exact position of a rasterized point was determined by checking the index of the single true value in the individual masked files. This cell index corresponded to the location of the ras-

terized point, and in turn, the value of each feature at that particular cell index was extracted and added to the feature list for that particular point. This was repeated for each sample location, and the resulting Python lists comprised the training set: target values, and the feature values that corresponded to them. A similar process was followed for creating a prediction set, except this set did not have target values.

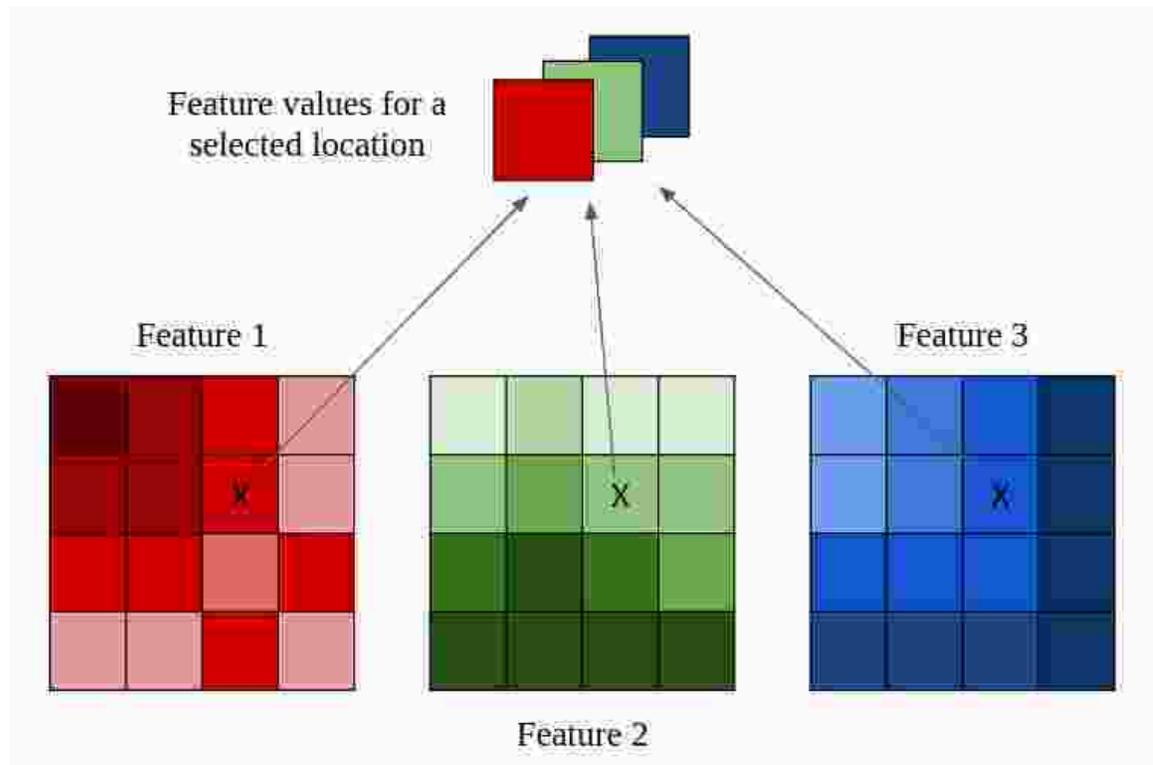


Figure 3.5.: Visual Representation of Grid Based Features and How Feature Values for a Single Location Are Extracted.

### Refining the Featureset

The initial feature set served as a starting point, but the feature set was to be iteratively modified and trimmed by training a prototype model and examining the feature importances that the trained model output. The feature importance scores output by the model were a measure of how decisive a feature was in splitting the data at tree nodes: was it used often, and how cleanly did it split the data on those

occasions Breiman et al. (1984). To begin trimming the featureset, the model was trained and feature importances would be examined, relative to each other. Typically, some features stood out as strong predictors, and others stood out as poor predictors. If a feature, TWI for example, scored poorly, several more models would be trained to determine why. First the model would be trained again, but without features similar to the poor scoring feature: in the case of TWI, the other hydrologic features would be withheld. This would check if TWI was indeed a bad predictor, or if other features explained the same things that TWI explained. The process of evaluating a feature would be done iteratively, dropping other features, including them again, all at the judgment of the modeler, in an attempt to evaluate if a feature was worthwhile to include. In the end, it was the judgment of the modeler to include or remove a feature after testing.

In the interest of automating the prediction framework, it was assumed that features deemed useful in this particular case study would be useful to any field soil interpolation in the future. While this may or may not be true in any given case, (perhaps for soil pH, slope has no predictive power) it was assumed that by including all features important to this case study, some features would always stand out as predictive of the soil property.

### **3.1.6 Validation and Testing**

After finalizing parameters for the RF and OK models and producing the grid-based and field average predictions, a series of experiments was ran. The first experiment was an even footing comparison of RF and OK: the RF feature set included only distances to known points, with no topographic features. By building a spatial position only RF model, the advantage of any other explanatory variable was removed. This allowed for a direct comparison of RF and OK: any difference in accuracy had nothing to do with extra data, simply the ability of the method to make a prediction with the same data. The second experiment was an evaluation of relief alone

as an explanatory variable. The RF model was built using only relief-based features, without any spatial position data. If the model performed well, it would show that relief is related to soil property variation at a small scale, and would validate the use of relief-based features. Also, building a relief based model would allow for a relative comparison spatial position (the sole explanatory variable for experiment one) and relief, to see which feature was stronger. The third and last experiment was an RF model that used both the spatial position features from experiment one and the relief features from experiment two. This tested the viability of including more than one feature type in a field soil map prediction; an increase in prediction accuracy between the first second experiment RF results and the third experiment RF results could be directly linked to the inclusion of the relief features. Additionally, if spatial position and relief explain different variations in the percent OM, it would seem that the combined featureset model should make the strongest prediction seen in the study, providing a best case scenario.

For each model, in each experiment, a prediction for the entire area of interest based on the 28 training points was made. After the prediction was made, the raster cells closest to the 82 validation points were selected as the prediction for that particular validation point. Using this set of predicted and measured values, four common performance metrics were calculated: Mean Absolute Error (MAE), the standard deviation of the absolute Errors, Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE). MAE, calculated using the equation below, is exactly as it sounds: an average of the absolute errors, or absolute residuals. If a model had zero error, the MAE would be zero, so the smaller the MAE score, the better the performance.

Where :

$n$  = # of samples

$y$  = actual value

$\hat{y}$  = predicted value

$$MAE = \frac{1}{n} \sum_{i=1}^n |(\hat{y} - y)|$$

(3.2)

The standard deviation of the absolute errors was calculated using the equation below, to allow for statistical comparison between interpolation error sets. Using the MAE and the standard deviation of the absolute errors from two interpolations, a two sample t-test could then be ran between them, to test the null hypothesis that there is no difference between the mean absolute error of the two models under comparison. The statistical significance will be evaluated using p-value and the criteria that p at or below 0.05 would result in the rejection of null hypothesis.

Where :

$n$  = # of samples

$y$  = observed value

$\mu$  = average value

$$SD = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (y_i - \mu)^2}$$

(3.3)

Mean Absolute Percentage Error (MAPE) calculated using the equation below, is very similar to MAE, but rather than an average of the absolute errors, it is an average of the absolute percent errors. While MAE is useful, it is not always simple to get a grasp of the relative amount of error using it alone. By using the MAPE, an estimate of the percentage error in the interpolation can be reviewed.

Where :

$n$  = # of samples

$y$  = actual value

$\hat{y}$  = predicted value

$$MAPE = \frac{100\%}{n} \sum_{i=1}^n \left| \frac{(\hat{y}-y)}{y} \right| \quad (3.4)$$

While the MAE might tend to average out a few outlying high errors, Root Mean Squared Error (RMSE) calculated using the equation below, is particularly sensitive to individual large errors, as all the errors used to calculate RMSE are first squared. Because of this, it can be very interesting to compare RMSE and MAE; if the RMSE is very large in comparison to the MAE, this may indicate that there are several errors that are much larger than the rest.

Where :

$n$  = # of samples

$y$  = actual value

$\hat{y}$  = predicted value

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (\hat{y}-y)^2}{n}} \quad (3.5)$$

### 3.2 Objective 3

In order to make a means of generating high quality interpolations accessible to the public, a self serve webtool was created where users can upload their data and retrieve interpolations made by the ML framework. One challenge in creating this webtool was that the ML framework was not lightweight, either in processing or in

environmental constraints. Running on a desktop, the framework took between three to five minutes to download elevation data, create a feature set, train, predict, and write out data, depending on CPU load. This processing would use the entire core, and often slow down other processes. If this heavy processing load was put on the client's browser, their device would likely experience the same symptoms, diminishing the user experience. In the environmental sense, the framework required several specific dependencies (like gdal command line utilities) on the host machine that were unlikely to be available on a user device. To deal with this, the ML framework was split from the client run UI and hosted on a Purdue Linux server with 4GB of memory and 150 GB of disk space, running Ubuntu 14.04 and maintained by Purdue AgIT. Additionally, to remove another burden from the user, the ML parameters were set up for field soil mapping with limited sample data, which will aid the untrained user in using the tool. The webtool requests a user's geolocated soil sample data, either in .shp or .csv format. If the user chooses to upload their data in .shp format, the webtool requires them to include all required files, and allows them to select any numerical data property as the property to be interpolated. If the user chooses to upload a .csv, it has to include at least three columns of numerical data: latitude, longitude, and a property to interpolate (fig. 3.6). The user must define which row represents each mandatory field, but the webtool checks the data to make sure they're numerical.

	A	B	C	D	E
1	Latitude	Longitude	OM	CEC	PH
2	40.24683426	-85.14797936	3.5	11.1	6.8
3	40.24683042	-85.14850798	3.3	9.5	5.9
4	40.24682644	-85.14903678	3	7.1	6
5	40.24682259	-85.1495654	2.7	9.3	6.7
6	40.24681875	-85.15009402	3.2	14.6	7.7
7	40.24681477	-85.15062282	2.9	9.6	6.4
8	40.24681078	-85.15115144	7.1	15.7	6.6
9	40.24680694	-85.15168025	4	10.8	7.4
10	40.24680296	-85.15220887	4	9.7	6.9
11	40.24639797	-85.15220383	5.2	14.1	6.6
12	40.24640181	-85.15167503	5.3	12.8	7.1
13	40.24640566	-85.15114641	3.8	12.6	7
14	40.24640964	-85.15061779	3	9.7	6.1

Figure 3.6.: An Example CSV File, Containing Columns For Latitude, Longitude, and Properties of Interest.

Once the soil data is uploaded and verified, the user has the opportunity to review the data on a map. If the user is satisfied with the data, they draw their field boundary and ask to submit. If their boundary and soil data is verified, they submit their data to the server side, where it is processed and passed into the Python prediction framework. They then receive an email with their request ID, and can check the status of their request at any time using the webtool. Once their request is complete, they receive an email, and they can view the prediction output. Finally, the user downloads their interpolation as a GeoTIFF file.

The core framework of the webtool was the Reactjs JavaScript library (Facebook Open Source Contributors, 2019). The React library was used to create the backbone of the application because it has useful rendering properties, and can create a basic, starting point application with useful boilerplate. However, managing application state is somewhat complex with React alone, so Cerebraljs (CerebralJS Contributors, 2019) was used for state management. To save time and give the application a clean

cut look, Material-UI (Material-UI Contributors, 2019), another JavaScript library, was used for application components and formatting. To show geospatial data in the UI, Leafletjs (LeafletJS Contributors, 2019) is a useful library, but as it typically doesn't work well alongside React, React-Leaflet (Le Cam, 2019), a library that wraps Leaflet for React applications, was used primarily. Loading client files was difficult, but Shapefile (Bostock, 2016), another Javascript library made to read shapefiles in memory, made it simple to convert Shapefile data into GeoJSON format, a standardized vector data format for the web. This then could then be passed to Turfjs (TurfJS Contributors, 2019), a very useful geospatial library. Turf was used to validate user provided data, checking for valid shapes, reasonable area sizes, reasonable boundary lengths, and that all the sample points were within the field boundary. Finally, once all data was collected from the user, the Axios library (Zabriskie and Uraltsev, 2019) was used to communicate with the back-end.

The core framework of the back-end was a JavaScript Node Express application (Node.js Foundation, 2019b,a) that managed user requests. To keep users from overloading the server, express-rate-limit (Friedly, 2019), a node package, checked how frequently users submitted requests, and blocked fraudulent use. As the interpolation was processing intense, it was decided to have the server run on it's own, detached from the webtool, and email users when their request was done. The emailing was done using Nodemailer (Reinman, 2019), another package, and Google APIs (Google Developers, 2019), which allowed the server to send an email to the user from a professional looking account. The real machinery of the application however was a Python script, ran as a child process of the Node Express app. The script first used the Python geopandas library to perform similar checks on the user data as Turfjs in the webtool: reasonable area, valid shape, etc. After the validation was finished, the Requests Python library (Reitz, 2019) was used to fetch DEM tiles from the OpenTopography server. A collection of gdal command line operation were used for DEM processing, ran from the Python script with the subprocess Python library. Once the DEM was processed, the Python library rasterio was used to load the DEM

in and write out other raster data. While the heart of rasterio was actually gdal operations, rasterio had useful convenience functions that saved time when scripting. Anytime the raster data needed accessed, the raster would be loaded in and the band would be extracted as a numpy 2-D array, as numpy is exceptionally useful for array based data indexing and analysis. As previously stated, various packages from the scikit-learn library were used both for analysis and interpolation.

The entire development (the webtool, API, prediction framework scripts, and database) was first developed locally in a developing environment. After everything was complete, it was determined to containerize the entire application for deployment. This was accomplished by creating a docker container and customizing the virtual machine for the needs of the entire system (Docker Inc., 2019). A container that came with gdal utilities pre-installed was used as the base container, and the rest of the system dependencies were included in setup. Finally, after all development was complete, the container was deployed to a Purdue server. A domain name, <https://precision.ag.purdue.edu/soilinterpolation/>, was chosen, and the domain was rerouted to the server port, exposing the webtool and server.

## 4. RESULTS AND DISCUSSION

### 4.1 Objective 1: Establishing a Baseline

#### 4.1.1 Applying A Field Average

The numerical performance of the field average percent OM prediction held some interesting points (table 4.1). In terms of percentage error, the MAPE was 23.9%, so on average, the prediction was 23.9% off. Additionally, as the gap between the RMSE and MAE was not dramatic, this would seem to imply a relatively even spread of error values, rather than one or two large errors. As this was a prediction of the field average, this made sense.

Table 4.1. MAE Scores, The Standard Deviations of The Absolute Residual Sets, MAPE Scores, and RMSE Scores

<b>Experiment</b>	<b>MAE</b>	<b>STDEV</b>	<b>MAPE</b>	<b>RMSE</b>
Field Average	0.909	0.685	23.9%	1.136
Grid	0.973	1.005	24.4%	1.395
Ordinary Kriging	0.959	0.973	23.6%	1.362
Random Forest: Spatial features only	0.779	0.599	20.3%	0.981
Extra Trees: Spatial features only	0.779	0.604	20.3%	0.983
Random Forest: Relief features only	0.715	0.535	18.3%	0.891
Extra Trees: Relief features only	0.739	0.565	19.1%	0.928
Random Forest: Spatial and relief features	0.678	0.504	17.5%	0.843
Extra Trees: Spatial and relief features	0.688	0.524	17.9%	0.863

The visual prediction (fig. 4.1) offers little in the way of insights, as there is no variation in the predicted value of percent OM, 3.957. However, it does show the shape of the field, and offer a reference point for prediction comparisons (the color-map for all predictions is set to the same scale).

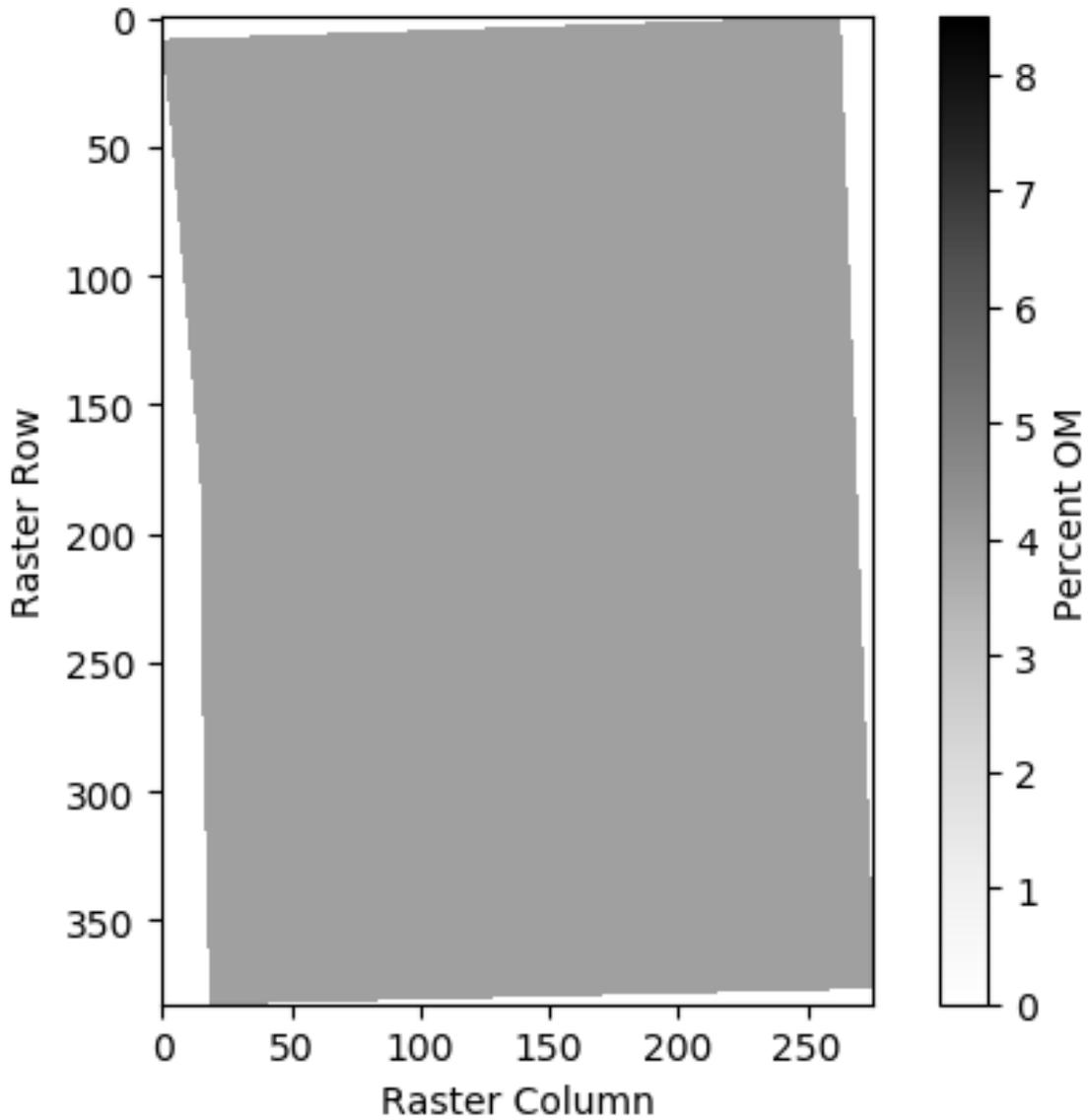


Figure 4.1.: Predicted Percent OM, Using The Field Average.

### 4.1.2 Applying A Grid Prediction

The MAE, MAPE, and RMSE were larger for the grid prediction than for the field average (table 4.1), though there was no significant difference in the absolute residual sets between the models (table 4.2 & table 4.3), which implies no significant performance difference. The gap between MAE and RMSE was a bit larger than the gap for the field average prediction, which implies that a bit more of the error was coming from larger values, but the amount was not alarming. These results implied that predicting the average value provided at least equal, if not better results than the nearest neighbor approach used in the grid interpolation. While this may not be the case for every field, it seems that in this case study, the grid size used was unable to pick up on the short scale variation present in the dataset. The points were simply not similar enough to their neighbors for the nearest neighbor approach to work well. While surprising, it seems as though, in this case, similar performance can be expected by predicting the average and predicting with a grid based interpolation.

Table 4.2. A Key For Interpreting Table 4.3, Where Each Interpolation Is Assigned A Unique Number.

<b>Experiment</b>	<b>Key</b>
Field Average	1
Grid	2
Ordinary Kriging	3
Random Forest: Spatial features only	4
Extra Trees: Spatial features only	5
Random Forest: Relief features only	6
Extra Trees: Relief features only	7
Random Forest: Spatial and relief features	8
Extra Trees: Spatial and relief features	9

Table 4.3. Two Sample T-test Results For Every Model Comparison; Created Using MEA And The Standard Deviation of The Absolute Errors. Any P-Values Significant At A 5% Level For A Comparison Are Displayed In Bold. The Null Hypothesis of The T-test Was That The Two Absolute Residual Sets Had The Same Mean.

	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>
<b>1</b>	1.00	X	X	X	X	X	X	X	X
<b>2</b>	0.64	1.00	X	X	X	X	X	X	X
<b>3</b>	0.70	0.93	1.00	X	X	X	X	X	X
<b>4</b>	0.20	0.15	0.16	1.00	X	X	X	X	X
<b>5</b>	0.20	0.14	0.16	1.00	1.00	X	X	X	X
<b>6</b>	<b>0.05</b>	<b>0.04</b>	<b>0.05</b>	0.47	0.47	1.00	X	X	X
<b>7</b>	0.08	0.07	0.08	0.66	0.66	0.78	1.00	X	X
<b>8</b>	<b>0.02</b>	<b>0.02</b>	<b>0.02</b>	0.24	0.25	0.65	0.46	1.00	X
<b>9</b>	<b>0.02</b>	<b>0.02</b>	<b>0.03</b>	0.30	0.30	0.74	0.55	0.90	1.00

An inspection of the visual prediction (fig. 4.2) reveals the characteristic grid shape of the grid based interpolation, but with a few minor deviations. Rather than a manually created grid, this prediction was created by assigning the value of the sample point nearest the the unknown point. As the sample grid was not a perfect rectangular grid, the prediction reflects this. For instance, two rows of samples in the center of the prediction were slightly to the left and right of each other, which resulted in a row of slightly diagonal grid boundaries through the center. Aside from this however, the prediction holds to the idea of the grid based interpolation, and provides a sound proxy.

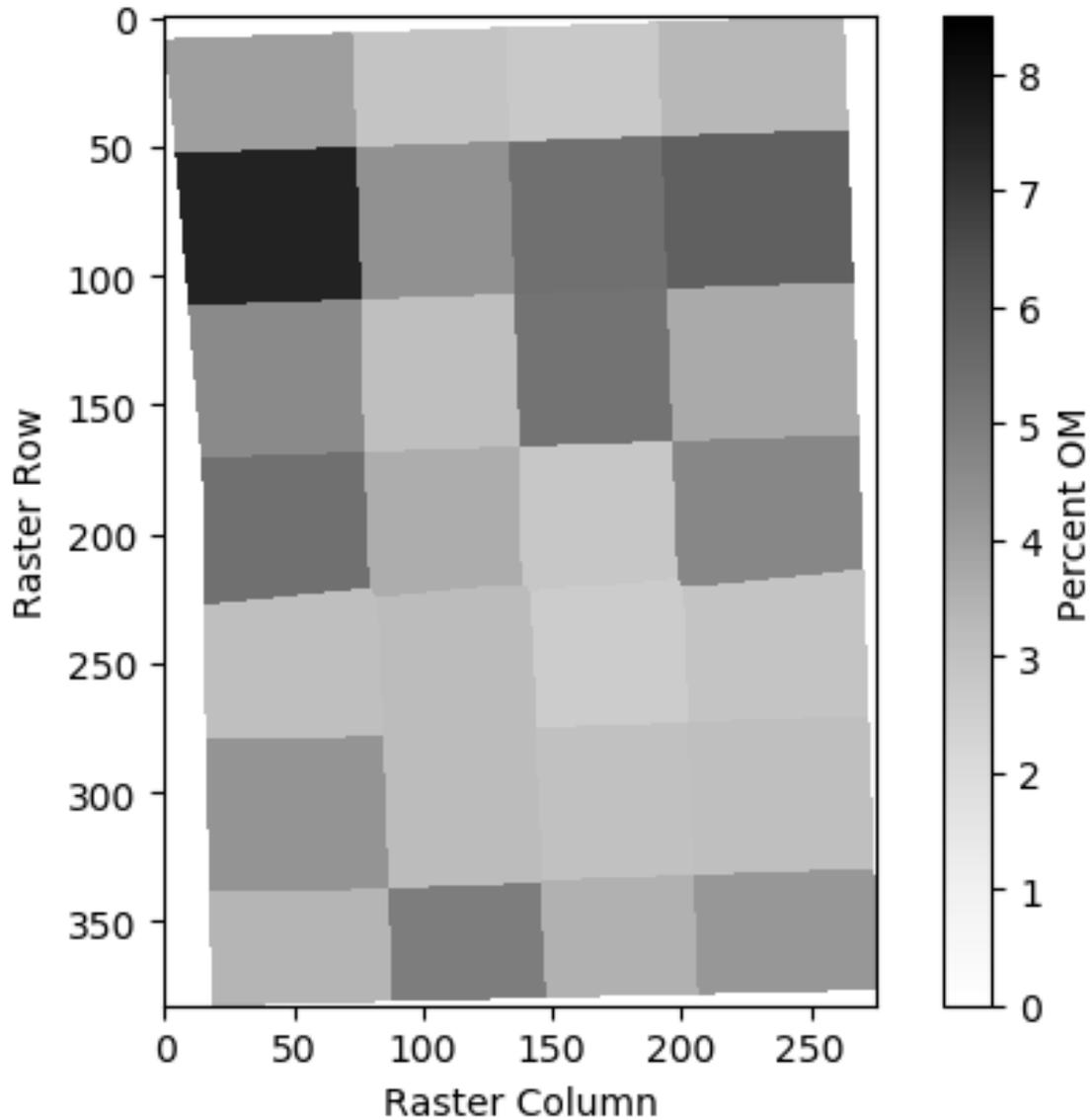


Figure 4.2.: A Grid Based Prediction of Percent OM.

### 4.1.3 Applying Ordinary Kriging

The histogram created to test the OK normality assumption (fig. 4.3), shows a positive skew in the data distribution of the 28 percent OM values. The skew was confirmed by the summary statistics, which reported a median value (3.55) lower than the mean (3.96).

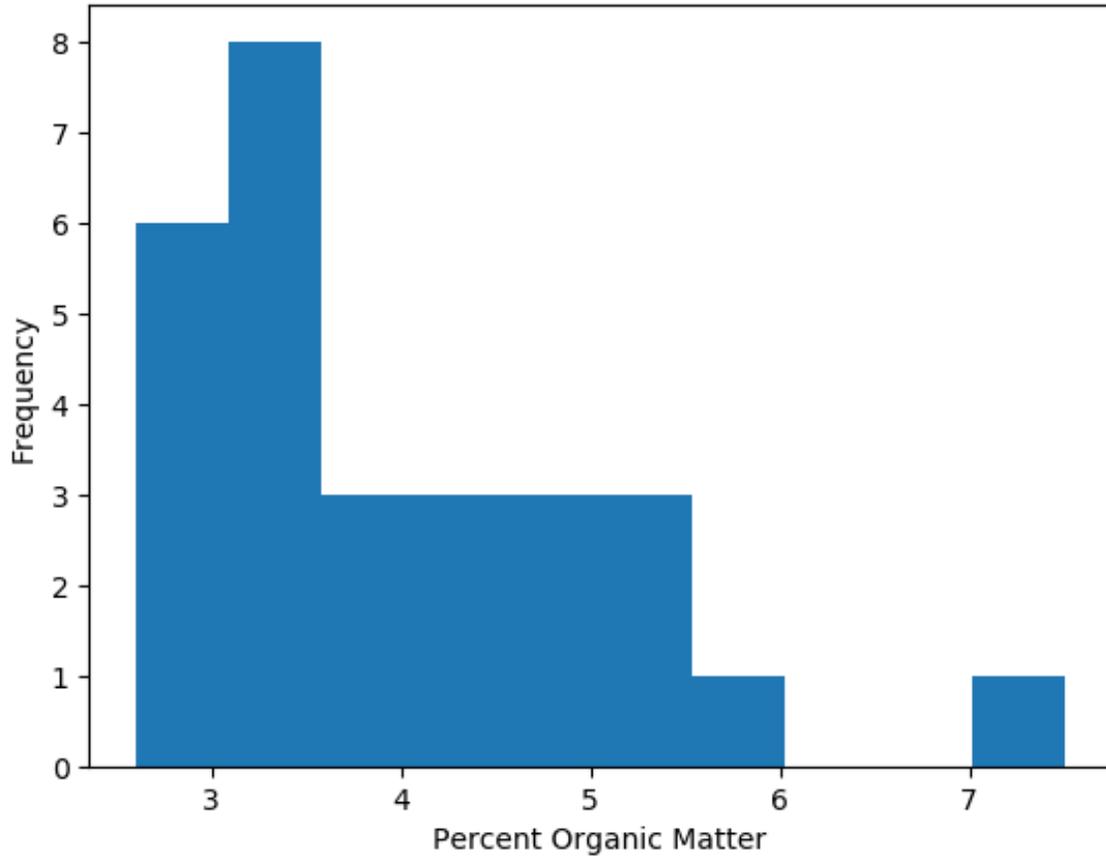


Figure 4.3.: Histogram of The Percent OM Values of The 28 Points Used For Interpolation.

This strong skew violated the OK assumption of normal distribution. Therefore, the test data was transformed using a Box-Cox transformation with a power of -1.6. While the result was visually rough (fig. 4.4), this was found to nearly converge the values of the mean (0.55) and median (0.54). With this numerical correction, the value distribution was considered sufficiently normal.

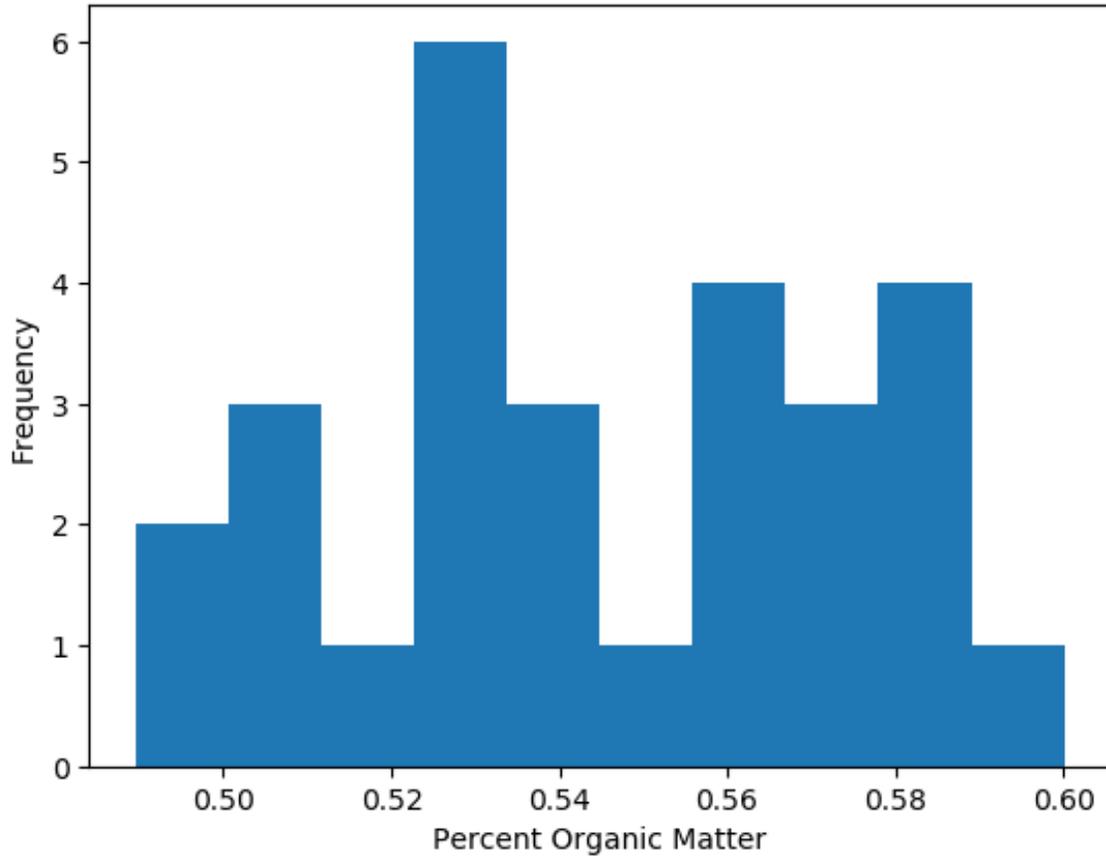


Figure 4.4.: Histogram of the Box-Cox Transformed Percent OM Values of The 28 Points Used For Interpolation.

The second assumption of OK, the points to be interpolated are not spatially clustered, was found valid as the points were sampled on a roughly even grid (fig. 4.5).

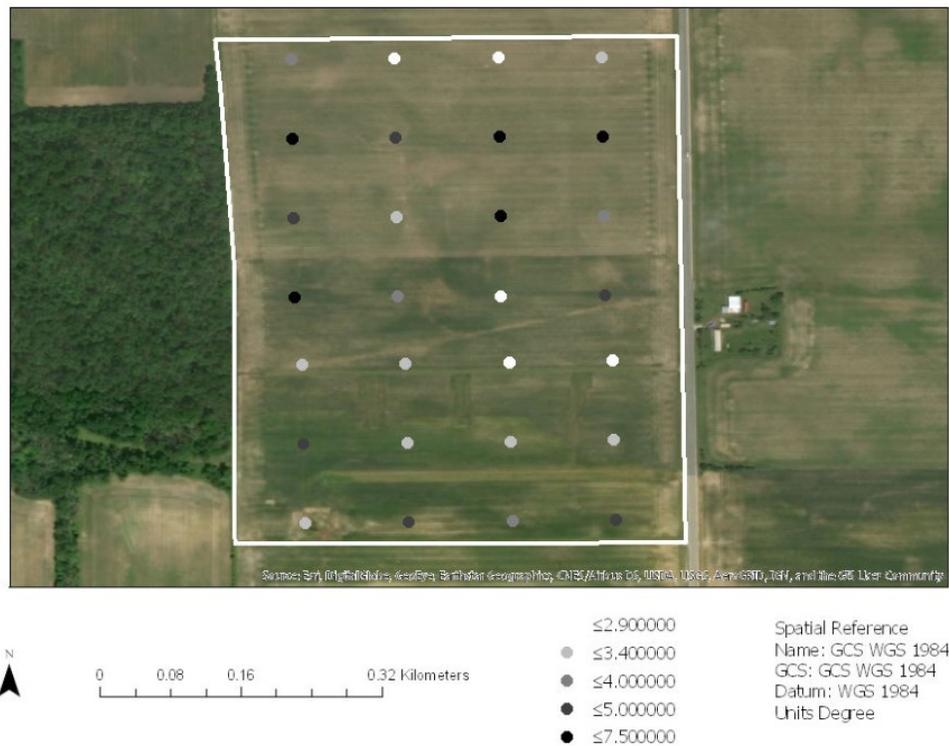


Figure 4.5.: Sampling Locations, Displaying Non-Clustered Layout.

To visually examine the third assumption of OK, that there are no spatial trends in the dataset, several polynomials of different orders were OLS fitted to the data (fig. 4.6). A strong N-S first order trend and a strong E-W second order trend was found in the testing data. To correct for this, a second order trend removal was conducted before interpolation in the geostatistical wizard, followed by kriging. After kriging, the second order trend was added back to the kriged result by the geostatistical wizard.

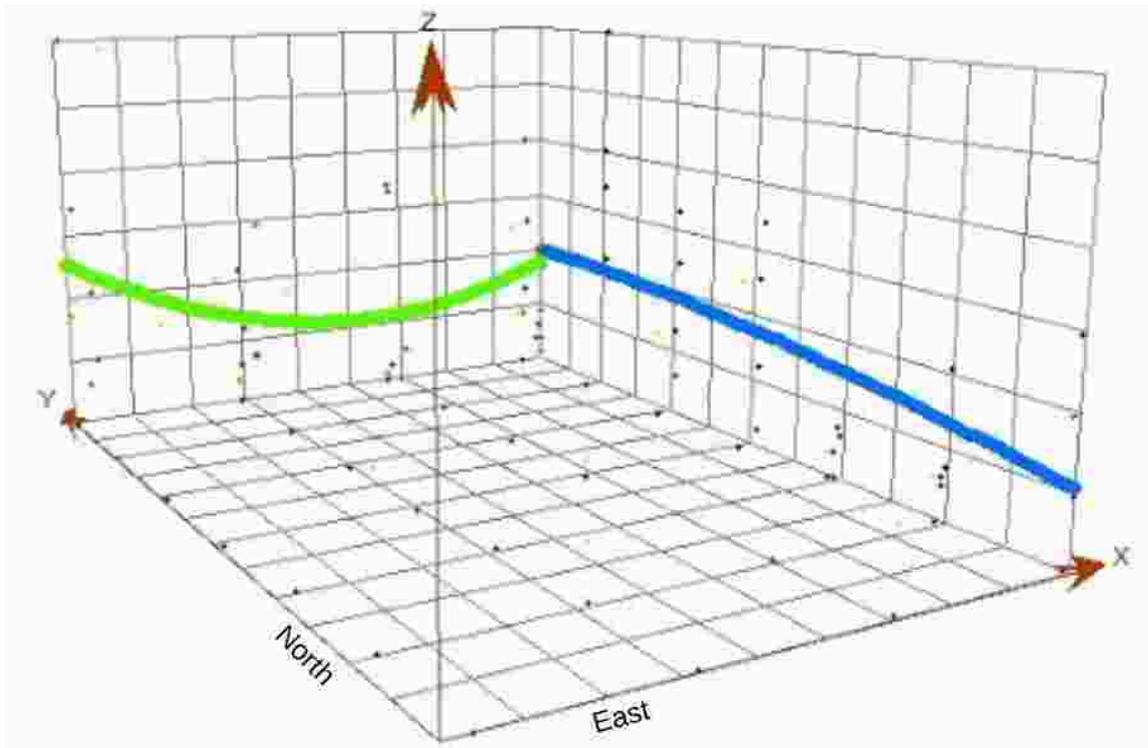


Figure 4.6.: A Trend Analysis of The Percent OM Dataset, Revealing First and Second Order Trends. The Three Dimensional Polynomial Is Shown As Two-Dimensional Slices: Where It Intersects The Northern Boundary of The Field (Green), and Where It Intersects The Eastern Boundary of The Field (Blue).

To check the assumption of stationarity, a Voronoi map of entropy was created for the dataset (fig. 4.7). The Voronoi map revealed a moderate amount of non-stationarity in the dataset. However, as there is no correction for non-stationarity, no corrections were made.

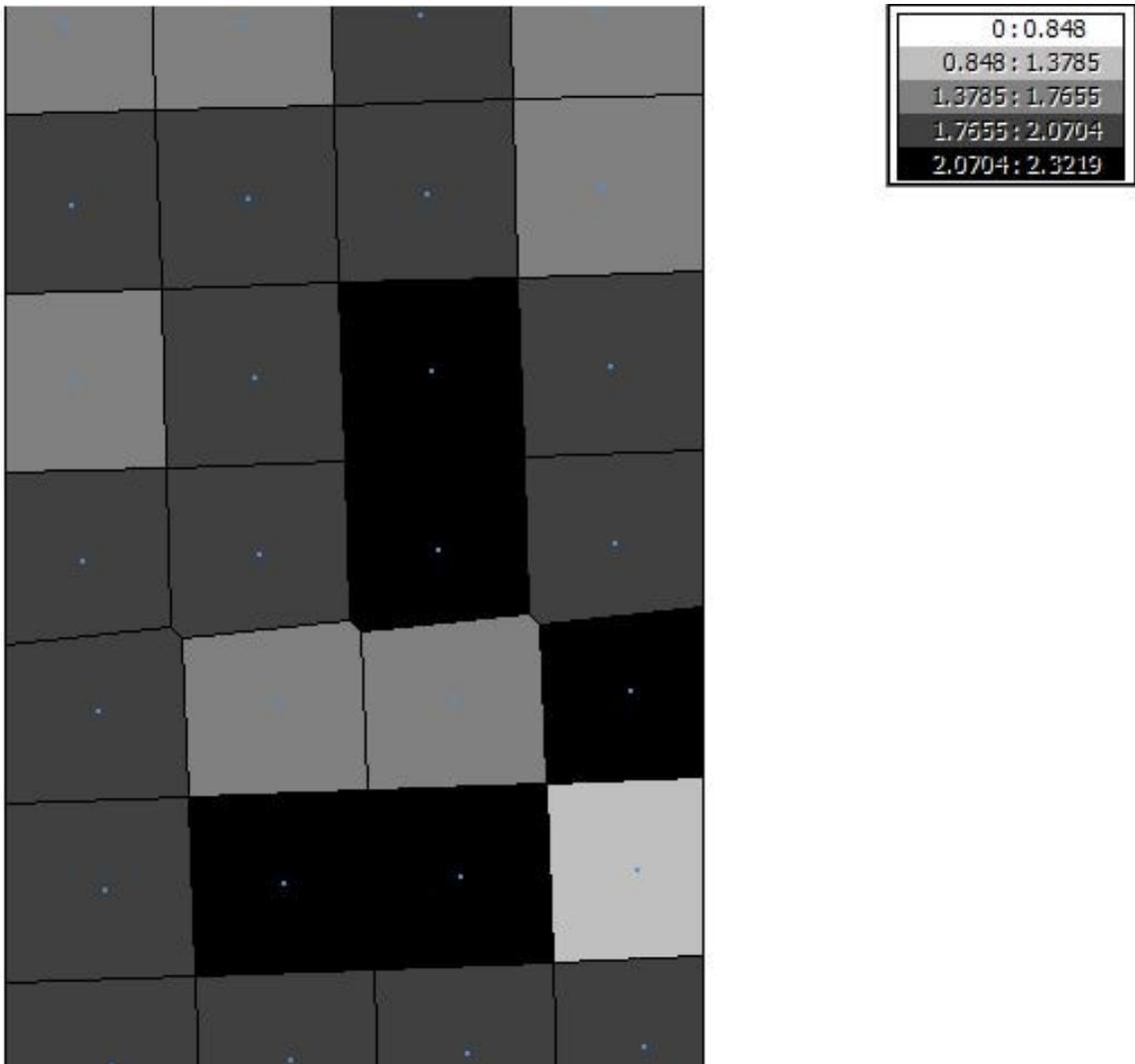


Figure 4.7.: A Voronoi Map of Entropy For The Percent OM Dataset.

Lastly, a semi-variogram of the data was constructed to check the assumption of spatial autocorrelation (fig. 4.8). The semi-variogram suggested the presence of low spatial autocorrelation in the dataset, as the spherical model was nearly flat; the value of the nugget (0.000195) seems to be the almost constant value of the model. While there may be many processes at play, it is highly likely that the main cause of this inability to easily model spatial autocorrelation with a spherical model is the direct result of the limited samples at hand. While densely sampled compared to most DSM efforts, the large nugget and almost non-existent partial sill (sill minus the nugget)

suggest that the samples are still taken at a resolution too coarse to capture the in field variation. If the area of interest had been larger, providing more samples, then it is possible the model would've been able to fit the spatial autocorrelation of the site more cleanly, regardless of the coarse resolution. As it is, the point cloud does seem to demonstrate lessening spatial autocorrelation as the lag increases, but four high values at a small lag seem to skew the model. Again, more points over a larger area would've likely reduced the impact of those outliers, which are likely caused by a single point with a value very different from it's four closest neighbors. The spatial autocorrelation of the dataset, using a spherical model, was perhaps not the best fit. However, the fitted model, with a nugget of 0.000195 and a range of 198 meters, was still used as a basis for OK interpolation.

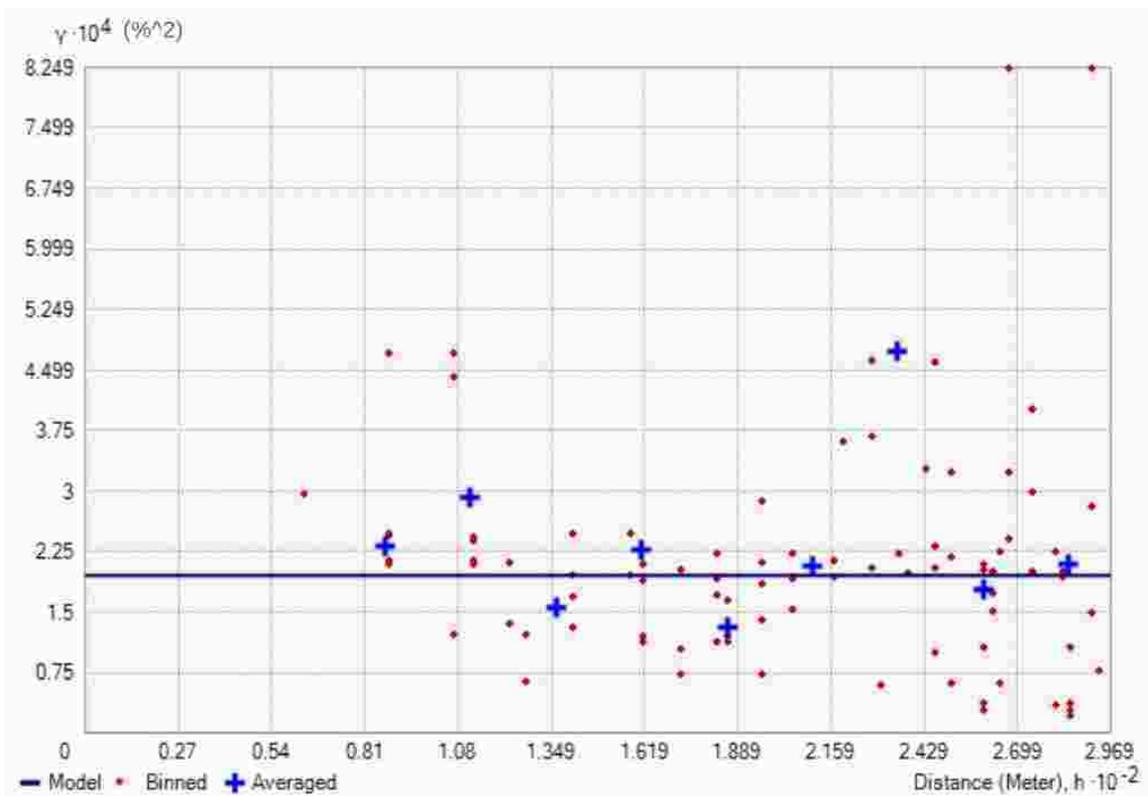


Figure 4.8.: Semi-Variogram and Accompanying Spherical Model of The 28 Percent OM Samples.

The performance measures used for evaluating OK interpolation and the grid interpolation were very similar. Also, the OK interpolation displayed a higher MAE, MAPE, and RMSE than the average prediction (table 4.1), though no significant difference in accuracy is detectable between the two (table 4.2 & table 4.3). Once again, it seems that in this case, it's just as good if not better to predict the average than to use the OK interpolation. This serves to reveal how weakly OK was able to model the data. As the semi-variogram spherical model was nearly flat at the value of the nugget, it shouldn't be a surprise that the model fitting was poor.

The visual results (fig. 4.9) at least seem to track with the general variation observable in Figure 3.1 in the sections above: a strong belt of low percent OM in the lower half of the field, increasing values in the top and bottom, and strong increases along the upper sides. The model also seems to predict with the full range of values in the dataset, from nearly 8 percent down to near 2 percent. On another note, there are some interesting artifacts clearly visible in the prediction. Overlapping circles of roughly similar sizes appear to layer and form the predicted values; these clearly represent the dependence of the model on distances from known points. The circular artifacts denote the range of influence that any one point has. It is likely that an increase in the number of points to predict with would result in the smoothing out of these artifacts, as limited nature of the prediction set seems to rely heavily on individual points.

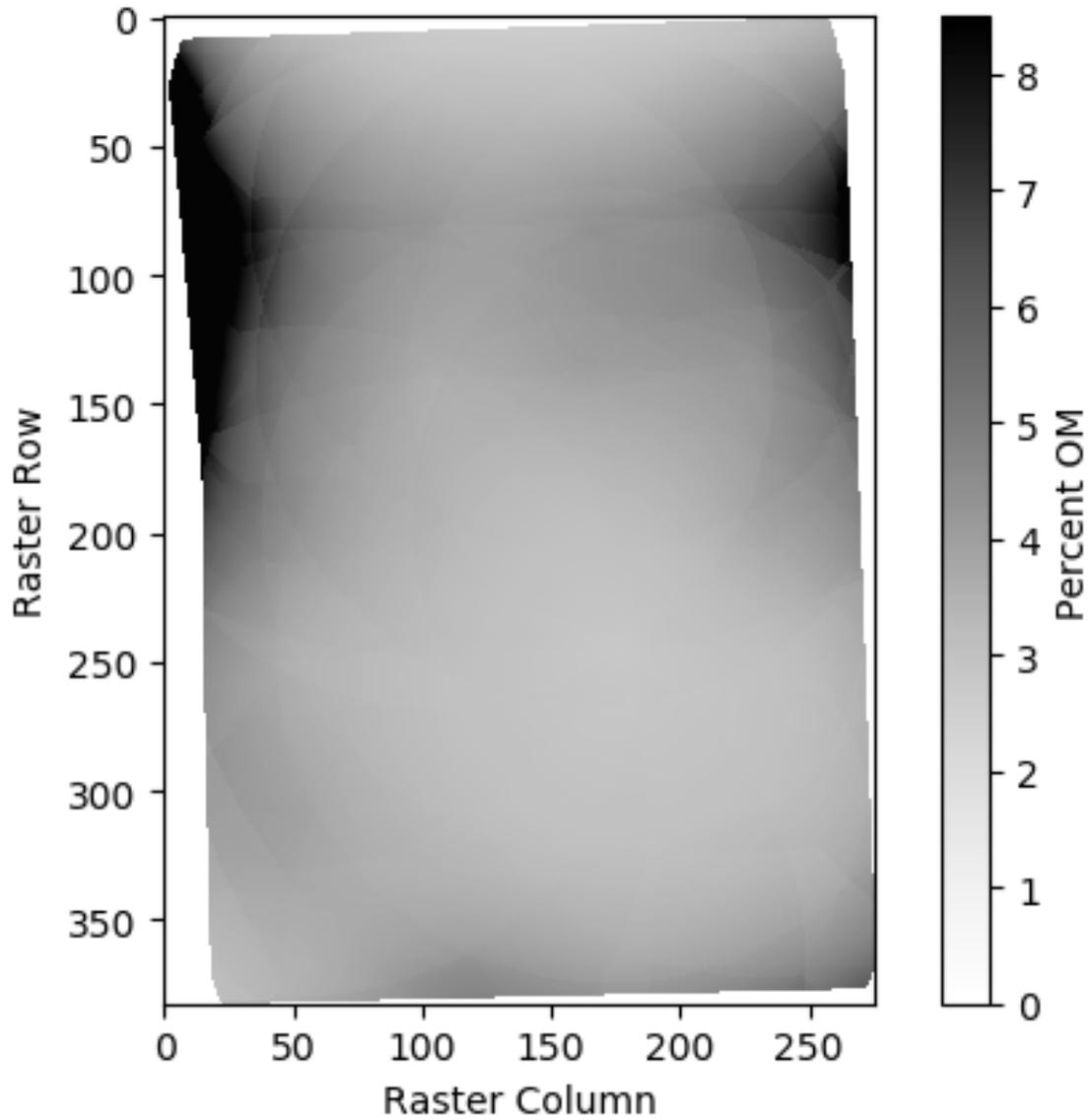


Figure 4.9.: Ordinary Kriging Percent OM Prediction.

The results of the OK and grid-based predictions indicate that both predictions are not significantly different than the field average prediction based on the t-test conducted using MAE and the standard deviation of the absolute residuals set (table 4.2 & able 4.3). In the case of the grid based prediction, it would seem that the sampling density was simply not high enough to capture the real variability in the field. If this is the case, the best way to make a sound grid-based prediction is higher

density sampling. In the case of OK, the prediction seems to have been hampered by at least one outlying value, which, as the dataset size was limited, was not dampened out. As there was evidence that several of the basic assumptions of OK (stationarity and spatial autocorrelation) were violated by the dataset, an informed modeler would likely experiment with other semi-variogram models, and perhaps even choose another interpolation method with less stringent assumptions. However, in the case of an average grower, it is likely that little data exploration would be done, and a grid prediction or OK would be used in a black box fashion. Clearly, this is a dangerous proposition, as this case study revealed that neither method guarantees a prediction better than a simple field average.

## 4.2 Objective 2: Develop an Improved Method

After some model prototyping and calculations, the RF parameters were set (table 4.4). Both the maximum tree depth (number of child nodes) and minimum observations in a leaf were set to 4. Less than 4 values in the prediction nodes made the prototype model more vulnerable to outlying values and increased the number of artifacts seen in the visual predictions. Once the depth was set, it remained to set the size of the ensemble: experimentation found that as few as 500 trees in prototype models could provide consistent performance from model to model, and with largely converged performance. However, as there was no downside to train a larger ensemble, the largest forest computationally practical to train was used, 4,000 trees. The last parameter, the maximum number of features considered for a split, was set at 50 percent, as this provided a balance between overfitting and not being able to model the data. Under 50%, the prototype model performance began to decrease, and above 50%, stronger artifacts began to appear in the visual results. As both RF variants were nearly identical, they both seemed optimized at the same parameter values, and values discussed above were used for both variants.

Table 4.4. The Random Forest Model Parameters.

<b>Parameter</b>	<b>Value</b>
Maximum Depth	4
Minimum number of observations in a node	4
Number of trees in the forest	4000
Maximum number of features used for a split	50%

#### 4.2.1 Experiment 1: Spatial Position

To represent the spatial position, the euclidean distance of every raster cell to every known point was used as the feature set. Figures 4.10, 4.11, and 4.12 are examples of these 28 distance rasters.

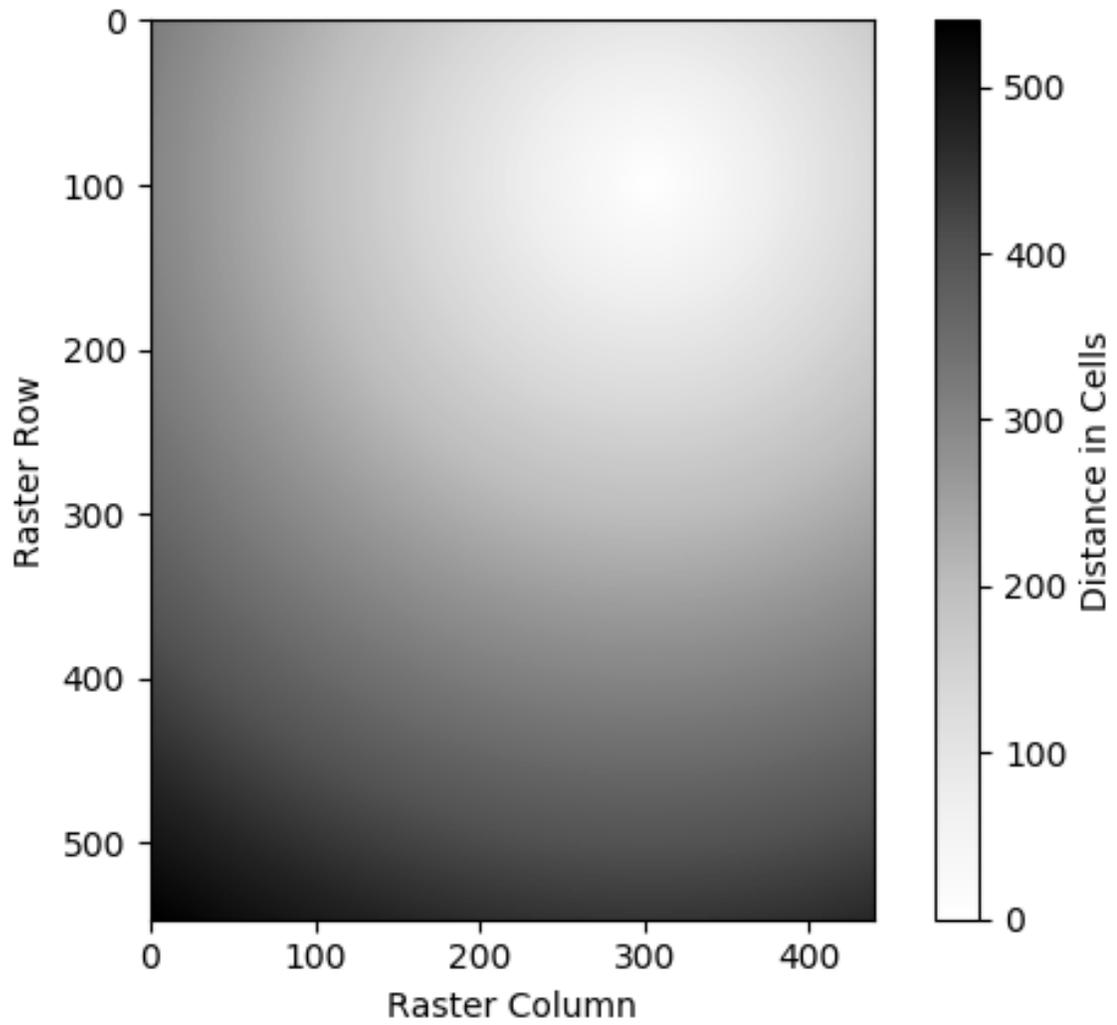


Figure 4.10.: Euclidean Distance To A Sample Point, Measured In Raster Cells.

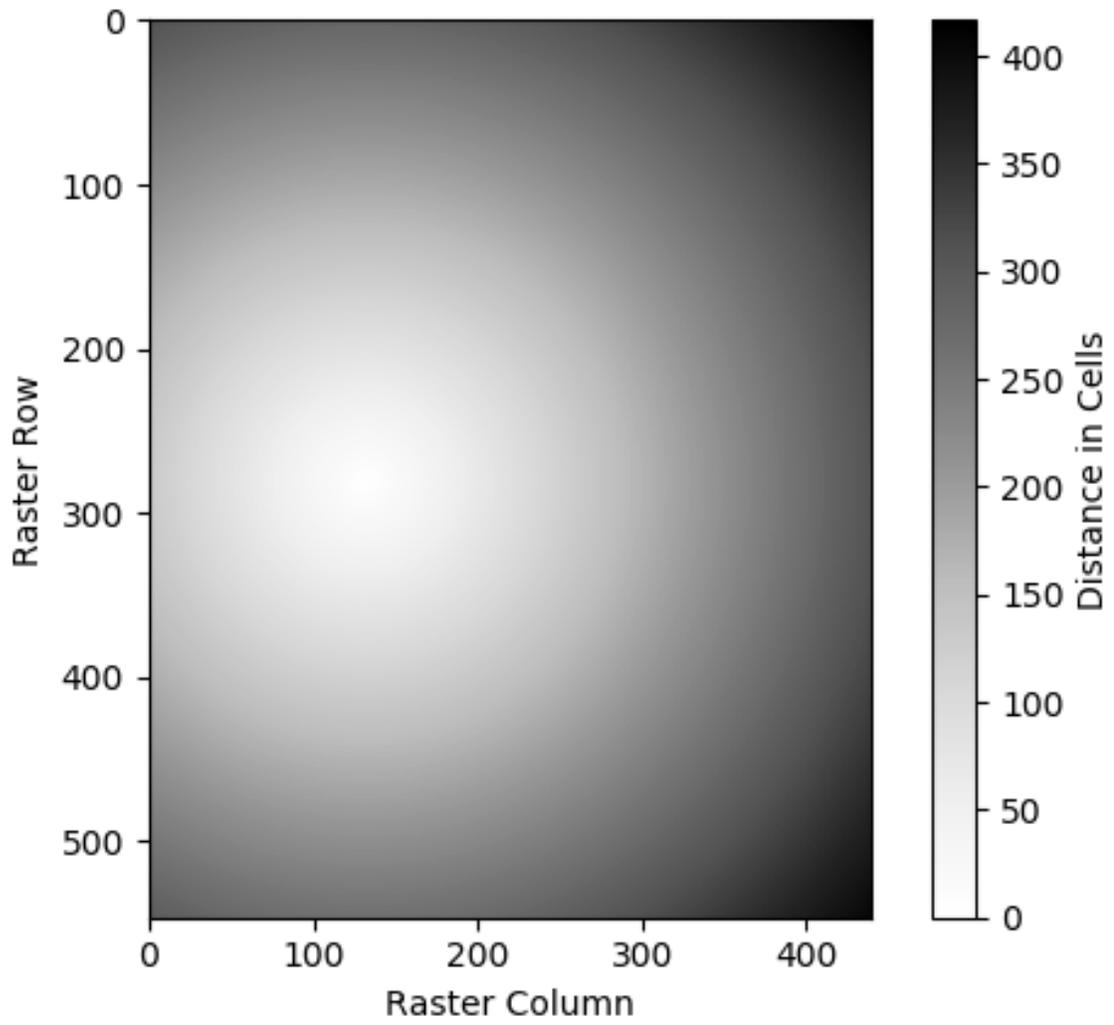


Figure 4.11.: Euclidean Distance To A Sample Point, Measured In Raster Cells.

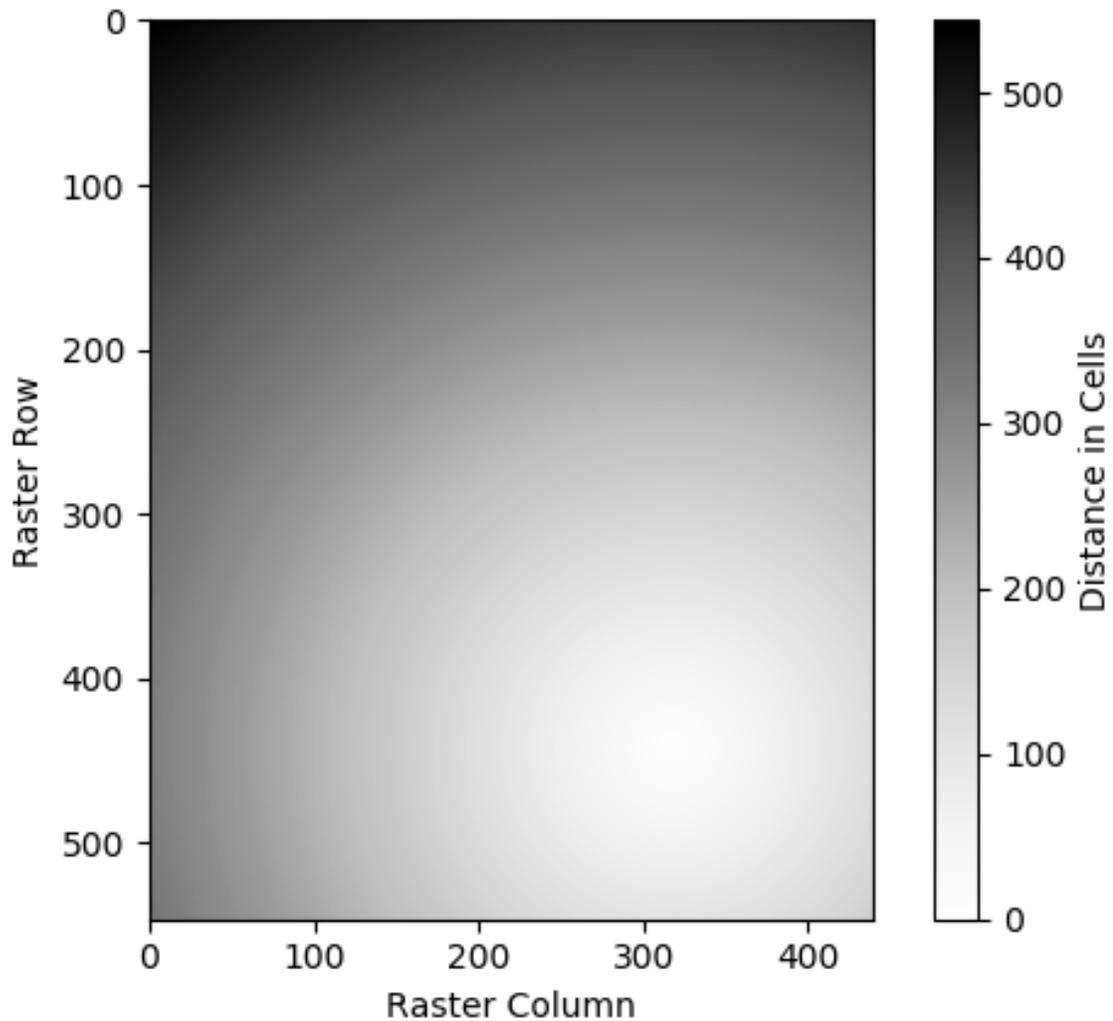


Figure 4.12.: Euclidean Distance To A Sample Point, Measured In Raster Cells.

While both RF models bested the three benchmark methods in MAE, MAPE, and RMSE (table 4.1), the performance difference between them was insignificant at 5% (table 4.2 & table 4.3). The MAE revealed that the average prediction was about 0.8 off, or in terms of percent error (MAPE), 20.3%. The gap between the MAE and the RMSE for the variants was small, implying a relatively even amount of error. As the field average made no insight into variability, any model that could slightly follow the data trends would outperform it, which is what happened in this case. The grid prediction was unable to compensate for variations occurring below

the scale of sampling, and it seems the RF models were able to detect overall trends occurring below the dataset scale. OK, while typically the method of choice for DSM benchmarking, did not meet basic modeling assumptions. Though normality and trends were corrected, OK lacked mechanism to compensate for non-stationarity in the dataset, as well as for undetectable spatial autocorrelation. Stationarity forms the basis of all linear modeling, and in the absence thereof, there is no simple recourse. This particular dataset had an area of relatively uniform values in the lower half, coupled with an area of high variation in the upper half. Combined with a few outlying values, this violated the assumption of distance/variation stationarity OK required in order to model the relationship at hand and resulted in a fitted model with almost no spatial variation explained by distance at all (fig. 4.8). The RF algorithms, on the other hand, are not tied to assumptions of linearity, normality, trends, etc. Of course, data that fits these criteria is simpler for any method to model, but the nature of the RF algorithm allows it to adapt to non-stationary data. In this case, this flexibility seems to have given RF a slight advantage over OK.

One second interesting insight is the fact that the performance between the two RF models was virtually identical. Not only was the performance difference between the two statistically insignificant, but the values were nearly the same as well. In this case it would seem that one prediction is likely as good as the other.

The visual results also hold some interesting insights. Both the RandomForest variant prediction (fig. 4.13),, and the ExtraTrees prediction (fig. 4.14), seem to hold with the general flow of values visible in the points (refer to Figure 3.1). Additionally, both predictions seem to generally agree with each other and the OK prediction (fig. 4.9). One key visual difference between the RF and OK predictions is the fact that the RF results don't predict the full range of sample values, while the OK results did. In particular, the RF predictions seem to predict the lowest values, but never predict up to the highest value, 7.5 (judging by the colormap). While this may be caused by many things, it is likely a combination of the data value distribution and the nature of the RF algorithm. As revealed in the histogram of the data values (fig. 4.3), the

data values had a strong positive skew, with a single outlier at 7.5. The nature of the RF algorithm inadvertently muffles this outlier in several ways. First, the prediction of a single tree is the average value of the sample values in the prediction node. As this forest was tuned to allow no less than four training values in a prediction node, the single high value would have always ended up in a prediction node with no less than three other values, ergo, the averaging during prediction would have reduced the predicted value. Additionally, as each tree was trained with a bootstrapped sample, the outlier may not have been used in the construction of all trees. Then as all the predictions of all the trees are averaged, the outlier would have been further diluted, cutting the highest value of the prediction considerably. Conversely, at the lower end of the values, as there were many points nearly equal the lowest value, the RF models easily predicted the lowest value. This illustrates a noteworthy trait of the RF algorithm: it can only predict values within the range of the training set, and depending on the model parameters and the data value distribution, it may tend to cut out values at the edge of the range (Hengl et al., 2018). Thus, when examining the field average prediction (fig. 4.1) and the RF predictions (figs. 4.13 & 4.14) side by side, it can be seen that the RF predictions are very like the average, but with a few slight deviations in select areas, where the RF prediction seemingly follows the trend of the data slightly.

Another point of interest is in the artifacts visible (and not visible) in the predictions. The RandomForest prediction holds artifacts very similar to those found in the OK prediction (fig. 4.9): overlapping circles. This clearly indicates their shared dependence on distance to known points, as each circle centers on where a known point would be. However, beyond the shape, the artifacts on each are very different. The OK artifacts are roughly the same size, quite numerous, and no one artifact seems to stand out much more than the others. However, the RandomForest prediction showcases fewer artifacts of differing sizes, obviously favoring some known points more than others. This distinction results from the nature of the RF algorithm, and highlights the ability of the algorithm to handle non-stationarity in the dataset. OK is forced

to assume all points are important and have similar effects at similar distances, while the RF algorithm can detect that some points are highly predictive of an area and others give little insight into their neighbors. The two very obvious artifacts in the RandomForests prediction (fig. 4.13) correspond to two very important split points in the data: the feature (distance to point X), and the value (the critical distance). Any points within X distance of the lower point seems to have a higher likelihood of being a low value, and any points within X distance of the upper point seems to have a higher likelihood of having a high value. At this point, the lack of artifacts visible in the ExtraTrees prediction becomes rather interesting. While the RandomForest prediction displays clear split points as circular artifacts in the data, the ExtraTrees prediction is smooth and void of similar artifacts. As the algorithms are identical except for the choosing of split points, the smoothing effect must be derive from this. This holds intuitively: ExtraTrees chooses split points from random values within the range of each feature, and RandomForest chooses split points from observed values in each variable. Thus, when the dataset is limited, RandomForest will have a limited amount of split point to choose from, and important splits are bound to appear in a good many trees, then in the prediction. However, as ExtraTrees can choose from infinite values within the range of the feature, the chances that any one integer value will become a common split point is greatly reduced. This had a dramatic smoothing effect in this limited data case, and clearly illustrates the difference between the two models. Despite the visual differences between the two predictions, it's worth noting that the two predictions are clearly very similar, as the very similar numerical results attest.

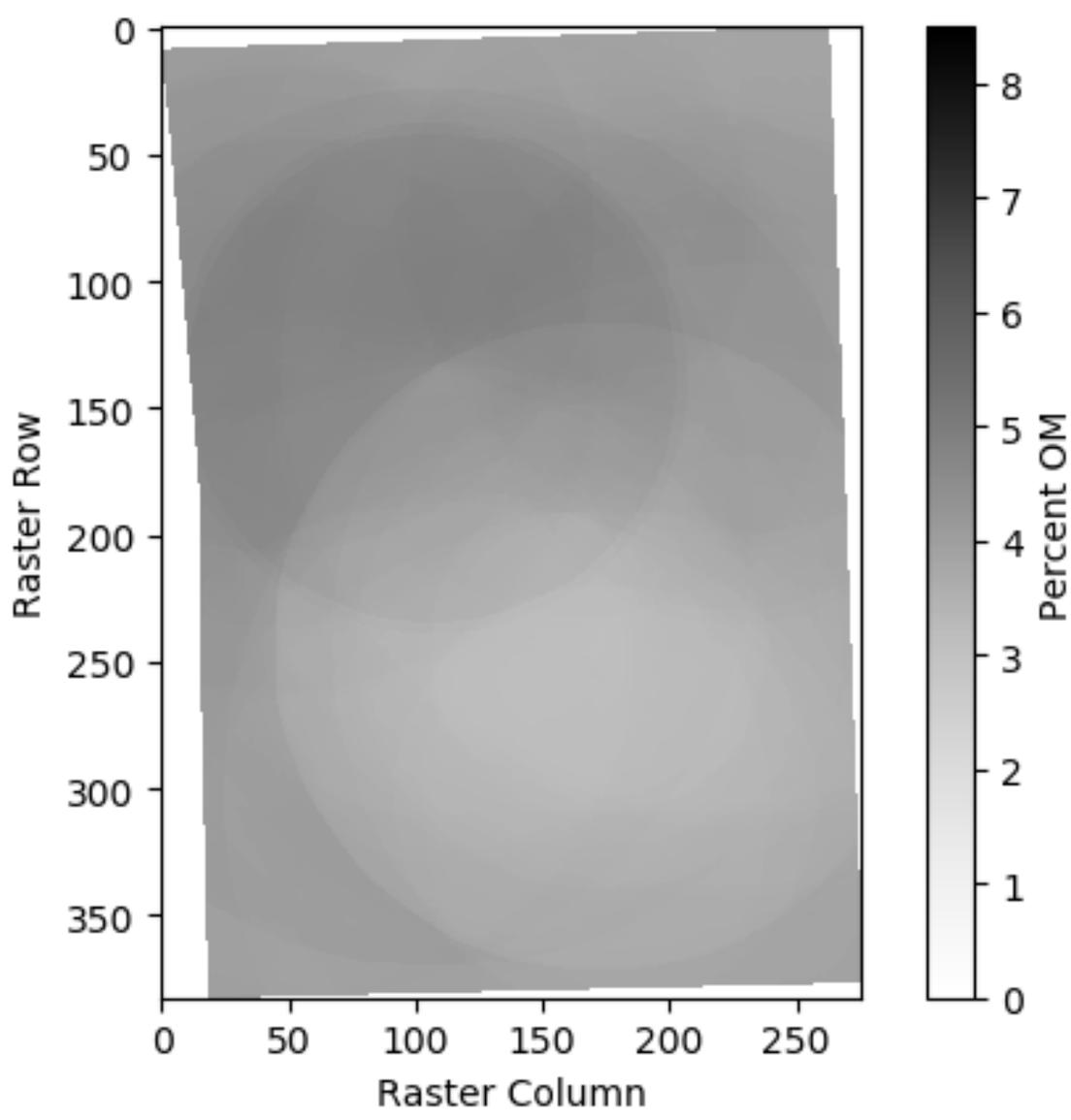


Figure 4.13.: Random Forests Percent OM Prediction, Based On Spatial Position only.

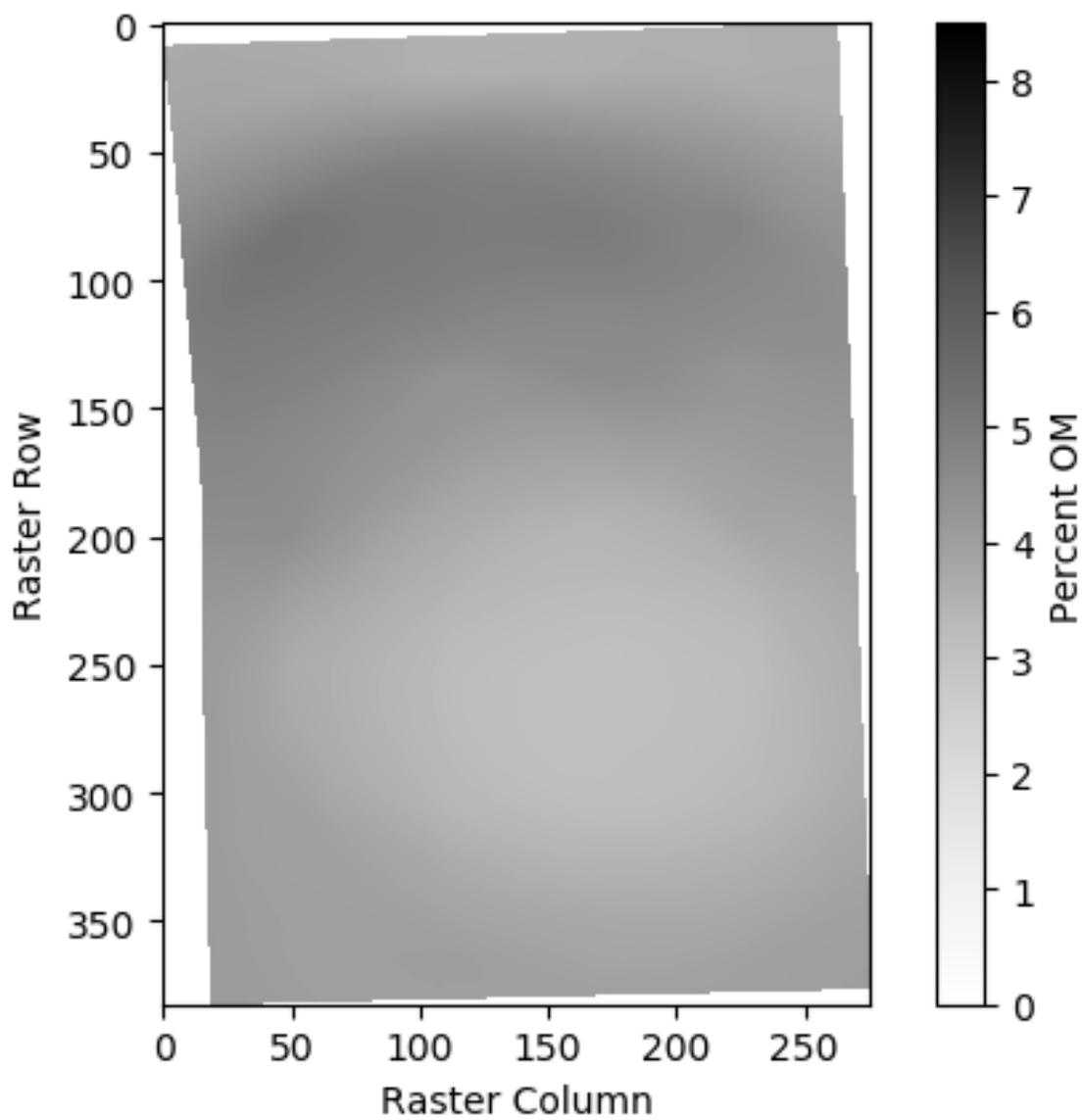


Figure 4.14.: Extra Trees Percent OM Prediction, Based On Spatial Position only.

### 4.2.2 Experiment 2: Relief

To examine the performance of relief as a feature relative to spatial position, only features representing aspects of relief were included in the feature set of the RF models. After the initial feature set had been created, the Feature Importance output from the scikit-learn packages (discussed in methods) was used to evaluate each feature and refine the feature set. The first focus was on the three hydrologic features included in the initial feature set: TWI, Flow Accumulation, and Horizontal Distance to Stream Network. Interestingly, each of the hydrologic features contributed very little to the model, judging by the feature importances. Presumably, in the case of this particular field, the hydrologic features included failed to represent the true processes at hand, or perhaps hydrological processes did not have a strong spatially variable impact on soil OM formation. This was unexpected, but led to the removal of all three hydrologic features from the final feature set. Elevation (fig. 4.15), on the other hand, was consistently useful to the model, and was included in the final feature set. In the case of slopes and curvatures, adding sets calculated at several different neighborhood sizes seemed to partial out processes occurring at different scales, and many of the neighborhoods (especially the larger neighborhoods) were very useful to the model. After tuning, it was determined to use topographic derivatives created at three neighborhood sizes: 5x5, 55x55, & 115x115. As the raster resolution was 1.52 m, the actual size of the neighborhoods considered was 7.6x7.6 m, 83.6x83.6 m, & 174.8x174.8 m. Figures 4.16 and 4.17 show slope calculated over the 5x5 and 115x115 neighborhoods. The final trimming of the feature set was determining which types of curvatures to include in the feature set. After testing, it was determined that tangential curvature was largely redundant, and only profile and planform curvatures were included in the feature set. Figures 4.18, 4.19, and 4.20 are a selection of the 6 curvature features included in the feature set. After all feature trimming was complete, the relief-based feature set was comprised of 10 features: elevation, slope at 7.6 m, profile curvature at 7.6 m, plan curvature at 7.6 m, slope at 83.6 m, profile

curvature at 83.6 m, plan curvature at 83.6 m, slope at 174.8 m, profile curvature at 174.8 m, and plan curvature at 174.8 m

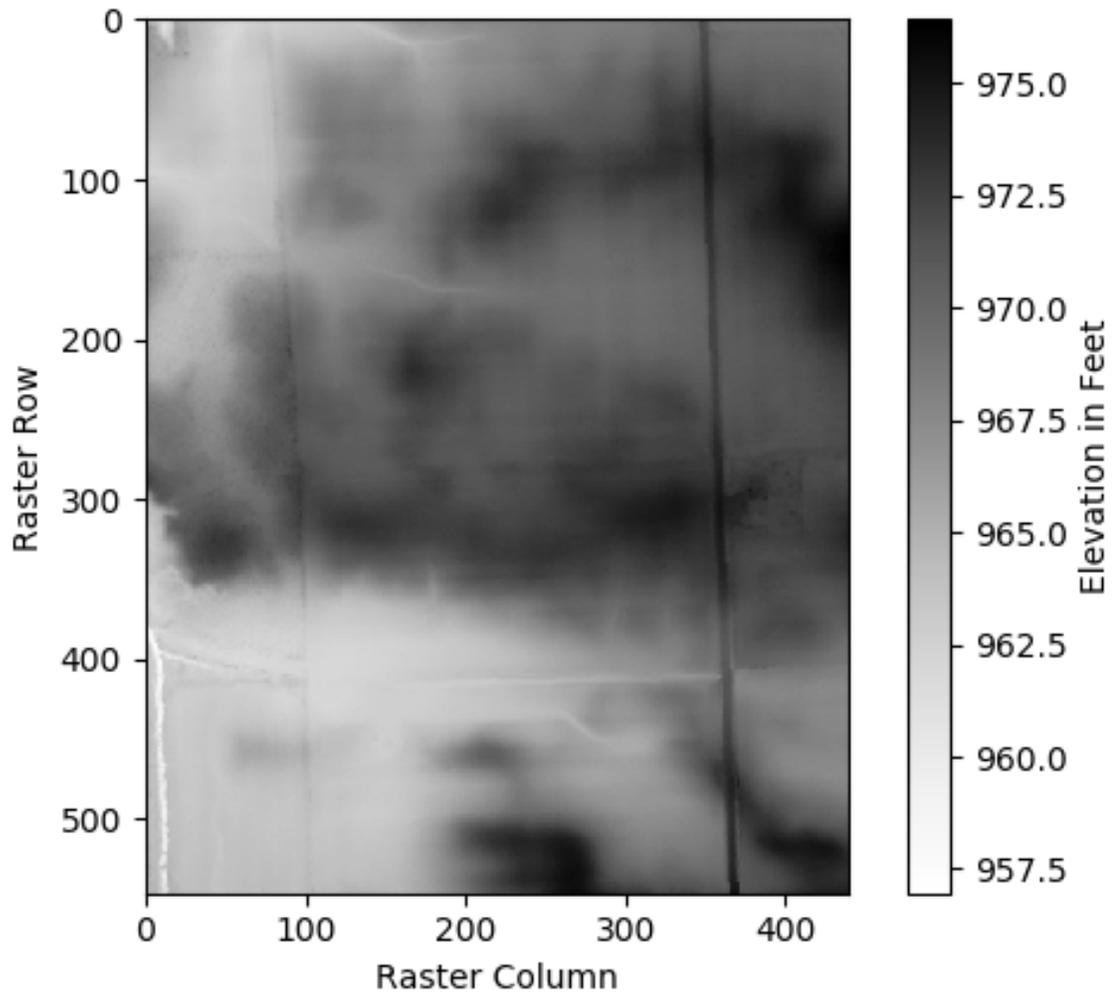


Figure 4.15.: The Elevation of The Site (in Feet).

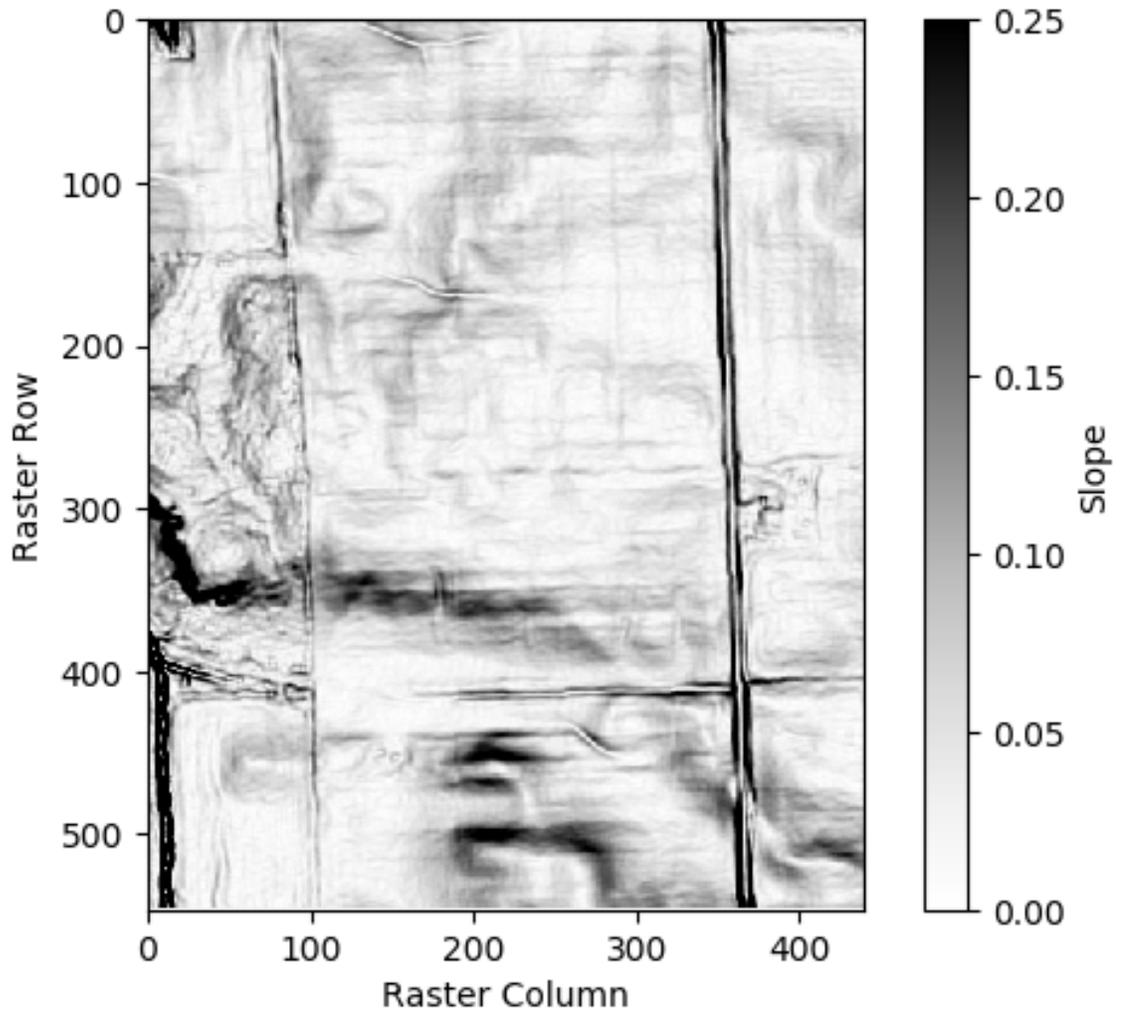


Figure 4.16.: The Slope of The Site In Percent, Calculated At 7.6 m Resolution. Note: While Slope Reaches 100% In Some Areas, Color Scale Has Been Set To 25% Max For Higher Contrast Viewing.

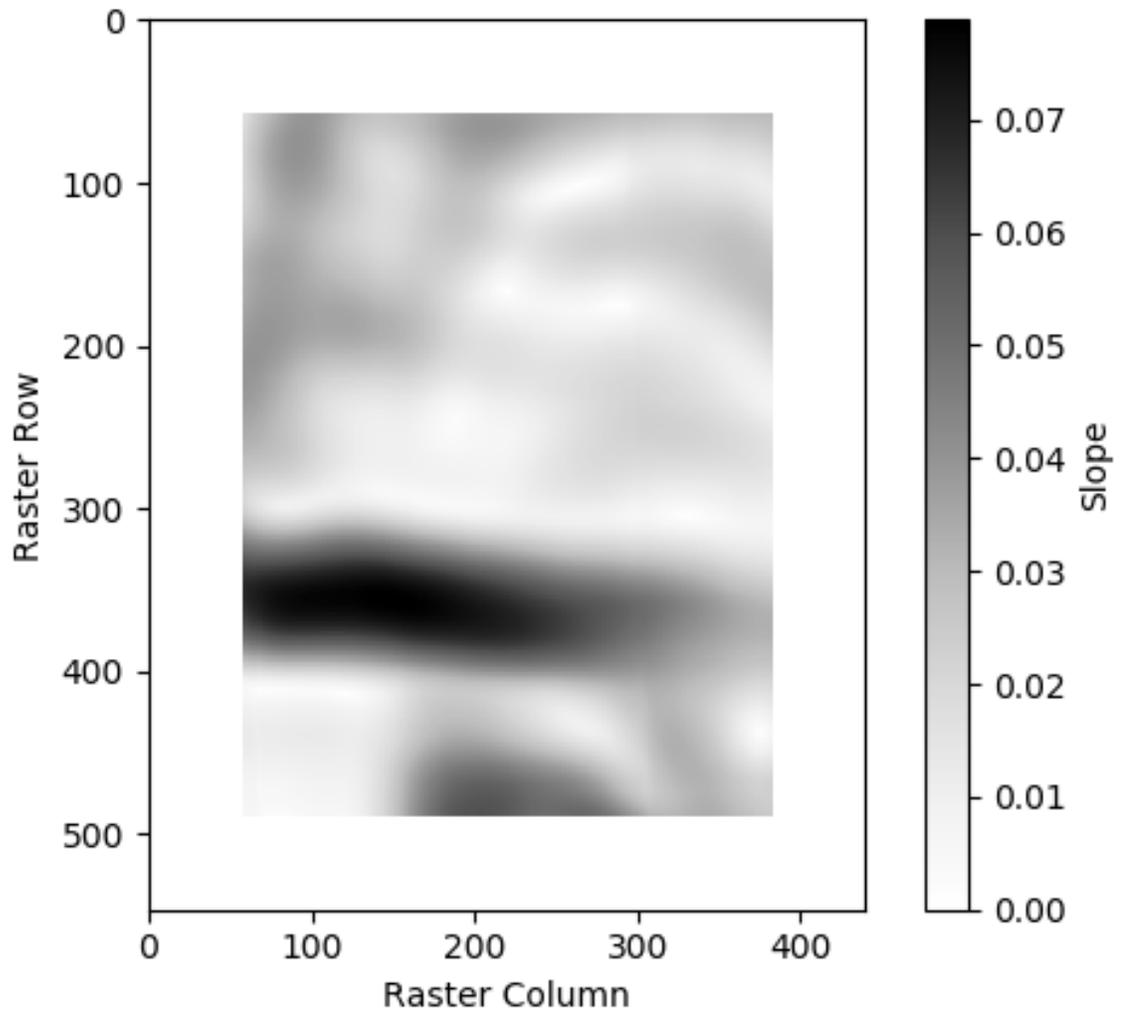


Figure 4.17.: TThe Slope of The Site In Percent, Calculated At 174.8 m Resolution. Note: Here Color Scale Is Set To True Value Range, Much Smaller Than The Previous Figure.

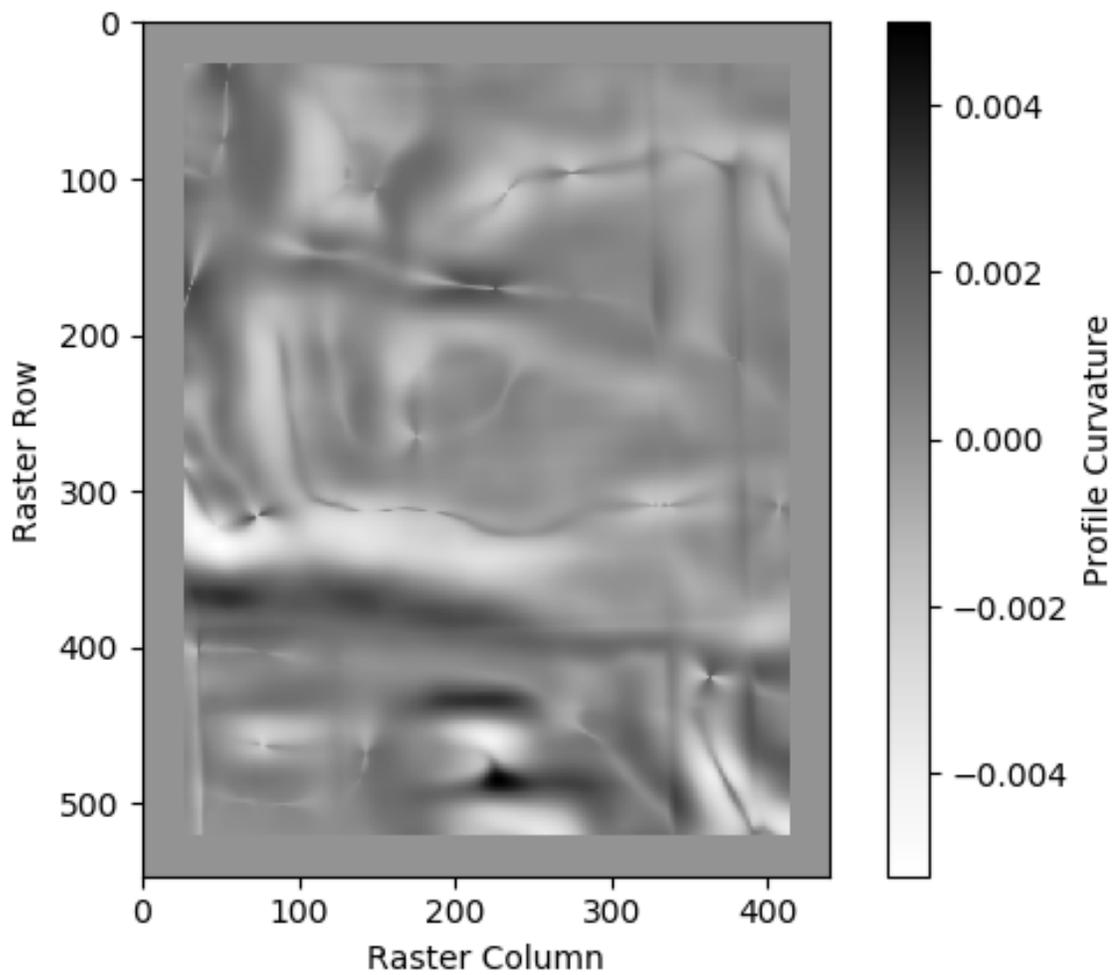


Figure 4.18.: The Profile Curvature, Calculated At A Scale of 83.6 m.

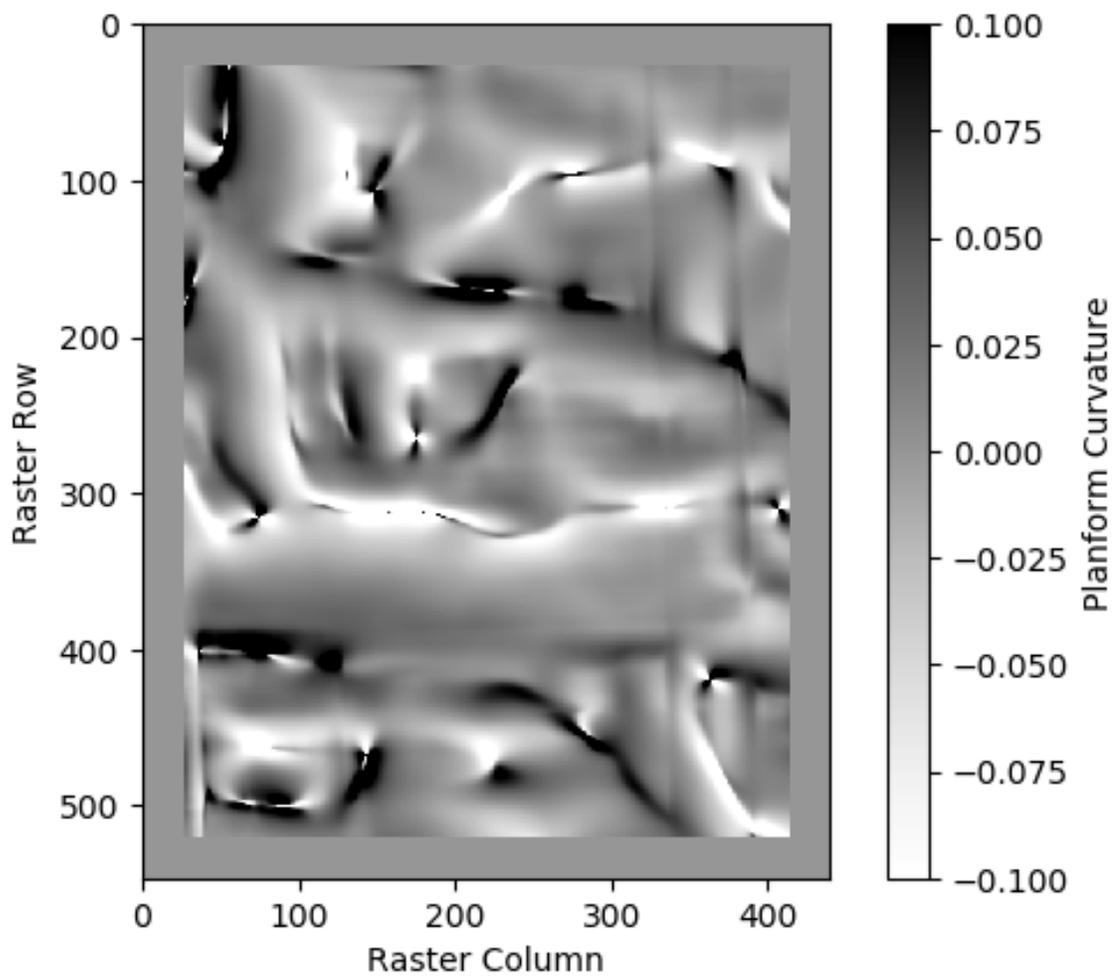


Figure 4.19.: The Planform Curvature, Calculated At A Scale of 83.6 m. Note: Here The Color Scale Is Set To -0.1-0.1 For Viewing Contrast, But The True Value Range Is Much Larger.

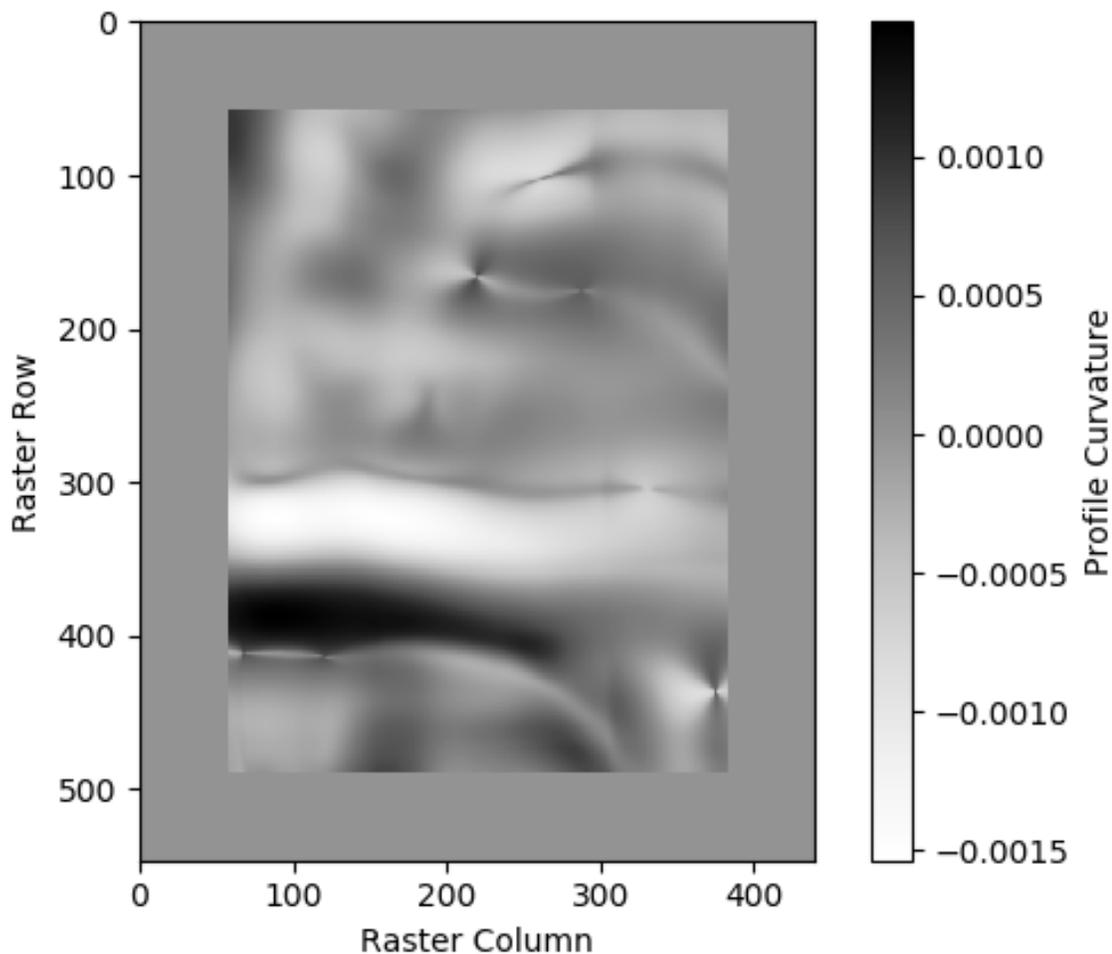


Figure 4.20.: The Profile Curvature, Calculated At a Scale of 174.8 m.

The results (table 4.1) revealed that the relief based models performed well, with lower scores than the three benchmarking methods. This difference was significant, in case of the RandomForest variant, and modestly significant (at 10%) in the case of the ExtraTrees variant (table 4.2 & table 4.3). While they also outperformed the spatial position based models from experiment one, the difference between them was not significant. This result seems to demonstrate that relief was a stronger explanatory variable of soil percent OM than spatial position, but only just. The strong performance of relief was somewhat surprising, given how little obvious relief the site in question had. Within the field boundary, the elevation had a maximum

change of only 4 m, with an average slope of 1.1 degrees. However, even these gentle variations seem to have a strong relationship with percent OM in the soil, as the relief based predictions lowered both the model error in comparison to previous interpolations.

Another interesting observation is that there was a slight performance gap between the ExtraTrees and RandomForest variants (though the performance difference between them was still statistically insignificant). The under performance of the ExtraTrees variant indicated that the extra randomness did seem to slightly impact the ability of the model to fit the relationships at hand. This gap may be due to the stronger relationships observable in the relief based features. In conditions where features explain little of the property of interest, additional randomness holds the model back from overfitting, but perhaps in better modeling conditions, this extra caution is more of a liability. This illustrates the need to test several models before using one exclusively, as each may be used to benchmark the others.

The visual results (figs. 4.21 & 4.22) again display interesting characteristics. Like the predictions from experiment one, both predictions fail to predict the maximum value present in the dataset, likely for the same reason discussed previously, that RF is unable to predict outside the training value range (Hengl et al., 2018). Both predictions again seem to follow the general flow of values present in the sample points (fig. 3.1), the OK prediction (fig. 4.9), and the results from experiment one (fig. 4.13 & 4.14). However, while the general flow of values seem to be the same (higher values in the top and bottom, with a swathe of low values through the lower half), the predictions seem to have more sharply defined transitions and zones. For example, consider the ExtraTrees predictions from experiments one & two (figs. 4.14 & 4.22). While both predictions seem smooth, with no sharply defined boundaries or artifacts, the experiment two prediction seems to display a much higher level of detail. The upper half of the experiment one prediction for example, displays a sort of wide band of higher values stretching across. While the experiment two prediction generally agrees, the same area displays half a dozen distinct areas of variation. This

difference is likely driven by two phenomenon. The first is that the experiment two results are simply more accurate than the experiment one results, and the experiment two prediction better displays the short range variation present in the real world. However ideal the first reason, the second cause is that both the experiment one and two predictions display strong artifacts, which are derived directly from the base features. The wide band of high values in experiment one looks very similar to the even, sweeping values from the distance features (figs. 4.10, 4.11, & 4.12). The areas of variation visible in the experiment two results, however, seem to track well with the variation present in the 83.6 m curvatures (figs. 4.18 & 4.19).

Like before, the artifacts present in the predictions are stronger in the RandomForest variant and softer in the ExtraTrees variant, but in this case, both predictions do display these artifacts more clearly. The most obvious of them all is the diagonal band of high values that cuts across the top of the RandomForest prediction. This seems to line up neatly with a similar structure visible the 83.6 m curvatures (figs. 4.18 & 4.19). In addition to this particular artifact, several others like it seem to be present in the prediction. Additionally, the RandomForest prediction displays a large number of much smaller linear artifacts that appear to run through the entire prediction, even overlapping with the larger artifacts in many places. This is clear evidence of important split points in the data, as the ExtraTrees prediction fails to predict these smaller artifacts with such definition. However, while the softening effect of the ExtraTrees variant does improve the visual appeal of the prediction, the RandomForest variant still outperformed. This issue of performance is a reminder that a pretty result may not always be a better result, as in this case. In any case, the results of this experiment are in line with previously reported results ( Moore et al. (1993); Zhu et al. (1996); Hengl et al. (2004, 2017); Ramcharan et al. (2018)), that relief could be considered as a viable predictor for field soil mapping.

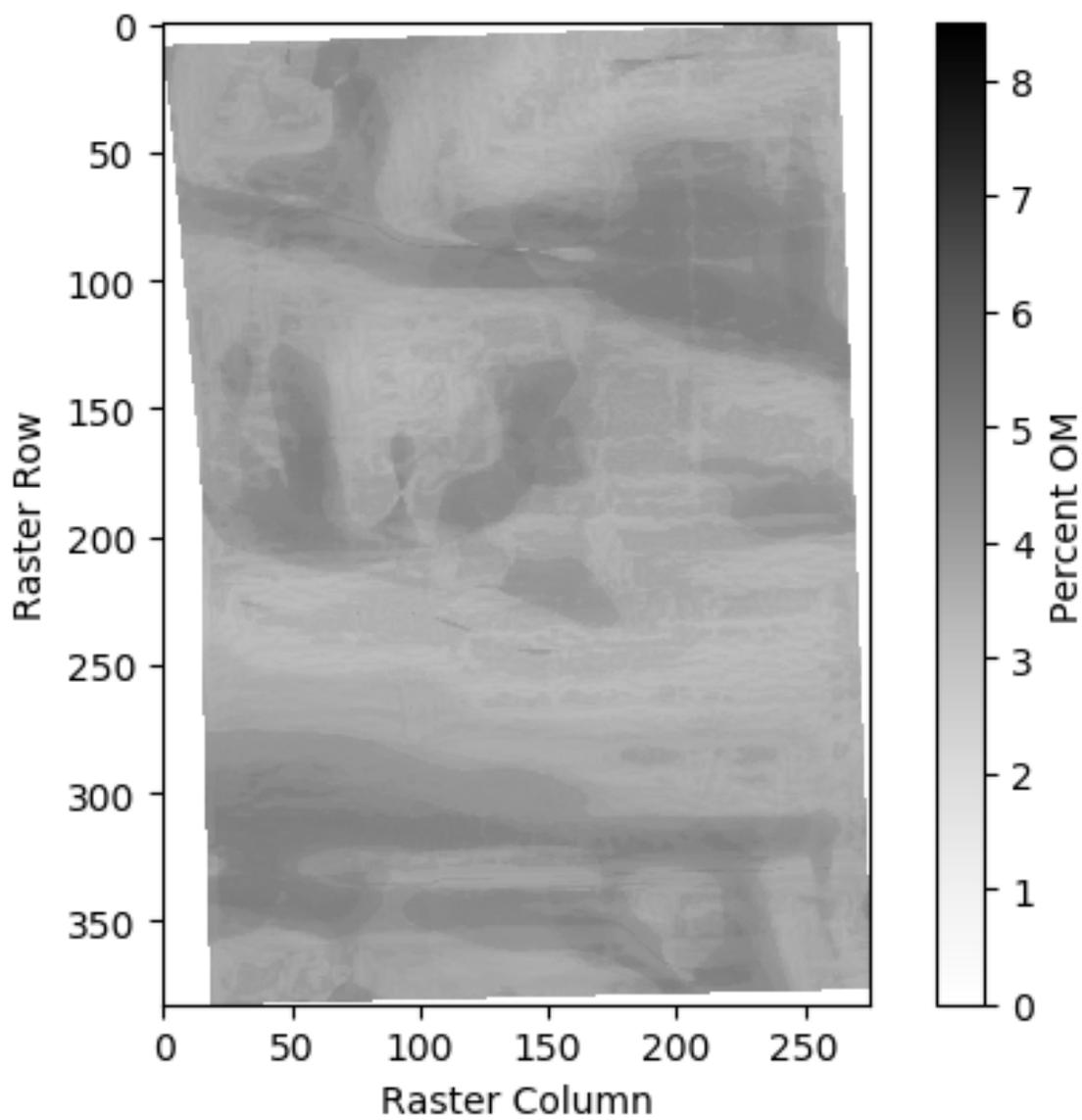


Figure 4.21.: Random Forests Percent OM Prediction, Based On Relief Only.

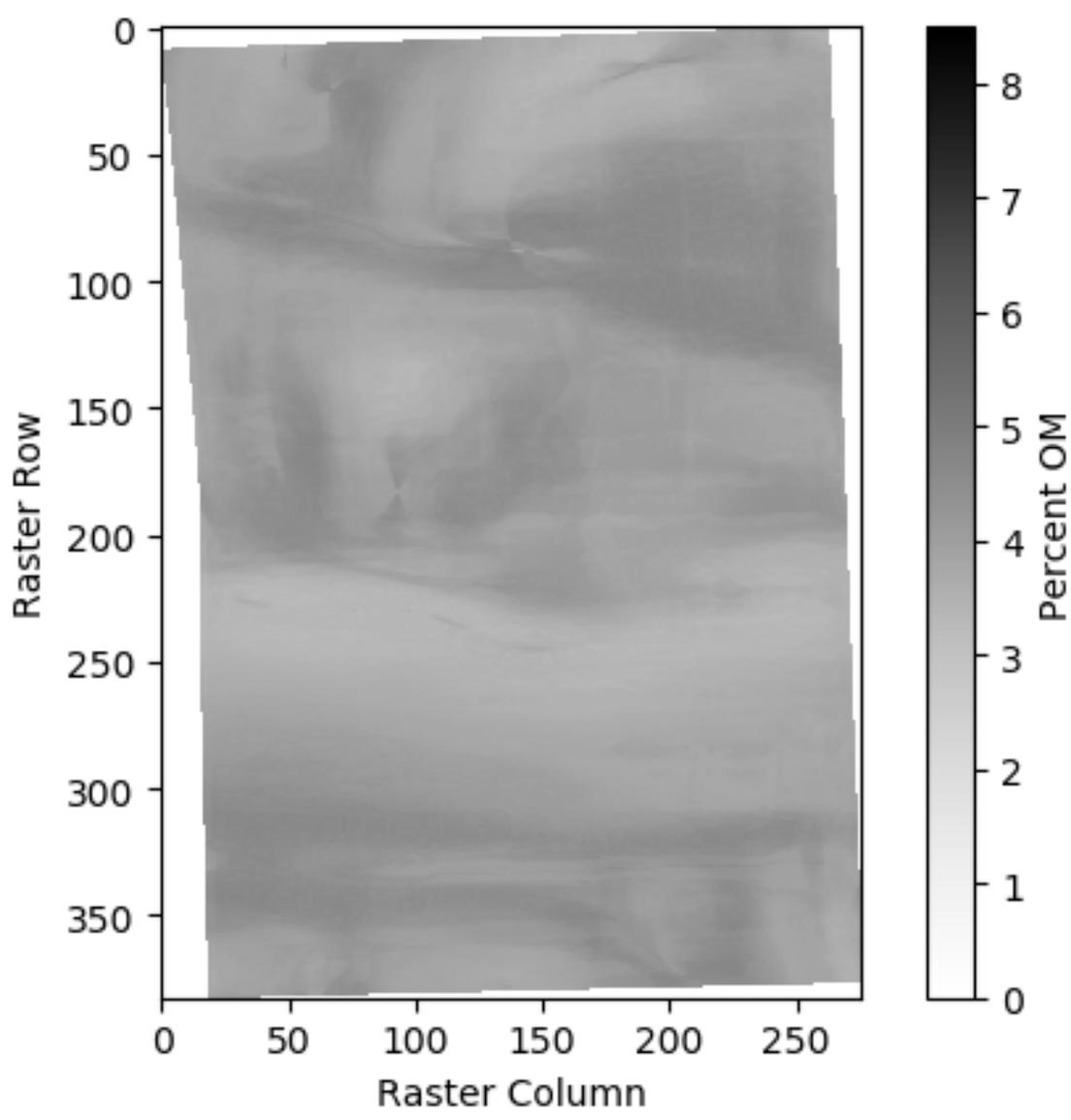


Figure 4.22.: Extra Trees Percent OM Prediction, Based On Relief Only.

### 4.2.3 Experiment 3: Spatial Position and Relief

Both variants performed better than any previous test in MAE, MAPE and RMSE (table 4.1). The performance gap between the three benchmarking methods and both variants were both significant, but as before, there was no significant variation between the RF experiments (table 4.2 & table 4.3). Like in experiment two, the Extra-Trees variant slightly under performed when compared to the RandomForest variant (though not significantly), which again points to a slightly weaker fit to the data. Interestingly, though both variants outperformed the previous best MAE, MAPE, and RMSE, the relief only models from experiment two were not far off. This was unexpected, as it was thought that the inclusion of two explanatory variables would explain much more of the value variation than one alone. The limited gains could perhaps be explained by several factors. The first possible explanation is that there were simply too many features in the feature set, which hampered model fitting. There were 28 spatial position features from experiment one, and 10 relief features from experiment two, so the number of features in the experiment three set, 38, well outnumbered the number of training points: 28. Another explanation could be that some of the same variations in the feature set were explained by both relief and spatial position, which would cut down on the performance gains. Whatever the reason for the modest gains, they still testify to the power of utilizing more than one explanatory variable.

The visual results (figs. 4.23 & 4.24) look very familiar, as they boast a strong resemblance to the visual results from experiment two. This resemblance is provided by the presence of the strong curvature artifacts present in each, and points to relief as a stronger predictor than spatial position. That said, the RandomForest variant displays at least one clearly visible circular artifact that corresponds to the lower artifact visible in the RandomForest spatial position only prediction (fig. 4.13). So, even if relief plays a stronger role in the prediction, artifacts alone show that spatial position is still a strong predictor. Apart from the artifacts, the experiment

three predictions appear smoother than the experiment two predictions, with some of the strong variation in the upper left tamed. The ExtraTrees prediction appears smoother, and the effects of the combined feature set is far more obvious between it and the ExtraTrees prediction from experiment two. The major relief artifacts in the lower half of the ExtraTrees prediction are almost invisible, as are almost all of the smaller relief artifacts that were visible in the experiment two prediction. Apart from artifacts, the two predictions do seem to be a blend of the experimental results from before, and, as the numerical results attest, this does seem to be a strength.

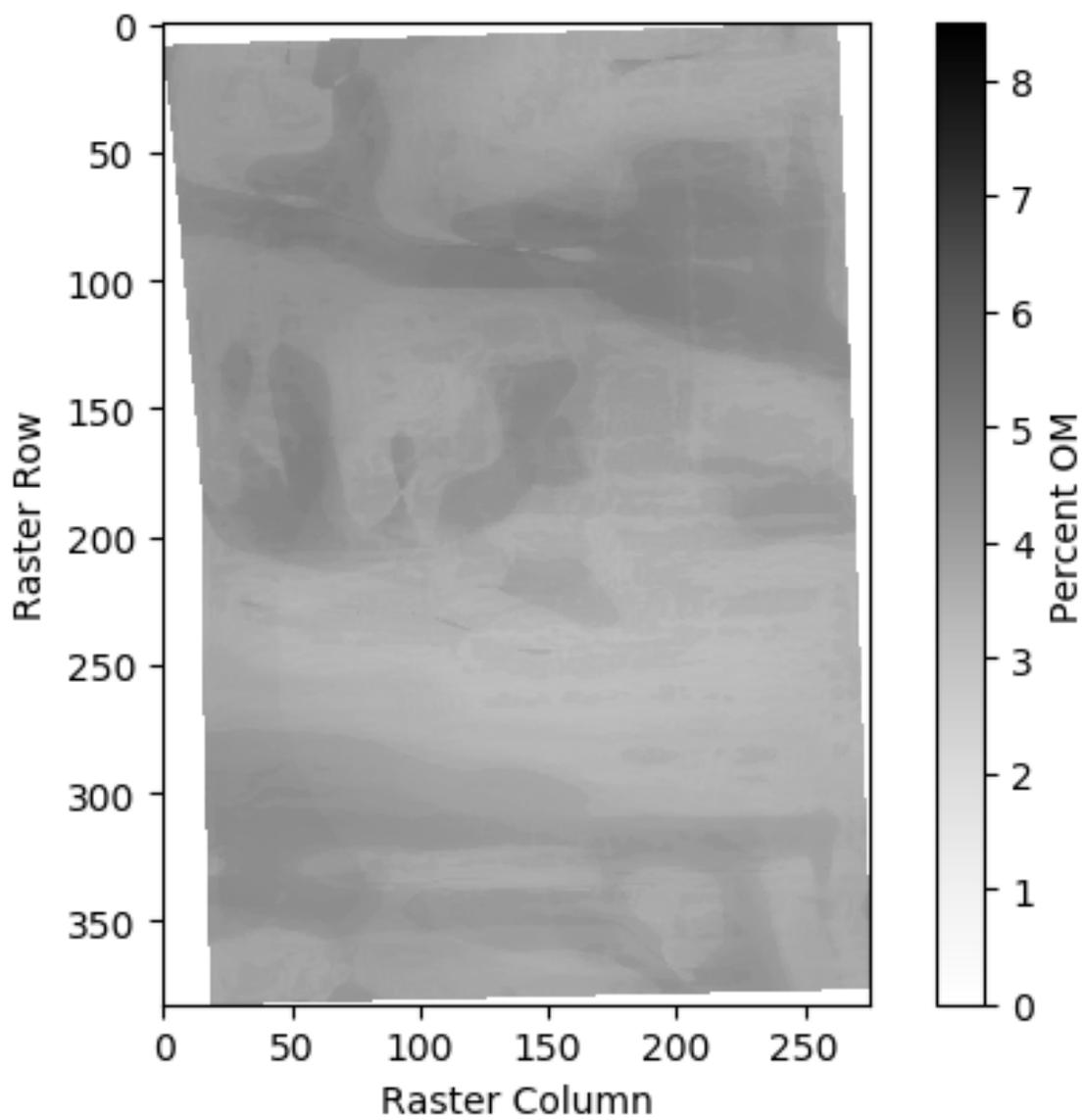


Figure 4.23.: Random Forests Percent OM Prediction, Based On Spatial Position And Relief.

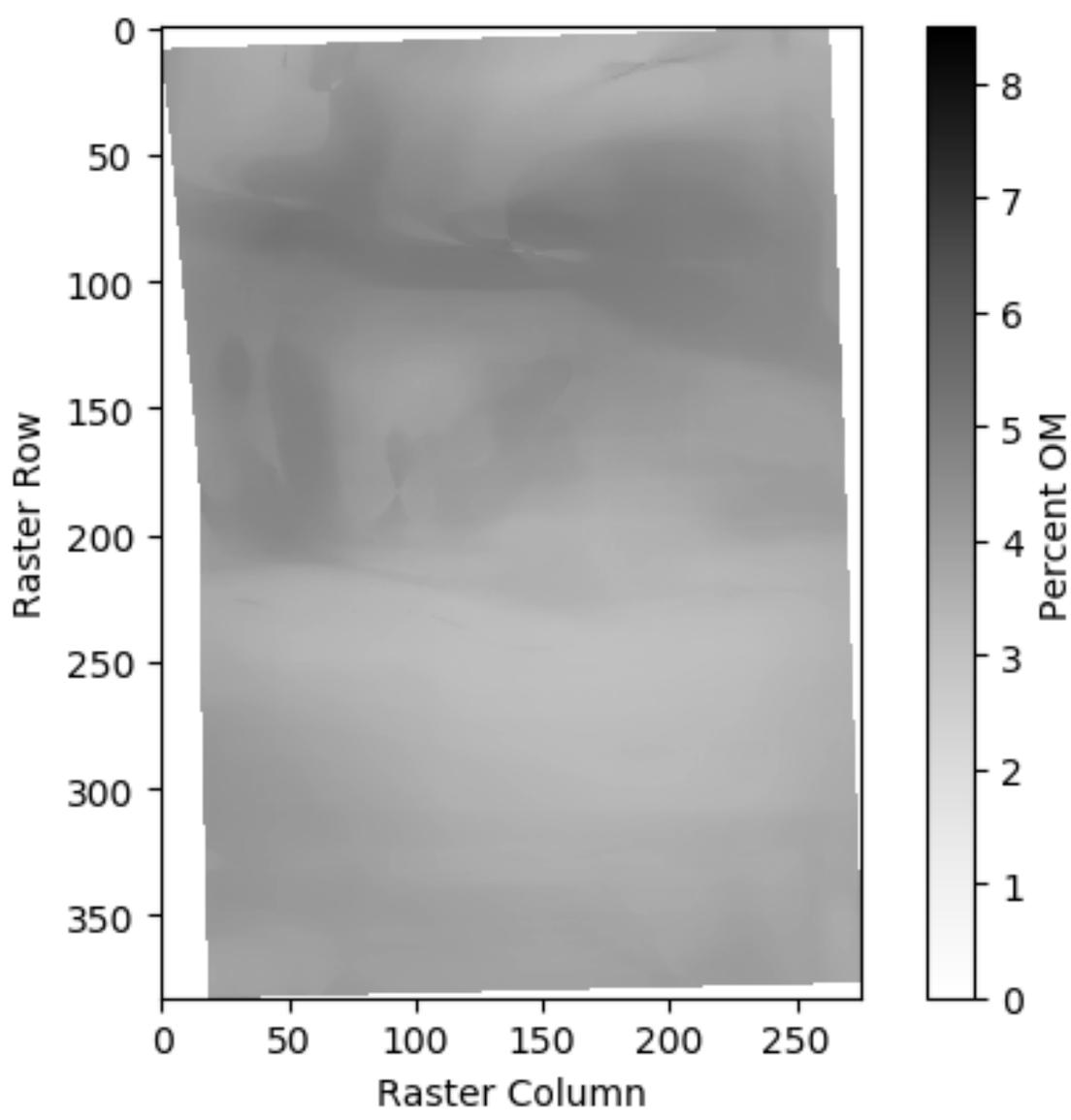


Figure 4.24.: Extra Trees Percent OM Prediction, Based On Spatial Position And Relief.

### 4.3 Objective 3: Make Accessible

Once the RF framework was proven as a viable interpolation method, work on a tool to make it accessible began. To be a drop-in replacement for OK or other interpolation methods, the tool had to require only two inputs from the user: referenced soil sample data, and a field boundary. In return, the user should receive only a prediction back. It was determined to create a simple JavaScript webtool to handle the inputs and output UI, and handle the computationally intense processing on the server.

#### 4.3.1 User Input

The landing page of the webtool (fig. 4.25), along with a brief introduction, offers the user two choices: uploading soil data for a new interpolation, or checking the status of a previous request. The application itself is a JavaScript React application, and all the graphics seen in the UI are implementations of the Material-UI design library. The React library provides a modular, low weight, UI with fast render times, and has a strong online support base. The Material-UI library provides a suite of pre-built HTML components like buttons, text blocks, cards, selection controls, etc. Using pre-made components saved time and also allowed for simpler uniformity throughout the application.

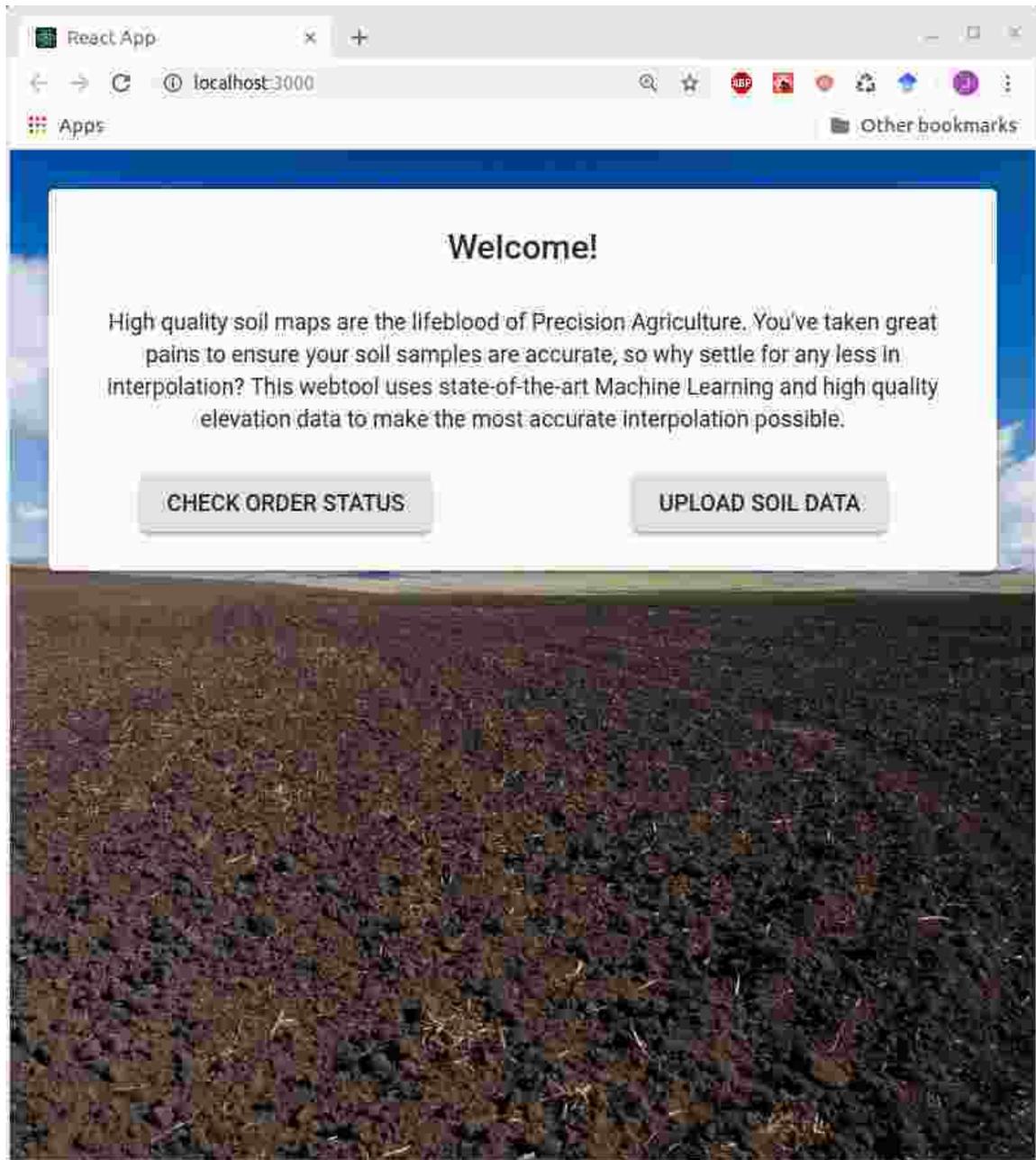


Figure 4.25.: The Landing Page of The UI.

To accept user soil data, the UI had to accept the two most common file types for field soil samples: shapefiles, and .csv files. Shapefile data is comprised of several files: geometry locations (.shp), geometry properties (.dbf), geometry projection (.prj), etc. Thus, the shapefile upload page (fig. 4.26) required the user to upload more than one

file before continuing. To simplify the application, it was assumed (and stated) that all data would be in the WGS84 projection, which is the projection most used for soil sampling. This assumption meant that the only files required from the user were a .shp file and a .dbf file. If the files were successfully loaded and of the correct type, the application would then allow the user to select the soil property that they were interested in interpolating. The application would then check that the property was indeed a numerical value and that the geometry was valid (not too small an area, not too large an area, enough points, etc.) before allowing the user to continue.

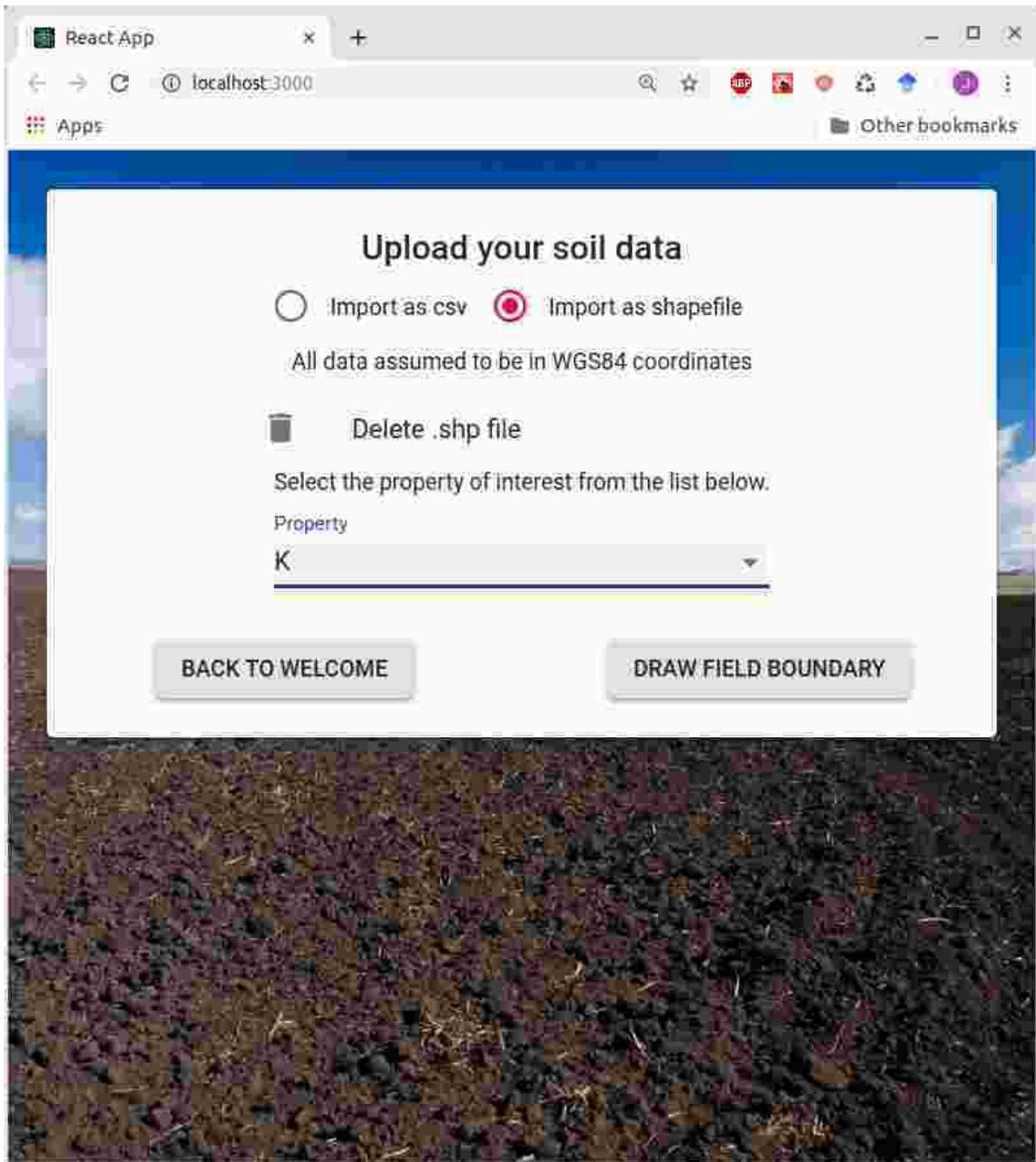


Figure 4.26.: The Shapefile Upload Page Of The UI.

If the user elected to instead upload a .csv file, they were given another upload page (fig. 4.27). The .csv format stores information about soil samples in a flat file format: each line corresponds to a sample, with latitude, longitude, and property values all in a row (fig. 3.6). After the user selected a file for upload, the application

checked that it met basic requirements (at least ten samples (the minimum for more than one split when modeling), no missing columns in any rows, correct file type) and the user was allowed to select if the file had a header or not. If the file had a header, the selection fields displayed the column values of the first row; if not, column indices were provided. The user was then prompted to select the columns that corresponded to the point Latitude, Longitude, and the property they were interested in interpolating. Upon the selection of each, the application verified that the column values were numerical and within the correct value ranges for each field. After all fields were selected, the application verified that the geometry was valid and allowed the user to continue to field boundary creation.

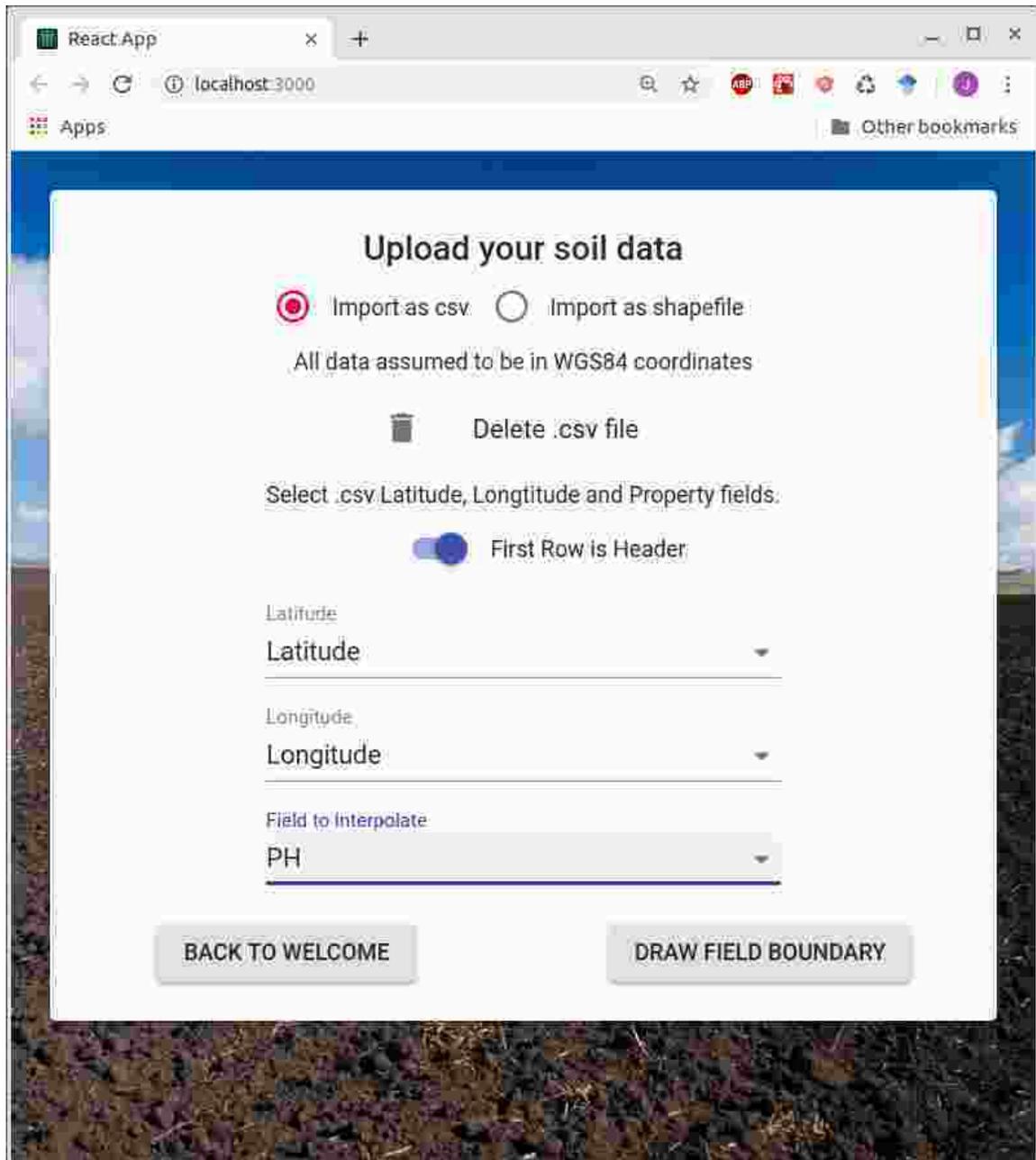


Figure 4.27.: The CSV Upload Page Of The UI.

While many users would be able to upload shapefiles of their field boundary, it was decided to streamline the application by simply asking users to draw their field boundary on a map (fig. 4.28). As field boundaries are typically not highly complex, this cut time and complexity from the user experience. Additionally, this allowed the

user to view their soil sample points and make sure they had rendered as expected. If an error was spotted (ex: the latitude and longitude were switched), the user could then return to the soil data upload page and correct the error. Once the user had finished drawing their boundary, the application would verify the geometry (not too big, not too small, no crosses, contains all points, etc.).

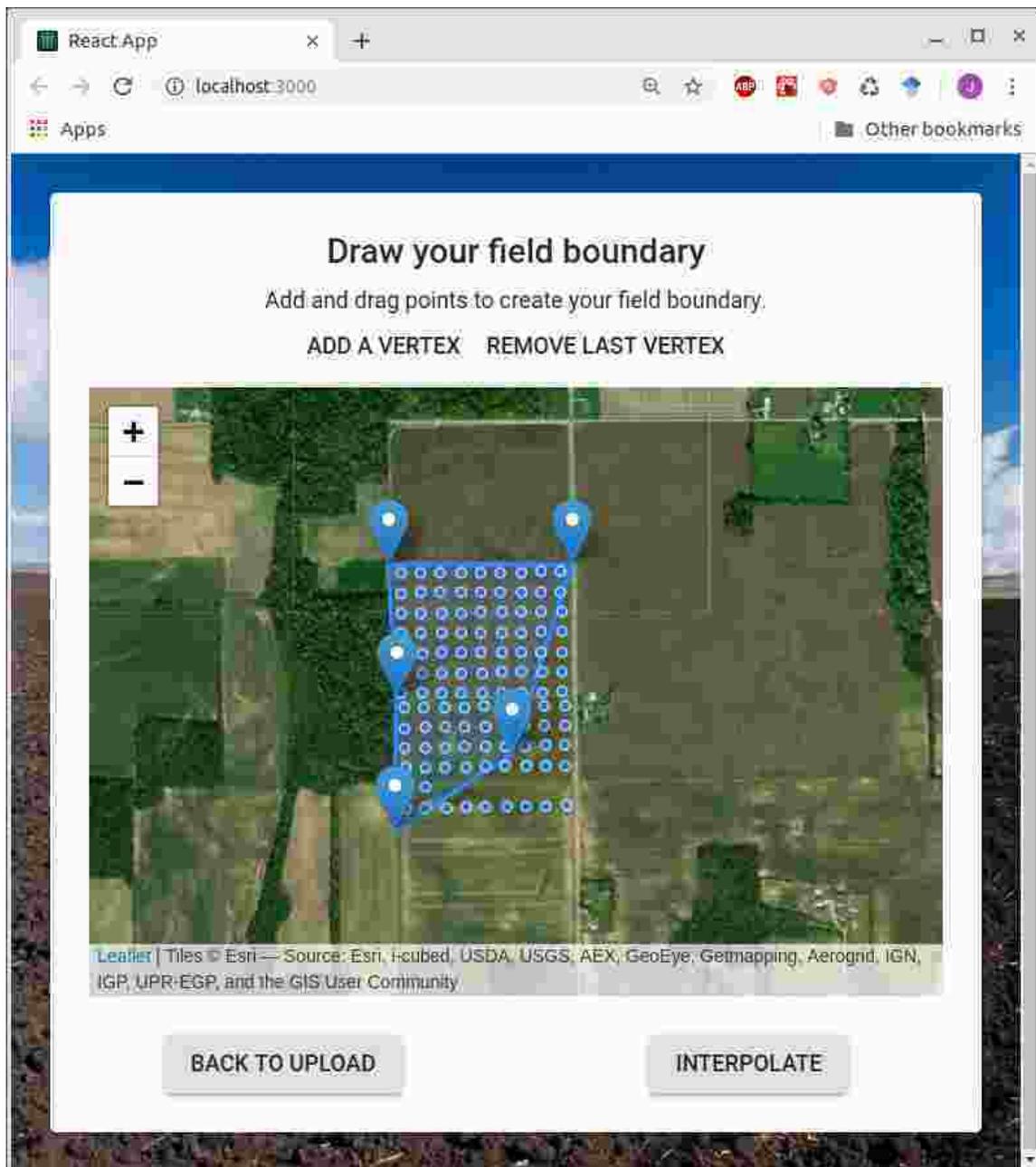


Figure 4.28.: The Field Boundary Creation Page Of The UI.

Once everything had been verified, they would be allowed to submit their request to the server. Figure 4.29 displays the page that verifies the entered email associated with the request.

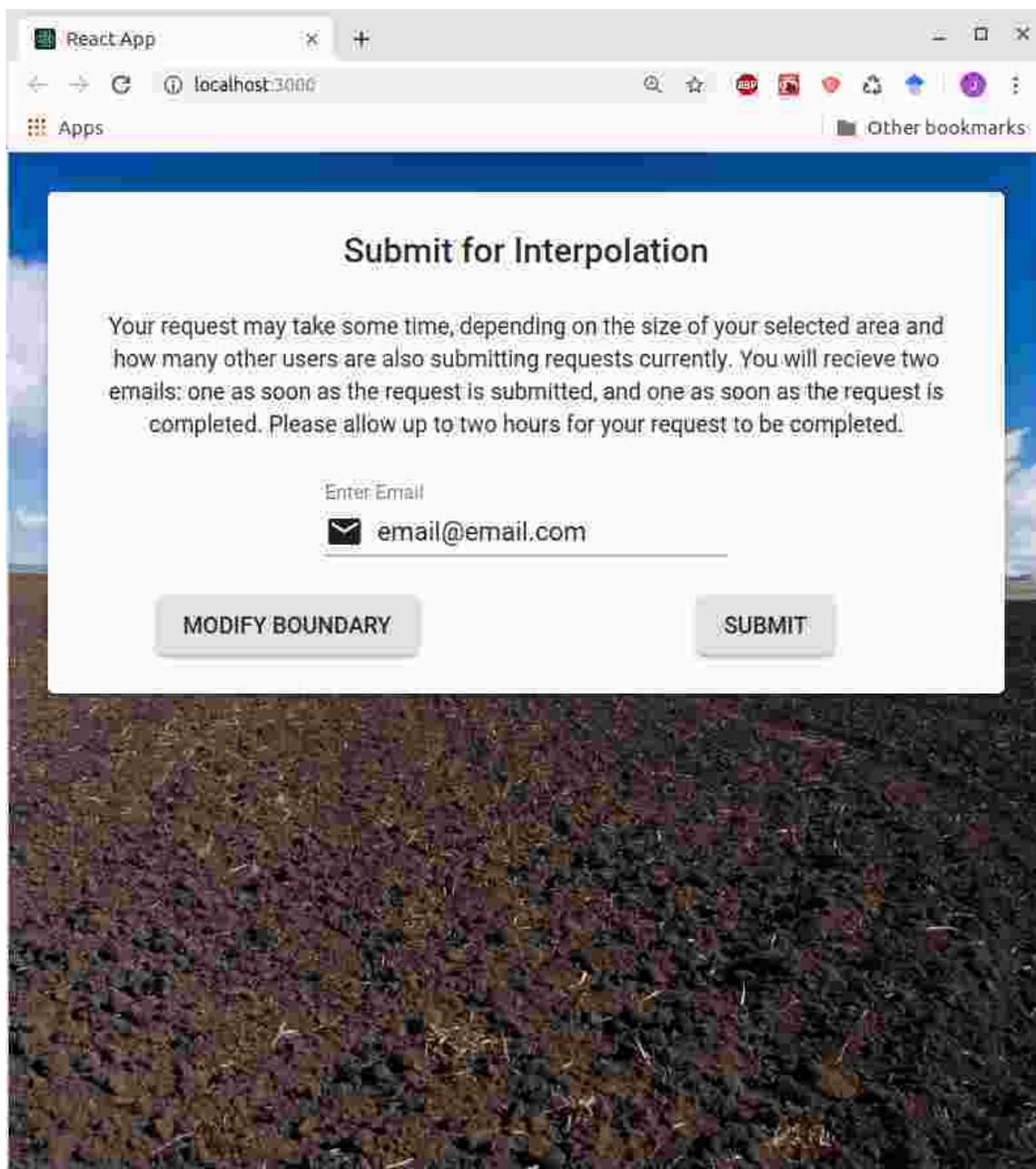


Figure 4.29.: The Submission Page Of The UI.

### 4.3.2 Processing

The first portion of the server encountered by a request is a Node Express based api. This api verifies the legitimacy of the request by checking the items included, creates a unique request id, then performs several tasks. The first task is the creation of a task management document in a MongoDB database (MongoDB, Inc., 2019). The MongoDB database holds the request data and information, like the time it was received, the soil data, the user email, etc. This can be referenced at any time by the server, and also serves as a log of activity for the administrator. After the successful creation of a database document for the request, the api sends an email to the user, informing them that their request had been received, giving them the request id, and informing them that the request was currently being processed. The last short term task was sending a response back to the webtool, informing the user that the request had failed or was in process, and giving them the request id. This resolves the request for the webtool, which then transitions to the status check page, (fig. 4.30), and the the actual processing can begin. Thus, while processing continues on the server, the user can prepare and submit other requests, leave the page, or even clear the browser cache.

The processing begins after the status is sent back to the webtool. First, the api starts the Python RF framework as a child process and passes it the request id. The Python script uses this id to retrieve the soil and boundary data from the database, and checks it over for validity. If at any time an error is caught, the error is logged on the database and the child process is closed. If the data is valid, the server uses the field boundary to determine which DEM tiles to download from the OpenTopography public repository. This is done by performing a spatial check on all the bounding boxes of all the raster tiles available for Indiana: if a buffered version of the field boundary crosses or is within any tile, that tile is downloaded. All the tiles are subsequently merged and clipped to the buffered field boundary bounding box. This elevation raster is the first feature created by the framework, and is used as the

snap raster for all subsequent features. After the DEM feature is complete, it is used as the basis to create the rest of the topographic features: the slope and curvatures at differing scales. Once this is complete, the field soil sample points are rasterized individually and used as the basis to create rasters of euclidean distance to each soil sample, the second set of features. Once the entire feature set has been created, the feature values corresponding to cells in the same position as the rasterized points are extracted and stacked into a list of features and training values ready for training.

This training set is then used to train an instance of the ExtraTreesRegressor (selected due to it's caution against overfitting), which is tuned to the same parameter values described in the experiments above. Once the model is trained, the feature values corresponding to each cell position in the snap raster are pulled and stacked into feature sets. This much larger set is fed into the model, which uses the feature values at each cell to make a prediction. Once the predictions are complete, the list of predictions is reshaped back into a 2-d array (of the same shape as the snap raster), the prediction is clipped to the field boundary, projected to WGS84, and written out as a .tif file into an exposed directory.

The Python script finishes by logging the results back into the database and informing the parent process, the Node Express api, that it's finished without errors. After this, the api kills the Python script and emails the user, informing them that the request they submitted is complete, their data can be retrieved, and reminding them of their request id. Once this is finished, the api process finally ends.

### 4.3.3 Output

At any point during, after, or before the server processing, the user can look up the status of a request using the Check Status page (fig. 4.30). Immediately after a request has been submitted, that particular request id is displayed above an input box, but the user can input any past request id they choose. Once the user submits a check status request, the api begins it's second process. Like the first, it checks the

validity of the request, and returns an error if something went wrong. If the request is valid, it checks the database for a task management document corresponding to the submitted request id. If one is found, it reads the document to check if processing is complete, if the results have been pruned from the server, etc. The status of the request is then returned to the webtool, along with a path to the output if the request is complete. This status is then displayed to the user, who can then check again or view the server output.

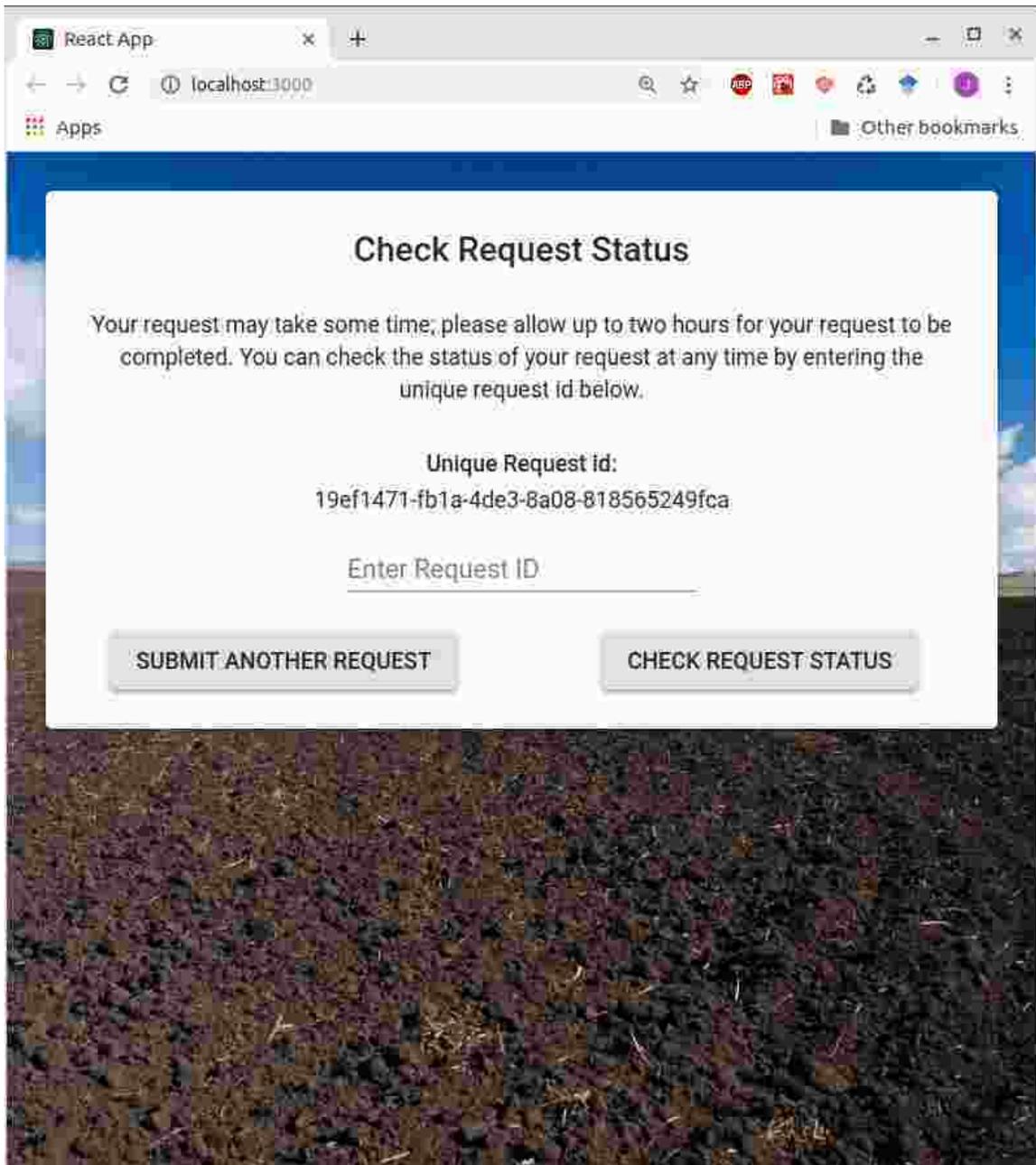


Figure 4.30.: The Status Display Page Of The UI.

After selecting to view their data, the user is given the output display page of the webtool (fig. 4.31). Here, the user can visually review the output of the server, and decide if they'd like to download the data or not. If they choose to download, the download button will link them to their GeoTIFF file hosted on the server, which will

automatically be downloaded. After downloading is complete, the user can return to the landing page (fig. 4.25), where they can choose to either submit a new request or check another request status.

The dockerized source code for the webtool and server can be retrieved from <https://github.com/jafiecht/soil-docker.git>.

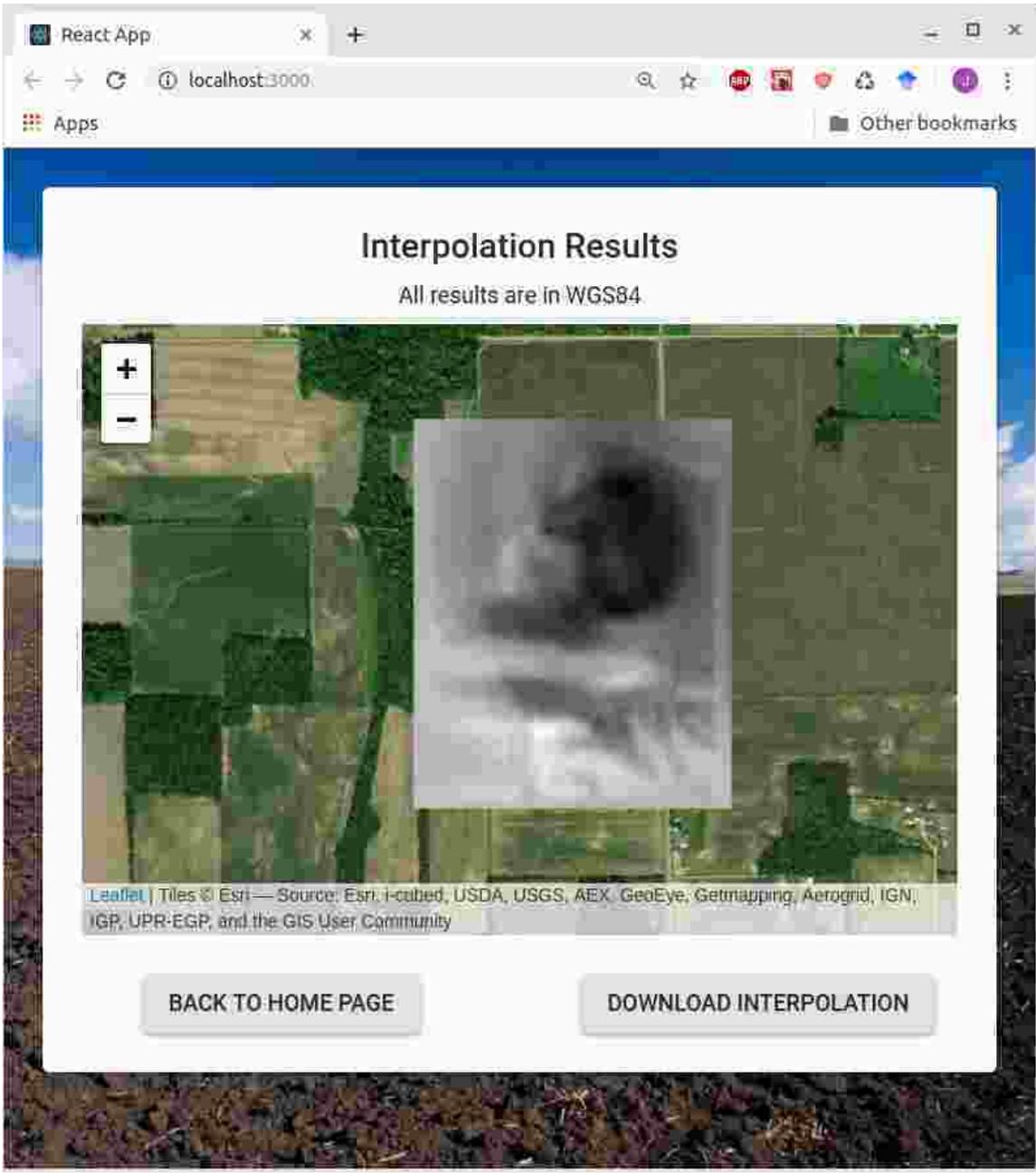


Figure 4.31.: The Output Display Page Of The UI.

## 5. CONCLUSIONS AND RECOMMENDATIONS

### 5.1 Objectives 1 & 2

One of the first conclusions illustrated by this study is the challenge of creating a quality interpolation from field soil samples. The process of fitting the OK model revealed a host of problems associated with low data: outliers affected stationarity, spatial autocorrelation between the points was limited, data values were skewed and trending, etc. While these particular characteristics were unique to this percent OM dataset, it's likely that some field soil sample datasets display similar issues. In fact, many fields are sampled at much lower densities, and may display these characteristics even more strongly. Modeling complex datasets like this well enough to make a sound interpolation is a challenge for any model, and for a model with a large set of assumptions like OK, it can be even more difficult. That said, the application of OK in this study had one considerable optimization ignored: the ability of the modeler to fit and trail different semi-variogram models beyond spherical, like stable or linear. It is possible that another model may have been able to pick up some level of spatial autocorrelation, which could have improved the OK prediction. In this particular study, the ability to change the model type was ignored in order to somewhat mimic the "black box" application of OK, where the user performs limited model intervention. The results of this study question the widespread use of OK for field soil interpolation if a user plans to leave all model parameters at a default. Similar problems appeared with the use of a grid based interpolation. While low data is not necessarily a problem for the grid based method, sparse sampling schemes unable to pick up property variation are, as was seen with the prediction of percent OM. Again, as many fields are sampled at even lower densities, the results of this study prompt the modeler to pause and consider the dataset at hand before using

a grid based interpolation. Surprising though it is, in some cases, it may simply be more accurate to predict a whole field average.

For a model with limited assumptions, experiment one showed the capability of an RF algorithm tuned for low data. RF proved itself capable in the field soil interpolation, as it outperformed the field average, the grid interpolation, and OK in model error, and presented insignificant model bias. While by no means flawless (unable to predict outside the training values, for one), RF seemed to present a viable drop-in replacement for other interpolative methods. It's flexibility and lack of need for case by case tuning make it an ideal method for field soil map interpolation.

Additionally, experiment two and three proved the viability of relief as a predictor of soil properties. Even in this case study, with it's subtle topographic features, relief proved to be a stronger predictor than even distance to known points. While untested in this study, logic would say that relief would be an even stronger predictor in a site with more dramatic relief. The results from this study, coupled with the availability of high resolution elevation data in Indiana and elsewhere, suggests that models utilizing relief, and possibly other public datasets, should be more widely used for field soil interpolation. While this does add complexity to the modeling process, the higher accuracy gained by inclusion seems worthwhile.

Future studies should more thoroughly evaluate RF models in other conditions and sites. This study, for instance, focused on only one soil property, percent OM, but the dataset used had more properties that could've potentially been tested. It's possible that other soil properties in this dataset would be more in line with the OK assumptions, or presented less spatial variability, and could've behaved very differently during prediction. Another interesting study would be the performance of RF at different dataset sizes (number of samples). Is there a threshold, high or low, where RF loses the performance edge? And of course, changing sites entirely would be very interesting: would a site with stronger topography yield a relief dataset more predictive of soil properties? Or, how would the model perform in a site with more dramatic soil property variation? A final interesting study would be a model comparison with

the soil sample scheme tailored to give traditional methods their best conditions. OK kriging would have likely been able to detect more spatial autocorrelation if there had been more sampled points within a short distance of each other. Or, perhaps, rather than a grid based interpolation, triangular elements could be fitted to the data, like LIDAR datasets or finite element analyses. These new methods would still be linear, but would break the mold from another side. While RF did perform well in this study, further benchmarking like these described would more fully validate it's performance.

## 5.2 Objective 3

As is the case for any program or software, there are many ways that the exposed prediction framework could be improved. The lowest hanging fruit is of course the inclusion of other DEM datasets. While the Indiana LIDAR generated DEM is high resolution and ideal for precision agriculture, there are other, more widespread DEMs available in different regions. A future improvement would be to catalog the best DEMs available in the contiguous United States and make prediction possible beyond just Indiana. This would of course require more extraction of tile boundaries and the investigation of which sites have public tile repositories, but would greatly expand the reach of the work. Another low hanging fruit is the optimization of the server-side processing. The processing is computationally expensive, and takes a considerable amount of time (*i*. 5 minutes for the dataset described), but has never been greatly optimized. For example of possible optimization, on a multi-core machine, the single thread Python script will max one core, but leave the others at normal operating loads. The Python scripts could be adapted to run some functions as multi-thread, which could cut the processing time immensely.

Beyond these two, of course, the webtool does lack the polish of a professionally developed site, and much time and energy could be expended into making the UI more intuitive and user friendly. It would be feasible, for instance, to accept data in

other projections than WGS84, which could save some users quite a bit of headache. Additionally, a more useful help page could better explain answers to frequently asked questions. Above all, it should be extensively trialed by agronomists and growers with real world experience and data. Incorporating more user feedback would truly refine the tool to what users need.

## REFERENCES

- Aitkenhead, M. and Coull, M. (2016). Mapping soil carbon stocks across scotland using a neural network model. *Geoderma*, 262:187–198.  
<http://dx.doi.org/10.1016/j.geoderma.2015.08.034>.
- Atherton, B., Morgan, M., Shearer, S., Stombaugh, T., and Ward, A. (1999). Site-specific farming: A perspective on information needs, benefits, and limitations. *Journal of Soil and Water Conservation*, 54(2):455–461.
- Behrens, T., Zhu, A.-X., Schmidt, K., and Scholten, T. (2010). Multi-scale digital terrain analysis and feature selection for digital soil mapping. *Geoderma*, 155(3-4):175–185. <http://dx.doi.org/10.1016/j.geoderma.2009.07.010>.
- Bostock, M. (2016). Shapefile. <https://github.com/mbostock/shapefile>.
- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.  
<http://dx.doi.org/10.1023/A:1010933404324>.
- Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and regression trees*. Routledge, New York.  
<https://doi.org/10.1201/9781315139470>.
- Bui, E. N., Loughhead, A., and Corner, R. (1999). Extracting soil-landscape rules from previous soil surveys. *Soil Research*, 37(3):495–508.  
<http://dx.doi.org/10.1071/s98047>.
- Burgess, T. M. and Webster, R. (1980). Optimal interpolation and isarithmic mapping of soil properties. *Journal of Soil Science*, 31(2):333–341.  
<http://dx.doi.org/10.1111/j.1365-2389.1980.tb02085.x>.

- Capellades, M. A., Reigber, S., and Kunze, M. (2009). Storm damage assessment support service in the u.s. corn belt using rapideye satellite imagery. *Proceedings of SPIE - The International Society for Optical Engineering*, 7472. <http://dx.doi.org/10.1117/12.830393>.
- CerebralJS Contributors (2019). Cerebraljs. <https://cerebraljs.com/>.
- Cressie, N. (1990). The origins of kriging. *Mathematical Geology*, 22(3):239–252. <http://dx.doi.org/10.1007/bf00889887>.
- Docker Inc. (2019). Docker. <https://www.docker.com/>.
- Erickson, B., Lowenberg-DeBoer, J., and Bradford, J. (2017). 2017 precision agriculture services dealership survey results. Retrieved from <https://agribusiness.purdue.edu/files/file/croplife-purdue-2017-precision-dealer-survey-report.pdf>.
- Erickson, B. and Widmar, D. A. (2015). 2015 precision agriculture services dealership survey results. Retrieved from <https://agribusiness.purdue.edu/files/resources/2015-crop-life-purdue-precision-dealer-survey.pdf>.
- Esri (2018). Arcmap 10.6.1. <http://desktop.arcgis.com/en/arcmap/>.
- Facebook Open Source Contributors (2019). Reactjs. <https://reactjs.org/>.
- Ferguson, R., Hergert, G., Schepers, J., Gotway, C., Cahoon, J., and Peterson, T. (2002). Site-specific nitrogen management of irrigated maize. *Soil Science Society of America Journal*, 66(2):544–553. <http://dx.doi.org/10.2136/sssaj2002.5440>.
- Franzen, D. W. and Peck, T. R. (1995). Field soil sampling density for variable rate fertilization. *Journal of Production Agriculture*, 8(4):568–574. <http://dx.doi.org/10.2134/jpa1995.0568>.

- Friedly, N. (2019). `express-rate-limit`.  
<https://www.npmjs.com/package/express-rate-limit>.
- GDAL/OGR contributors (2019). GDAL/OGR geospatial data abstraction software library. <https://gdal.org/>.
- Gillies, S. and Perry, M. (2019). Rasterio.  
<https://rasterio.readthedocs.io/en/stable/>.
- Google Developers (2019). Google apis.  
<https://developers.google.com/apis-explorer/>.
- Goovaerts, P. (1999). Geostatistics in soil science: state-of-the-art and perspectives. *Geoderma*, 89(1-2):1–45. [http://dx.doi.org/10.1016/s0016-7061\(98\)00078-0](http://dx.doi.org/10.1016/s0016-7061(98)00078-0).
- Gunduz, N. and Fokoue, E. (2015). Robust classification of high dimension low sample size data. *arXiv e-prints*.  
<https://ui.adsabs.harvard.edu/abs/2015arXiv150100592G>.
- Guo, P.-T., Li, M.-F., Luo, W., Tang, Q.-F., Liu, Z.-W., and Lin, Z.-M. (2015). Digital mapping of soil organic matter for rubber plantation at regional scale: An application of random forest plus residuals kriging approach. *Geoderma*, 237-238:49–59. <http://dx.doi.org/10.1016/j.geoderma.2014.08.009>.
- Helling, C. S., Chesters, G., and Corey, R. B. (1964). Contribution of organic matter and clay to soil cation-exchange capacity as affected by the ph of the saturating solution1. *Soil Science Society of America Journal*, 28(4):517–520. <http://dx.doi.org/10.2136/sssaj1964.03615995002800040020x>.
- Hengl, T., Heuvelink, G. B., and Stein, A. (2004). A generic framework for spatial prediction of soil variables based on regression-kriging. *Geoderma*, 120(1-2):75–93. <http://dx.doi.org/10.1016/j.geoderma.2003.08.018>.
- Hengl, T., Jesus, J. M. D., Heuvelink, G. B. M., Gonzalez, M. R., Kilibarda, M., Blagoti, A., Shangguan, W., Wright, M. N., Geng, X., Bauer-Marschallinger, B.,

- and et al. (2017). Soilgrids250m: Global gridded soil information based on machine learning. *Plos One*, 12(2):e0169748.  
<http://dx.doi.org/10.1371/journal.pone.0169748>.
- Hengl, T., Nussbaum, M., Wright, M. N., and Heuvelink, G. B. (2018). Random forest as a generic framework for predictive modeling of spatial and spatio-temporal variables. *PeerJ*, 6:e5518. <http://dx.doi.org/10.7717/peerj.5518>.
- Heung, B., Ho, H. C., Zhang, J., Knudby, A., Bulmer, C. E., and Schmidt, M. G. (2016). An overview and comparison of machine-learning techniques for classification purposes in digital soil mapping. *Geoderma*, 265:62–77.  
<http://dx.doi.org/10.1016/j.geoderma.2015.11.014>.
- Hoskinson, R. L., Rope, R. C., Blackwood, L. G., Lee, R. D., and Fink, R. K. (2004). The impact of soil sampling errors on variable rate fertilization. *International Precision Agriculture Conference*.
- Keeney, D. R. and Corey, R. B. (1963). Factors affecting the lime requirements of wisconsin soils 1. *Soil Science Society of America Journal*, 27(3):277–280.  
<http://dx.doi.org/10.2136/sssaj2002.5440>.
- Kempen, B., Brus, D. J., Heuvelink, G. B., and Stoorvogel, J. J. (2009). Updating the 1:50,000 dutch soil map using legacy soil data: A multinomial logistic regression approach. *Geoderma*, 151(3-4):311–326.  
<http://dx.doi.org/10.1016/j.geoderma.2009.04.023>.
- Le Cam, P. (2019). React-leaflet. <https://react-leaflet.js.org/>.
- LeafletJS Contributors (2019). Leafletjs. <https://leafletjs.com/>.
- Mansuy, N., Thiffault, E., Par, D., Bernier, P., Guindon, L., Villemaire, P., Poirier, V., and Beaudoin, A. (2014). Digital mapping of soil properties in canadian managed forests at 250m of resolution using the k-nearest neighbor method. *Geoderma*, 235-236:59–73. <http://dx.doi.org/10.1016/j.geoderma.2014.06.032>.

- Material-UI Contributors (2019). Material-ui. <https://material-ui.com/>.
- McBratney, A., Mendonca Santos, M., and Minasny, B. (2003). On digital soil mapping. *Geoderma*, 117(1-2):3–52.  
[http://dx.doi.org/10.1016/S0016-7061\(03\)00223-4](http://dx.doi.org/10.1016/S0016-7061(03)00223-4).
- Mcbratney, A., Webster, R., and Burgess, T. (1981). The design of optimal sampling schemes for local estimation and mapping of regionalized variables: Theory and practice. *Computers & Geosciences*, 7(4):335–365.  
[http://dx.doi.org/10.1016/0098-3004\(81\)90078-9](http://dx.doi.org/10.1016/0098-3004(81)90078-9).
- McBride, J., Van den Bossche, J., Wasserman, J., Jordahl, K., Wolf, L., and Rocklin, M. (2019). Geopandas. <http://geopandas.org/index.html>.
- Minasny, B. and McBratney, A. (2016). Digital soil mapping: A brief history and some lessons. *Geoderma*, 264:301–311.  
<http://dx.doi.org/10.1016/j.geoderma.2015.07.017>.
- MongoDB, Inc. (2019). Mongoddb. <https://www.mongodb.com/>.
- Moore, I., Gessler, P., Nielsen, G., and Peterson, G. (1993). Soil attribute prediction using terrain analysis. *Soil Science Society of America Journal*, 57(2).  
<http://dx.doi.org/10.2136/sssaj1993.572npb>.
- Nawar, S., Corstanje, R., Halcro, G., Mulla, D., and Mouazen, A. M. (2017). Delineation of soil management zones for variable-rate fertilization. *Advances in Agronomy*, 143:175–245. <http://dx.doi.org/10.1016/bs.agron.2017.01.003>.
- Node.js Foundation (2019a). Expressjs. <https://expressjs.com/>.
- Node.js Foundation (2019b). Nodejs. <https://nodejs.org/en/>.
- NumFOCUS Foundation (2019). Numpy. <https://numpy.org/>.

- Oliver, M. A. and Webster, R. (1986). Semi-variograms for modelling the spatial pattern of landform and soil properties. *Earth Surface Processes and Landforms*, 11(5):491–504. <http://dx.doi.org/10.1002/esp.3290110504>.
- Padarian, J., Minasny, B., and McBratney, A. B. (2019). Using deep learning for digital soil mapping. *SOIL*, 5(1):79–89. <http://dx.doi.org/10.5194/soil-5-79-2019>.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Ramcharan, A., Hengl, T., Nauman, T., Brungard, C., Waltman, S., Wills, S., and Thompson, J. (2018). Soil property and class maps of the conterminous united states at 100-meter spatial resolution. *Soil Science Society of America Journal*, 82(1):186–201. <http://dx.doi.org/10.2136/sssaj2017.04.0122>.
- Reinman, A. (2019). Nodemailer. <https://nodemailer.com/about/>.
- Reitz, K. (2019). Requests. <https://2.python-requests.org/en/master/>.
- Schloeder, C., Zimmerman, N., and Jacobs, M. (2001). Comparison of methods for interpolating soil properties using limited data. *Soil Science Society of America Journal*, 65(2):470–479. <http://dx.doi.org/10.2136/sssaj2001.652470x>.
- Shaikhina, T., Lowe, D., Daga, S., Briggs, D., Higgins, R., and Khovanova, N. (2019). Decision tree and random forest models for outcome prediction in antibody incompatible kidney transplantation. *Biomedical Signal Processing and Control*, 52:456–462. <http://dx.doi.org/10.1016/j.bspc.2017.01.012>.
- Smith, M., Goodchild, M., and Longley, P. (2018). *Geospatial Analysis: A Comprehensive Guide to Principles Techniques and Software Tools*. Winchelsea Press, Winchelsea.

- Smith, M. P., Zhu, A.-X., Burt, J. E., and Stiles, C. (2006). The effects of dem resolution and neighborhood size on digital soil survey. *Geoderma*, 137(1-2):58–69. <http://dx.doi.org/10.1016/j.geoderma.2006.07.002>.
- State of Indiana (2011). 2011-2013 indiana statewide lidar. Retrieved from <http://dx.doi.org/10.5069/G9959FHZ>.
- Subburayalu, S., Jenhani, I., and Slater, B. (2014). Disaggregation of component soil series on an ohio county soil survey map using possibilistic decision trees. *Geoderma*, 213:334–345. <http://dx.doi.org/10.1016/j.geoderma.2013.08.018>.
- Taghizadeh-Mehrjardi, R., Sarmadian, F., Minasny, B., Triantafyllis, J., and Omid, M. (2014). Digital mapping of soil classes using decision tree and auxiliary data in the ardakan region, iran. *Arid Land Research and Management*, 28(2):147–168. <http://dx.doi.org/10.1080/15324982.2013.828801>.
- TurfJS Contributors (2019). Turfjs. <https://turfjs.org/>.
- Vaysse, K. and Lagacherie, P. (2017). Using quantile regression forest to estimate uncertainty of digital soil mapping products. *Geoderma*, 291:55–64. <http://dx.doi.org/10.1016/j.geoderma.2016.12.017>.
- Wollenhaupt, N. C., Mulla, D. J., and Crawford, C. G. (1997). Soil sampling and interpolation techniques for mapping spatial variability of soil properties. In Pierce, F. and Sadler, E., editors, *The State of Site-Specific Management for Agriculture*, pages 19–53. American Society of Agronomy, Crop Science Society of America, Soil Science Society of America, Madison, WI. <http://dx.doi.org/doi:10.2134/1997.stateofsitespecific.c2>.
- Zabriskie, M. and Uraltsev, N. (2019). Axios. <https://www.npmjs.com/package/axios>.

- Zhu, A.-X., Band, L. E., Dutton, B., and Nimlos, T. J. (1996). Automated soil inference under fuzzy logic. *Ecological Modelling*, 90(2):123–145.  
[http://dx.doi.org/10.1016/0304-3800\(95\)00161-1](http://dx.doi.org/10.1016/0304-3800(95)00161-1).
- Zhu, A.-X., Burt, J. E., Smith, M., Rongxun, W., and Jing, G. (2008). The impact of neighbourhood size on terrain derivatives and digital soil mapping. In Zhou, Q., Lees, B., and Tang, G.-a., editors, *Advances in Digital Terrain Analysis*, pages 333–348. Springer Berlin Heidelberg, Berlin, Heidelberg.  
[http://dx.doi.org/10.1007/978-3-540-77800-4\\_18](http://dx.doi.org/10.1007/978-3-540-77800-4_18).
- Zhu, A. X., Hudson, B., Burt, J., Lubich, K., and Simonson, D. (2001). Soil mapping using gis, expert knowledge, and fuzzy logic. *Soil Science Society of America Journal*, 65(5):1463–1472. <http://dx.doi.org/10.2136/sssaj2001.6551463x>.
- Zhu, A. X., Liu, J., Du, F., Zhang, S. J., Qin, C. Z., Burt, J., Behrens, T., and Scholten, T. (2015). Predictive soil mapping with limited sample data. *European Journal of Soil Science*, 66(3):535–547. <http://dx.doi.org/10.1111/ejss.12244>.
- Zimmerman, J. and Shallenberger, J. (2016). Gis topographic wetness index (twi) exercise steps. Retrieved from  
<https://www.srbc.net/pennsylvania-lidar-working-group/docs/twi-srbc.pdf>.