GEOSPATIAL SEMANTIC PATTERN RECOGNITION IN VOLUNTEERED GEOGRAPHIC DATA USING THE RANDOM FOREST ALGORITHM

by

Richard Wen

Bachelor of Environmental Studies

Honours Geomatics, Computer Science Minor

University of Waterloo, 2014

A thesis

presented to Ryerson University

in partial fulfilment of the

requirements for the degree of

Master of Spatial Analysis

in the program of

Spatial Analysis

Toronto, Ontario, Canada

Author's Declaration

Abstract

Geospatial Semantic Pattern Recognition in
Volunteered Geographic Data Using the Random Forest Algorithm

by
Richard Wen
B.E.S (Honours Geomatics, Computer Science Minor), University of Waterloo, 2014
Master of Spatial Analysis, Ryerson University

The ubiquitous availability of location technologies has enabled large quantities of Volunteered Geographic Data (VGD) to be produced by users worldwide. VGD has been a cost effective and scalable solution to obtaining unique and freely available geospatial data. However, VGD suffers from reliability issues as user behaviour is often variable. Large quantities make manual assessments of the user generated data inefficient, expensive, and impractical. This research utilized a random forest algorithm based on geospatial semantic variables in order to aid the improvement and understanding of multi-class VGD without ground-truth reference data. An automated Python script of a random forest based procedure was developed. A demonstration of the automated script on OpenStreetMap (OSM) data with user generated tags in Toronto, Ontario, was effective in recognizing patterns in the OSM data with predictive performances of ~71% based on a class weighted metric, and the ability to reveal variable influences and outliers.

Acknowledgements

I would like to express my sincere gratitude to my advisor Dr. Claus Rinner for his guidance, help, funding, support, and the freedom that I was granted for this thesis. The meetings and conversations I had with Dr. Rinner were always interesting, and opened up my mind to new ideas and possibilities.

I would like to thank Dr. Eric Vaz and Dr. Tony Hernandez in the Department of Geography at Ryerson University for their support and review of this thesis. Special thanks to Dr. Lu Wang for chairing the thesis defense and Dr. Victoria Fast for reviewing the thesis defense contents and providing feedback. I would also like to thank The Ryerson Centre for Cloud and Context-Aware Computing (RC4) for providing me with a remote server for development and experimentation. I am grateful to the Geothink Social Sciences and Humanities Research Council (SSHRC) Partnership Grant and the Yeates School of Graduate Studies at Ryerson University for funding provided to me for the duration of the thesis.

I am also grateful to be granted the opportunity to work alongside my colleagues at the Centre for Global Health Research (CGHR) at St. Michael’s Hospital. Although the work was not directly related to my thesis, the environment was inspiring, and supportive; working at CGHR enabled me to develop and sharpen skills that were very helpful in the progress of my thesis.

I am grateful for the continued support of my friends and family. Lastly, I would like to say thanks *Hilary* for the years of unconditional encouragement and support.

Table of Contents

List of Tables

List of Figures

List of Acronyms

CAS	Complex Adaptive Systems
CNND	Class-specific Nearest Neighbour Distance
CSV	Comma Separated Values
LIDAR	Light Detection and Ranging
MDI	Mean Decrease Impurity
MDS	Multi-Dimensional Scaling
OOB	Out-of-bag
OSM	OpenStreetMap
SVM	Support Vector Machines
VGD	Volunteered Geographic Data
VGI	Volunteered Geographic Information

CHAPTER 1: INTRODUCTION

Volunteered Geographic Information (VGI), a type of user-generated content, utilizes the engagement of private citizens through the internet to create geospatial data that are free and widely available to the public (Goodchild, 2007). VGI provides an efficient and cost-effective approach to generating geospatial data in masses to fill incomplete datasets, enhance community knowledge of spatial information, and improve geographic collaboration through the World Wide Web as demonstrated by OpenStreetMap (OSM) – a successful online VGI platform that generates geospatial data through volunteer contributors that digitize and annotate geographic objects. The created geospatial data within VGI systems (Fast & Rinner, 2014), termed Volunteered Geographic Data (VGD) in this thesis, suffers from data quality issues due to user misinterpretation and inexperience (Koukoletsos et al., 2012). The reliability of VGD is questionable without efforts to assess the accuracy, errors, and completeness of generated data. It is also impractical to manually evaluate and assess VGD due to the abundance of content generated by volunteers. An automated method to evaluate and assess VGD is required to evaluate the reliability and utility of the data.

The emergence of artificial intelligence and rapid improvement of computer hardware has enabled powerful machine learning algorithms, a system that is able to learn without being explicitly programmed, to be used for complex, non-linear, tasks that are infeasible and impractical for humans to perform (Rao, 2005). Machine learning algorithms have been used to discover structure in large volumes of data such that it may recognize data patterns, reduce multidimensional data for improved understanding, and perform predictions on unseen data (Kodratoff & Michalski, 2014). Random forests, a method of machine learning based on the assembling of decision tree predictors, have been effectively used to detect patterns in large-scale, varying, and erroneous data (Breiman, 2001). Random forests have been used commonly in geospatial applications such as classification of remotely sensed imagery, and modelling of spatial phenomena such as disease spread and species distribution (Furlanello et al., 2003; Rodriguez-Galiano et al., 2012). The erroneous and irregularly distributed nature of VGD creates an issue for common machine learning algorithms – namely, neural networks and Support Vector Machines (SVM) – and often require adjustments to the input data or algorithms to achieve better results (Hastie et al., 2009). Random forests are more forgiving to errors and irregularly distributed data without the need for rigorous efforts to tune parameters and preprocess input data (Lancashire et al., 2009). The efficiency and robustness of the random forest algorithm provides a suitable approach in the pattern recognition of VGD.

The random forest algorithm, and machine learning algorithms in general, require variables that describe the data in order to search for useful patterns. Geospatial semantic variables, variables that hold meaning in geographic space, provide values to quantitatively describe geographic objects such that both machines and humans can understand. For example, a distance variable can describe the proximity of two geographic objects in space, and an area variable can describe the size of the geographic objects. Geospatial semantic variables can be used by machine learning algorithms to determine patterns in VGD as these variables may improve the predictive power of machine learning models generated from the algorithms. For example, if the area and the distance to the nearest bus stop and road are given for an unknown geographic object, then it is possible to assign a certainty of what the geographic object may be based on the data provided. The combination of geospatial semantic variables and a random forest algorithm can discover VGD patterns in an automated and efficient way that can enhance manual human analysis of VGD.

1.1 Volunteered Geographic Data

VGD is dynamic in nature and is often criticized for its reliability due to data quality issues from errors, uncertainty, misinterpretations, and inexperience (Sui et al., 2012). VGD is valuable in that it provides cost effective and unique data in masses with often high temporal resolutions that may not be feasible or practical to produce by other means (Feick & Roche, 2012). Many efforts have been made towards the quality assessment of VGD but these studies often utilize ground-truth reference data or a metric developed by experts (Mooney et al., 2010; Koukoletsos et al., 2012; D'Antonio et al., 2014). VGD also suffers from unpredictability such that automated metrics or methods by experts for one study area are not transferable to other areas, even if the same users generated data for all of the study areas (See et al., 2013). Machine learning algorithms are potentially suitable for VGD as they can be flexible, automated, customized, and adapted to the dynamic nature and variable quality of VGD.

1.2 OpenStreetMap

OSM, an online VGI platform, enables users with an internet connection to digitize geographic objects in the form of points, lines, and polygons, followed by the annotation of tags to describe the geographic objects by common feature classes such as roads and restaurants (Haklay, 2008). A tag consists of a key/value pair in which the key is a broad category of geographic objects and the value further details the geographic object being tagged. A few examples of basic tags include "amenity=hospital", "road=secondary", and "place=neighbourhood". These tags are tied to geographic objects in space, and can be utilized to find meaningful patterns in different areas. The voluntary nature and success of OSM has led to large and freely-available volumes of VGD that is up to date and continuously changing, but not without questionable quality from human errors and inconsistencies. Many efforts have been made to utilize ground truth data to directly assess the quality and structure of OSM data, but automated approaches may be more practical when ground truth data are non-existent or incomplete (Girres & Touya, 2010). The geospatial characteristics, voluntary nature, and large quantities of OSM data provide a suitable real-world dataset to demonstrate the effectiveness and utility of a machine learning algorithm on classified VGD (Gislason et al., 2006).

1.3 Machine Learning and Pattern Recognition

Machine learning, a subfield that grew from computer science, involves predicting outputs using computer algorithms that perform classification or regression in a supervised or unsupervised manner (Hastie, et al., 2009). Classification refers to the prediction of qualitative values and regression refers to the prediction of quantitative values. For example, the prediction of temperature is a regression problem, while the prediction of forest cover type is a classification problem. Training data is data in which the computer algorithm uses to recognize patterns from. Supervised machine learning uses training data that has the desired output known for each sample. Unsupervised machine learning uses training data that does not have the desired output known for each sample. Unsupervised machine learning is often used for clustering in which the goal is to create groups in which samples are similar within the groups and each group is dismilar with other groups (Kodratoff & Michalski, 2014). For example, if a dataset contains explanatory variables that describe the forest cover type, a supervised algorithm requires that the forest cover type be known while an unsupervised algorithm does not. VGD would require supervised learning, where the desired output is known from the data generated by volunteers, and classification in the case of OSM as the desired outputs are tags that describe qualitative values. The random forest algorithm has the ability to perform supervised machine learning for VGD and classification for OSM data to create an automated pattern recognition model that has uses in enhancing the understanding of variable influences and data quality.

Pattern recognition, a branch of machine learning originating from engineering, is focused on the discovery of regularities in data with higher focus on efforts to explain the regularities (Bishop, 2006). The explanation of discovered patterns is important in addition to creating a model that has the ability to predict outputs from unseen data samples with high accuracy. Probability is crucial to the explanation of discovered patterns in the data. Probability explains how likely a predicted output is from a machine learning model. Probability, in this case, can defined as the likelihood of a prediction output and may be interpreted as how certain or uncertain the model is about one or more predictions given a set of training data (Feller, 1968). Similarity is also important to explaining patterns in the data. Similarity explains how regular or irregular one or more samples are in the data, and can be determined using quantitative measures suitable for each particular machine learning model. Similarity measures produce higher values when the dependency of two comparable data samples increases (Goshtasby, 2012). The random forest algorithm applies the concepts of probability and similarity, which enables it to explain discovered patterns in VGD with a quantitative approach.

1.4 Geospatial Semantics

Semantics, the study of meanings, provides a foundation for the transfer of knowledge between human users and machines (Christopher, 1998). The utilization of semantic data with random forests enables the machine to recognize meaningful patterns that are understood by both the user and the machine. Geospatial semantics are the meanings of referents in geographic space. Spatial relationships and spatial reasoning form the foundation of geospatial semantics with extensions from cognitive science (Janowicz et al., 2012). The use of semantics includes the application of logic in the form of triples, which is a subject-predicate-object structure that both humans and machines can understand (Christopher, 1998; Rohloff et al., 2007). For example, *"Person A knows Person B"* where *Person A* is the subject, *knows* is the predicate, and *Person B* is the object. Semantics can be incorporated in a geospatial context such that the predicate refers to spatial terms. For example, *"Place A is near Place B"* where *Place A* is the subject, *is near* is the predicate, and *Place B* is the object. Geospatial semantics allow machines to understand semantic data at the geographic scale in order to provide a framework for interoperability, knowledge standards, and information extraction. It is possible to create geospatial semantic variables from geographic data to provide quantitative information for the machine to learn from, and human users to interpret.

1.5 Research Objectives

The main objective of this research was to discover and explain patterns in VGD with automated random forest models trained on geospatial semantic variables using OSM data in the City of Toronto, Ontario, as a demonstration. The research required that the following tasks be achieved in order to fulfill the main objective:

(1) Extraction of geospatial semantic variables from OSM data;

(2) Empirical determination of the optimal number of decision trees for a random forest trained on the full Toronto OSM data with extracted geospatial semantic variables;

(3) Training of several random forests with optimized parameters to evaluate performance stability on unseen Toronto OSM data using F1 scores;

(4) Calculation of variable importance, variable contributions, proximities, and probabilities for parameter optimized random forest trained on the full Toronto OSM data; and

(5) Automation of Steps 1-4 such that the research can be reproduced for multi-class vector-based VGD not limited to OSM data or the Toronto area

CHAPTER 2: LITERATURE REVIEW

2.1 Nearest Neighbours

Nearest neighbours, defined as objects in space that are within a close proximity of each other, can serve as predictive variables that contribute to patterns existing in geospatial data (Anselin, 1990). The concept of nearest neighbours is related to Tobler's First Law of Geography which states that "everything is related to everything else, but near things are more related than distant things." (Tobler, 1970, p. 236). Complex Adaptive Systems (CAS) theory suggests that multiple independent components can interact to form complex behaviours (Holland, 1992). CAS can be applied to geographic space where interactions between local geographic objects can create complex global behaviours that model real-world phenomena (Miller, 2004). The nearest neighbours can be seen as the local geographic objects that form the complex global behaviours of a select study area. When the number of geographic objects is large, it is reasonable to assume that geographic objects are similar when they are close in space (Duda & Hart, 1973). When most or all geographic objects reside within a defined subspace, the nearest neighbour distances are likely to not vary greatly as the distances are likely to approach the average distance (Beyer et al., 1998). This concept can be applied to the predictability of target geographic objects that are spatially related to neighbouring geographic objects.

Nearest neighbour distances are globally meaningful, such they improve the predictability of target geographic objects, when most or all neighbouring geographic objects do not fall within a few subspaces. When the neighbouring geographic objects reside in a few subspaces, the nearest neighbours distances are globally non-meaningful as the distances are likely to approach a random distribution. Figure 1 exemplifies the nearest neighbour distance concept applied to globally meaningful and non-meaningful nearest neighbours. The image on the right does not hold useful distances for predicting each target geographic object as the distances of the neighbouring geographic objects would be relatively random due to the existence of neighbouring geographic objects at only one subspace. Providing the distances of neighbouring geographic objects does not improve the predictability of each target geographic object. Thus, the existence of target geographic objects is likely not affected by the existence of the neighbouring geographic objects. The image on the left exemplifies an opposite effect. Neighbouring geographic objects exist inside each subspace, which causes the nearest neighbour distances to approach an average value. Thus, the predictability of each target target geographic object is improved by knowing the distance to its neighbouring geographic objects. Geospatial semantic variables using nearest neighbour distances may potentially contribute to the predictive power of machine learning algorithms, and simplify geographic understanding of spatial patterns for human users if the nearest neighbour distances are meaningful. For example, given an unknown geographic object and its distance to the nearest bus stop, it is possible to say with certainty that the unknown geographic object is a school if most or all schools in the study area have small distances to bus stops. The concept of nearest neighbor distances have been used in geographic applications for imputation of missing or erroneous values (Ohmann & Gregory, 2002), clustering (Fritz et al., 2013), predictions on unseen data (Hu et al., 2014), and the study of spatial relationships (Clark & Evans, 1954).

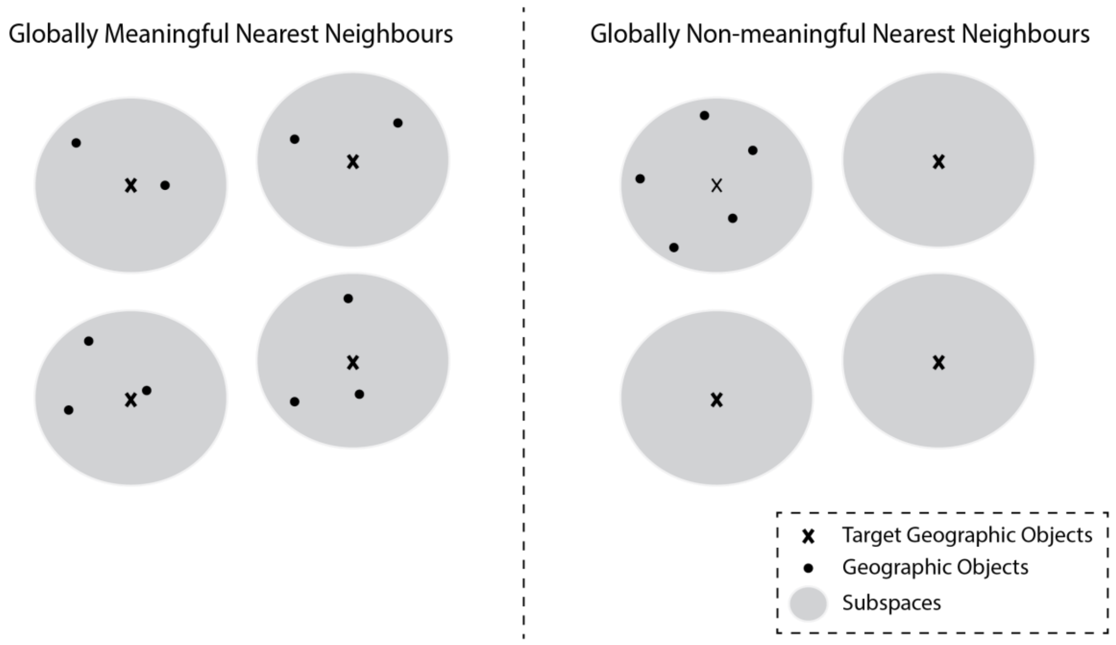
Globally Meaningful Nearest Neighbours
Globally Non-meaningful Nearest Neighbours
Target Geographic Objects
Geographic Objects
Subspaces

Figure 1: Globally Meaningful and Non-Meaningful Nearest Neighbour Distances.

2.2 Neural Networks

Neural networks, a machine learning mechanism modeled after neurons responsible for human learning, are used to learn complex data patterns and perform predictions on unseen data. Neural networks require normalized and meaningful data that can be interpreted by machines to search for possible patterns and structures in data (Nielson, 2015). The naïve concept of a biological neural network involves minor adjustments of the synaptic connections between the neurons such that there is a spike of energy sent through the axon, once a certain threshold is met (Lancashire et al., 2009). An artificial neural network attempts to model the biological neural network such that the learning mechanisms of the neuron can be incorporated using a machine with logical algorithms. An artificial neuron models the neuron through accepting a number of inputs, often normalized, standardized, or scaled, to produce a single aggregate output.

Artificial neural networks are built upon an architecture based on an input layer, an output layer, and multiple hidden layers with a collection of artificial neurons in each layer, where increasing the number of artificial neurons increases the complexity of the model (Maltarollo et al., 2013). The hidden layer is responsible for the synaptic-based learning of the artificial neural network as it aggregates the input values, determines if a threshold is met, and passes values to an output. Figure 2 presents the naïve concept of the artificial neural network modeled after a biological neural network. Diagram A presents the main components of a neuron, Diagram B presents a machine representation of the neuron, Diagram C presents the synaptic connections that make adjustments, and Diagram D presents the synapses with the input/output layers (the hidden layers are in the center with hidden nodes represented by lighter shaded circles). Weights are used to represent synapses, where the artificial neural network makes adjustments to the weights such that it begins to output expected results given appropriate inputs and example outputs. Multiple artificial neurons are connected to form a neural network of complex activation interactions in order to weigh inputs to arrive at an output decision.

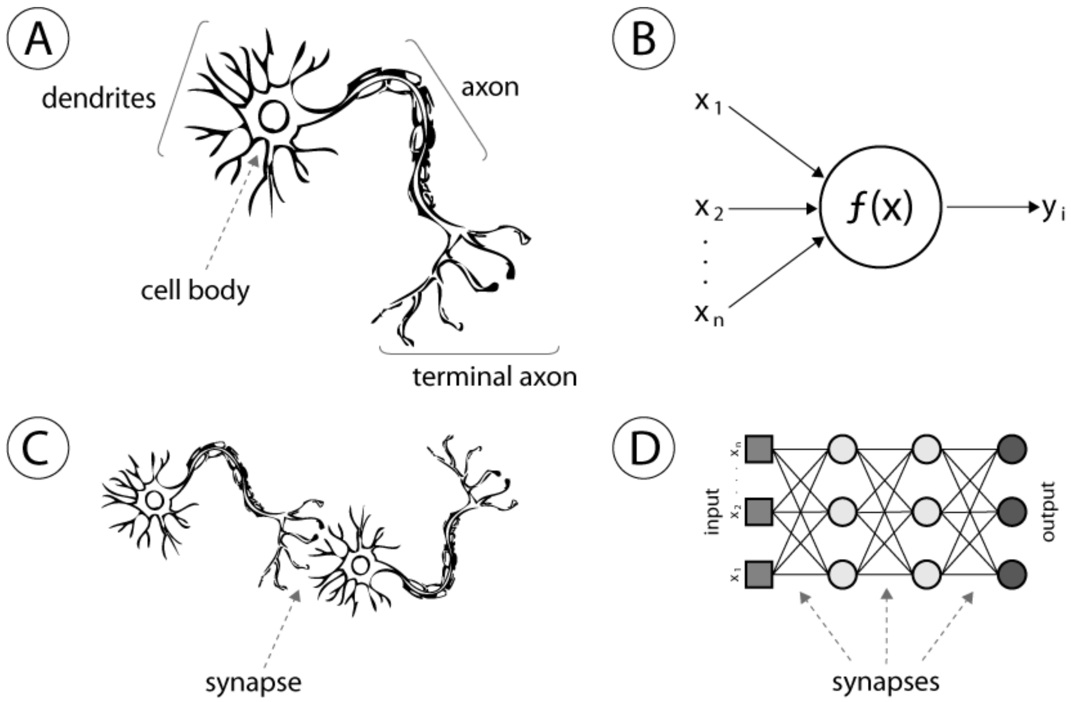

Figure 2: Naïve Concept of Artificial and Biological Neural Networks. Reprinted from *Artificial Neural Networks - Architectures and Applications* (p. 205) by K. Suzuki, 2003: InTech. Copyright 2003 by InTech under the Creative Commons Attribution License. Reprinted with permission.

A neural network may be represented as a collection of activating artificial neurons based on an aggregation of weights that express the importance of each input variable. When the aggregated weights meet a certain threshold, the artificial neuron is activated and the information is fed forward to make a decision or to become input values for another connected artificial neuron. The activation of an artificial neuron with weighted sums is described in Equation 2.1 by Nielson (2015):

$$f(w,x,b) = \begin{cases} 0 \; if \; \mathrm{w} \cdot \mathrm{x} + \mathrm{b} \leq 0 \\ 1 \; if \; \mathrm{w} \cdot \mathrm{x} + \mathrm{b} > 0 \end{cases} \quad \textbf{(2.1)}$$

where w and x is the weight and the input value vectors respectively, and b is the bias which determines the activation sensitivity of the artificial neuron. The activation of an artificial neuron need not depend on a weighted sum and other activation functions can be utilized; for example, a logistic sigmoidal function or a hyperbolic tan function (Hornik et al., 1989).

The connections between artificial neurons, bias adjustments, hidden layers, and activation functions may be adjusted to create neural networks suited for specific applications. For example, a neural network may be customized to learn using a back-propagating procedure that repeatedly adjusts the weights of the network such that a measure of difference between the outputs and the training targets (the actual outputs from the training data) are minimized (Rumelhart et al., 1988). The backpropagation procedure uses an error or a cost function, often a form of the mean squared error between a prediction and its target values such that it is solved by gradient descent in which the combination of weights that minimizes the error metric (the global minimum) becomes the solution (Cross et al., 1995; Rojas, 1996). Artificial neural networks have been utilized in geographic applications to predict forest attributes (Corne et al., 2004), mine land use patterns (Hagenauera & Helbicha, 2011), automate quality assurance of road networks (Jilani et al., 2013), and predict weather station data (Reusch & Alley, 2002).

2.3 Support Vector Machines

A SVM is a mathematical algorithm that seeks to define an optimized separating hyperplane decision function given multi-dimensional example data based on vectors closest to the defined hyperplane (Noble, 2006). The separating hyperplane seeks an optimally defined boundary, between the planes of the support vectors, which separates and classifies clusters of data. Formally, the support planes can be defined in Equation 2.2 (Cristianini & Shawe-Taylor, 2000):

$$\langle w \cdot x \rangle + b = 1 \tag{2.2}$$

where w is a weight vector, b is the bias, and x describes the support vectors (the training examples closest to the hyperplane). A linearly optimized separating hyperplane may be found by determining the largest minimum distance, the margin, between the support planes. The margin M can be defined in Equation 2.3 (Hastie et al., 2009):

$$M = \frac{2}{||w||} \tag{2.3}$$

where $||w||$ is the norm of the weight vector. Thus, the optimized linearly separating hyperplane that provides the largest margin between training example data can be computed using Equation 2.4 (Hastie et al., 2009):

$$\min_{w,b} f(w) = \frac{1}{2}||w||^2 \,, subject\ to\ y_i(\langle w \cdot\ x_i \rangle - b) \geq 1\ for\ every\ i \tag{2.4}$$

where b is the bias, w is the weight vector, and y_i represents each label at index i, and x_i represents the variables belonging to a label at index i. Equation 2.4 is subject to constraining the data points such that they do not fall inside the margin between the support planes. An example of the maximum separating hyperplane for a binary classification problem is seen in Figure 3. The optimally separating hyperplane is the hyperplane that maximizes the distance of the margin given the planes determined by support vectors. Class 1 and class 2 are maximally split in this example by the boundary of the linearly separating hyperplane by solving for an optimization problem in two dimensional space. Assuming that the data are linearly separable, the most basic linear SVM involves computing the support vector planes and the maximum separating hyperplane to determine a decision boundary which optimally separates classified binary input examples.

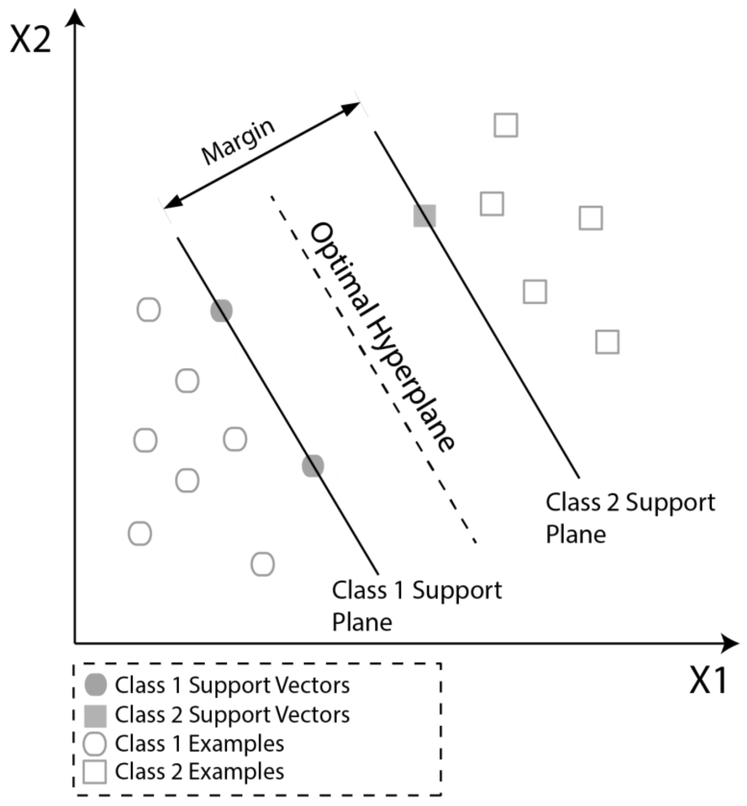

Figure 3: Support Vector Machine Margin Optimization.

The SVM algorithm has a strong theoretical foundation that has been known to produce high accuracy results not limited to binary classes (Cristianini & Shawe-Taylor, 2000). A common one-versus-one scheme can be implemented to handle multiclass data, where a set of binary SVMs are trained for all possible combinations of classes equal to a number defined by Equation 2.5 (Mather & Brandt, 2009):

$$N = \frac{n(n-1)}{2} \tag{2.5}$$

where N is the number of SVMs trained, and n is the number of classes. Another approach is to consider a one-versus-rest scheme in which a SVM is trained for each class, where each SVM fits each class against all other classes (Knerr, Personnaz, & Dreyfus, 1990); only the class being fitted (positive) and all other classes not belonging to that class (negative) are considered for each SVM classifier. In comparison to a one-versus-rest scheme, a one-versus-one scheme would optimize computational cost cases as each SVM classifier would only need to process a training sample size equal to the maximum of training samples between two classes instead of the complete training data. The one-versus-rest scheme also has the possibility to produce unclassified cases leading to lower accuracies as opposed to the one-versus-one scheme (Pal, 2005). Predictions are made using a majority vote method in which the most frequently voted class becomes the predicted outcome. SVM algorithms have been successfully utilized in geographically related applications to classify land cover from remotely sensed imagery (Pal & Mather, 2005), perform natural hazard assessment of flood occurrences (Tehrany et al., 2015), and predict potential disease absence and presence distribution (Guo, Kelly, & Graham, 2005).

2.4 Decision Trees

The decision tree is a non-parametric algorithm that seeks to learn simple rules for defining a pattern in the data by recursively partitioning the training data space until certain criterion is met (Liaw & Wiener, 2002). In the classification problem, a decision tree can be constructed by determining and splitting nodes by a measure of information gain based on impurity, and setting stopping criterion to halt the partitioning of nodes (Quinlan, 1984). A decision tree is built with nodes and connected paths between nodes that represent learned rules from the training data. Each node contains a learned rule associated with a variable from the training data, and the path taken is based on the split proposed from the rule. The structure of a decision tree is composed of a root node, which has zero incoming paths and one or more outgoing paths, an internal node, which has one incoming path and two or more outgoing paths, and a leaf node, which has one incoming path and zero outgoing paths (Tan, Steinbach, & Kumar, 2006). The decision for a class is obtained with the leaf node. An example visualizing the structure of the decision tree is shown in Figure 4. The structure of the decision is composed of nodes as seen by the ellipses, rectangular boxes, and directional paths that connect between nodes as seen by the arrows. Each node is a learned rule about the training data that enables a path to be taken depending on the input data for prediction. The questions demonstrate an example of the decision rules for which mode of transportation to take to reach a destination.

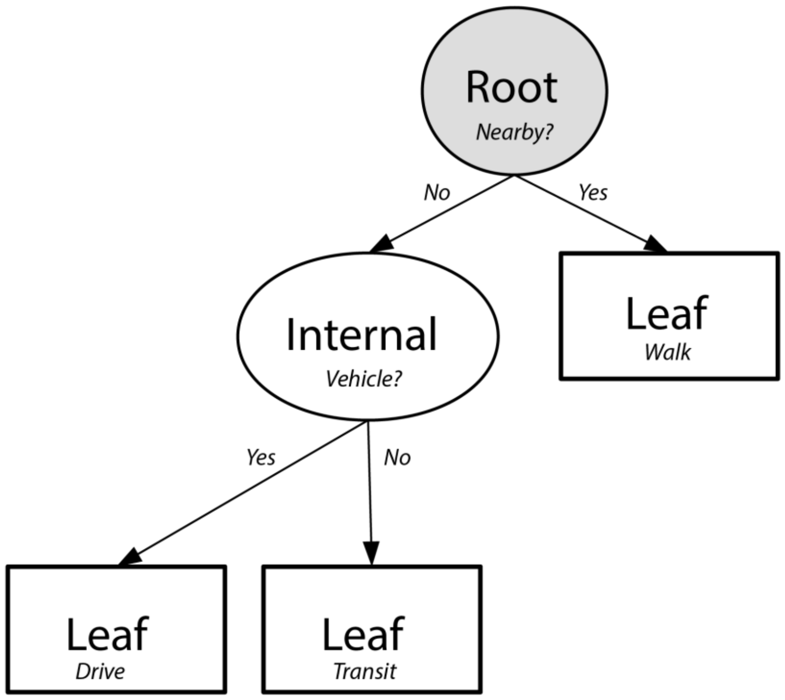

Figure 4: Decision Tree Structure.

The algorithm to construct a decision tree starts with determining a root node with a measure of information gain based on impurity, then splitting from the root node to create either internal or leaf nodes using the same measure until a predefined stopping criterion is met. Some stopping criteria examples are stopping when meeting a training sample threshold for determining nodes, or when all leaf nodes are pure. However, it is not suggested to stop when all leaves are pure as each class would have one leaf node, resulting in an increase in the complexity and computational time, while leading to overfitting of the training data (Rao, 2005). In order to compute information gain, a measure of impurity for a node is required. The common impurity measures are entropy, Gini impurity, or classification error. These common measures of impurity are approximately consistent with each other in terms of the performance of the decision tree, but may vary for individual splits inside the tree (Rokach & Maimon, 2005). Entropy, Gini impurity, and classification error for a node are defined in Equations 2.6, 2.7, and 2.8 respectively, according to Tan et al. (2006):

$$Entropy(t) = -\sum_{i=0}^{c-1} p(i|t)log_2 p(i|t) \tag{2.6}$$

$$Gini(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2 \tag{2.7}$$

$$Classification\ error(t) = 1 - \max_i[p(i|t)] \tag{2.8}$$

where $p(i|t)$ is the fraction of training samples for class i at node t, and c is the number of classes. The impurity measures range between 0 and 1 for entropy, and 0 and 0.5 for Gini and classification error, where the maximum value is obtained when p is equal to 0.5 which describes the classes as being uniform. The information gain based on an impurity measure can then be calculated using Equation 2.9 according to Tan et al. (2006):

$$Infomation\ Gain = I(v_{parent}) - \sum_{j=1}^{k} \frac{N(v_j)}{N(v_{parent})} I(v_j) \tag{2.9}$$

where I is the measure of impurity function of a node, v_{parent} is the parent node, k is the number of variables, j is an index for a variable, N is the total number of training samples for a node, and v_j is the child node from a variable. The maximum information gain given a split from the child node determines if the split will be chosen, which is interpreted as minimizing the weighted average of the impurities of child nodes given that the impurity of the parent is static (Rao, 2005). The information gain is often a good measure for determining the relevance of a variable to the outcome in the training data, but overfitting occurs when the variable has a large number of distinct values (Hastie et al., 2009).

The advantages of the decision tree algorithm include that it is robust as it accepts both continuous and discrete values, may handle outliers and noise, and results can be visualized using familiar graphics such as a flowchart. Decision trees have been used for non-parametric analysis and interpretation of geographic data for Light Detection and Ranging (LIDAR) variable delineation (Crasto et al., 2015), land cover classification (Chasmer et al., 2014), terrain prediction (Abdallah, 2010), and spatial modelling of disease (Stevens & Pfeiffer, 2011). However, the main disadvantage of decision trees is that they often overfit to the training data without methods of adjustment such as pruning or preprocessing the input training data before the training phase (Kotsiantis, 2013).

2.5 Random Forests

Random forests utilize multiple decision trees and randomization to reduce overfitting and provide better generalization of the training data. The random forest algorithm is based on an ensemble method in which a collection of classifiers are trained and a majority vote is done for the trained classifiers to produce predictions on unseen data (Dietterich, 2000). Thus, a random forest can be defined as an ensemble based classifier that is composed of multiple decision trees in which each tree classifier is trained on a random selection of variables and training samples. The reduction in overfitting is based on the idea that the random selection of variables and training samples varies the input training data for each decision tree classifier trained, and that the collective predictions produced by the decision trees would generalize better to the entire training dataset than a single decision tree. As the number of trees increases, the generalization error is proven to decrease based on the Law of Large Numbers, where the aggregate of the decision trees is expected to be closer to the actual outcome as more tree classifiers are trained (Breiman, 2001). Random forests have been used for several geographic applications including complex land cover classification (Ghimire et al., 2012), disaggregation of geodemographic data from multiple data sources (Stevens et al., 2015), and ecological species modelling (Prasad et al., 2006). Random forests also have the ability to compute useful metrics that are not limited to predictions; these are explained below and include a measure of variable importance, measures of proximity for the training data, an out-of-bag (OOB) error estimate, and an interpretive variable contributions measure for each sample.

The variable importance measure is based on the Mean Decrease Impurity (MDI). MDI seeks to evaluate a variable’s importance by computing the sum of the weighted impurity decreases for all nodes of the variable in the forest and averaging it over the sample size of the trees in the forest (Louppe et al., 2013); ranking these values will provide insight into which variables being used in the random forest contribute the most information for the predictions. The MDI for variable importance is a useful method for obtaining insight about the structure of the training data, however, it is biased such that it places preference on variables with a high number of categories, and is unstable when variables are correlated as once one variable is used for a node in a tree, the other is considered uninformative despite both variables providing the same information gain (Strobl et al., 2008; Calle & Urrea, 2010). Variable importance measures are useful for providing insight on the most influential variables of the training data, and for variable selection to eliminate dimensionality in the training samples when there is an abundance.

A measure of proximity or similarity between samples in the training data may also be computed by counting the number of times a pair of training samples reach the same leaf for each decision tree classifier, and normalizing it by the number of tree classifiers in the forest (Louppe, 2014). The proximity values range from 0 to 1, where values close to 1 indicate similarity (samples reach similar leaves overall), and values close to 0 indicate dissimilarity (samples reach different leaves overall). The proximity measure is affected by the depth of each decision tree as shallow trees increase the probability of pairs of samples reaching the same leaves, and the number of decision trees as the increase in decision trees provides more information about the structure between sample pairs (Hastie et al., 2009). Outliers can be detected in the training samples as a sample is considered an outlier if its average proximity value with respect to other training samples is small; the average proximity can be extended for class specific outliers as well (Louppe, 2014). The proximity measures of a random forest can be used for anomaly detection, clustering, and exploratory visualization with methods such as Multi-Dimensional Scaling (MDS) (Kruskal, 1964).

The OOB error estimate seeks to provide a measure to evaluate the performance of the random forest without setting aside a portion of the data into a testing set. At each randomized sampling of training a decision tree, approximately a third of the samples are left out of training, known as the OOB data, and used to calculate a predictive score for that tree (Breiman, 2001). The average of the OOB data is used to produce an error estimate on the generalization of the random forest and is often as accurate as using cross validation given that enough trees are trained in the forest (Liaw & Wiener, 2002). The OOB error estimate can be used for the selection of random forest models to optimize parameters that affect its performance such as the number of trees or the depth of each tree.

The variable contributions from the random forest predictions can be used to interpret the influence of each predictor variable for individual samples. The variable contributions are computed from the aggregation of local variable contribution increments for the random forest (Palczewska et al., 2014). Variable contributions improve the interpretability of the black-box random forest algorithm as it details the influence of each variable for individual training samples, as well as unseen testing samples.

2.6 A Comparison of Neural Networks, SVMs, and Random Forests for VGD

A robust and automated method to assess the structure of VGD, without using expert knowledge, may provide an attractive way to assess and evaluate the reliability and quality of VGD. The emergence of improved hardware has enabled machine learning to be scalable and feasible for large quantities of data, including geospatially referenced data. Machine learning algorithms may adapt to the dynamic and highly temporal nature of VGD such that they may learn and make adjustments to new incoming data (Kodratoff & Michalski, 2014). Many machine learning algorithms also have the ability to perform functions to achieve multiple goals, such as performing both predictions and clustering of the data (Rao, 2005). Machine learning algorithms also have great potential to assess non-linear patterns and structures in VGD as there is an absence of strict standard structure, and the nature of VGI is dynamic and unpredictable.

Random forests provide robustness, ease-of-use, interpretability, efficiency, and scalability for assessing the structure of VGD compared to neural networks and SVMs. Neural networks require the proper adjustment of hyper parameters, activation functions, and hidden layers to be effectively used for a particular application or case study, and often require an extensive amount of input data preprocessing before being fed into a constructed network

(Lancashire et al., 2009). Neural networks are also known to overfit and are not guaranteed to produce a solution with a global optimum (Nielson, 2015). Overfitting occurs when a model fits the training data perfectly but fails to perform adequately on predictions for unseen test data, if they consist of high dimensions. However, the amount of customizability of neural networks may be an attractive variable as the connections, artificial neuron behaviours, and direction of data flow often can be modified to produce results with high accuracies depending on the application. SVMs often provide high accuracies with good generalization, possess a strong theoretical foundation, are more robust to high dimensional training data, and produce a solution with a guaranteed global optimum, but are not scalable to excessively large quantities of data (Rao, 2005). SVMs also do not require the many parameter optimizations of the neural network; for example, it is often sufficient to empirically adjust the parameters of c, a penalty parameter, and gamma, a kernel coefficient or similarity function such as linear or polynomial, to optimize the performance of the SVM (Cristianini & Shawe-Taylor, 2000). Both neural networks and SVMs are black-box models such that they do not provide informative and interpretive details on how they arrived at a solution, and often increase in complexity as parameters are adjusted or more classifiers are used (Lantz, 2013); for example, increasing the number of hidden layers in a neural network would drastically increase the complexity of the model, and increasing the number of SVM classifiers in the case of multiclass data would also increase the complexity of the model. Random forests, in comparison to neural networks and SVMs, provide more interpretability in that they can produce measures of variable importance, variable contributions, and proximities that make the algorithm transparent and understandable (Palczewskaet al., 2014). These interpretable measures provide insight into the variables for the entirety of the data as well as individual instances, and enable the ability to perform outlier detection. Random forests are also scalable compared to neural networks and SVMs, as each tree can be trained in parallel without dependency on order (Breiman, 2001); thus, the addition of processing cores will improve the training speed of the random forest drastically. Random forests also do not require that the input data be preprocessed to be robust to outliers and missing data, and can handle high dimensional data due to the randomization and aggregation procedures (Louppe, 2014); therefore, random forests may obtain good results even if parameters are not optimized, and when prior or expert knowledge about the data is not available. The variables (input variables) to a random forest can be continuous, discrete, or categorical and do not require standardization, normalization, or scaling that may produce bias in results (Breiman, 2001). Random forests, compared to neural networks and SVMs, provide a method that is robust, informative, and scalable to the dynamic nature and growing volume of VGD.

CHAPTER 3: METHODOLOGY

3.1 Data

The implementation and testing of the random forest classifier for VGD required a geospatial dataset that changed continuously, potentially contained errors, and obtainable in a standard file format. OSM data provided different freely accessible sources to obtaining usable and up-to-date datasets from several areas. The data were downloaded from Mapzen's Metro Extracts server which freely distributes city-sized portions of weekly updated OSM data (Mapzen, 2016). An OSM dataset in GEOJSON format organized by IMPOSM specifications, was downloaded for the city of Toronto, Ontario, in geographic coordinates (World Geodetic System 1984). A GEOJSON file contains one or more spatial objects described by geometry types and geometric properties in a key/value structure (Butler et al., 2008). IMPOSM is an importer for OSM data organized based on tags, which provides files based on the tag categories: points (amenities, places, and transport points), lines (minor roads, main roads, motorways, railways, and waterways), and polygons (administrative, buildings, land uses, aero ways, and water areas) (Omniscale, 2016). A few of the GEOJSON files were selected based on IMPOSM key categories: places, amenities, transport areas, aero ways, transport points, and roads (simplified to 50 meter tolerances). This research uses Toronto OSM data as a demonstration dataset, but the methods used can be applied to any vector-based geospatial dataset with class or discrete values and accessible geometry data.

The selected files represented data with a total of 70,898 geographic objects in the City of Toronto detailed in Table 1. Point, polygon and line geometry types existed in the data. The majority of the geographic objects reside in the transport points and roads datasets, while the least amount of geographic objects resided in the transport areas dataset. The tag values are all the available classes to be predicted.

Table 1: OpenStreetMap Data for City of Toronto, Ontario from Mapzen.

Key Category Dataset	Tag Values Available	Geometry Type	Count
Amenities	fire_station, fuel, hospital, library, police, school, townhall, university	Point	1507
Places	city, county, hamlet, locality, neighbourhood, suburb, town, village	Point	760
Transport Areas	aerodrome, apron, helipad, platform, station, terminal	Polygon	72
Aero Ways	runway, taxiway	Line	438
Transport Points	aerodrome, bus_stop, crossing, gate, halt, helipad, level_crossing, motorway_junction, station, subway_entrance, terminal, tram_stop, turning_circle	Point	21,309
Roads	disused, monorail, motorway, motorway_link, preserved, primary, primary_link, rail, secondary, secondary_link, subway, tertiary, tertiary_link, tram, trunk, trunk_link	Line	46,812

The distribution of the 49 classes (tag values) in the Toronto OSM data is visualized in Figure 5. Class in this figure signifies the classified tag value (of a key/value pair) given to a geographic object in the data. For example, given a key/value pair of “road=secondary”, the class would be “secondary”. Class imbalance exists in the data and leads to misinterpreted accuracies as the most frequent classes are prioritized.

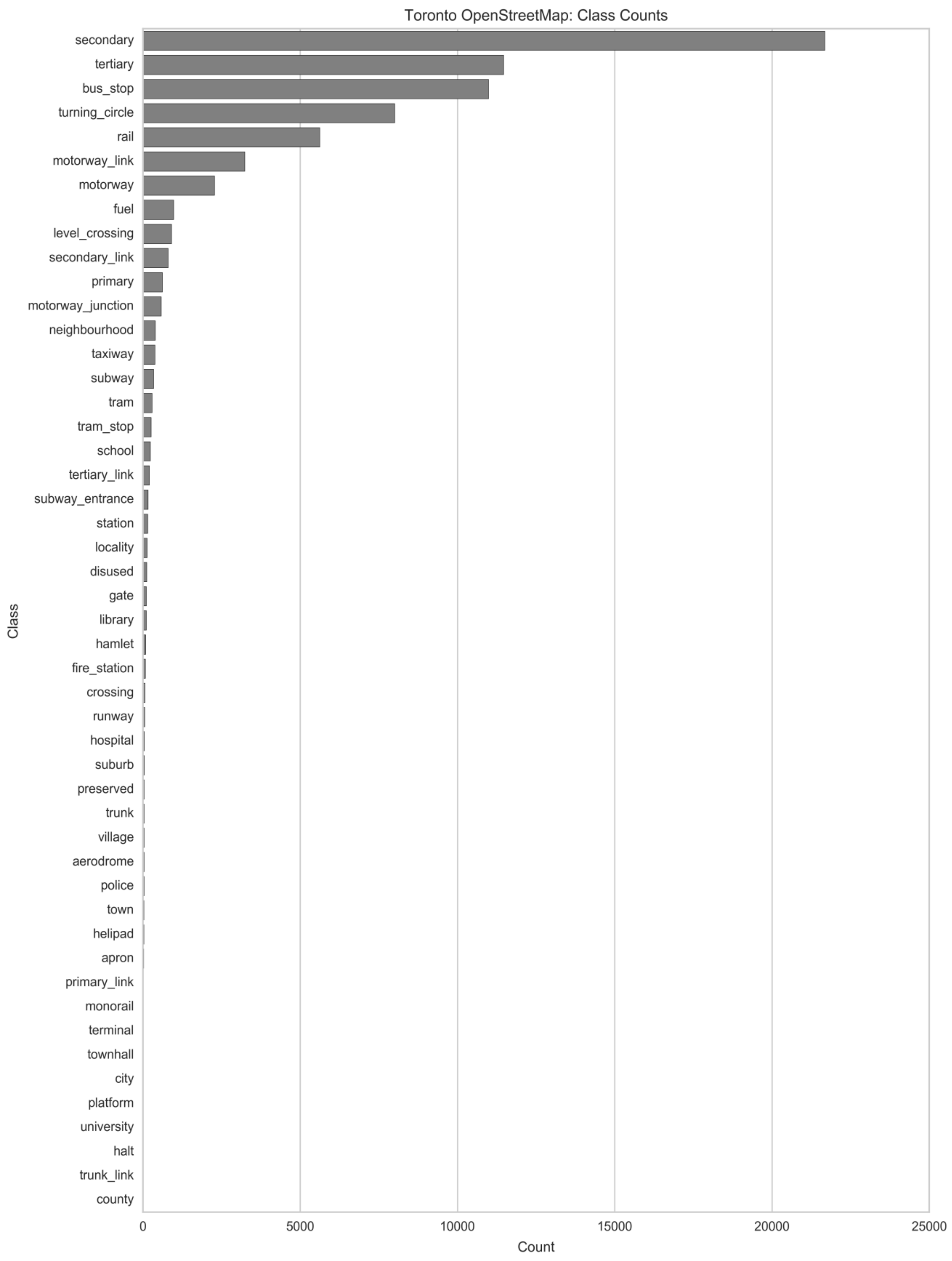

Figure 5: Class Distribution of Toronto OpenStreetMap Data.

3.2 Development Libraries and Modules

The methods for this research included using the programming language Python 3 (Python Software Foundation, 2016) to automate the procedure such that it may be reproduced and customized for any geospatially referenced dataset with classified data. The open source Python *libraries* and modules used to automate each procedure were pandas, geopandas, scipy, numpy, Rtree, seaborn, scikit-learn, treeinterpreter, pickle, and joblib. Pandas is a library used for high-performance manipulation of data structures in a variety of formats (McKinney, 2010). Geopandas is an extension of pandas to enable manipulation of geospatial data (GeoPandas developers, 2014). Scipy is a collection of open source software for scientific computing which are often used for statistical analysis and numerical algorithms (Jones, et al., 2001). Numpy is a library for efficient numerical computation scalable to large quantities of data (van der Walt et al., 2011). Rtree is a spatial indexing library for fast and efficient spatial queries using rectangular bounding boxes (The Toblerity Project, 2014). Seaborn is a statistical visualization library with an interface and attractive graphics that is based on the matplotlib plotting library in Python (Waskom, 2015). Scikit-Learn is a library for data mining and data analysis with a variety of machine learning algorithms available (Pedregosa et al., 2011). Treeinterpreter is a package that enables the variable contributions to be computed for individual prediction instances of a random forest (Saabas, 2015). Pickle is a module for serializing and de-serializing Python object structures which enables the objects to be stored to and loaded from the hard disk. Joblib is a set of tools for pipelining with useful code for parallelized computing and fast disk-caching (Varoquaux, 2009). These open source Python libraries were used in combination to create automation code for reproducible processing and versatility for any vector-based geospatial data in a standard format. Continuum Analytics Anaconda Python was used for easier installation of dependent libraries and modules not available in the standard Python 3.5 installation (Continuum Analytics, 2015). The purposes of each of the utilized libraries for this research are detailed in Table 2. The anaconda installed column states if the library or module is installed by using Continuum Analytics Anaconda Python. The base python column states if the library or module is available from a standard Python 3.5 installation.

Table 2: Purpose of Python Development Libraries and Modules.

Library/Module	Purpose	Anaconda Installed	Base Python
Pandas	Non-spatial data manipulation	Yes	No
Geopandas	Spatial data manipulation	No	No
Scipy	Statistical calculations and formulas	Yes	No
Numpy	Numerical calculations	Yes	No
Rtree	Optimized line and polygon queries	No	No
Seaborn	Plotting	No	No
Scikit-learn	Random forest models and machine learning metrics	Yes	No
treeinterpreter	Variable contribution calculations	No	No
Pickle	Saving of processed results	No	Yes
Joblib	Parallelized computing	No	No

3.3 Extraction of Geospatial Semantic Variables

The geometric characteristics extracted using the following approach included area, length, vertices, and representative coordinates. The data were first read and projected into planar coordinates (North American Datum 1983 Universal Transverse Mercator Zone 17 North) suitable for Toronto, Ontario, using geopandas before the extraction of geospatial semantic variables to allow representative geometric calculations. The area, length, and representative coordinates are available through the geopandas library, while the vertices were calculated by iterating each geographic object and counting the number of points that the geographic object contains. The representative coordinates are extracted as columns of x and y coordinates that represent the centres of points, lines, and polygons with the added constraint that they fall within the geometries. The representative points provide information on the spatial distribution of the geographic objects, which may allow the random forest to expose potential positional patterns of the geographic objects. The vertices provide information on the complexity of each geographic object, which may be useful for differentiating between classes that contain line or polygon type geographic objects.

The Class-specific Nearest Neighbour Distance (CNND) of each class was extracted using an Euclidean metric for each geographic object. The CNND for each geographic object involved spatially optimized queries for polygons and lines with an R-tree from the Python library rtree. An R-tree utilizes minimum bounding rectangles for each geographic object for efficient spatial querying of nearest neighbours. R-trees are based on the assumption that another geographic object in space is not a nearest neighbour if its minimum bounding rectangle does not intersect the minimum bounding rectangle of the geographic object under consideration (Guttman, 1984). Thus, the number of required euclidean distance measurements are reduced to the number of bounding boxes that intersect the minimum bounding rectangle. The first class-specific nearest neighbour was used for each geographic object to provide information of spatial relationships between classes to the random forest algorithm. The main focus was to enable the random forest to determine if the geospatial proximity of a geographic object to neighbouring geographic objects was a discriminative phenomenon for improving the predictability of a class. The first neighbour for the nearest distances was used as geospatial semantic variables. Due to the often large quantities of VGD, it is a reasonable assumption that CNNDs will be approximately normally distributed if a global spatial pattern in relation to each class of geographic objects exists (Beyer et al., 1998). An example of the first CNND, considering three classes and a geographic object is seen in Figure 6. Only the first nearest neighbour is considered for each class object in geographic space. These distances are represented as cells in a table where the each column represents the distances to each of the first nearest classes.

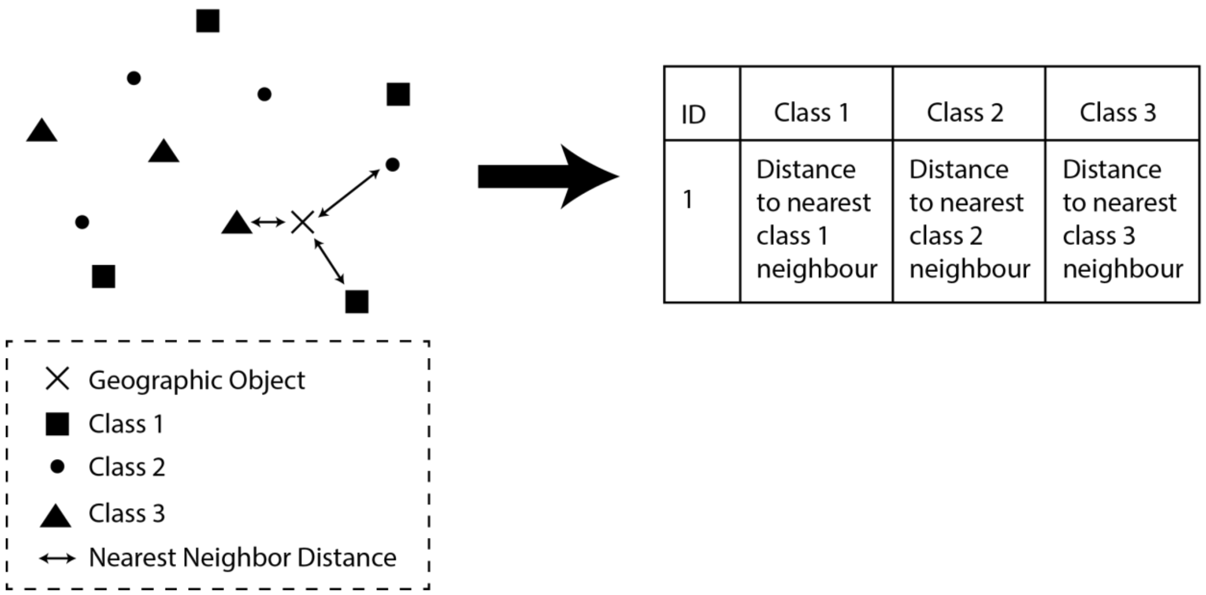
ID
Class 1
Class 2
Class 3
1
Distance to nearest class 1 neighbour
Distance to nearest class 2 neighbour
Distance to nearest class 3 neighbour
Geographic Object
Class 1
Class 2
Class 3
Nearest Neighbor Distance

Figure 6: First Class-specific Nearest Neighbour Distance Example for Three Classes.

The structure of the extracted geospatial semantic variables is shown in Table 3. The data structure of the extracted geospatial semantic variables is formatted as a standard table. Each column represents an extracted geospatial semantic variable, and the rows represent geographic objects. The CNNDs have one column per unique class in the data. These variables are used in the random forest models to predict the classes of the Toronto OSM data.

Table 3: Structure of Extracted Geospatial Semantic Variables.

Area	Length	Vertices	X	Y	Class-1 to Class-n Distance..
Area of the geographic object	Length of the geographic object	Number of vertices for the geographic object	Representative x axis coordinate	Representative y axis coordinate	Class-specific nearest neighbour distance..

3.4 Multicollinearity Reduction of Variables

Multicollinearity reduction was performed to provide improved stability in variable importance measures. Recall that the variable importance is obtained by computing the sum of the weighted impurity decreases for all nodes of the variable in the forest and averaging it over the sample size of the trees in the forest (Louppe et al., 2013). The variable importance values of a random forest are unstable and misrepresentative when one or more variables are highly correlated (Dietterich, 2000). In order to provide improved understanding of variable importance metrics without varying values among variables, multicollinearity was reduced by removing variables in order from the left-most column to the right-most column.

The algorithm for the multicollinearity reduction procedure is seen in Figure 7. The removal procedure was done by selecting first the left-most column and removing all other variables that are highly correlated. Variables were considered highly correlated with each other if the Pearson's correlation coefficient was less than or equal to -0.7 or greater than or equal to 0.7. The remainder of the variables that were not removed variable were then subject to the same removal procedure until all remaining variables were not highly correlated with each other. Area, length, vertices, and representative coordinates x and y were examined first, as these are the primary characteristics of geographic objects. The CNNDs were ordered based on the frequency of each class from highest frequencies first to lowest frequencies last. The assumption for the ordering of classes is that a class object is less prone to erroneous classification by OSM contributors if the frequency of the class is high, which means that the class is more common and likely to be agreed upon. It is also likely that classes with high frequencies tend to be encountered more often, which may provide utility when manual or interactive examination of the data is done. If this process was applied to data other than from OSM, a different approach to ordering may be required.

```
Ordered Multicollinearity Reduction Algorithm

Set $Remain$ to be all available variables in a desired order
For each variable $Var1$ in $Remain$:
    Set $Other$ to be all $Remain$ variables that are not $Var1$
    For all other variables $Var2$ in $Other$:
        Compute Pearson correlation $Corr$ between $Var1$ and $Var2$
        if $Corr$ < = -0.7 or $Corr$ > = 0.7:
            Remove $Var2$ from $Remain$
Output $Remain$ contains the variables that are not highly correlated with each other
```

Figure 7: Ordered Multicollinearity Reduction Algorithm.

3.5 Empirical Optimization of Random Forest Parameters

The random forest's performance and ability to provide adequate interpretive results is affected by the number of trees trained and the number of variables per tree for the forest. The number of trees to experiment on were based on a study by Oshiro et al. (2012) that suggested that a number of trees between 64 and 128 was found to be a reasonable range for trade-offs between computation time and accuracy. The set of tree parameters to experiment on were 64, 96, and 128 in order to observe the performance stability of the random forest by evaluating low and high ranges. The number of variables per tree were set to scikit-learn's default value of the square root of the total number of variables available, which is advised to be suitable for classification data (Pedregosa et al., 2011). The OOB error, one minus the OOB score, was used to evaluate which random forest model would perform better as it enabled the use of the entire dataset instead of subsets. It is expected that the highest number of trees would be the best performing random forest parameter. An entropy based measure of impurity was used, and balanced class weights were employed to account for class imbalances in the input data. The *balanced class weights* parameter adjusts weights for each class to be proportional to the class counts before allowing each tree classifier to train on the randomized sample. The balanced class weights can be defined according to Equation 3.1:

$$W_c = \frac{N}{n(c_f)} \tag{3.1}$$

where w is the weight for a class c, N is number of training samples, n is the number of classes, and c_f is frequency of class c. The rest of the random forest parameters were set to scikit-learn defaults. The parameters used for the random forest are shown in Table 4. These parameters were used in the random forest experiments to optimize and select the best performing random forest model. Only the number of tree classifiers changes in the experiments, which total to four different random forest models. Out of the four random forest models, one is selected based on its OOB error. The construction of each random forest model involved parallel computing that built decision trees in parallel based on the number of available computing cores.

Table 4: Parameter Settings for Random Forests.

Parameter	Setting
Number of Tree Classifiers	64, 96, 128
Measure of Impurity	Entropy
# of Variables for Best Split	$\sqrt{\# \, of \, features}$
Maximum Depth of Trees	Until all nodes are pure or until minimum samples for nodes are reached
Minimum Samples of Internal Node Split	2
Minimum Samples in Leaves	1
Minimum Weighted Fraction of Input Samples	0
Use of Bootstrap Samples for Building Trees	True
OOB Score	True
Number of Jobs	Number of available cores
Re-use Previous Solution	False
Class Weights	Balanced

3.6 Evaluation of Random Forest Performance

The evaluation of the performance of the random forest model after selecting an empirically optimal number of trees involved empirical cross-validation experiments and F1 scores. Stratified cross-validation was performed where the Toronto OSM data were divided into a number of folds (partitions) evenly based on set values, where each even fold had approximately the same class distributions as the full dataset. Two-fold, five-fold, and ten-fold cross-validation was used to evaluate the random forest, where a fold was used as testing data and the rest of the folds were used as training data until all possible folds had been used for testing. A random forest model was created and trained with the optimal number of trees for each cross-validation test. This enabled different variations of the Toronto OSM data to be tested for and evaluated tests of performance stability in relation to missing data. For example, in the two-fold cross-validation test, approximately 50% of the data was used for testing, which meant that approximately 50% of the data used for training is assumed to be missing. The percentages of training and testing data for each selected cross-validation test are seen in Table 5. The cross-validation folds determine the splits in the dataset. One fold is used for the testing data, and the rest is used for training until all possible testing and training combinations are evaluated. This allows the random forest to be trained on subsets of data to observe its performance in the case of missing and varying data.

Table 5: Percentages of Training and Testing Data for Cross-validation Tests.

Cross-validation Folds	Percentage of Data Used for Training	Percentage of Data Used for Testing
2	50%	50%
5	80%	20%
10	90%	10%

The F1 score was utilized to evaluate and quantify the quality of the random forest predictions for each class. The F1 score is the harmonic mean of precision and recall, where a value of 1 indicates the perfect ideal model and a value of 0 indicates a completely incorrect model. Precision is the proportion of positively predicted cases that are actually positive, and recall is the proportion of negatively predicted cases that are actually negative. A model with good performance will attempt to maximize the precision and recall simultaneously, resulting in a F1 score of close to 1. The F1 score is defined in Equation 3.2 according to Powers (2011):

$$F1 = 2\frac{pr}{p+r} \; where \; p = \frac{tp}{tp+fp} \; and \; r = \frac{tp}{tp+fn} \tag{3.2}$$

where $F1$ is the F1 score, p is the precision, r is the recall, tp is number of true positives, fp is the number of false positives, and fn is the number of false negatives. The F1 score was produced for each class and the class weighted average of the F1 scores was used to determine a measure of performance for each cross validation test.

3.7 Assessment of Data Patterns for Parameter Optimized Model

An assessment of the structure of the Toronto OSM data was done using probabilities, proximities, variable importance, and variable contributions using the selected model with the empirically optimal number of trees trained on the entire Toronto OSM dataset.

Probabilities were extracted from all random forest predictions on the entire training data of the Toronto OSM dataset to measure how certain the random forest was for each predicted class by averaging the within-class probabilities. The probability for each prediction is the mean of the decision tree classifiers probabilities in the random forest, and the probability for a class in a single decision tree classifier is the proportion of samples of the same class in a leaf (Bostrom, 2007). The weighted average probability for a predicted class in the random forest is expressed in Equation 3.3:

$$\overline{P_c} = \frac{1}{n}\sum_{i=1}^{n} P_i \tag{3.3}$$

where $\overline{P_c}$ is the weighted average probability for a predicted class c, n is the number of classes, and P_i is the probability for a predicted class instance i. Probabilities range from 0 to 1, where 0 indicates that the random forest model is completely uncertain for an individual prediction, and 1 indicates that the random forest model is completely certain for an individual prediction.

Proximities were calculated between two sample instances in the random forest using Equation 3.4 according to Louppe (2014):

$$proximity(x_1, x_2) = \frac{1}{M}\sum_{m=1}^{M}\sum\nolimits_{t \in T_m} 1(X_1, X_2 \in X_t) \tag{3.4}$$

where x_1 and x_2 are sample instances to compare, M is the number of decision tree models in the forest, t is a leaf in the forest, and T_m is the set of leaf nodes in a m decision tree. The proximity can be interpreted as how often the two sample instances both end up in the same leaf node for each tree in the forest (Breiman, 2003). A proximity matrix for samples within each class was constructed to calculate outlier measures. The proximity matrix was used to compute outlier measures using Equation 3.5 as described by Breiman & Cutler (2004):

$$outlier(n_c) = \frac{N}{\sum_{k_c}^{K}[proximity(n_c, k_c)]^2} \tag{3.5}$$

where n_c is a sample instance of class c, k_c is all other sample instances of class c, K is the total number of k_c, and N is the total number of n samples. The outlier measure is then normalized by subtracting every outlier value for n instances of each class c by the median of all outlier measures inside the same class c, and dividing by the absolute deviation from the median. Large values of the outlier measure indicate that the sample instance is an outlier, where values of greater than 10 are suspected of being an outlier (Breiman, 2003).

Variable importance values were obtained for each variable in the random forest to understand the influence of variables with regards to all the classes. The measure of variable importance is the Mean Decrease Impurity (MDI), based on a measure of impurity such as entropy, Gini, or classification error (see Section 2.4) as defined in Equation 3.6 according to Louppe et al. (2013):

$$Imp(X_m) = \frac{1}{N_T}\sum_{T}\sum\nolimits_{t \in T: v(s_t) = X_m} p(t)\Delta i(s_t, t) \tag{3.6}$$

where X_m is a variable, T is a decision tree structure, N_T is the sample size for a decision tree T, t is a node in the decision tree, s_t is the split for node t, $v(s_t)$ is the variable used to split s_t, and $p(t)\Delta i(s_t, t)$ is the weighted impurity decreases for nodes t of variable X_m given an impurity measure i. The values range from low to high, with higher values indicating that the variable is more important and lower values indicating that the variable is less important; ranking these values provided insight into the influence of variables for all the classes combined.

Variable contributions were calculated for individual prediction instances of the training data to obtain insight into the variable influence of each class. A local variable contribution increment, representing the change in probability for a class between a child and a parent node for a variable, is defined in Equation 3.7 according to Palczewska et al. (2014):

$$LI_f^c = \begin{cases} Y_{mean}^c - Y_{mean}^p & if\ split\ of\ p\ is\ for\ f \\ 0 & otherwise \end{cases} \tag{3.7}$$

where LI is the local variable contribution increment, f is a variable, c is the child node, p is the parent node, Y_{mean}^c is the fraction of training samples in a child node, and Y_{mean}^p is the fraction of training samples in a parent node. The sum of the local variable contribution increments for each decision tree classifier produces the variable specific contribution. The variable contribution of a training sample can be defined in Equation 3.8 according to Palczewska et al. (2014):

$$FC_i^f = \frac{1}{T}\sum_{t=1}^{T} FC_{i,t}^f \tag{3.8}$$

where FC_i^f is the variable contribution of a training sample, $FC_{i,t}^f$ is the sum of local variable specific contribution increments, f is a variable, T is the total number of trees in the forest, t is a tree in the forest, and i is a training sample. Similar to the variable importance, the variable contributions can be ranked from highest to lowest to provide insight into a variable's contributions to a prediction, where high values indicate high contribution and low values indicate low contribution. The median of the variable contributions for a predicted class was used to obtain information on the influence of variables for each class.

CHAPTER 4: RESULTS

4.1 Automated Python Script

A Python script was produced that automated the methodology. The automation flowchart is shown in Figure 8. The script was versatile in that it will accept any number of spatially enabled data files readable by the geopandas library, allowing for a mixture of geometry types (lines, points, or polygons). The source of the data files can be either local, online, or in a compressed zip format. The required user input configuration for the script to function included the definition of a workspace on the local hard drive, and the definition of one or more experiments containing at minimum the following: a data source and the column name containing the classes to be trained and tested on by the random forest. The column name of the prediction classes must be unique for each data file, and exist in all of the data files from the data source. A workspace may have multiple experiment definitions. The default configuration is set to be consistent to the methodology described in this research. However, the user may choose to experiment with customized settings. The Python script can be executed through the command line by calling the *main.py* file in the current directory and passing the path to a configuration file as the first argument: *python main.py path\\to\\config.txt*.

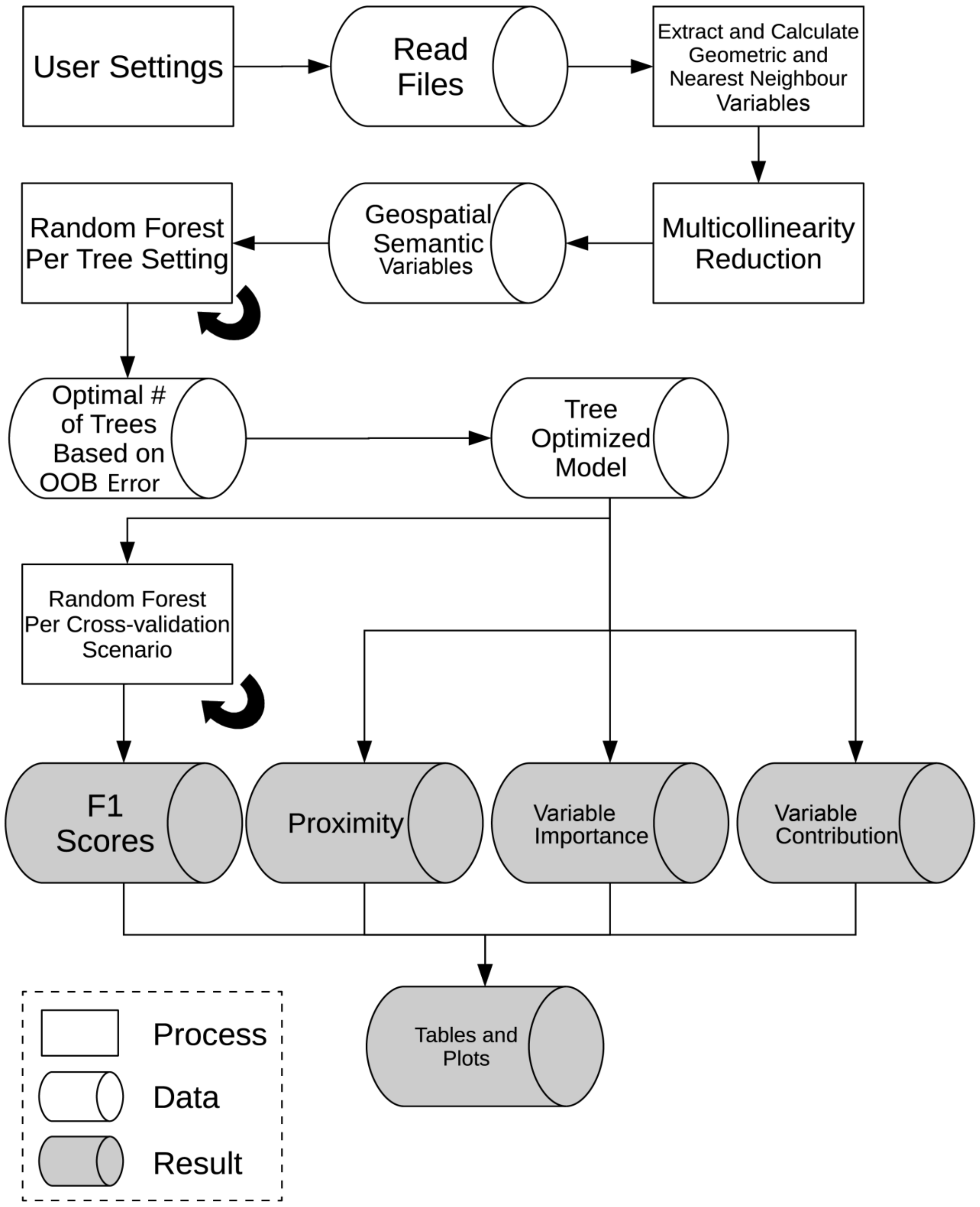
User Settings
Read Files
Extract and Calculate Geometric and Nearest Neighbour Variables
Multicollinearity Reduction
Geospatial Semantic Variables
Random Forest Per Tree Setting
Optimal # of Trees Based on OOB Error
Tree Optimized Model
Random Forest Per Cross-validation Scenario
F1 Scores
Proximity
Variable Importance
Variable Contribution
Tables and Plots
Process
Data
Result

Figure 8: Automated Python Script Flowchart.

The complete range of options for user configuration of the script is seen in Table 6. The user configuration settings for the python script are organized by subcategories: Settings, Plot, Analysis, Experiment, and Forest. Each setting and experiment may have analysis, plot, and forest options. The nested configurations defined inside the *settings* category will be considered first before being set to default values seen in the table.

Table 6: Automated Python Script Options.

Configuration	Default	Description
		Settings
workspace	No Default	Local workspace path e.g. 'path/to/ws'
save_tables	True	Whether to generate tables or not
save_plots	True	Whether to generate plots or not
cores	-1	Multiprocessing cores; -1 to use all available cores
		Plot
plot_style	'whitegrid'	Seaborn style: 'darkgrid', 'dark', 'white', 'ticks'
plot_color	'gray'	Plot color e.g. 'red', 'black', 'blue', '#0000FF'
plot_dpi	300	Quality of produced plots
plot_ext	'.png'	Format of produced plots e.g. '.pdf', '.jpeg'
		Analysis
cross_validation_tests	[2, 5, 10]	The cross validation folds of each test to perform for performance evaluation
high_correlations	[-0.7, 0.7]	Negative and positive ranges respectively to determine if variables are highly correlated; A pair of variables are considered highly correlated if it is less than the negative or greater than the positive range defined
outlier_value	10	If outlier measure > outlier_value value, a geographic object will be considered an outlier.
persist	True	Whether to save progress to the local disk to avoid reprocessing of results in subsequent executions
		Forest*
n_estimators	[10, 64, 96, 128]	Number of decision trees to use
criterion	['entropy']	Measure of impurity: 'entropy', 'gini'
class_weight	['balanced']	Class weighting system to use: 'balanced_subsample', None
		Experiments
title	''	Title of the experiment e.g. 'Toronto OSM'
src	No Default	Data path e.g. 'path/to/ds.shp', 'https://serv.ds.zip'
filter	[]	Use files that have these file patterns only eg. ['.shp', 'water', 'roads']
id	[]	Identifier columns to include in tables e.g. ['id']
keep_columns	[]	Column names to include for training
epsg	'4326'	EPSG to project the data to
units	'units'	Unit measurement of spatial reference system
target	No Default	Column name with classification data e.g. 'type'

* The forest options are the same as scikit-learn's *RandomForestClassifier* (http://scikit-learn.org) parameters. The user is not limited to the ones in the table, but options have to be enclosed in square brackets like so: *[param]*.

An example of user defined configuration for the Toronto OSM data used in this paper is seen in Figure 9. This can be stored in a text-like file, where the double square brackets *[[]]* defines a nested category. The analysis and forest categories can be nested in the *settings* category, or in each experiment. An experiment can be provided with any naming alias compatible with file naming standards. In this example, *toronto_osm* is the experiment alias used to identify our Toronto OSM processing experiment for the random forest. An example of the minimal required configuration file is seen in Figure 10.

```
[settings]
workspace = 'G:\\user\\Desktop\\workspace'
save_tables = True
save_plots = True
cores = -1
[[plot]]
plot_style = 'whitegrid'
plot_dpi = 600
plot_ext = '.png'
[[analysis]]
cross_validation_tests = [2, 5, 10]
high_correlations = [-0.7, 0.7]
outlier_value = 10
persist = True
[[forest]]
n_estimators = [10, 64, 96, 128]
criterion = ['entropy']
class_weight = ['balanced']

[toronto_osm]
title = 'Toronto OpenStreetMap'
src = 'https://s3.amazonaws.com/metro-extracts.mapzen.com/toronto_canada.imposm-geojson.zip'
filter = ['places', 'amenities', 'transport_areas', 'aeroways', 'transport_points', 'roads_gen1']
id = ['osm_id']
keep_columns = []
epsg = '26917'
units = 'meters'
target = 'type'
```

Figure 9: Example of Configuration for Toronto OSM Data.

```
[settings]
workspace = 'path/to/local/folder'

[an_experiment]
src = 'path/to/data/or/link'
```

Figure 10: Example of a Minimal Configuration.

The result after completed execution of the automated Python script is an organized directory structure with processed files, plots, and tables inside the specified workspace as seen in Figure 11. Persistence in the figure refers to the saving of progress such that the script does not have to re-process previously processed data again in subsequent runs. Pickled data refers to processed Python readable files saved on the hard disk that are used for persistence; these are stored in the memory folder. The experiment folders are determined from user settings for each project definition. The specific plots and tables generated are detailed in Table 7. The generated plots and tables from executing the python script are detailed in this table. Each plot and table is saved as a file with the experiment naming alias at the beginning of the file name followed by the trailing name. Each table will be saved as a Comma Separated Values (CSV) file.

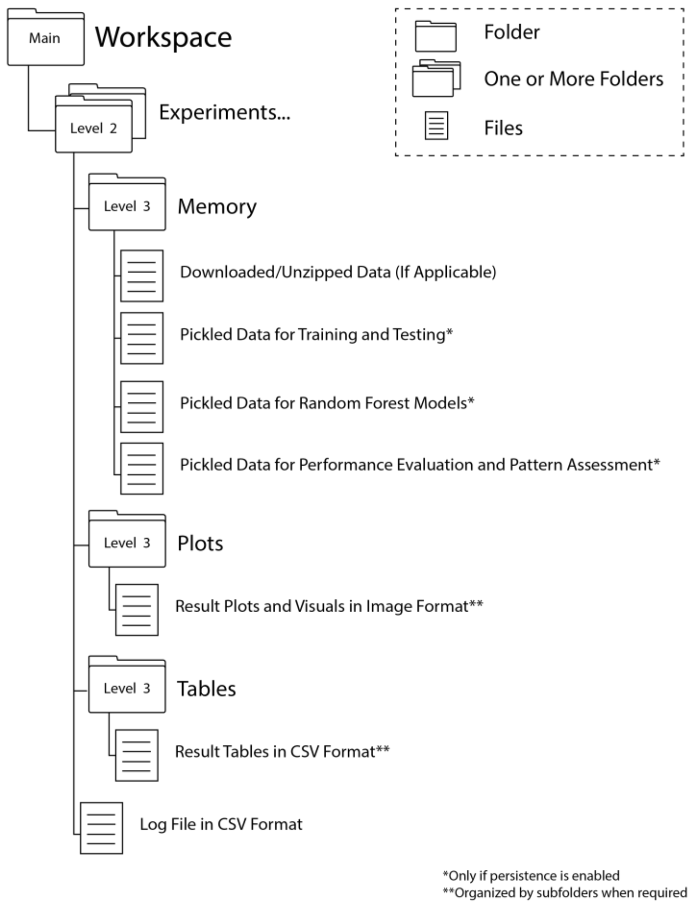
Main
Workspace
Folder
One or More Folders
Files
Level 2
Experiments...
Level 3
Memory
Downloaded/Unzipped Data (If Applicable)
Pickled Data for Training and Testing*
Pickled Data for Random Forest Models*
Pickled Data for Performance Evaluation and Pattern Assessment*
Level 3
Plots
Result Plots and Visuals in Image Format**
Level 3
Tables
Result Tables in CSV Format**
Log File in CSV Format
*Only if persistence is enabled
**Organized by subfolders when required

Figure 11: Structure of Results from Automated Python Script.

Table 7: Generated Plots and Tables of Automated Python Script.

Trailing Name of File	Quantity	Description
	Plots Folder	
f1scores_line	1	Line plot of F1 Scores for each cross validation test
feat_importances_bar	1	Bar plot of variable importances for optimized forest
mean_prob_bar	1	Bar plot of mean probabilities of each class for optimized forest
multicorr_reduction_bar	1	Bar plot of variables removed for multicollinearity reduction
outlier_classes_box	1	Box plot of outlier measures for each class
	Variables Folder Under Plots Folder	
class_hist	1	Histogram of class distributions
near_box	1	Box plot of first CNND
area_hist	1	Histogram of area distributions for all geographic objects
len_hist	1	Histogram of length distributions for all geographic objects
vtx_hist	1	Histogram of vertex distributions for all geographic objects
	Outliers Folder Under Plots Folder	
*class*_outlierclass_contrib_bar	# of outlier classes	Bar plots of variable contributions for each outlier class
	XY Folder Under Plots Folder	
*class*_repxy_joint	# of classes	Maps of representative coordinates for each class
	Tables Folder	
f1_scores	1	Table of F1 scores for each cross validation test
feat_importances	1	Table of variable importance for optimized forest
feats	1	Table of variables before multicollinearity reduction
multicorr_reduction	1	Table of multicollinearity reduction details per variable
neardist_feats	1	Table of class-specific nearest neighbour distance variables before multicollinearity reduction
outlier_measures	1	Table of outlier measures for all geographic objects
outlierclass_contrib	1	Table of all variable contributions for each outlier geographic object
param_optimize	1	Table of parameter optimization details for the optimized forest
prob	1	Table of probabilities of the optimized forest for all the geographic objects
rf_dataset	1	Table of variables used to train each random forest after multicollinearity reduction

4.2 Geospatial Semantic Variables

The distributions of the processed area, lengths, and vertices for the Toronto OSM data are shown in Figure 12, Figure 13, and Figure 14 respectively. The Toronto OSM data consisted of mostly small sized, short, and non-complex geographic objects as most of the geographic objects in the data are points. The variables used to train the random forest were not normally distributed and had a large number of outliers per variable. The area, length, and vertices had large portions of values in the lower range. The dynamic and irregular nature of the variables was expected and reflected upon the nature of VGD.

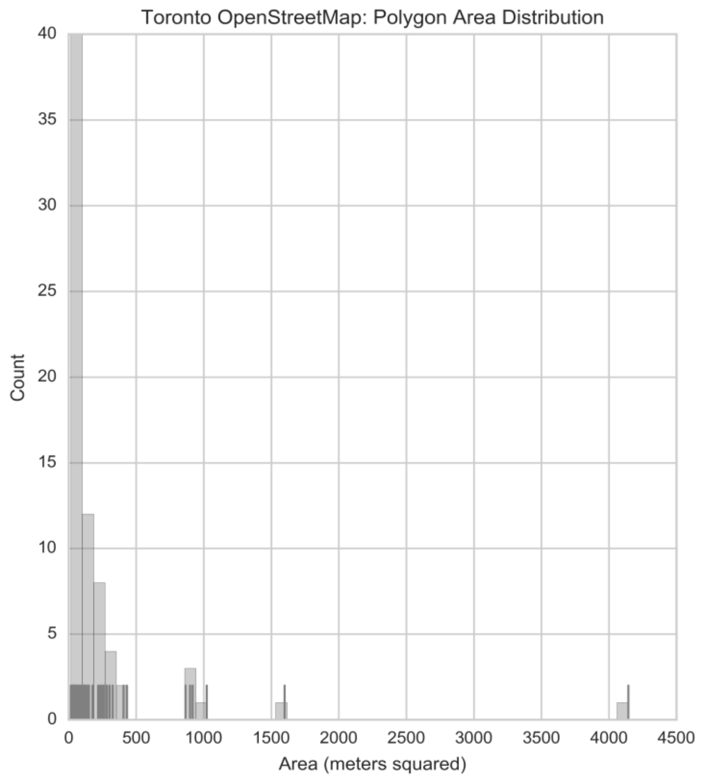

Figure 12: Distribution of Area Variable.

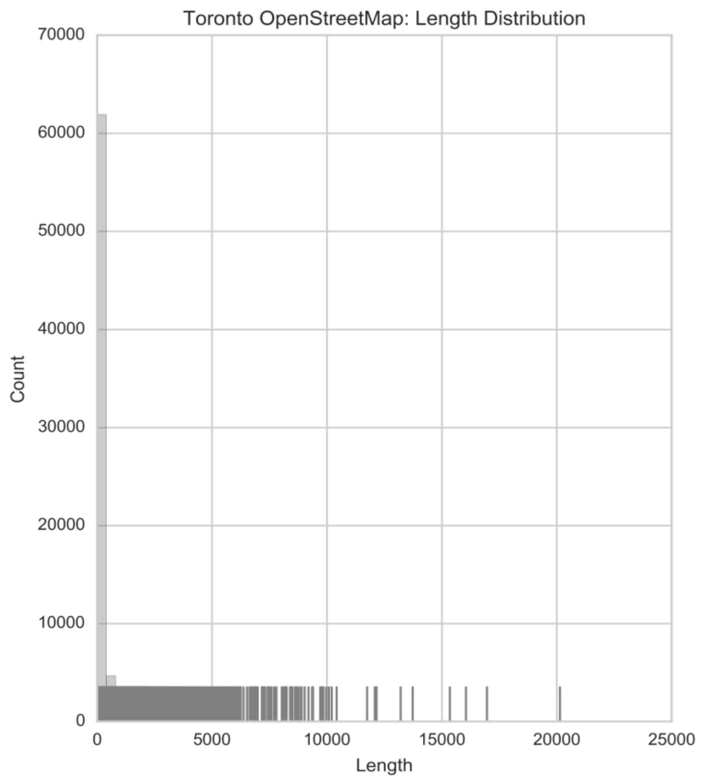

Figure 13: Distribution of Length Variable.

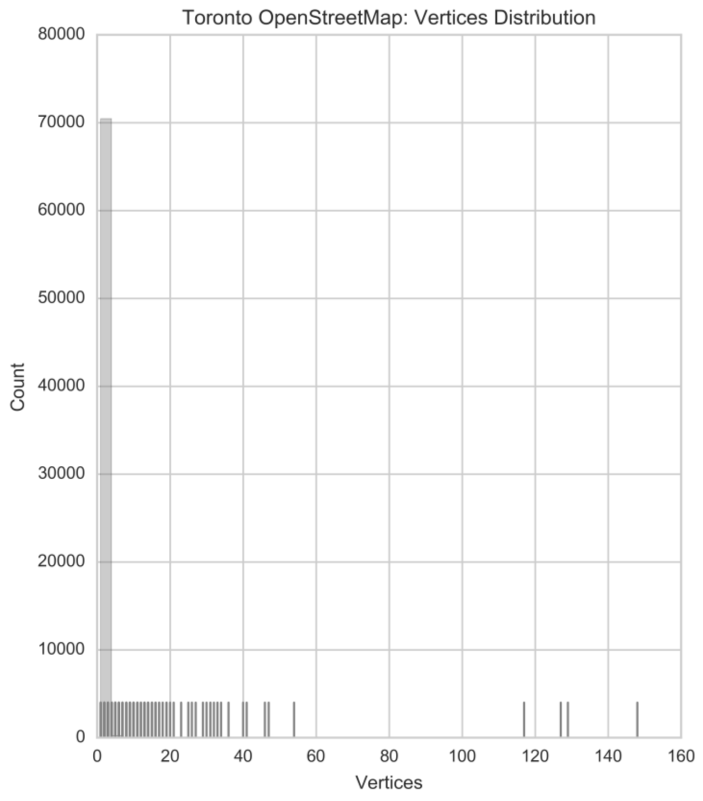

Figure 14: Distribution of Vertices Variable.

A sample map of the representative coordinates are available in Figure 15. The distribution of the x and y coordinates are also shown as histograms near the map. The representative coordinate variables were extracted to enable the random forest to detect an informative spatial pattern if they exist. For example, if most of the representative points for a class were near the downtown area, then the coordinates x and y are informative as it improves the predictability of the class according to location. The representative coordinates had differing densities throughout the study area, but most of the densities were located near or within downtown Toronto, Ontario.

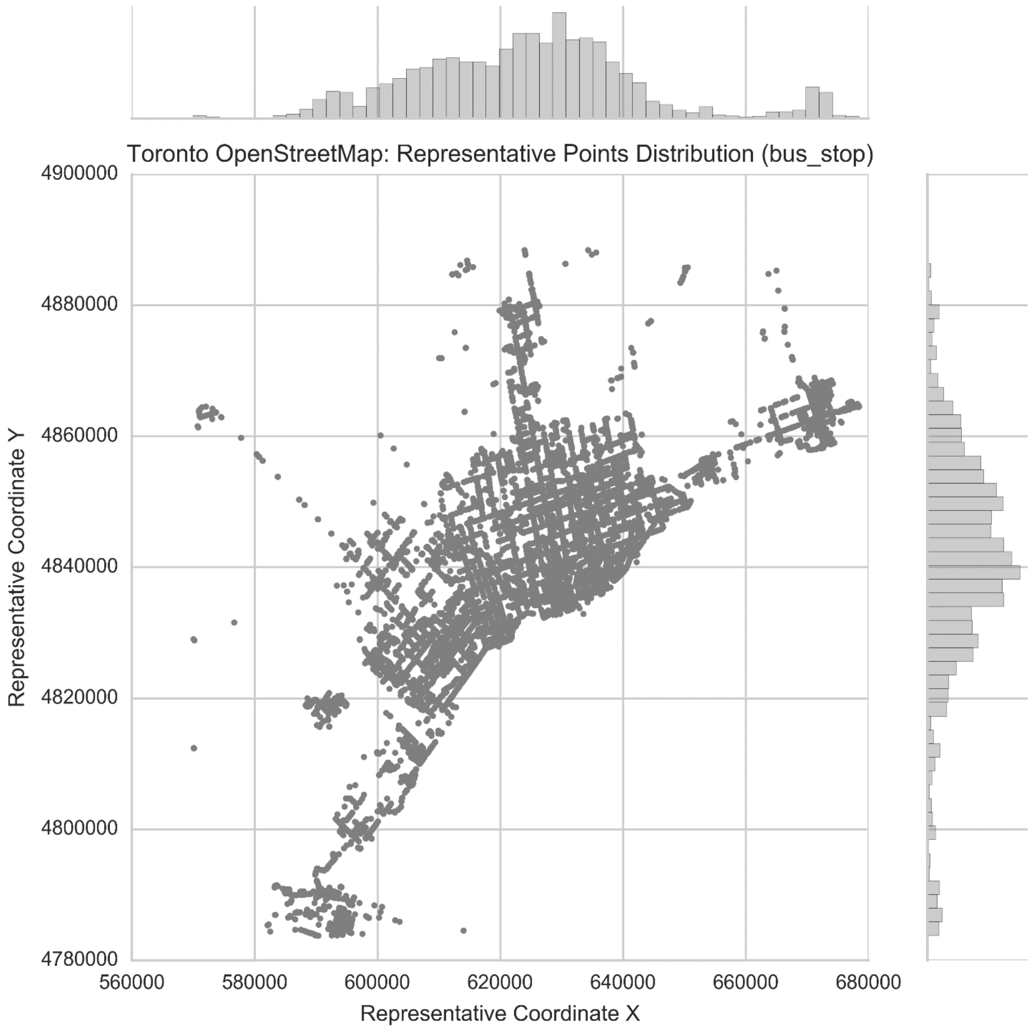

Figure 15: Sample Map of Representative Points.

The CNND variables are detailed as box plots in Figure 16. Most of the CNNDs were irregular distributed and contained many outliers. Most CNNDs were approximately below 40,000 meters with the exception of CNNDs for classes of trunk, trunk_link, and county.

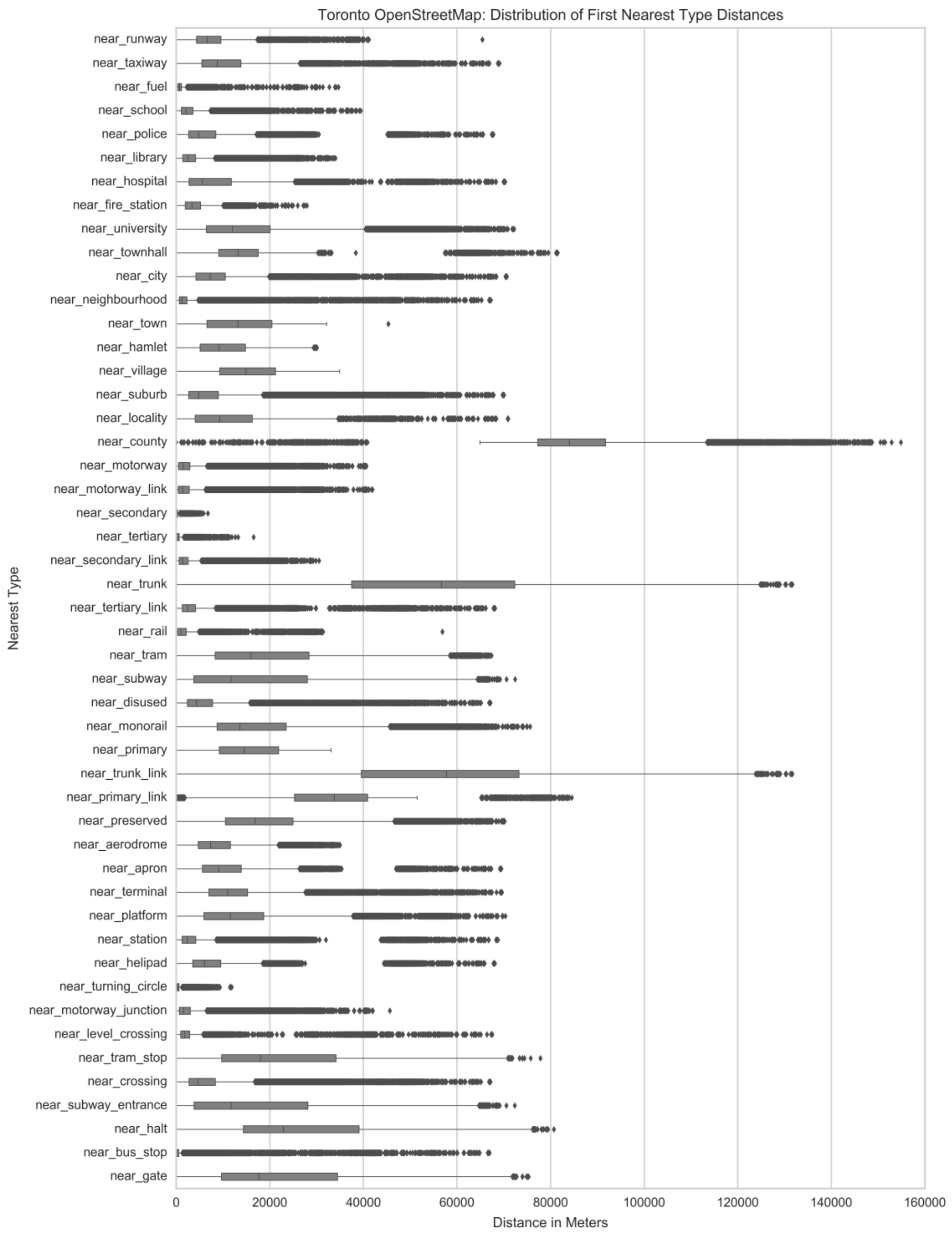

Figure 16: Distribution of Class-specific Nearest Neighbour Distance Variables.

An example of the CNNDs for sample amenities in the Toronto OSM data is seen in Figure 17. The police station is seen as relatively close to libraries compared to fuel stations and schools when considering the nearest neighbour. Fuel station, library, police station, and school are the values portion of the *amenity* key in reference to the key/value structured OSM tags. These are the classes when referring to the CNNDs.

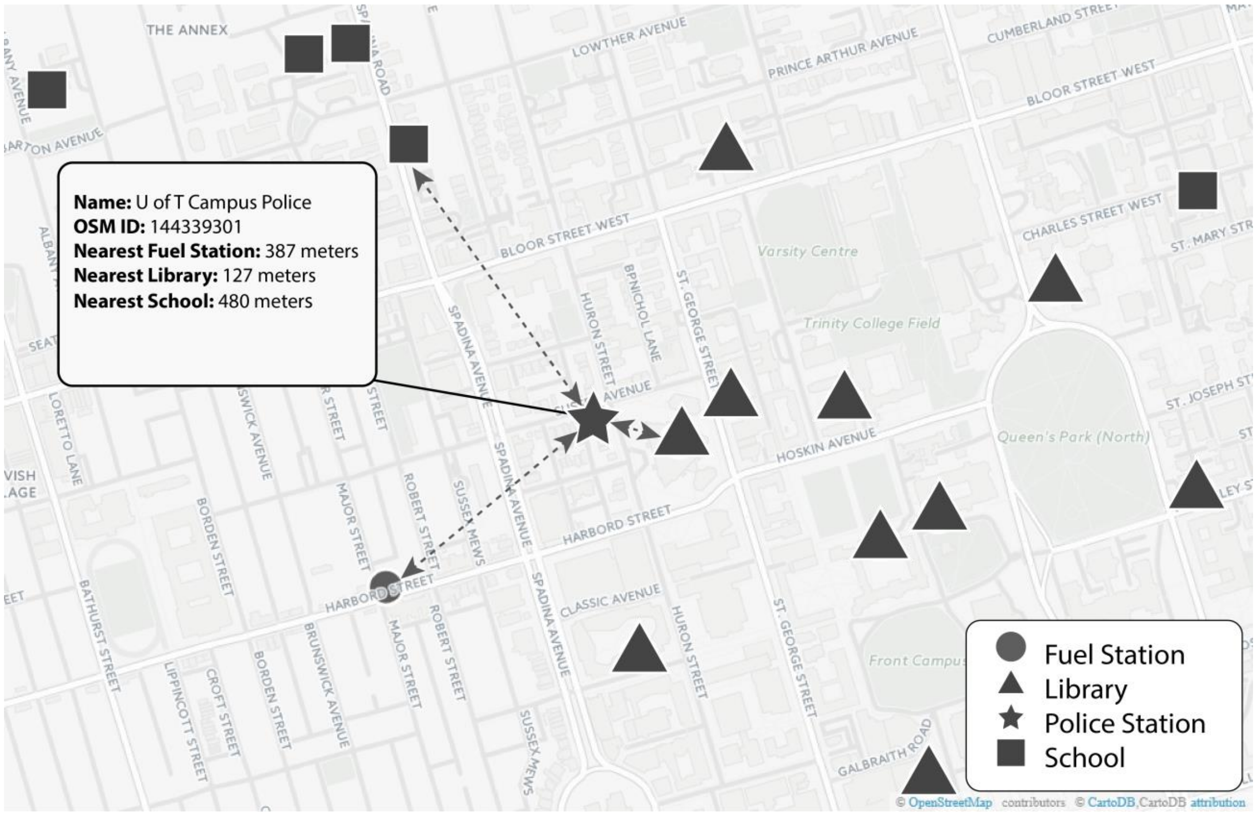
Name: U of T Campus Police
OSM ID: 144339301
Nearest Fuel Station: 387 meters
Nearest Library: 127 meters
Nearest School: 480 meters
Fuel Station
Library
Police Station
School
© OpenStreetMap contributors © CartoDB, CartoDB attribution

Figure 17: Example of Class-specific Nearest Neighbour Distance Variables for Amenities.

4.3 Multicollinearity Reduction

Several variables were removed for multicollinearity reduction to improve the stability of the Toronto OSM variable importance results. It is important to first perform multicollinearity reduction as a measure of impurity at each split of the random forest determines informative variables for the best split at each node. When one of two highly correlated variables are chosen for a split, the other variable becomes uninformative even when both variables should be interpreted to have approximately equal information gain (Strobl et al., 2008). Thus, the randomness of the algorithm will distort variable importance measures between the pair of variables leading to misinterpretation. The multicollinearity reduction of each removed variable for each variable kept are shown in Table 8. The kept variables are shown in order relative to the reduction procedure. For each variable kept, one or more variables were removed if they had a pairwise correlation value below -0.7 or above 0.7. The number of variables removed is shown in Figure 18. Ordered multicollinearity reduction was performed to reduce the unstable variable importance measures, where the geometric variables (representative coordinates, area, length, and vertices) were considered first for pairwise variable removal, followed by the CNND variables ordered by class frequencies. After performing the multicollinearity reduction in a recursive manner, the resulting variables had pairwise correlations between -0.7 and 0.7. The variables near_bus_stop, near_subway, and near_disused had the highest number of pairwise correlations and would produce unstable variable importance results when included with their highly correlated variables.

Table 8: Kept and Removed Variables from Multicollinearity Reduction.

Kept variable	Removed variable (Pairwise Correlation < -0.7 or > 0.7)
repx	near_trunk, near_trunk_link
repy	near_trunk, near_trunk_link
length	
area	
vertices	
near_secondary	
near_tertiary	
near_bus_stop	near_level_crossing, near_neighbourhood, near_tertiary_link, near_station, near_city
near_turning_circle	
near_rail	near_level_crossing, near_station
near_motorway_link	near_motorway, near_motorway_junction
near_fuel	
near_secondary_link	
near_primary	near_town, near_primary_link, near_halt
near_taxiway	near_apron, near_terminal
near_subway	near_tram, near_tram_stop, near_subway_entrance, near_gate, near_town, near_monorail, near_halt
near_school	
near_locality	near_hamlet, near_town
near_disused	near_neighbourhood, near_station, near_crossing, near_suburb, near_university
near_library	
near_fire_station	
near_runway	
near_hospital	
near_preserved	near_university
near_village	near_tram_stop, near_hamlet
near_aerodrome	
near_police	near_station
near_helipad	
near_townhall	
near_platform	near_monorail
near_county	near_trunk, near_trunk_link

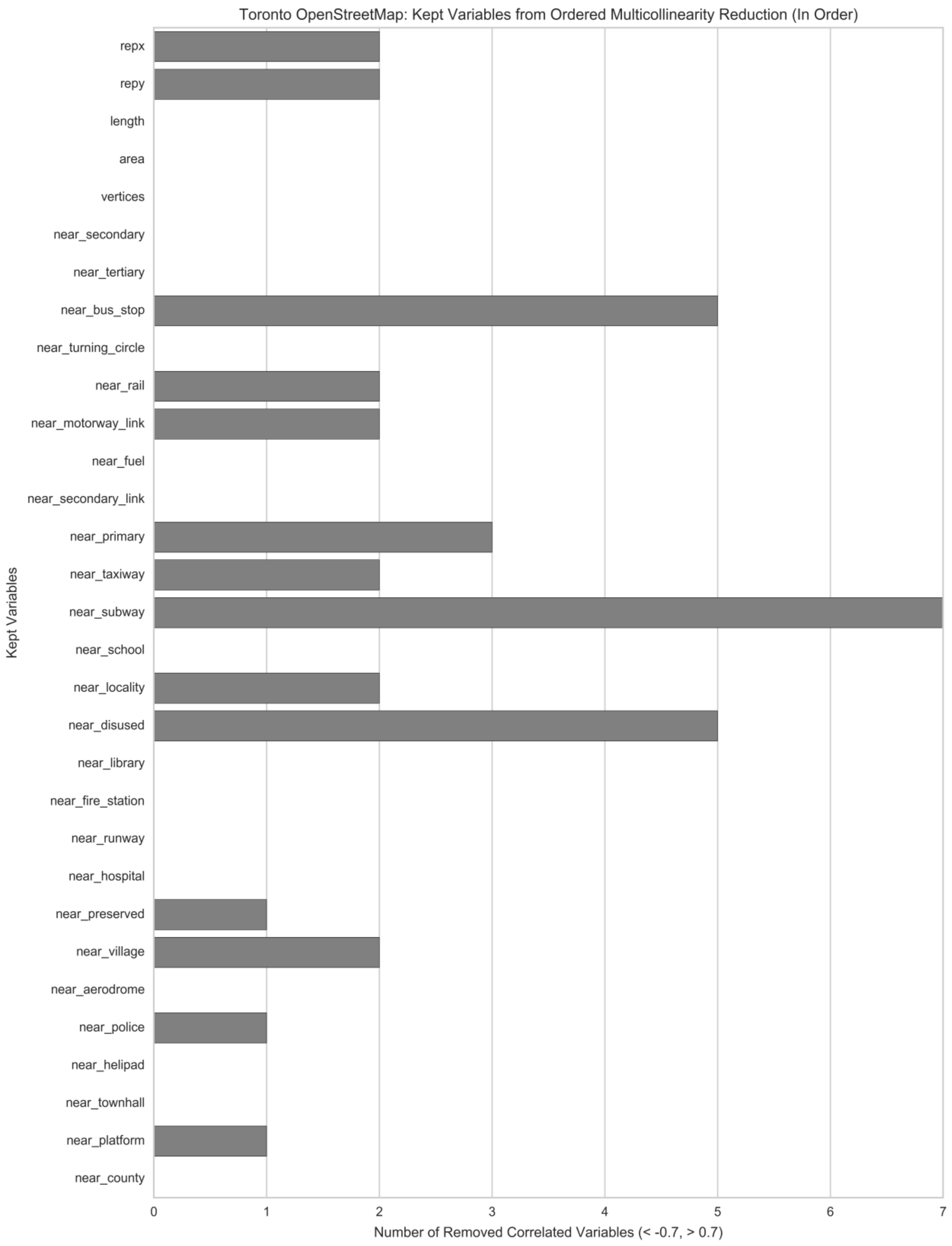

Figure 18: Kept Variables from Ordered Multicollinearity Reduction.

4.4 Parameter Optimization

The parameter optimization of the random forest algorithm was done empirically to obtain an adequate balance between performance and computational cost for the Toronto OSM data. A random forest is constructed from a number of decision trees, and the aggregate of these trees provides a prediction that tends to avoid overfitting. Increasing the number of decision trees in the random forest provides improvements in predictive performance at the cost of computational resources. The determination of the best number of trees was to train random forests with a number of tree parameters (64, 96, and 128) advised by Oshiro et al. (2012) to obtain the lowest OOB error. The parameter optimization for the number of decision trees for the random forest models is shown in Table 9. The number of trees parameter was experimented on to search for the lowest OOB error. The ideal model contains a low OOB error and a high fit. The table shows the change in OOB error with the change in the number of decision trees used for a random forest. The random forest had relatively higher reductions in the OOB error from 0.166 to 0.162 when using 128 decision trees versus 96 decision trees respectively. The OOB error had minor reductions in the OOB error from 0.167 to 0.166 when using 64 decision trees versus 96 decision trees respectively. Each decision tree's computational resource usage is mostly dependent on the number of rows in the data, and number of variables (Pedregosa et al., 2011). The random forest's approximate computational resource usage is the sum of all decision tree resource usages. Thus, increasing the number of decision tree classifiers will significantly raise the computational resource usage. The large volume of Toronto OSM data used, 70,535 geographic objects, and the corresponding large data volume made it necessary to empirically optimize parameters such that adequately low OOB errors are obtained, but at reasonable training times. The optimized number of trees parameter selected for the Toronto OSM data was

128 as it provided the lowest OOB error and had reasonable training times in the minutes scale. For example, it took approximately 10 minutes to test all decision tree parameters including the extraction of geospatial semantic variables.

Table 9: Parameter Optimization Results for Random Forests.

Number of Decision Trees	OOB Error	Fit
64	0.167	0.999
96	0.166	0.999
128	0.162	0.999

4.5 Cross-validation Performance

The random forest algorithm provided adequate predictive performance for the Toronto OSM data based on cross-validation tests. The cross-validation test performances based on mean class weighted F1 are shown in Figure 19. The figure shows an automated line plot of the change in mean class weighted F1 scores for each cross validation test defined by the folds. The cross validation tests differ by the different folds defined to evaluate the performance and stability of the parameter optimized random forest model to different variations of subset data. The cross-validation tests selected were 2-fold, 5-fold, and 10-fold to evaluate the random forest's class weighted F1 score stability in the possibility of large portions of missing and varying data. Each random forest trained in a cross-validation fashion used the optimized 128 trees parameter. The mean of the mean class weighted F1 scores for all folds was approximately 0.71, which revealed that the random forest provided adequate predictive performance even under class imbalance and outlier expectations. It also revealed that the random forest performance was fairly stable as the F1 scores did not fluctuate drastically for each cross-validation test. The random forest had the ability to remain robust and stable in its predictive performance for the missing, irregular, and varying assumptions of the Toronto OSM data.

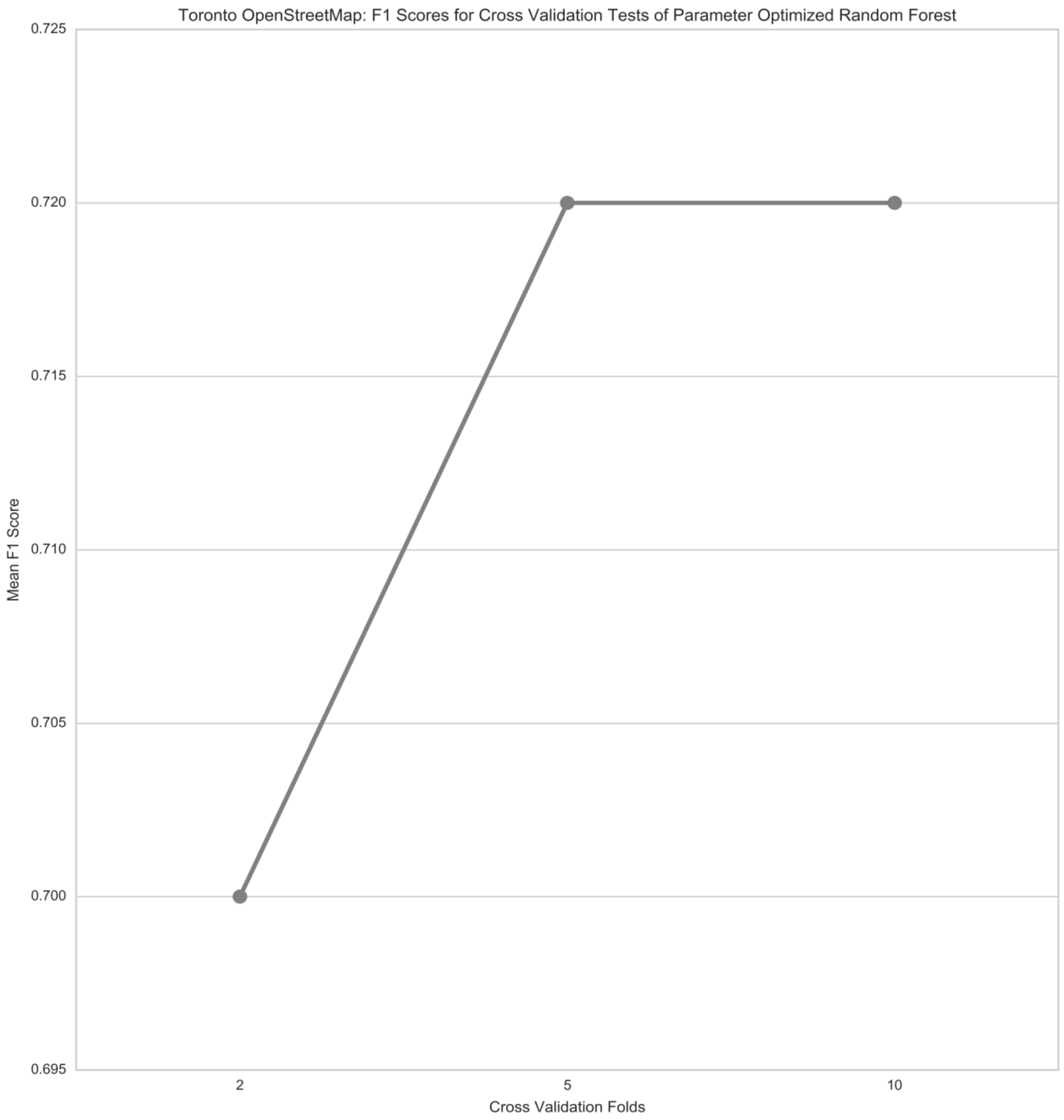

Figure 19: Mean Class Weighted F1 Scores per Cross-validation Test.

4.6 Prediction Probabilities

The data patterns were assessed based on probability, variable importance, outlier measures, and variable contributions of outlier classes. The mean probability of each class, for the parameter optimized random forest model is shown as an automated bar plot in Figure 20. The mean probabilities for each class are sorted from highest to lowest (top to bottom). Probabilities for each class also provided informative data on the predictability of classes in the Toronto OSM data. All classes had greater than 0.6 mean probability, which provided a measure of certainty in the predictions of the random forest with 128 decision trees trained on the entire Toronto OSM dataset. Classes that had lower probabilities had patterns that were less predictable or less uniform when the randomized decision trees split the data for training in the random forest. Classes with higher probabilities were more predictable and had patterns that were more uniform or more frequent as there was an agreement among decision trees in the forest even when the data was randomly subsampled. Notable classes that had low probabilities were station, library, university, police, neighbourhood¸ fire_station, village, suburb, helipad, town, townhall, city, and county. Notable classes with high probabilities were gate, taxiway, and rail.

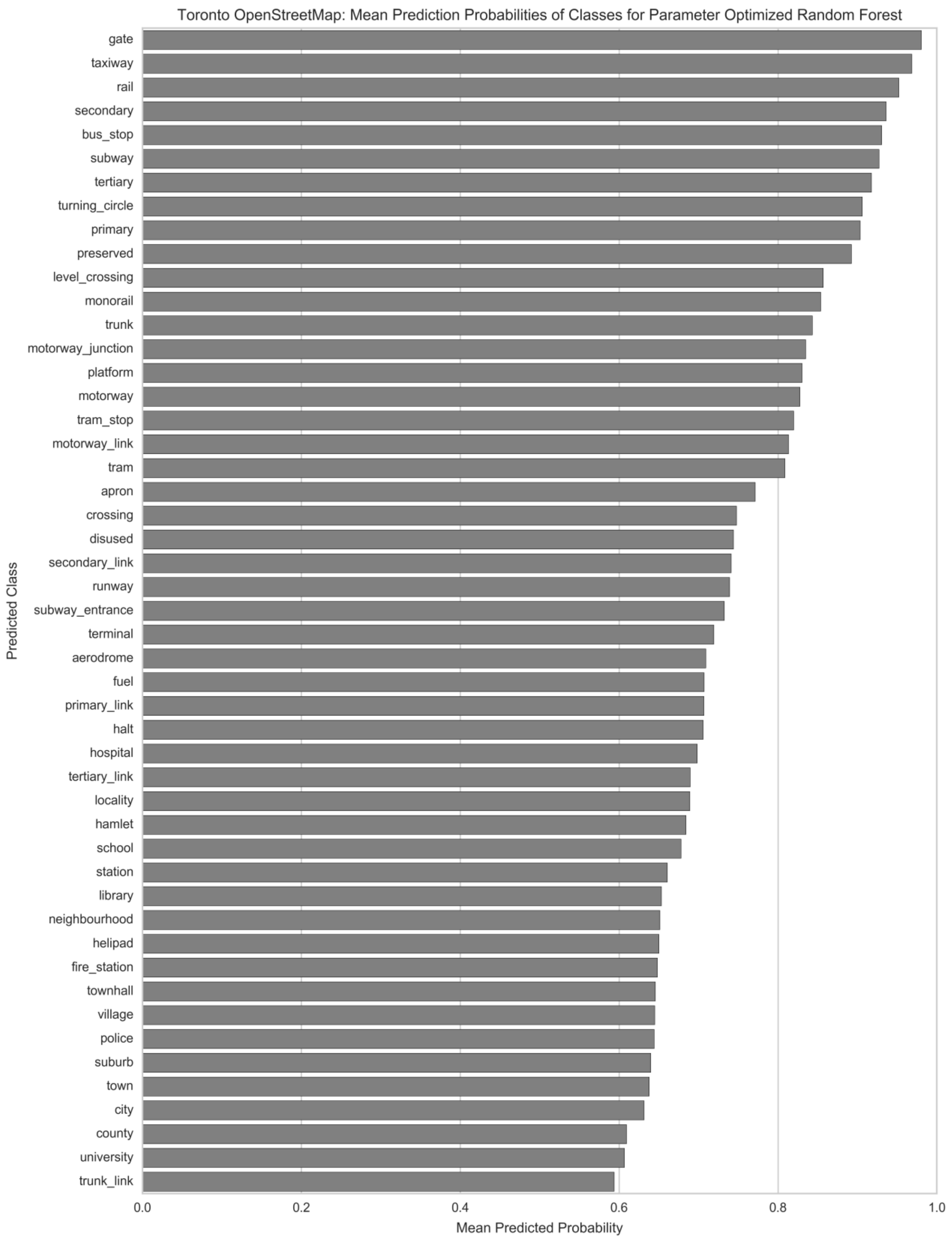

Figure 20: Mean Prediction Probabilities per Class for Parameter Optimized Random Forest.

4.7 Variable Importance

The variable importance measures for the random forest with 128 decision trees trained on the entire Toronto OSM dataset provided insight into the influence of variables on the predictive pattern of the data. The variable importance measures are shown as an automated bar plot of variable importance for the parameter optimized random forest in Figure 21. The variable importance measures are sorted from highest to lowest (top to bottom). The most influential variables of the random forest were length and vertices. Length and vertices had distinctive values that provided the best split of data according to an entropy-based measure of impurity for the entirety of the data. Other notable variables were near_subway, near_secondary, near_rail, near_bus_stop, near_motorway_link, and near_primary in which the variable importance measures were also relatively high. The classes near_secondary, and near_primary refer to the road classes in the Toronto OSM dataset.

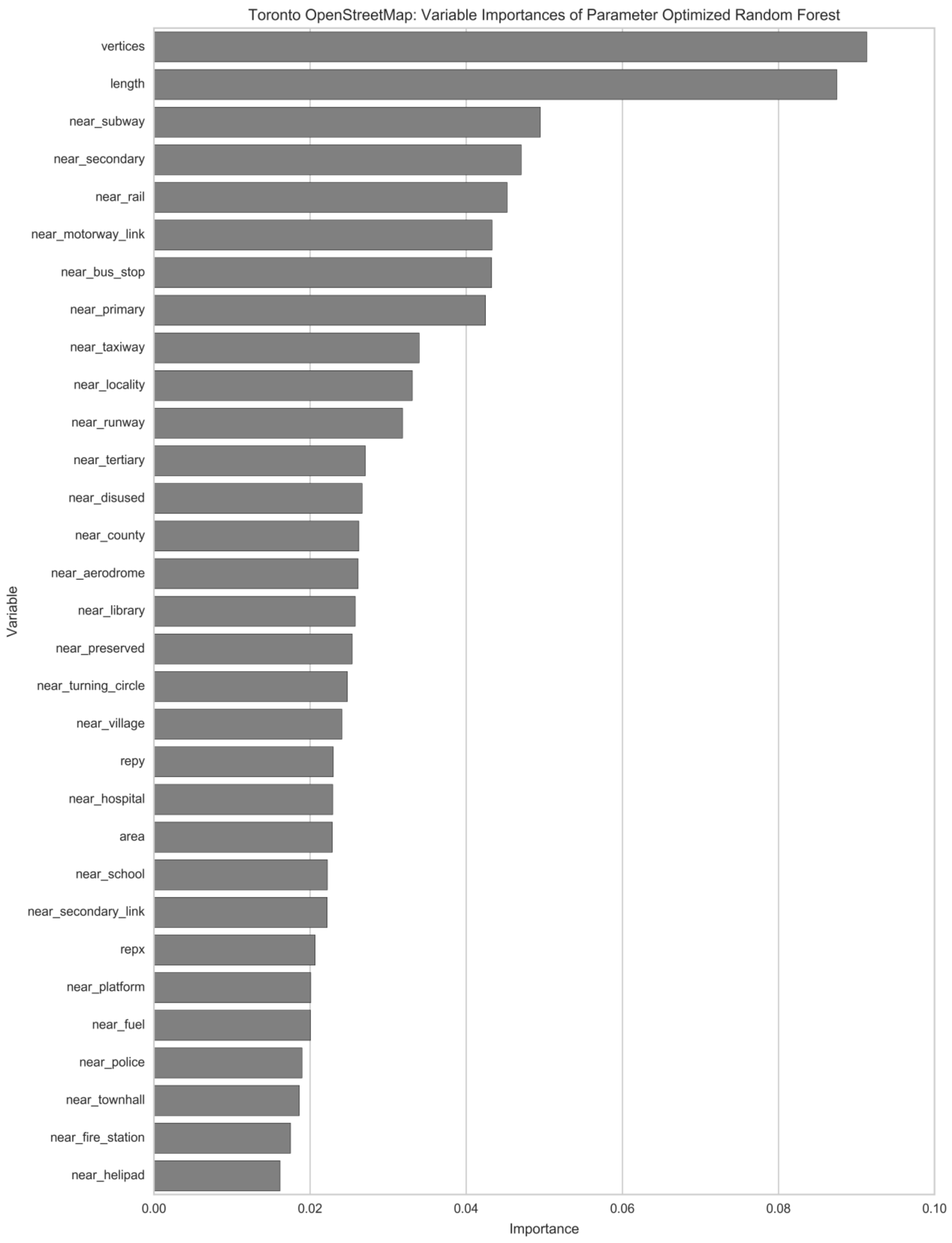

Figure 21: Variable Importance for Parameter Optimized Random Forest.

4.8 Outliers and Variable Contributions

Outliers were identified by creating proximity matrices for each class and calculating outlier measures for all geographic objects. The outlier measures using the parameter optimized random forest model are seen as an automated box plot of outlier measures for each class in Figure 22. Outlier values greater than 10 are considered outlier geographic objects. The school and hospital classes were the only classes that had outliers as all of the outlier measures above the value of 10 were in the school and hospital classes. These outliers may be further inspected with variable contribution metrics.

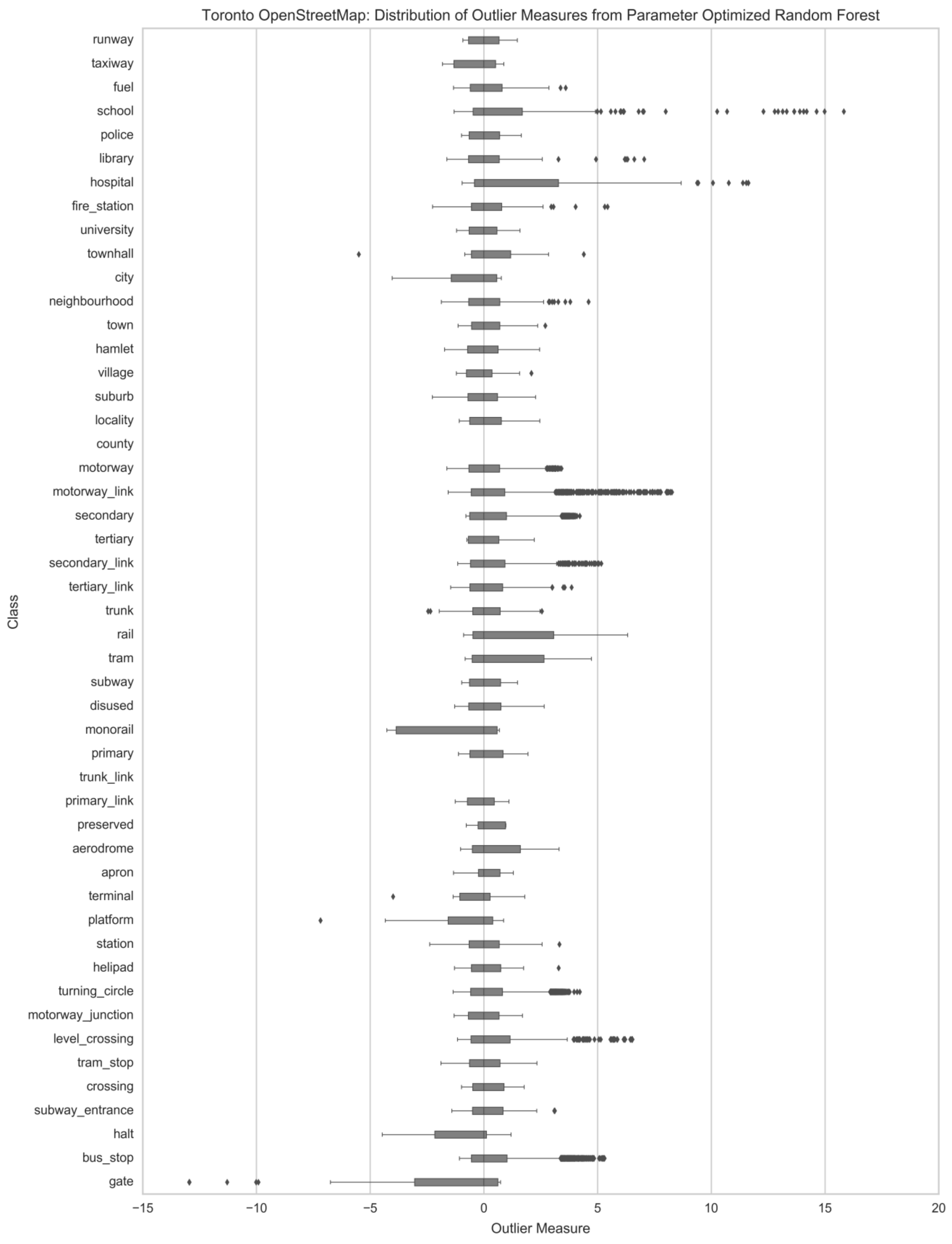

Figure 22: Outlier Measures per Class for Parameter Optimized Random Forest.

Variable contributions were calculated for the outlier school and hospital classes to determine which variables influenced the outlying values in the classes. The variable contributions for the outlier school and outlier hospital classes are seen as automated bar plots of variable contributions using the parameter optimized random forest model in Figure 23 and Figure 24. The variable contribution values are sorted from highest to lowest (top to bottom). The length and vertices variables did not contribute to the school and hospital outlier predictions as geographic objects in the classes have point geometry which contained only 1 vertex and had length and area values of zero. The highest contribution was from the variable near_bus_stop (the distance to the nearest bus stop) for the school class, and near_secondary (the distance to the nearest secondary road) for the hospital class. Other notably high contributions for the school class outliers were from the variables near_primary, near_secondary, near_subway, near_motorway_link, and near_turning_circle. The high contribution variables were mostly related to transportation type geographic objects.

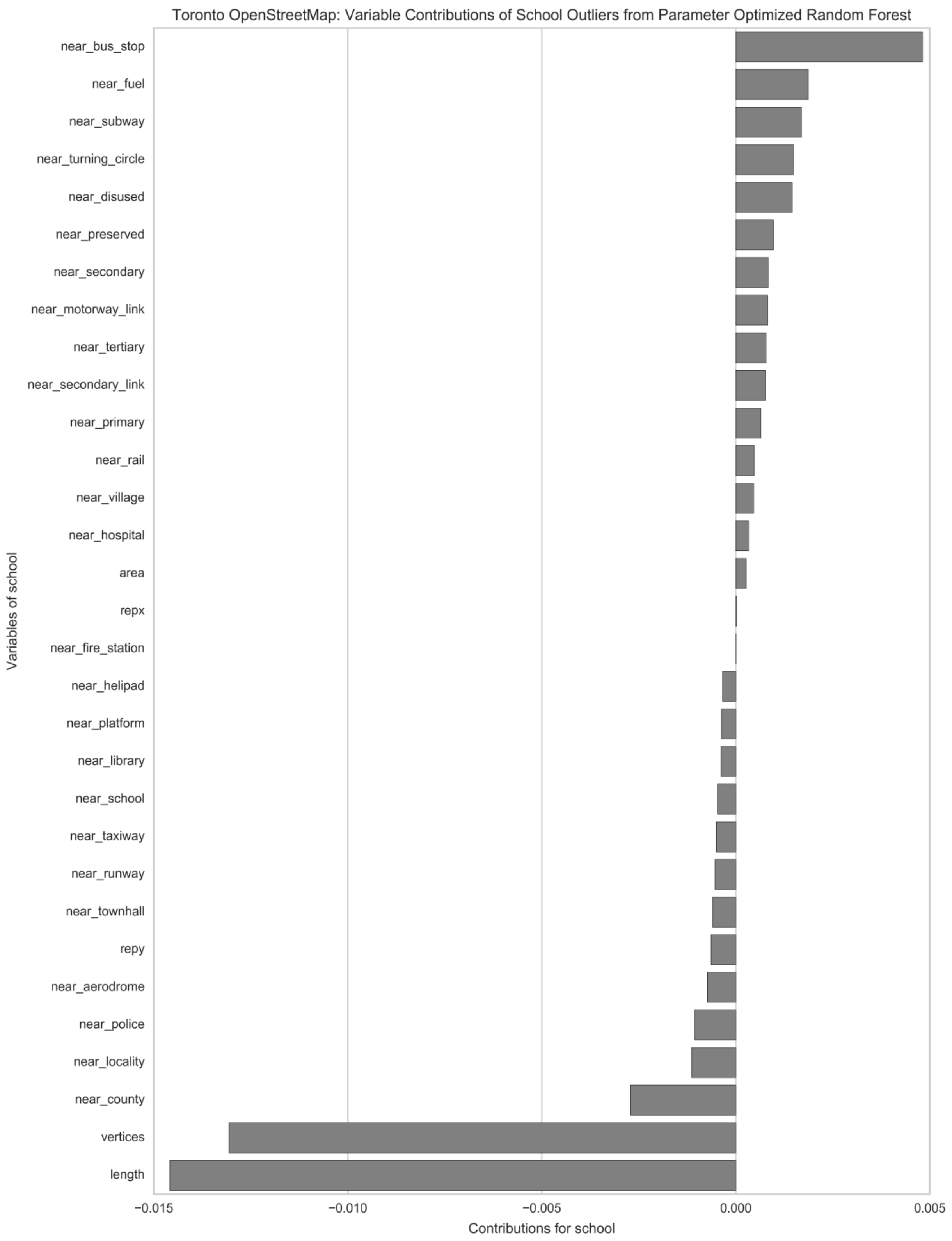

Figure 23: Variable Contributions for Outlier School Class.

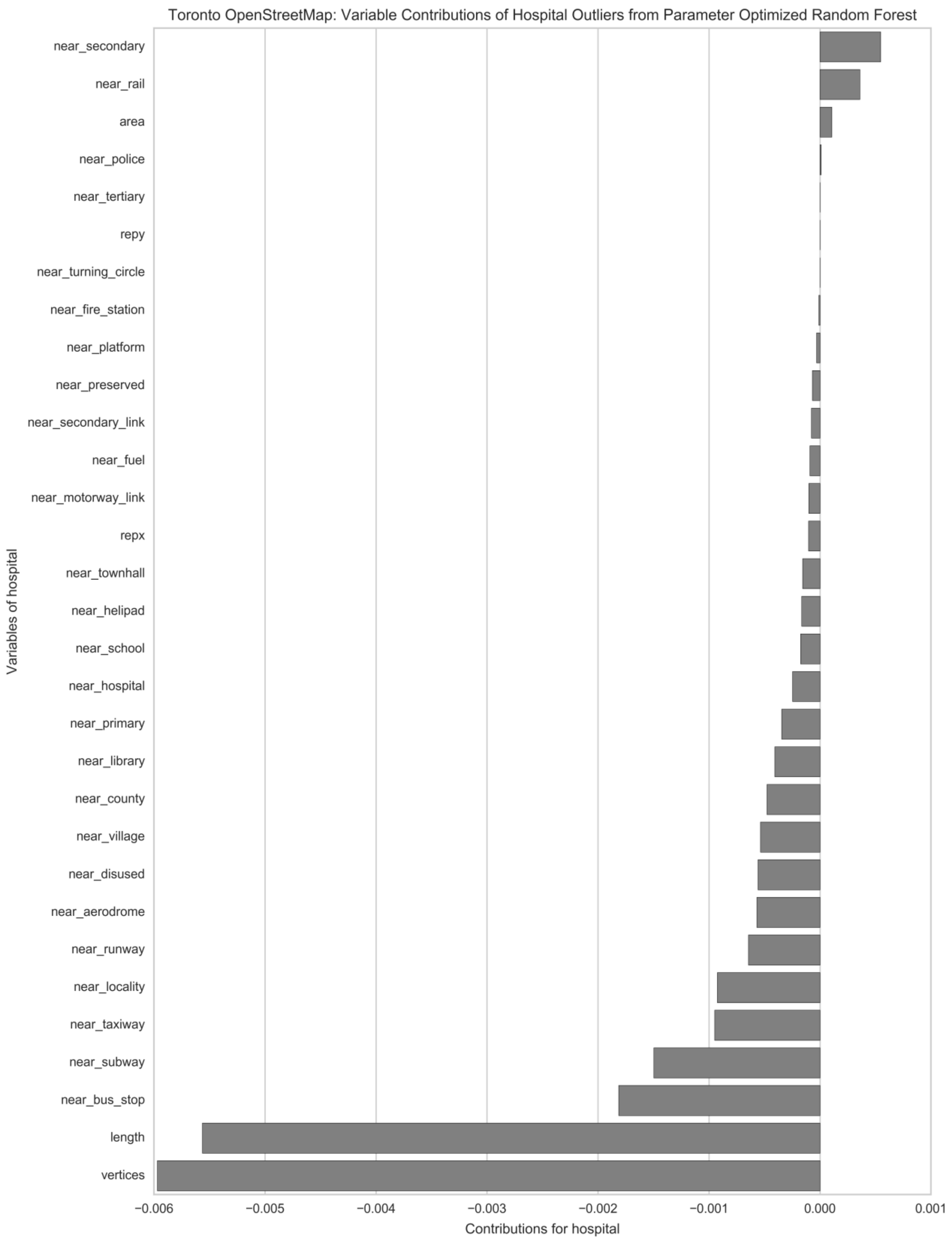

Figure 24: Variable Contributions for Outlier Hospital Class.

CHAPTER 5: DISCUSSION AND CONCLUSION

The nature of VGD is that it is dynamic, voluminous, and unique, but often erroneous and difficult to understand. This research used a random forest to reveal the internal structure of large quantities of VGD, while enabling prediction and corrective action that are relatively scalable. The results from the random forest may be used to guide and direct attention to erroneous and significant data, understand the patterns in the data, and provide automated predictions or suggestions with a metric of confidence for unseen data. It also has the ability to update and retrain new data, which is highly effective for the high temporal resolution of VGD. The versatility, robustness, and automated efficiency of the random forest may be applied in a variety of ways to enhance the data quality and understanding of VGD.

5.1 Recommendations and Auto-completion

The random forest provides prediction probabilities and outlier measures that could be utilized to perform suggestions of classes for automated or supported data quality cleaning and enhancement. Each prediction has an associated probability that can be used to provide a measure of confidence for the random forest. The large quantities of classes make it difficult to select the best suited class for each geographic object, and understanding the standards and structures for each class may quickly become tedious for users – especially given that most of these contributors are often editing and producing data voluntarily. Providing predictions and probabilities enables contributors and editors to receive class suggestions with a quantified certainty for each geographic object that is created or edited. A notable example of class suggestions for improving VGD quality is *OSMRec*, an automated recommendation tool for categories in OSM based on an SVM algorithm (Karagiannakis et al., 2015). It is also possible to fill in incomplete or unclassified geographic objects using the predictions from the random forest with supportive probabilities for each prediction to evaluate the quality of the data completion procedure. Suggestions may also be made for erroneous data that may need correcting by using the outlier measures along with the probabilities to focus attention to particular geographic objects in the abundance of data available. The suggestion of classes and outliers for geographic objects have the potential to improve the rate of VGD production and quality, create consistency in the classes of the VGD, and increase the ease of selecting the best suited classes for each geographic object.

5.2 Information and Knowledge Support

The variable importance measures obtained from the random forest not only directed attention to which variables influence the overall predictive pattern, but enabled the detection of redundant and non-informative variables. Important variables may be utilized to create statistical summaries of the variable values for each class to provide descriptive information for each class. These class descriptions may enable better understanding of the geography in the study area, as well as inform unexperienced data contributors and editors of the characteristics of each class. For example, the top 5 or 10 variables could be selected to provide summary statistics on the nearest distances to each class to reveal if a class has other classes nearby or further away. When deciding which class to assign to the geographic object, the contributor or editor now has more information guiding their decision and the source of that information may be traced back to the variable importance for analysis. The variable importance measures provide an understanding of the overall data structure, and may prove to be especially effective when contributors or editors are inexperienced with the geography and structure of the study area.

Variable contributions can also be utilized for individual predictions to provide descriptive information about variables at a localized scale. For each prediction, an associated set of variable contributions can be used to further inform the data contributor or editor of the geographic object. The values of the variables with the highest contribution may be used as informative descriptors for the single geographic object. This enables the data contributor or editor to be informed of how the random forest made the suggestion and whether it would be logical in the experience of the user to classify the geographic object with the recommended classes. Both the variable importance measures and the variable contributions attempt to reduce the amount of variables such that the abundance of data may be transformed to informative knowledge support to guide human decisions in data contribution and editing.

5.3 Example Application for OpenStreetMap Outliers

A working example application can be to support human data inspection using outlier measures and variables contributions on a web map. The hospital and school outliers are seen in Figure 25 and Figure 26 respectively. The outliers were detected using random forest outlier measures and the top five contributing variables were selected based on the random forest variable contribution metrics. Each geographic object can be interactively selected to display the name, id, and the top five contributing variable values. Closer inspection reveals that the hospitals were hospital wings that were a further than normal from secondary roads, and that the schools were historical geographic objects that were uncommonly far from bus stops, fuel stations, and subways. In addition to understanding the outliers without inspecting the entire dataset manually, the outlier hospitals and schools draw attention to the enhancement of OSM tagging standards. Some example suggestions for improving OSM tagging standards may be to include hospital wings as a separate tag value or a more detailed tag value, and to incorporate historical geographic objects as a separate OSM project so that volunteers can digitize historical data. As exemplified, the volunteer or editor did not have to understand the random forest algorithm, manually inspect a large sample of objects to find the outliers, or look through irrelevant variables.

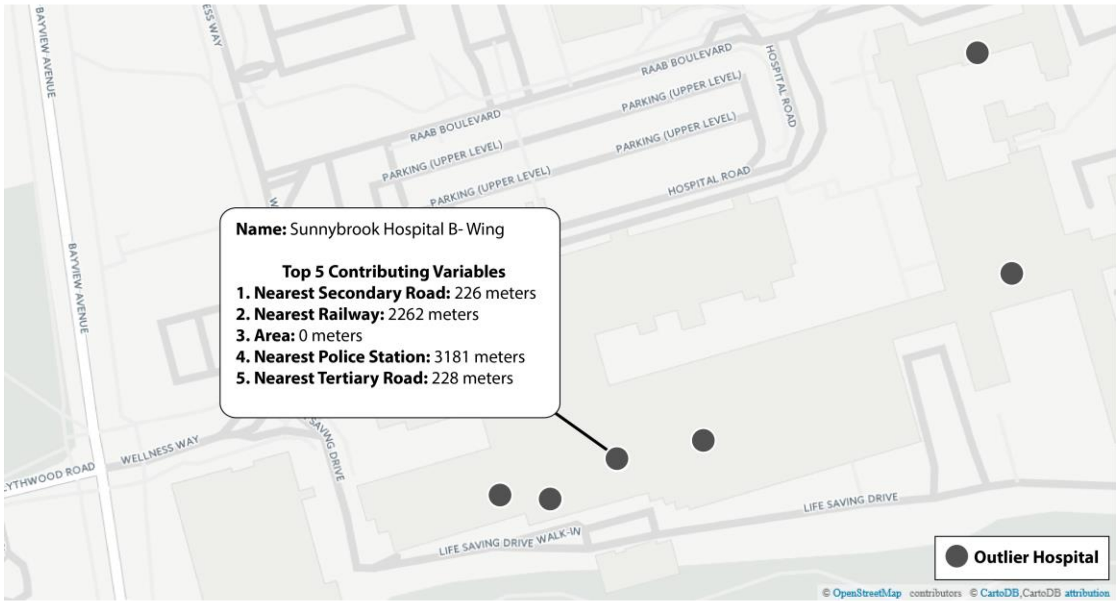
BAYVIEW AVENUE
RAAB BOULEVARD
PARKING (UPPER LEVEL)
HOSPITAL ROAD
WELLNESS WAY
LIFE SAVING DRIVE
Name: Sunnybrook Hospital B- Wing
Top 5 Contributing Variables
1. Nearest Secondary Road: 226 meters
2. Nearest Railway: 2262 meters
3. Area: 0 meters
4. Nearest Police Station: 3181 meters
5. Nearest Tertiary Road: 228 meters
● Outlier Hospital
© OpenStreetMap contributors © CartoDB,CartoDB attribution

Figure 25: Example of Top Five Variable Contributions for Outlier Hospitals.

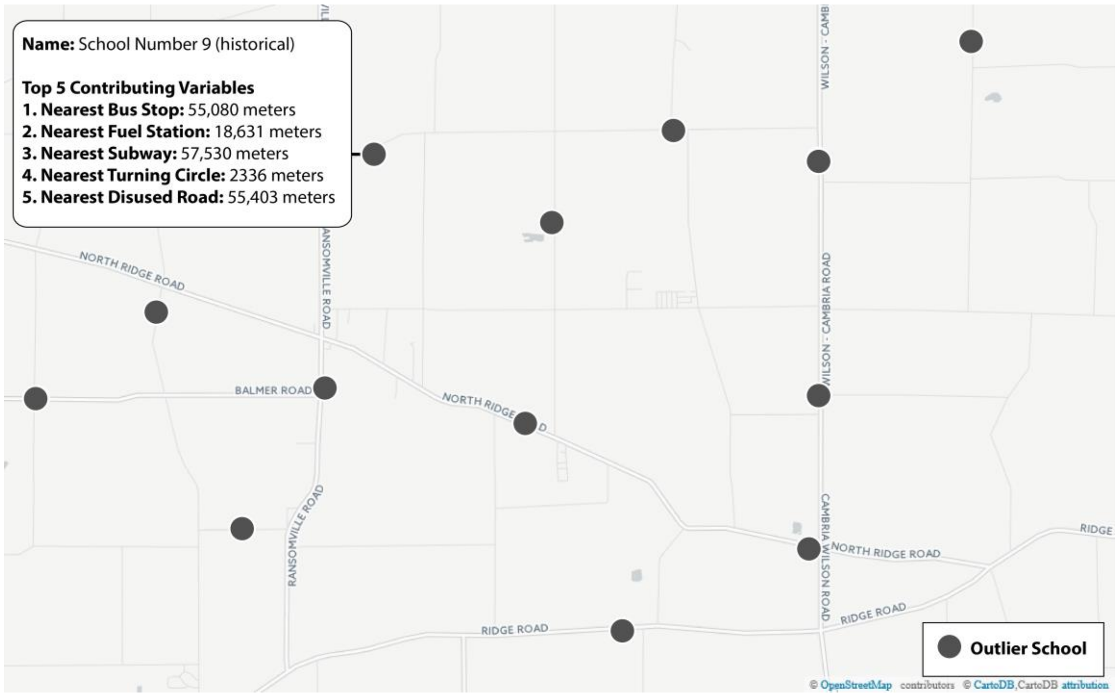
Name: School Number 9 (historical)
Top 5 Contributing Variables
1. Nearest Bus Stop: 55,080 meters
2. Nearest Fuel Station: 18,631 meters
3. Nearest Subway: 57,530 meters
4. Nearest Turning Circle: 2336 meters
5. Nearest Disused Road: 55,403 meters
NORTH RIDGE ROAD
RANSOMVILLE ROAD
BALMER ROAD
WILSON - CAMBRIA ROAD
RIDGE ROAD
● Outlier School
© OpenStreetMap contributors © CartoDB,CartoDB attribution

Figure 26: Example of Top Five Variable Contributions for Outlier Schools.

5.4 Limitations

The random forest models built in this research did not consider the inclusion of raster data from remote sensed imagery to improve the quality of predictions. When editing or contributing VGD, users often have a base map that contains raster based data of remote sensed imagery of the earth. These remote sensed images guide the user in creating the geographic objects and the classes in which they are assigned. The inclusion of raster data from remote sensed images of the earth that were used as a base map for contributors and editors may potentially produce informative variables that add to the predictive performance of the random forest.

Temporal data from user editing histories were also not incorporated in the random forest models in this research. The behaviour of data contributors and editors may aid in the detection of outliers or strengthen the confidence of each class prediction from the random forest. This behaviour may be extracted as data from user contributing and editing histories throughout time. Neis et al (2012) utilized OSM user editing and contributing history to detect forms of vandalism and abnormal user behaviour that could lead to erroneous and irregular OSM data. The historical data may also be used as a measure of confidence and quality in the assigned classes of the geographic objects. The high temporal resolution of VGD and the variables that may be extracted from the temporal data may prove to be potentially informative in the detection of outliers and anomalies in the data, but could not be considered in this research.

The first nearest neighbour spatial relationship was chosen in this research, but other spatial relationships were not experimented with. Several other spatial relations such as distance bands, k-nearest neighbours, and polygon contiguities were not considered (Anselin, 1990).

Spatial relationships other than the nearest neighbour method may be studied and experimented with to better understand the spatial pattern of the data in relation to predictive performance and computational time.

5.5 Conclusion

This research investigated the use of the random forest algorithm for VGD using Toronto OSM data as a case study. The random forest algorithm was able to achieve an adequate predictive performance of approximately 0.71 based on a F1 score metric for the Toronto OSM data. It was also possible to reveal the influences of each variable used in the random forest model using variable importance measures and variable contributions to better understand patterns in the data. Outliers were detected using a measure based on calculating proximity metrics from the decision trees in the random forest. The results show that the random forest algorithm produces capable models for VGD as it not only predicts unseen data, but also provides quantifiable and interpretable measures to explain the patterns in the data.

The interpretability introduced by variable importance measures and variable contributions of the random forest are especially useful in improving VGD quality in VGI and understanding the patterns in the large quantities of data. Suggestions of classes and outliers may be made based on the predictions of the random forests with confidence measures based on prediction probabilities. Variable contributions and variable importance further supplement these suggestions by providing data contributors and editors with information on the characteristics of each class, and why it made the suggestion. These suggestions may also be used for auto-completion of data with explainable variables and confidence measures for each automated completion task. The improvement of knowledge for data contributors and editors lead to the enhancement in VGD quality, and the automated capabilities of the random forest potentially improve the rate in which VGD may be produced.

However, raster data such as remotely sensed imagery and temporal data such as user contribution history were not utilized in this research. Exploration of other spatial relationships were also not experimented with. The random forest algorithm would potentially benefit from the use of raster data, temporal data, and possibly other spatial relationships as they would produce variables that may be calculated as informative to improve the prediction results. Further work in exploring the spatial relationships in VGD, and the incorporation of raster and temporal data is needed to improve the random forest algorithm to enhance pattern recognition for VGD.

REFERENCES

Abdallah, C. (2010). Spatial distribution of block falls using volumetric GIS–decision-tree models. *International Journal of Applied Earth Observation and Geoinformation, 5*(12), 393-403.

Anselin, L. (1990). SPATIAL DEPENDENCE AND SPATIAL STRUCTURAL INSTABILITY IN APPLIED REGRESSION ANALYSIS. *Journal of Regional Science, 30*(2), 185-207.

Beyer, K., Goldstein, J., Ramakrishnan, R., & Shaft, U. (1998). When Is “Nearest Neighbor” Meaningful? In C. Beeri, & P. Buneman, *Database Theory — ICDT’99* (pp. 217-235). Berlin: Springer Berlin Heidelberg.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* Springer.

Bostrom, H. (2007). Estimating class probabilities in random forests. *Machine Learning and Applications*, 211-216.

Breiman, L. (2001). Random Forests. *Machine Learning, 45*(1), 5-32.

Breiman, L. (2003). *Manual–setting up, using, and understanding random forests V4.0.* Retrieved from https://www.stat.berkeley.edu/~breiman/Using_random_forests_v4.0.pdf

Breiman, L., & Cutler, A. (2004). *Random Forests*. Retrieved March 23, 2016, from Random Forests: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#outliers

Butler, H., Daly, M., Doyle, A., Gillies, S., Schaub, T., & Schmidt, C. (2008, June 16). *The GeoJSON Format Specification*. Retrieved from GeoJSON: https://datatracker.ietf.org/doc/draft-ietf-geojson/

Calle, M. L., & Urrea, V. (2010). Letter to the editor: stability of random forest importance measures. *Briefings in bioinformatics*, 1-4.

Chasmer, L., Hopkinson, C., Veness, T., Quinton, W., & Baltzer, J. (2014). A decision-tree classification for low-lying complex land cover types within the zone of discontinuous permafrost. *Remote Sensing of Environment*, 73-84.

Christopher, P. (1998). General Semantics on the Internet. *et Cetera*, 450-463.

Clark, P. J., & Evans, F. C. (1954). Distance to nearest neighbor as a measure of spatial relationships in populations. *Ecology, 35*(4), 445-453.

Continuum Analytics. (2015, November). *Anaconda Software Distribution*. Retrieved from Continuum Analytics: https://continuum.io/

Corne, S. A., Carver, S. J., Kunin, W. E., Lennon, J. J., & Hees, W. W. (2004). Predicting Forest Attributes in Southeast Alaska Using Artificial Neural Networks. *Forest Science*, 259-276.

Crasto, N., Hopkinson, C., Forbes, D. L., Lesack, L., Marsh, P., Spooner, I., & van der Sanden, J. J. (2015). A LiDAR-based decision-tree classification of open water surfaces in an Arctic delta. *Remote Sensing of Environment, 164*, 90-102.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to Support Vector Machines and other kernel-based learning methods.* Cambridge: Cambridge University Press.

Cross, S. S., Harrison, R. F., & Kennedy, R. L. (1995). Introduction to neural networks. *The Lancet*, 1075-1079.

D'Antonio, F., Fogliaroni, P., & Kauppinen, T. (2014). VGI edit history reveals data trustworthiness and user reputation. *17th AGILE International Conference on Geographic Information Science.* Spain.

Dietterich, T. G. (2000). Ensemble Methods in Machine Learning. *Multiple classifier systems, 1857*, 1-15.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis* (Vol. 3). New York: Wiley.

Fast, V., & Rinner, C. (2014). A Systems Perspective on Volunteered Geographic Information. *International Journal of Geo-Information, 3*(4), 1278-1292.

Feick, R., & Roche, S. (2012). Understanding the Value of VGI. In D. Sui, S. Elwood, & M. Goodchild, *Crowdsourcing geographic knowledge* (pp. 15-29). Dordrecht: Springer.

Feller, W. (1968). *An Introduction to Probability Theory and Its Applications.* London: John Wiley & Sons.

Fritz, C. E., Schuurman Nadine, R. C., & Lear, S. (2013). A scoping review of spatial cluster analysis techniques for point-event data. *Geospatial health*, 183-198.

Furlanello, C., Neteler, M., Merler, S., Menegon, S., Fontanari, S., Donini, A., . . . Chemini, C. (2003). GIS and the Random Forest Predictor: Integration in R for Tick-Borne Disease Risk Assessment. *Proceedings of the 3rd international workshop on distributed statistical computing*, (pp. 1-11). Vienna.

GeoPandas developers. (2014). *Geopandas*. Retrieved March 10, 2016, from Geopandas: http://geopandas.org/

Ghimire, B., Rogan, J., Galiano, V. R., Panday, P., & Neeti, N. (2012). An evaluation of bagging, boosting, and random forests for land-cover classification in Cape Cod, Massachusetts, USA. *GIScience & Remote Sensing*, 623-643.

Girres, J. F., & Touya, G. (2010). Quality Assessment of the French OpenStreetMap Dataset. *Transactions in GIS, 14*(4), 435-459.

Gislason, P. O., Benediktsson, J. A., & Sveinsson, J. R. (2006). Random Forests for land cover classification. *Pattern Recognition Letters*, *27*(4), 294-300.

Goodchild, M. F. (2007). Citizens as sensors: the world of volunteered geography. *GeoJournal*, 211-221.

Goshtasby, A. A. (2012). Similarity and dissimilarity measures. *Advances in Computer Vision and Pattern Recognition*, 7-66.

Guo, Q., Kelly, M., & Graham, C. H. (2005). Support vector machines for predicting distribution of Sudden Oak Death in California. *Ecological Modelling, 182*(1), 75-90.

Guttman, A. (1984). *-trees: a dynamic index structure for spatial searching* (2 ed., Vol. 14). ACM.

Hagenauera, J., & Helbicha, M. (2011). Mining Urban Land Use Patterns from Volunteered Geographic Information Using Genetic Algorithms and Artificial Neural Networks. *International Journal of Geographical Information Science*, 963-982.

Haklay, M. M. (2008). OpenStreetMap: User-Generated Street Maps. *Pervasive Computing*, 12-18.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2009). *Elements of Statistical Learning: Data mining, inference, and prediction.* New York: Springer.

Holland, J. H. (1992). Complex adaptive systems. *Daedalus*, 17-30.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 359-366.

Hu, L., Aixin, S., & Liu, Y. (2014). Your neighbors affect your ratings: on geographical neighborhood influence to rating prediction. *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval* (pp. 345-354). New York: Association for Computing Machinery.

Janowicz, K., Scheider, S., Pehle, T., & Hart, G. (2012). Geospatial Semantics and Linked Spatiotemporal Data - Past, Present, and Future. *Semantic Web*, 1-13.

Jilani, M., Corcoran, P., & Bertolotto, M. (2013). Automated Quality Improvement of Road Network in OpenStreetMap. *Association of Geographic Information Laboratories in Europe* (pp. 1-4). Leuven: AGILE.

Jones, E., Oliphant, T., Peterson, P., & others. (2001). *SciPy: Open source scientific tools for Python*. Retrieved March 10, 2016, from SciPy: Open source scientific tools for Python: http://www.scipy.org/

Karagiannakis, N., Giannopoulos, G., Skoutas, D., & Athanasiou, S. (2015). OSMRec Tool for Automatic Recommendation of Categories on Spatial Entities in OpenStreetMap. *Proceedings of the 9th ACM Conference on Recommender Systems* (pp. 337-338). ACM.

Knerr, S., Personnaz, L., & Dreyfus, G. (1990). Single-layer learning revisited: a stepwise procedure for building and training a neural network. *Neurocomputing*, 41-50.

Kodratoff, Y., & Michalski, R. (2014). *Machine Learning: An Artificial Intelligence Approach* (Vol. 3). San Mateo: Morgan Kaufmann.

Kotsiantis, S. B. (2013). Decision trees: a recent overview. *Artificial Intelligence Review, 39*(4), 261-283.

Koukoletsos, T., Haklay, M., & Ellul, C. (2012). Assessing data completeness of VGI through an automated matching procedure for linear data. *Transactions in GIS, 16*(4), 477-498.

Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika, 29*(1), 1-27.

Lancashire, L., Lemetre, C., & Ball, G. R. (2009). An introduction to artificial neural networks in bioinformatics--application to complex microarray and mass spectrometry datasets in cancer studies. *Briefings in Bioinformatics*, 315-329.

Lantz, B. (2013). Black Box Methods – Neural Networks and Support Vector Machines. In B. Lantz, *Machine Learning with R* (pp. 205-242). Packt Publishing Ltd.

Liaw, A., & Wiener, M. (2002). Classification and Regression by randomForest. *R News*, *2*, 18-22.

Louppe, G. (2014). *Understanding Random Forests: From Theory to Practice.* Belgium: University of Liege.

Louppe, G., Wehenkel, L., Sutera, A., & Geurts, P. (2013). Understanding variable importances in forests of randomized trees. *Advances in Neural Information Processing Systems*, 431-439.

Maltarollo, V. G., Honório, K. M., & Ferreira da Silva, A. B. (2013). Applications of Artificial Neural Networks in Chemical Problems. In K. Suzuki, *Artificial Neural Networks - Architectures and Applications* (pp. 203-223). Rijeka: InTech.

Mapzen. (2016, March 10). *Metro Extracts*. Retrieved from Mapzen: https://mapzen.com/data/metro-extracts/

Mascaro, J., Asner, G. P., Knapp, D. E., Kennedy-Bowdoin, T., Martin, R. E., Anderson, C., . . . Chadwick, K. D. (2014). A Tale of Two ‘‘Forests’’: Random Forest Machine Learning Aids Tropical Forest Carbon Mapping. *PLOS one, 9*(1).

Mather, P., & Brandt, T. (2009). *Classification Methods for Remotely Sensed Data* (2 ed.). Boca Raton: CRC Press.

McKinney, W. (2010, March 3). Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*, (pp. 51-56). Retrieved from pandas: http://pandas.pydata.org/

Miller, H. J. (2004). Tobler’s First Law and Spatial Analysis. *Annals of the Association of American Geographers, 94*(2), 284-289.

Mooney, P., Corcoran, P., & Winstanley, A. C. (2010). Towards Quality Metrics for OpenStreetMap. *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems* (pp. 514-517). New York: ACM.

Neis, P., Goetz, M., & Zipf, A. (2012). Towards Automatic Vandalism Detection in OpenStreetMap. *ISPRS International Journal of Geo-Information, 1*(3), 315-332.

Nielson, M. A. (2015). *Neural Networks and Deep Learning.* Determination Press.

Noble, W. S. (2006). What is a support vector machine? *Nature Biotechnology*, 1565-1567.

Ohmann, J. L., & Gregory, M. J. (2002). Predictive mapping of forest composition and structure with direct gradient analysis and nearest-neighbor imputation in coastal Oregon, USA. *Canadian Journal of Forest Research, 32*(4), 725-741.

Omniscale. (2016, March 3). *Imposm Imports OpenStreetMap data into PostgreSQL/PostGIS databases*. Retrieved from Imposm: http://imposm.org/

Oshiro, T. M., Perez, P. S., & Augusto, J. (2012). How many trees in a random forest? In P. Perner, *Machine Learning and Data Mining in Pattern Recognition* (Vol. 7376, pp. 154-168). Berlin: Springer.

Pal, M. (2005). Multiclass Approaches for Support Vector Machine Based Land Cover Classification. *Proceeding of Computer Vision and Pattern Recognition: Neural and Evolutionary Computing*. Map India.

Pal, M., & Mather, P. M. (2005). Support vector machines for classification in remote sensing. *International Journal of Remote Sensing, 26*(5), 1007-1011.

Palczewska, A., Palczewski, J., Robinson, R. M., & Neagu, D. (2014). Interpreting random forest classification models using a feature contribution method. *Integration of Reusable Systems*, 193-218.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 2825-2830.

Powers, D. M. (2011). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies, 2*(1), 37-63.

Prasad, A. M., Iverson, L. R., & Liaw, A. (2006). Newer Classification and Regression Tree Techniques: Bagging and Random Forests for Ecological Prediction. *Ecosystems*, 181-199.

Python Software Foundation. (2016). *Python*. Retrieved from Python: https://www.python.org/

Quinlan, J. R. (1984). Induction of decision trees. *Machine Learning, 1*(1), 81-106.

Rao, C. R. (2005). *Handbook of Statistics: Data Mining and Data Visualization* (Vol. 24). Elsevier.

Reusch, D. B., & Alley, R. B. (2002). Automatic weather stations and artifical neural networks: Improving the instrumental record in West Antarctica. *Monthly Weather Review*, 3037-3034.

Rodriguez-Galiano, V. F., Ghimire, B., Rogan, J., Chica-Olmo, M., & Rigol-Sanchez, J. P. (2012). An assessment of the effectiveness of a random forest classifier for land-cover classification. *ISPRS Journal of Photogrammetry and Remote Sensing, 67*, 93-104.

Rohloff, K., Dean, M., Emmons, I., Ryder, D., & Sumner, J. (2007). An Evaluation of Triple-Store Technologies for Large Data Stores. *On the Move to Meaningful Internet Systems*, 1105-1114.

Rojas, R. (1996). *Neural networks: a systematic introduction.* Berlin: Springer.

Rokach, L., & Maimon, O. (2005). Top-Down Induction of Decision Trees Classifiers—A Survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 35*(4), 476-487.

Rumelhart, D. E., Geoffrey, E., & Williams, R. J. (1988). Learning representations by back-propagating errors. In T. A. Polk, & C. M. Seifert, *Cognitive Modeling* (pp. 533-536). Cambridge: The MIT Press.

Saabas, A. (2015). *treeinterpreter*. Retrieved March 16, 2016, from Python Package Index: https://pypi.python.org/pypi/treeinterpreter/0.1.0

See, L., Comber, A., Salk, C., Fritz, S., van der Velde, M., Perger, C., . . . Obersteiner, M. (2013). Comparing the Quality of Crowdsourced Data Contributed by Expert and Non-Experts. *PLOS one*.

Stevens, F. R., Gaughan, A. E., Linard, C., & Tatem, A. J. (2015). Disaggregating Census Data for Population Mapping Using Random Forests with Remotely-Sensed and Ancillary Data. *PloS one, 10*(1).

Stevens, K. B., & Pfeiffer, D. U. (2011). Spatial modelling of disease using data- and knowledge-driven approaches. *Spatial and Spatio-temporal Epidemiology, 2*(3), 125-133.

Strobl, C., Boulesteix, A. L., Kneib, T., Augustin, T., & Zeileis, A. (2008). Conditional variable importance for random forests. *BMC bioinformatics, 9*(1).

Sui, D., Elwood, S., & Goodchild, M. (2012). *Crowdsourcing geographic knowledge: volunteered geographic information (VGI) in theory and practice.* Springer.

Tan, P.-N., Steinbach, M., & Kumar, V. (2006). *Introduction to Data Mining.* Pearson Education.

Tehrany, M. S., Pradhan, B., Mansor, S., & Ahmad, N. (2015). Flood susceptibility assessment using GIS-based support vector machine model with different kernel types. *CATENA, 125*, 91-101.

The Toblerity Project. (2014). *Rtree: Spatial indexing for Python*. Retrieved March 10, 2016, from Rtree: http://toblerity.org/rtree/

Tobler, W. R. (1970). A computer movie simulating urban growth in the Detroit region. *Economic geography, 46*, 234-240.

van der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science & Engineering*, 22-30.

Varoquaux, G. (2009). *Joblib: running Python functions as pipeline jobs*. Retrieved March 17, 2016, from Joblib Documentation: https://pythonhosted.org/joblib/index.html

Waskom, M. (2015). *Seaborn: statistical data visualization*. Retrieved March 10, 2016, from seaborn: https://stanford.edu/~mwaskom/software/seaborn/