Predicting Method of Transportation

COMP 598 – Final Project

Alireza Saberi
260358148

Jean Merheb-Harb
260324745

Teng Long
260616355

ABSTRACT

In this report, authors have approached the "method of travel classification" task using different classifiers and preprocessing steps. The dataset is an augmented version of Concordia University's private TRIP Dataset combined with gaseous pollutants data of Montreal OpenData and weather network data. The performances of logistic regression, feed forward neural network, Support Vector Machine (SVM), and random forest have been compared. Random forest shows the best accuracy 72.7%, followed by neural network (54%), logistic regression (53.5%), and SVM with radial basis function (RBF) kernel (52.7%), respectively.

1. INTRODUCTION

In machine learning, classification is the problem of assigning a new observation to a category that it belongs. In this project, we tackle the problem of classifying method of travel, *i.e.,* walk, bicycle, car, public transit, car and public transit from augmented dataset dataset of Concordia University private TRIP Dataset [1] with gaseous pollutants data of Montreal OpenData [2] and weather network data [3]. TRIP dataset is a subset of cellphone GPS information of 242 users recorded at 698400 geographical points. The gaseous pollutants data of Montreal OpenData contains atmospheric concentrations of gaseous pollutants measured on the Island of Montreal and the climate information gathered from weather network web site.

The geographical information of TRIP dataset includes latitude, longitude, altitude, speed, direction, h_accuracy, and v_accuracy. The gaseous pollutants data includes NO, NO2, PM10, O3, and SO3, while weather network data includes minimum and maximum temperature, and wind speed.

In this project, after segmenting the GPS data points into individual trips, we augment the trip data with the gaseous pollutants data and climate information. The new dataset includes 11890 examples with travel methods.

We attempted logistic regression as our baseline learner, fully connected Feed-forward Neural Network, linear, RBF, poly kernel, and sigmoid kernel SVMs, and random forest to tackle this classification.

The rest of the paper is organized as follows: Section 2 contains the preprocessing methods that we tried to use. Feature selection method is described in Section 3. A brief overview algorithm selection is presented in Section 4. The optimization and parameter selection are in Section 5 and 6. The Testing and validation are presented in Section 7. And Finally Section 8 contains the discussion and conclusion part

2. PREPROCESSING

The raw data are GPS points that are not necessarily defined a specific travel. Thus, we segmented the GPS points into different travels using the algorithm proposed by Zhang et al., [10] in which travels are identified based on changes in the position, the speed values, and the heading changes. Because different travel methods (*e.g.,* walk, bicycle, car, and *etc.*) follow different speed patterns, it is possible to detect each travel based on its speed variations. Using this algorithm, we performed point segmentation in two phases:

1. We checked all points separately for each user and segment them based on the time-stamps, in which a trip must have been done in one day. Then, starting from the first trace to the end, if the traveled distance from that trace to its fifth consecutive trace is less than 5 meters, we break the points and start a new segment from here. To calculate the traveled distance between two traces, we used Haversine[1] formula in which the distance is formulated in the forms of two-argument inverse tangent function that measure the great circle distance between these two points on the Earth.

$$\begin{aligned} haversine(\frac{d}{r}) = haversin(\Theta_2 - \Theta_1) \\ +cos(\Theta_2)cos(\Theta_1)haversin(\lambda_2 - \lambda_1) \end{aligned} \tag{1}$$

   where:

$$haversin(\Theta) = sin^2(\frac{\Theta}{2}) = \frac{1 - cos(\Theta)}{2} \tag{2}$$

   In which d is the distance between the two points, r is the radius of the sphere, Φ_1, ϕ_2 are latitude of point 1 and latitude of point 2, and λ_1, λ_2 are longitude of point 1 and longitude of point 2, respectively.

2. We check the speed of all points detected by the first step and if it is smaller than 0.5 m/s and/or if the changes in heading magnitude is larger than 100 degrees, we go to the next point, otherwise segment it from here.

   We start with 698400 raw travel points and after performing the segmentation and filtering, we end up working with 11 travel points.

3. FEATURE SELECTION

We aim to reduce the number of features by selecting those features that are important for the prediction of the mode of

[1]This formula for calculating the distance between two points based on their GPS information (i.e., latitude and longitude) is recommended by Bob Chamberlain (rgc@jpl.nasa.gov) of Caltech and NASA's Jet Propulsion Laboratory

Table 1: The accuracy before and after feature selection to display the impact of different features.

	Accuracy		
Algorithm	**Before adding pollutant/weather**	**Before feature selection**	**After feature selection**
Logistic regression	0.523	0.530	0.535
Random Forest	0.688	0.709	0.727
Linear SVM	0.524	0.524	0.524
Poly kernel SVM	0.517	0.522	0.524
RBF kernel SVM	0.522	0.527	0.527
Sigmoid kernel SVM	0.522	0.522	0.522
Neural network	n/a	n/a	n/a

transport. We must understand the influence of each feature individually on the system [5]. Thus, we use filter method [7] to perform the feature selection by applying a univariate criterion on each feature separately based on the assumption that there is no interaction between our features. Consequently, we apply Kruskal-Wallis rank sum test on data of each feature and calculate the p-value as a measure of how effective this feature is as an independent, separate feature.

The empirical cumulative distribution function (CDF) of the p-values for our 15 initial features are plotted in Figure 1. It shows that around 80% of the features have p-value smaller than 0.05 and around 50% of the features have p-values close to zero. Moreover, Figure 2 also shows the amount of p-value for different features. We select those with p-value smaller than 0.001 and end up with 16 out of 18 features.

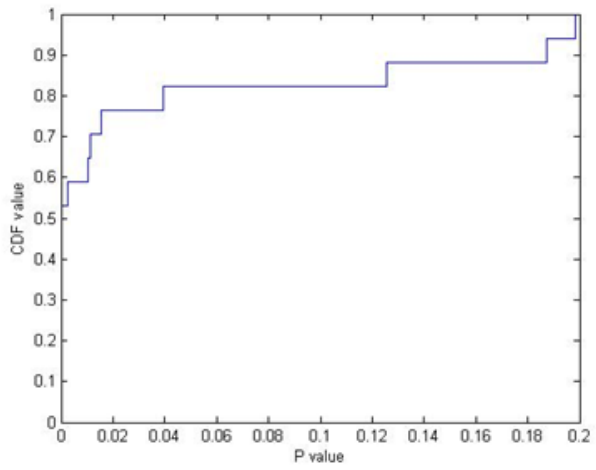

Figure 1: The cumulative distribution function (CDF) of the p-values for initial features.

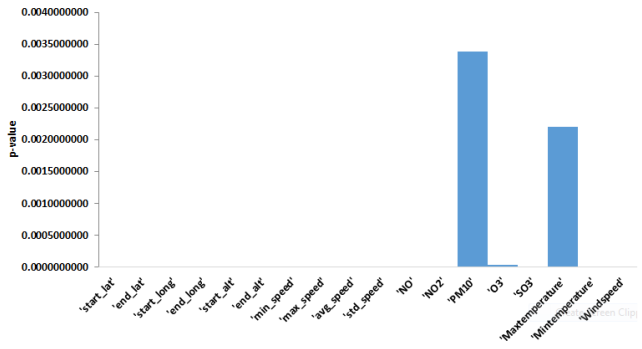

Figure 2: The p-values for 15 initial features.

As shown in Table 1, augmenting with gaseous pollutants and weather features improves the accuracy. Moreover, feature selection increases the accuracy. However, for SVM, regardless the choice of kernels and feature sets, the prediction accuracy remain stable.

4. ALGORITHM SELECTION

4.1 Logistic Regression

Linear classifiers are the simplest classifier which we start with. The classification output is considered as a linear function of the features [8]. However, a linear classifier is not able to model the decision surface between the classes that are not linearly separable. In this study, we used logistic regression classifier, as a base-line method to compare with other more complex classifiers. Logistic regression can be categorized as a discriminative learning algorithm. Given the complexities of our dataset, modelling the conditional probability distribution of classes given features is far more feasible than modelling the joint probability of features and, which is the case in generative learning models. Therefore, logistic regression is used as the basic linear model for approaching this problem.

The data set shows an error rate of 46.51 percent with a linear classifiers after feature selection [8]. Considering the extra sets added to our dataset (e.g. pollutant, and weather information), we expected slightly increase accuracy in comparison with the original dataset. However, the difference is trivial.

4.2 Random Forest

We used `scikit-learn` [8] library's implementation of Random Forest. This algorithm creates many decision trees where the decisions are chosen from a random subset of features. Considering the random subset, the decision that is chosen is the one splitting the data in the most informative way. The following equation gives a quantitative way of evaluating of how informative a decision is.

$$IG(x) = H(D) - H(D|x)$$

where

$$H(D) = -\sum_i p_i log_2 p_i$$

and

$$H(D|x) = \sum_i P(x = v) H(y|x = v)$$

.

As more trees are created, key features will be chosen more often for the decision, causing the set of trees to become correlated. This will then give more precise results as more trees are created. A decision tree segments the data into certain classes. This means that data such as the GPS coordinates can then be divided into areas representing neighbourhoods and speed can be split into high versus low speeds, which can represent a person driving as opposed to walking. Since key differences between our classes are separable in such a way, we believe that random forests will perform very well.

4.3 Feed-forward Neural Network

Multi-layer neural networks trained by gradient descent are known to have the ability to learn complex, high dimensional and nonlinear models [7]. These characteristics make them attractive for many dataset problems, but our data has some features that can't be analyzed as linear variables. Variables such as GPS coordinates cannot be considered as values that increase or decrease, but as segmented ranges that represent key information. We therefore don't expect this classifier to work as well as the Random Forest, which will segment key data very well. In this study, we tested a multilayer fully connected neural network, trained by back propagation, as one of the simplest implementations of the neural network model. Although theoretical results have shown that any function can be approximated using only one hidden layer [mhaskar1992], some studies showed that adding more hidden layers to the neural net sometimes have positive impact on the total performance of the network [7]. Therefore, we tried to reach the optimal network structure considering both single and double hidden layers.

4.4 Support Vector Machines (SVM)

We used `scikit-learn` [9] library's implementation of SVM (Support Vector Machine) with a few different kernels (linear, RBF, poly kernel, and sigmoid). Given data belonging to one of two classes, this classifier tries to find a hyperplane which separates the data. Since there might be many of such hyperplanes, SVM tries to find the one that results in the largest margin. The different kernels find different types of hyperplanes, but all have the same goal of maximizing the margin between classes. For linear kernels, this is done by the following linear optimization:

$$\arg\min_{w,b} \|w\| \quad \text{subject to } y_i(w \cdot x_i - b) \geq 1$$

where w represents the normal vector of the hyperplane separating the two classes. There are many ways of handling more than two classes, such as a "one-vs-all" or a "one-vs-one" method. The `scikit-learn` algorithm we are using implements the "one-vs-one" approach described by Knerr *et al.* [6]

We don't expect to get very good results here either as this dataset is likely not linearly separable, as mentioned with the neural networks. A richer feature set will definitely help reduce the error rate though, but not by a large margin, as it faces the same problems as the neural network, where it can't analyze GPS coordinates very well.

5. OPTIMIZATION

5.1 Logistic Regression

Logistic regression is implemented using Newton-Rapshon gradient descent. The Newton-Raphson method is one of the most powerful optimization techniques which have been used successfully with logistic regressions cross entropy minimization [4]. In order to avoid over-fitting, an L1-norm regularization term has been added to the cost function. We used a "one-against-all" strategy for making the logistic regression work on a multi-class problem. Although a logistic regression can be considered as a neural network without any hidden layer, cost function for the fully connected neural network is the sum of square differences, whereas the cross entropy is minimized in the case of logistic regression.

5.2 Random Forest

As the random forest algorithm chooses a random subsets of data and chooses the decisions based on a fixed formula, there isn't much optimization to be done. The only hyper-parameter to be chosen is the number of trees to create. A too large number of trees will over

t the training data as it will find the perfect way to split features and predict the training set targets, otherwise, all optimization is done by choosing good decisions, which again, is chosen automatically.

5.3 Feed-forward Neural Network

The neural network is trained using the back propagation gradient descent, where the gradients are calculated from stochastic gradient descent. The cost function is the cross-entropy, where the cost is high for false positives and false negatives. In the stochastic strategy the network weights are updated each time a new sample reaches the network.

An epoch, in the training procedure, means passing through all the samples in the training set once. A heuristic approach has been taken toward assigning the learning rate: the learning-rate will be divided every time the cross-validation accuracy seems to stagnate. However, this decreasing rate will stop when the learning rate reaches the minimum amount of 0.01. That is, the algorithm begins with a large learning to converge faster and avoid local minima, and as we progress, the learning rate will be decreased to avoid divergence or oscillation around the expected minimum. As for the initial value of learning rate, it is assigned by cross validation. A few things to note about training neural networks is that the error rate quickly drops (with a high enough initial learning rate), then takes quite a few epochs before converging. Also, as accuracy on the training set keeps increasing, accuracy on the cross-validation set starts to drop after a certain time. This is due to overfitting, where the neural network keeps learning to predict the training set, but starts to learn features that are specific to the training set data alone, and not general ones. A good time to choose to stop training is when the cross-validation accuracy stops increasing. At this point, we will lose the general prediction ability of the classifier if we keep going.

5.4 Support Vector Machines

There wasn't any optimization required for any of SVM classifiers, as the `scikit-learn` library already provides great default values. However, there are a few hyper-parameters that we will discuss in the next section that show that different SVMs can take certain costs into account to deal with data in specific ways. This doesn't change that optimization is fixed, as the cost function is convex.

6. PARAMETER SELECTION

6.1 Logistic Regression

The primary parameter to be selected for Logistic Regression is the the inverse of regularization strength C. This parameter essentially controls how much to penalize the weights. Smaller values of C specify stronger regularization. A sweep of the values of C is shown in the Figure 3, and shows that the optimal value of C is 1.

6.2 Random Forest

As mentioned in the previous section, there only exists one hyper-parameter, which is the number of trees to create. We tried a quite few number of trees through cross-validation where we found out that performance began converging at around 80 trees, but accuracy kept slightly increasing until we got to 1000 trees, at which accuracy started dropping. From the Figure, we can see that performance at 2000 trees was lower than at 1000, which shows that overfitting had started (Figure 4). We also see that the training accuracy is

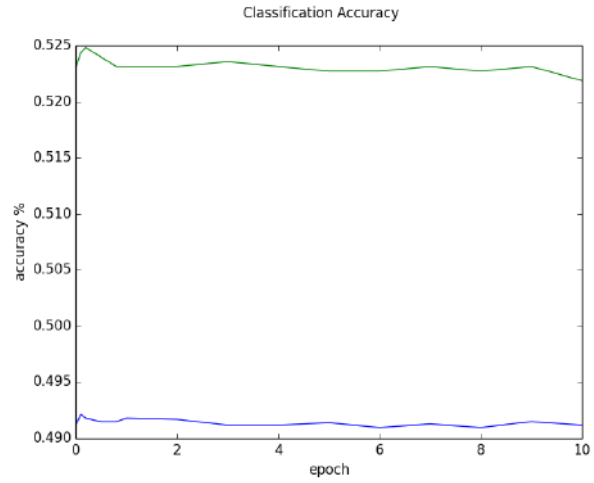

Figure 3: Logistic Regression - Train and validation accuracy for different values of lambda (regularization factor) in green and blue respectively.

almost 100% from the very beginning, which is completely normal as the random forest fits itself to the training data from the very beginning, while making the decisions.

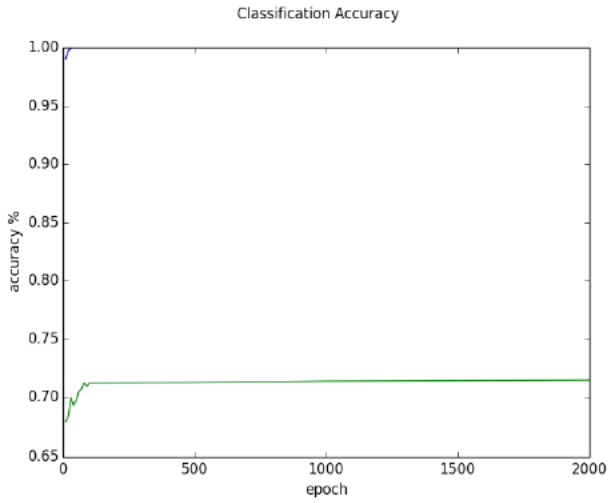

Figure 4: Random Forest - Train and validation accuracy for different number of epochs in blue and green respectively

6.3 Feed-forward Neural Network

In the optimization procedure, we mention that training accuracy increases constantly, while validation accuracy rises and then falls where the classifier starts to overfit on the training set. To optimize the test results, as validation starts to slow, then we stop training the classifier and take the weights as they are. We train our network using 70% of the data as training set, validate the network using another 20%, and at last test the final network using the remaining 1%.

The structure of the neural net was determined using cross validation. The neuron count in the first layer is determined by the number of features (input layer), and the number of neurons in the output layer is equal to the number of classes. Therefore, the only parameter to optimize is the number of hidden layers and their corresponding neuron counts. First, we begin by increasing the number of neurons in the first hidden layer from 10 to 60 neurons, with a step size of 10 neurons. Considering the fact that we only have up to 18 features, we decided to not to go further than 60 neurons in the first hidden layer. The cross validation is done by waiting until cross-validation accuracy starts to drop and the initial value of learning rate is set to 0.1. Continuous rising pattern have been observed by increasing the number of neurons. However, only slight efficiency difference was observed between from 40 neurons and up. Therefore, we fixed our first hidden layer units to 40 to avoid additional computation burden. Then, another layer is added to the network with both 30 and 40 neurons. The results showed that adding the second layer had no positive impact on the performance of the network compared to only single layer network.

6.4 Support Vector Machines

We tried using different C and gamma values on the “rbf”, “poly” and “sigmoid” kernels. Intuitively, the gamma parameter defines how far the in influence of a single training example reaches, with low values meaning that single training examples are meaningless, and the model tries to be as general as possible. The C parameter trades off misclassification of training examples against simplicity of the decision surface. A low C makes the decision surface smooth, while a high C aims at classifying all training examples correctly.

7. TESTING AND VALIDATION

One major issue that we faced was having unbalanced data. For all classifiers, we tried using them on both balanced and unbalanced data. The issue however, was that balancing was done by decreasing the size of the popular classes. This decreased the amount of training data available, which then affected performance, as the classifiers didn’t have enough data to generalize well.

7.1 Logistic Regression

Using 20% of the dataset as cross validation, we have reached accuracy of 52.5% percent for the baseline method, with 0.2 as the optimal L2 coefficient parameter. Figure 5 shows the confusion matrix for the logistic regression classifier, the matrix has been normalized row-wise to account for between class variability. The results show that the algorithm achieves its best performance for the class 2, which is “Car”. However, other classes show very poor results. The higher performance for the car class could be due to some features that are quite distinguishable from the others, such as the high speed.

7.2 Random Forest

We reached the highest accuracy rates using random forests. Again, using 20% of the dataset as cross validation and using the optimal number of trees, we got a performance of 71.7%.

Figure 6 shows the confusion matrix for the random forest. We can also see that most classes are often confused for class 3, which is the ’Public Transport’ class. This is due to the fact that the dataset is unbalanced and has many more points in the ’Public Transport’ class than the others. This lack of balance favours the class as a target, and therefore the classifier guesses it more often than not. We then tried balancing the data by decreasing the number of elements from each class except for the least common one, so that they all had the same number of instances. As we can see from Figure 7, the guesses are more balances, but the accuracy dropped. As mentioned earlier, we believe that this is due to the lower amount of data, which hurts the classifier’s ability to generalize well.

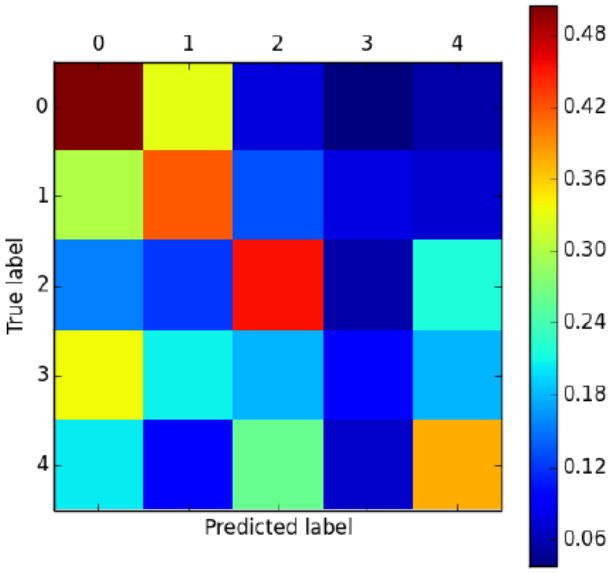

Figure 5: Logistic Regression - Confusion Matrix with balanced data

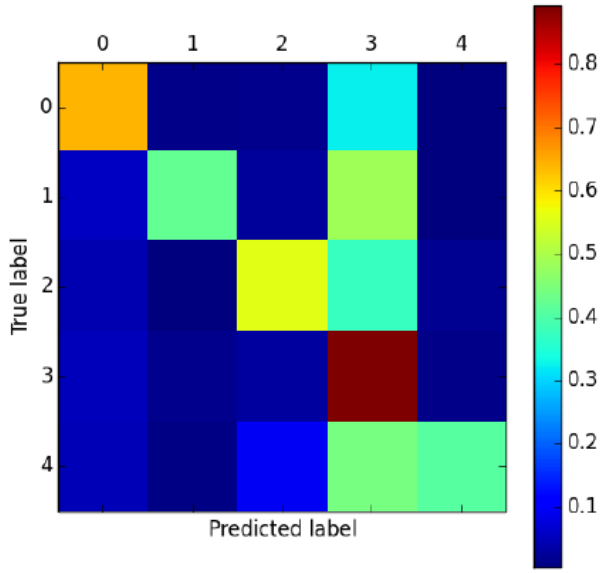

Figure 6: Random forest - Confusion Matrix with unbalanced data

7.3 Feed-forward Neural Network

The network was trained using 70% of the data and 20% as cross-validation. The optimum network has 50 hidden neurons in the first layer, and learning rate initialized at 0.1. Accuracy on the test set reaches 54% which is slightly higher than the accuracy reported for both the linear classifier and the SVM. The confusion matrix shows that the neural network didn't perform extremely well in any class, which we believe is because of the special properties of the data. Larger values in GPS coordinates do not represent any kind of increase in score. As mentioned earlier, this type of data needs to be segmented, which the neural does not do. This means that valuable columns are lost, as these will be deemed irrelevant, and the neural network will train using other, more linear, features.

7.4 Support Vector Machines

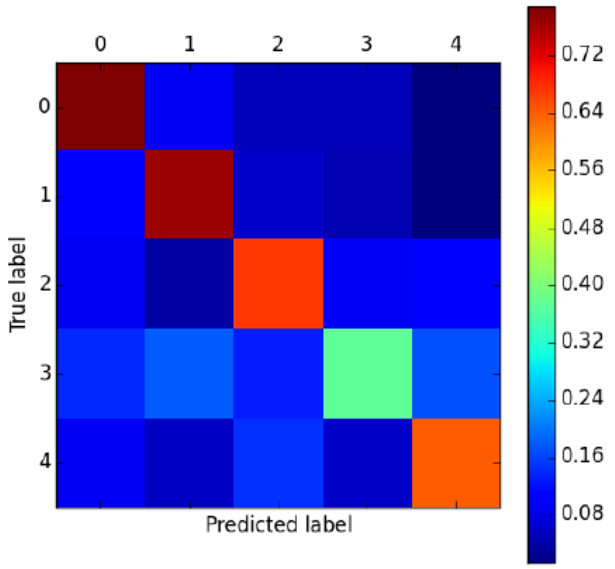

Figure 7: Random forest - Confusion Matrix with balanced data

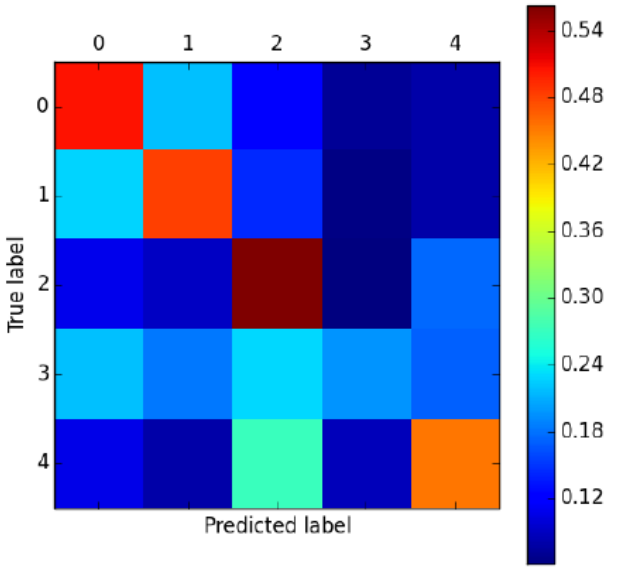

Figure 8: Feed-forward Neural network - Confusion Matrix with balanced data

Like the other classifiers, our SVMs were trained using 20% of the data as cross-validation. The higher percent accuracy observed was 52.7%, which was a result from using the "RBF" kernel with the hyper parameters decided from the best cross-validation result. We use cross-validation to find the a gamma of 1 and C of 0.5 were optimal. As with all other classifiers, the SVM was greatly affected by the unbalanced data. This resulted in the SVM classifying most data as "Public Transport", which was by far the most popular class. Since the class was very popular, guessing it as a target gave good, but very biased results. Again, we tried balancing the data, but then too much valuable information was lost, causing the algorithm to lose its ability to predict well.

8. DISCUSSION

In this study, we compared multiple classifiers to predict what mode of transportation was used, based on GPS and environment data. We believe that random forests would outperform all other classifiers, as they can efficiently segment location data into areas of the city. Furthermore, differences between the transportation methods should have clear separations, making the random forest a good choice to learn where to divide the classes. As expected, this classifier did outperform the others and by a large margin of around 20%.

One hurdle that proved hurtful was the lack of data in the dataset. The initial data which was composed of individual GPS and speed data at given timestamps was condensed into entire trips. Then, the surplus of certain classes created heavy biases for the classifiers. Tackling this issue meant to further decrease the number of data points to then balance it. However, this was very costly as data became lacking and made the classifiers lose generalization.

We have tested many other classification techniques such as logistic regression, feed-forward neural nets and SVM with many kernels. However, the accuracies were limited between 40-55%, with feed-forward neural nets with best parameter being the strongest at around 54 percent accuracy, and SVM with RBF kernel using the best parameter the weakest method with 52.7% accuracy.

At first, the dataset was comprised of pure GPS data, combining location and speed. As the classifiers lacked in accuracy, we augmented the data using weather and gaseous pollutants data from the OpenMontreal dataset. This information could be seen as valuable as people frequently change modes of transportation when certain situations arise. During heavy rain, cyclists and walkers will usually take public transport. Once the data was augmented, we saw an increase in accuracy of approximately 3%.

We have shown the confusion matrix for each of the discussed methods Figures 5 and Figure 8. The matrices are normalized row-wise to account for classes of different size. The matrices show the lack of balance caused heavy biases in most classifiers, but that random forests were least affected. Once balanced was restored, the classifiers gave more general predictions, but lost accuracy as the dataset was then too small.

As mentioned, the random forest performed the best amongst all the classifiers we tested. With the full methodology described in the previous sections, we were able to reach an accuracy of 72.7%, which is satisfactory, as the dataset was quite limited. In term of usage, it was also one of the simplest one, as the `scikit-learn` library has very simple functions, fit and score, which do most of the work. The only hyper-parameter to choose was the number of trees to build, which is also easy to find using cross-validation.

We hereby state that all the work presented in this report is that of the authors.

9. REFERENCES

[1] Gaseous pollutants data. http://faculty.concordia.ca/patterson/trip/, Accessed: December 2014.

[2] Montreal opendata. http://donnees.ville.montreal.qc.ca/dataset/rsqa-polluants-gazeux, Accessed: December 2014.

[3] Weather network. http://www.theweathernetwork.com/, Accessed: December 2014.

[4] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[5] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, Mar. 2003.

[6] S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In F. SouliÃl' and J. HÃl'rault, editors, *Neurocomputing*, volume 68 of *NATO ASI Series*, pages 41–50. Springer Berlin Heidelberg, 1990.

[7] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97(1-2):273–324, Dec. 1997.

[8] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] L. Zhang, S. Dalyot, D. Eggert, and M. Sester. Multi-stage approach to travel-mode segmentation and classification of gps traces. In *ISPRS Workshop on Geospatial Data Infrastructure: from data acquisition and updating to smarter services*, 2011.