**Predicting Traffic Congestion in Sao Paulo**

Tianhua Zhao, Qiaofang Deng, Ryan Eakins, Yijing Wang

**Abstract:**

We used various machine learning techniques and algorithms on a sample data set in order to determine relative performance on the data. With the goal of determining the best solution for this type of data. Naive Bayes, KNN, Linear Regression, SVM, decision tree, Adaboost and Neural Nets were all used to create the comparisons. With the conclusion that Linear Regression was best for this dataset.

**Keywords:**

Naive Bayes Regression Neural Networks Decision Tree Adaboost

**I Introduction**

In recent years, the behavior of the parcel delivery market in Sao Paulo has undergone significant changes due to many factors, one of them is the increase of the urban flow. The aim of this paper is to apply several machine learning algorithms to predict slowness from attributes such as 'time', 'Immobilized bus', 'Broken Truck', 'Vehicle excess', 'Accident victim', 'Running over', 'Fire vehicles', 'Occurrence involving freight', 'Incident involving dangerous freight', 'Lack of electricity', 'Fire', 'Point of flooding', 'Manifestations', 'Defect in the network of trolleybuses', 'Tree on the road', and 'Semaphore off', 'Intermittent Semaphore'. The methodology of the paper consists of Naive Bayes, KNN, Linear Regression, SVM, and Neural Networks trained on these features of notable occurrences to predict the increased slowness in traffic.

We expect that classification will perform better than regression when applied to this dataset because the target has range of possible values. Also, the group that collected the data used classification with a neural net and achieved high accuracy over the data.

Then, for the classification algorithms, we expect that the Neural Net or SVM algorithm will give us the most accurate prediction for this dataset because the information gain seems to be high for each attribute. On the other hand KNN might not perform well because two independent features could lead to traffic but those points might not be spatially near each other.

When determining the number of classes that results in the most accuracy we expect that the fewer classes we discretize into the more accurate we will be. Because one class is trivial to classify and will always have zero error, while given many classes the chance of randomly selecting the correct one decreases with the number of classes. However fewer classes means we are learning less about the data so even as test error is reduced it might not be optimal.

We believe that the conclusion of this paper will help delivery companies in Sao Paulo predict the most accurate traffic slowness.

**II Background**

Literature review:

Article title: How (not) to use Machine Learning for time series forecasting: Avoiding the pitfalls

This article talked about some of the common pitfalls of machine learning for time series forecasting. The author pointed out a common mistake in choosing accuracy metrics that results in high prediction accuracy while the model actually has almost no predictive power whatsoever.

People commonly use machine learning algorithms like linear regression or LSTM Network for time series forecasting. Time series data tend to have a high correlation in time and exhibit a significant autocorrelation. Data value at t+1 is likely to be close to the value at time t. Therefore, if we know the target value at time t, we can predict the value at t+1 well. It is not surprising that common error metrics such as mean squared error or R2 score would indicate a high prediction accuracy. Despite having a high prediction accuracy, most of the times, the model is not helpful to us at all. The author demonstrated this with a data set that is generated through a random walk process, which means that we cannot possibly predict if the value at time t+1 would go up or down compared to at time t. The stock market is also a good example. If we build a model for the stock price, a good R2 score or MSE simply means that we could predict the stock price (t+1 is likely to be close to t). However, the model doesn't predict the return. In fact, for a random walk process, it is not possible to predict the return.

So, when predicting time series data, we can not claim to have a good model simply because we got a good mean squared error or R2 score from the model. The model could have no predictive power in practice. We must be very careful when evaluating our model performance.

**III Dataset**

From the UCI Sao Paulo traffic data set the samples were extracted from the csv file. The data was collected at an intersection every hour for 5 days recording things like power outages, stalled trucks, and the target slowness in traffic as a percent. This led to a total sample size of 135 with each sample having 18 features with one being the target value.

**IV Method**

* Step 1: Modify data

The original data which we downloaded from the dataset website is all in one cell and the value of slowness is incorrect. Therefore, we separate them one by one and replaced the correct value of the attribute slowness.

* Step 2: Create a constant number named "TIMES"

We created this to be the number of times to run the experiment because the random data selection of training and testing. Every time running the code, the avg\_mean\_squared\_error for each algorithm is always different, sometimes the Regression gets the least error but sometimes Adaboost does. So, for accuracy, we run the code 100 times to get the average error from them and this behavior can make our result more representative of a statistical average than a possible one time anomaly.

* Step 3: Clean data

Clean data is important because it ensures that we can get the best model of each algorithm. For this case, we print the data info to make sure there are not any missing values. Then we used a heatmap to plot our dataset and pairplot to get a relationship between each of the samples. The heatmap represents the strength of interdependence of each sample with lighter colors representing less independence. Pairplot tells us if the attribute is linear or non-linear for the slowness. After analyzing we determine the data is independent and linearly related..

* Step 4: Transform data to metrics and separate them into a training and test set

We transformed the slowness from a float value to integers by multiplying them by 10 to eliminate the decimal and avoid input error for the KNN algorithm. Besides the real number y values, we also transformed y to a binary series(1 if greater than median, 0 if less than median). Both real number y and binary y are used to evaluate test errors. Then, we randomly ordered the whole data and choose 80% of them as training data and 20% of them as test data.

* Step 5: Run all algorithms 100 times (each time with a random selection of training data and testing data)
  + Algo 1: Get mean\_squared\_error by predicting test data by using Naive Bayes (including GaussianNB, BernoulliNB and MultinomialNB)
  + Algo 2: Get mean\_squared\_error by predicting test data by using KNN (including KNN = from 1 to 5)
  + Algo 3: Get mean\_squared\_error by predicting test data by using Linear Regression
  + Algo 4: Get mean\_squared\_error by predicting test data by using SVM (including kernel = linear and RBF)
  + Algo 5: Get mean\_squared\_error by predicting test data by using MLP (including hidden layer = one and two)

The explanation of parameter selection:

A neural net was used to compare the performance of discretization levels and neural net shapes. The target values were originally continuous values of traffic slowness in units of percent. They were discretized into two sets of targets; one for the nearest percent and one into a binary target with 0 less than 8 percent and 1 more than 8 percent. Then tree models were trained on a portion of the data set with each of the targets. The models were a NN with 10 neurons in the hidden layer, a NN with 40 neurons in the hidden layer, and a NN with two hidden layers of size 17 then 10. As would be expected the binary classification performed the best on all test sets. However unexpectedly the NN with 40 hidden neurons performed the worst for its targets on both sets.

* + Algo 6: Get mean\_squared\_error by predicting testing data by using Decision Tree

Tuning: by trying different max depth, we found that depth 10 is good.

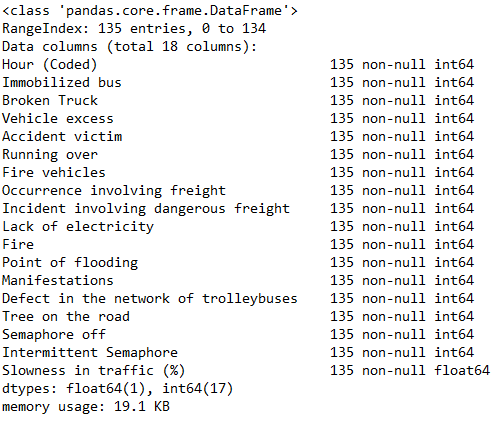
* + Algo 7: Get mean\_squared\_error by predicting testing data by using AdaBoost

Tuning: by trying different n estimators, we found that 50 is a good value.

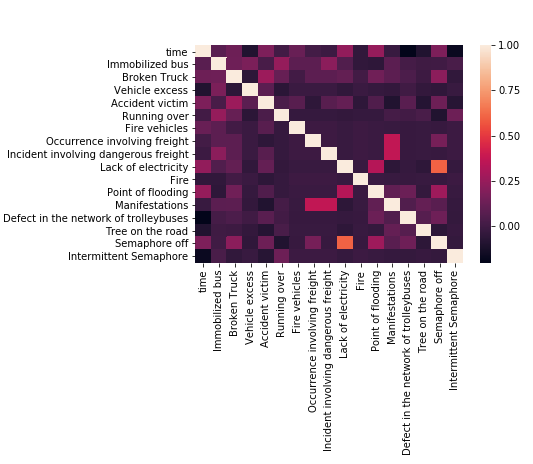
* Step 6: Print the result of average mean squared error after running 100 times

**V Results**

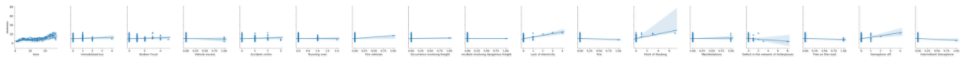
* Check the validation of the data including missing values, regression relationship and dependent relationship
  + Data info



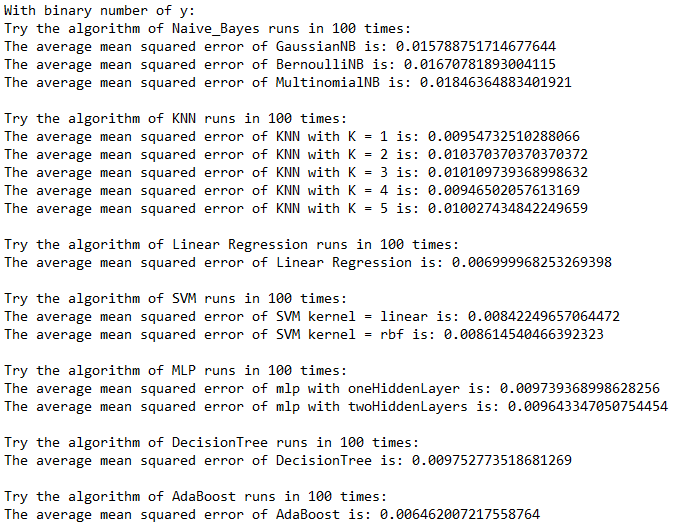
* + Dependent relationship:

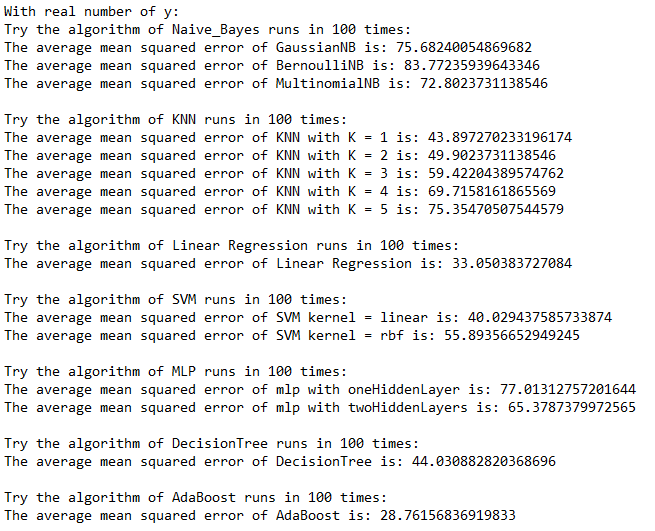


* + Regression relationship:



* The results of each algorithm running 100 times





**VI Conclusion**

We observed that the Linear Regression algorithm got the least error and is the best model for predicting the slowness with given attributes of the data. For Naive Bayes see empirically that using the Gaussian distribution resulted in the lowest error when compared to bernoulli and multinomial, likely because the slowness was fairly normally distributed over the length of a day. With KNN the error was always similar, this could be because the features are largely independent and most are binary. When a feature has a value there is usually traffic so checking the nearest neighbor or 5 nearest when there is an event doesn't really matter they all lead to traffic. Along with AdaBoost Linear Regression gave the lowest error, this makes sense because the target values were originally continuous and roughly linear with time. Similarly AdaBoost performs well because it can easily classify much of the traffic with one weak classifier then others can use other features that are less common to get more accuracy. Svm also performed alright because we would expect traffic is linearly separable along there being a traffic incident however it losses some accuracy because the exact percentage of traffic is likely not separable. MLP and Decision tree performed better than Naive Bayes but otherwise among the worst because this data is not suited to rough classification as it was originally continuous. From these results we can conclude when working with continuous target data regression is the best model even when trying to discretize the targets into classes.

**Reference**

[1] Ricardo Pinto Ferreira ¹, Carlos Affonso ² and Renato José Sassi ³ , 10th Brazilian Congress on Computational Intelligence (CBIC’2011), November 8 to 11, 2011, Fortaleza, Ceará Brazil

[2] “How (not) to use Machine Learning for time series forecasting: Avoiding the pitfalls” https://towardsdatascience.com/how-not-to-use-machine-learning-for-time-series-forecasting-avoiding-the-pitfalls-19f9d7adf424