Final Project

Traffic Prediction From Weather On I-94 in Minnesota

Course:	EECE5644
Professor:	Deniz Erdogmus
Names:	Rickard Stureborg
	Jeff Weintraub
	Nicholas Fresneda
Date:	June 19th, 2019

Abstract

In our final project, we worked on developing and comparing machine learning models for predicting traffic volume, given inputs of weather, time, temperature, and holiday timing data. We cleaned a large dataset in R and then implemented two versions of machine learning models; an SVM using python's scikit-learn library, and a neural network using python's keras library.

Data Cleaning

Before we could create any machine learning models from our dataset, we first had to edit the data such that it would be useful for the model, and our language of choice was R, as it is useful for this type of manipulation and visualization. After importing the data file to a data frame, it was examined by hand to see if there were any extreme outliers or erroneous data. There is a considerable amount of bad data in the set to begin with, including temperature readings of absolute zero, unrealistic rainfall and snowfall amounts, and duplicated time values. Code was written to remove all of these things from the dataset, although only complete duplicate rows were eliminated rather than all rows with duplicate time values, as there was no way to determine which of the two or more rows is correct. Once the data was prepped for editing, the following libraries were used in order to properly manipulate the data: ggplot2 (plotting), dplyr and plyr (general data manipulation), and lubridate (date and time manipulation).

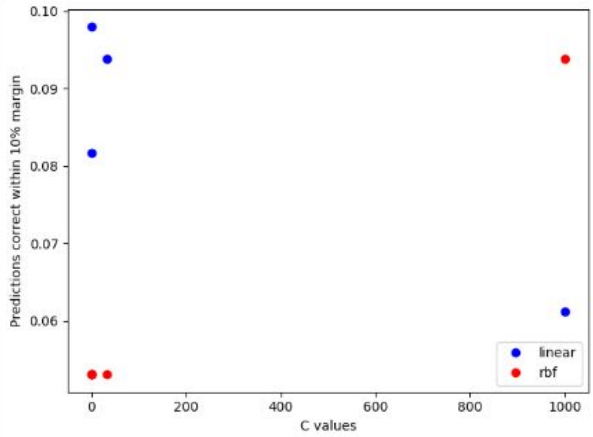
The first action taken was to fix the date and time column such that it could be converted into a numerical representation of the two. Once this was established, new columns representing the hour, day of the week, month, and year were all recorded as new columns, the first three being for input into the ML algo, and the last being for separating data. After plotting seven 1D scatterplots, each for every day of the week, it was determined that weekends are distinct from weekdays in terms of traffic but otherwise there was no difference between the two. Next, the issue of holidays was dealt with by plotting their traffic volume for all of the holidays on separate 1D scatter plots plotted next to each other such that visual groupings could be made for the data, if a holiday experienced traffic similar to a normal day, it was grouped in with normal days, and if a holiday experienced traffic far less than a normal day it was marked as such, such that a row's holiday column was either a 0 (normal day or high traffic holiday) or a 1 (low traffic holiday). While visual grouping isn't always a good thing it worked out well here due to the distinct nature of the holidays in our dataset, and some simple attempts at grouping statistically (testing for if a holiday's 95th percentile is less than the 68th percentile of a normal day).

Finally, the weather_main column was split into multiple columns, each a boolean representing if the day was a certain weather condition or not, as categorical data did not seem to work well with our models. Finally, all of the numeric columns were normalized and the data was exported to two CSV files, one with the original data and the other with the normalized data.

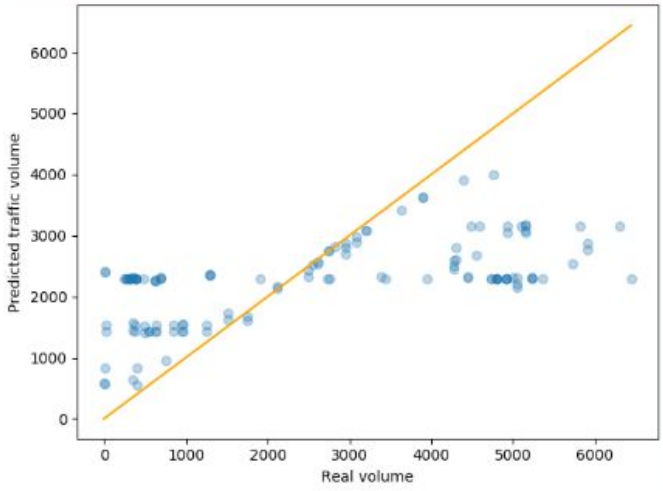
SVM Implementation and Results

For the SVM implementation, we used scikit-learn's SVR support vector regression framework. We implemented 5-Fold cross validation on the C parameter (penalty of errors) and the kernel function used (we compared the 'linear' and 'rbf' functions).

For validation, training, and testing, we picked 200 random samples from the dataset. The SVR sklearn implementation scales quadratically with the number of samples, making larger sample sizes infeasible for our project time frame. Because of this, at least in terms of efficiency, support vector regression would not be an ideal machine learning model for this data set. Below are the results of our 5 fold cross validation (which was limited because of the time complexity of the SVR). The 'linear' kernel function combined with a C error penalty of 31.6 proved to be the best hyper parameters for this model.



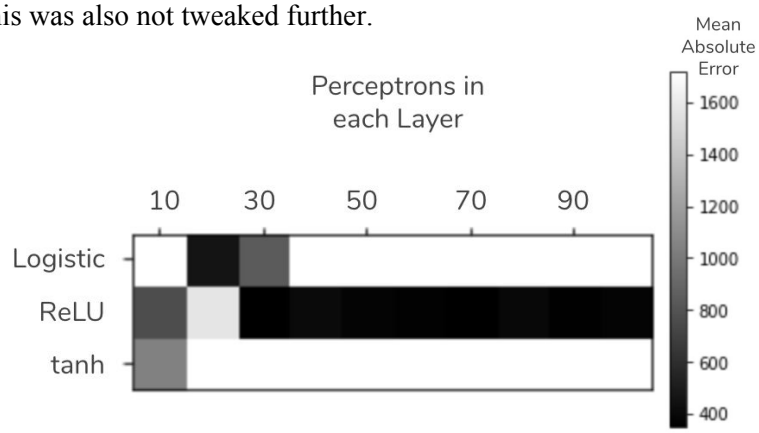
Below is a graph of the fitness of our final results. The x axis corresponds to actual traffic volume and y axis refers to the SVR predicted traffic volume. The orange line represents the results of an ideal perfect predictor (where predicted is equal to real traffic volume).



Multi-Layer Perceptron - Implementation and Result

Due to the complexity of the relationship between both the time-related features and weather-related features in our dataset to their corresponding labels, the model attempted above may not be suitable for this dataset. A more flexible model which can handle lots of complexity in the underlying data is a Neural Network. Specifically, a multi-layer perceptron with many hidden nodes can capture what other structural machine learning models can't.

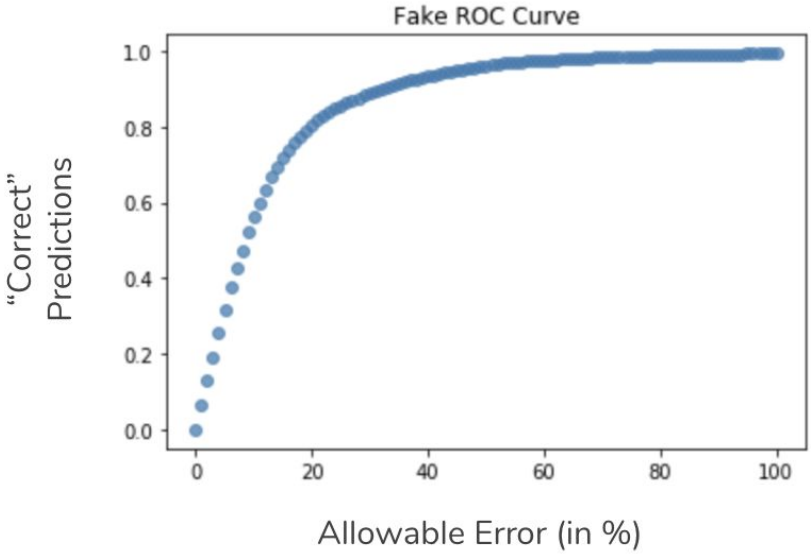
The package we used to build and train the neural net was sklearn's MLP Regressor. This package offers plenty of hyperparameters for tuning but abstracts a healthy amount of work in order to reduce the time it takes to code. We chose to use the Adam solver, since it is generally accepted as state-of-the-art and very robust for many different applications since it is essentially an extension to momentum and stochastic gradient descent. Based on this, the first hyperparameter we tuned for was the learning rate. We found a learning rate of 0.01 avoided issues with the error not converging during training and for this reason we used it for the rest of the models trained. Further, we found that there was little improvement in adding more than three hidden layers and therefore this was also not tweaked further.



We then decided to run grid-search on two more hyperparameters: Perceptrons in each layer and the activation functions. The result of these experiments can be seen on the right. Through limiting the amount of hyperparameters being tuned during this time, our computational expense was significantly reduced. One thing to note is that we did not use k-fold cross validation as it is much more computationally heavy. Instead, we split the data into training and validation data, training the data on the former and testing each individual hyperparameter set on the latter. Only during the final evaluation of the model would we use a third set of data, the testing set. This ensures we avoid any overfitting, both when training and when tuning. It may not be as effective as cross validation but it allows us to tune for more hyperparameters. The results showed 70 perceptrons with ReLU performed the best.

In order to evaluate this model, it is important to remember that it is a regression-based neural network and therefore is not aiming to classify traffic volume into "bins." One could design a problem such as this by classifying "high," "medium," or "low" traffic rather than exact values, but we thought this approach would be more advantageous. Since it is not a classification problem, an ROC curve doesn't make sense, but we can create an alike curve which is inspired by the measures an ROC curve attempts to show. In a traditional ROC curve the y-axis shows the true positive rate while the x-axis shows the false positive rate. That is, the rate of how many positive classifications were correct over the rate of how many positive classifications were incorrect. Instead, we can define an allowable margin of error for the prediction made by the model, and compare this to the rate of correct predictions the model makes with such a threshold.

The graph below shows exactly this:



Much like an ROC curve, a perfect model would have these points in the top left corner: with 100% accuracy on no allowable error (exact predictions necessary). Our model performs quite well here, and we can deduce that at 10% allowable error in predictions, the model predicts the correct traffic volume 58% of the time. At 20% allowable error, the model predicts the correct traffic volume even better, at 82%.

Conclusion

There is a distinct and predictable relationship between the traffic volume and weather and time. This relationship is complex and may not be best explained by a support-vector machine, but rather by models that can handle more complex relationships between data, such as multi-layer perceptrons. Even with fairly limited data pre-processing and hyperparameter tuning of a neural network, the results showed an impressive accuracy for predicting traffic volume.

If we were to have further resources and time devoted to this project, we would likely spend additional time on feature selection. Deciding which features to include in the model, and constructing cross-features through data mining techniques may prove to help a model find even better relationships in the data that aren't immediately apparent. Further, we would have liked to spend more time with hyperparameter tuning of these models since computational time limited the hyperparameters we could tune and the combinations we could test. With these two extensions we are confident a model could perform even better on this dataset.