Predicting Bike Usage in (Metro) Washington, D.C.

Anna M. Kot
M.S. Candidate in Analytics
Department of Analytics & Data Science
University of New Hampshire
Durham, NH 03824
amk1073@wildcats.unh.edu

This report was prepared for Data 802: Analytical Tools and Foundations – July 30, 2018.

I. Introduction

In this study, we investigate the use of several different learning algorithms, namely k-nearest neighbors, logistic regression, multiple linear regression, and naïve Bayes to predict bike usage in the (metro) Washington, D.C. and determine what, if any, are optimal conditions for maximum and/or minimum bike usage in (metro) Washington, D.C.

The original dataset consists of roughly 21,700 response variables. An analysis of the data is done, and a vector of features is collected and analyzed in Python. The features will be discussed in detail in a following section.

II. Data Collection

The Capital Bikeshare data used was obtained from Capital Bikeshare self-reported system data[1,2] in accordance with the Capital Bikeshare Data License Agreement. To reduce the number of outliers, Capital Bikeshare has processed the data to remove 1) trips that are taken by staff as they service and inspect the system, 2) trips that are taken to or from any of Capital Bikeshare "test" stations at Capital Bikeshare warehouses, and 3) any trips lasting less than 60 seconds (potentially false starts or users trying to re-dock a bike to ensure it is secure).

The weather data was obtained from an unknown third-party, containing hourly weather information for the (metro) Washington, D.C. area for the last five (5) years.

The Capital Bikeshare data and weather data was merged and subsequently cleaned to reflect two response variables: 'Total Count', a numerical variable denoting the total number of bikes rented per hour for each day, month, year combination and 'Bin Count', a categorical variable classifying the number of Capital Bikeshare bikes rented in (metro) Washington, D.C. into categories of volume: 'Low' and 'High'. Additionally, a 'Holiday' feature was created to denote if a specific day is considered a Federal U.S. holiday, temperature variables were converted from Kelvin to Fahrenheit, and the wind speed variable was converted from meters per second to miles per hour.

III. Features

The features that have been considered for the classification and regression algorithms are as follows:

- **Humidity**: Current humidity percentage
- **Rain (1H):** Rain volume for the last hour
- **Rain (3H):** Rain volume for the last three (3) hours
- **Snow (1H):** Snow volume for the last hour
- **Cloudiness:** Current cloudiness percentage
- **Weather (ID):** Categorial weather condition
- **Weather (Main):** Categorical weather parameter (e.g., rain, snow, extreme, etc.)
- **Weather (Description):** Categorical weather condition within the group (e.g., light rain, heavy rain, etc.)
- **Weather (Icon):** Categorial weather icon ID
- **Temperature (F):** Current temperature in degrees Fahrenheit (converted from Kelvin)
- **Minimum Temperature (F):** Minimum temperature in degrees Fahrenheit (converted from Kelvin). This is a deviation from temperature that is possible for large cities and megalopolises geographically expanded
- **Maximum Temperature (F):** Maximum temperature in degrees Fahrenheit (converted from Kelvin). This is a deviation from temperature that is possible for large cities and megalopolises geographically expanded
- **Wind (Degrees):** Wind direction in degrees
- **Wind (MPH):** Wind speed in miles per hour (converted from meters per second)
- **Year:** 2016 and 2017
- **Month:** Respective month value (i.e., 1-12)
- **Day:** Respective day value (i.e., 1-31)
- **Hour:** 24-hour cycle (i.e., 0-23)
- **Holiday:** Federal U.S. holiday classifier (True/False)

All features, unless otherwise noted, are numeric.

IV. Data Separability

The features in the previous section were further analyzed to account for missing data, superfluous data, and redundancy. As the features 'Rain (1H)', 'Rain (3H)', 'Snow (1H)' were missing data for more than half of the dataset, they were removed from the analysis. Subsequently, 'Weather (Icon)' was also removed as it was considered a superfluous feature.

With the four features removed, our dataset of 21,715 response variables remained intact. In reviewing the count of missing data rows with the prior mentioned features removed, 942 rows of data were up for potential removal. As this is a small portion of the data, the 942 rows containing missing data were removed, resulting in a dataset with 20,773 response variables.

A heatmap showing the correlation between features identified the three (3) features denoting temperature values to be highly correlated. To reduce redundancy, 'Minimum Temperature (F)' and 'Maximum Temperature (F)' were removed from the analysis.

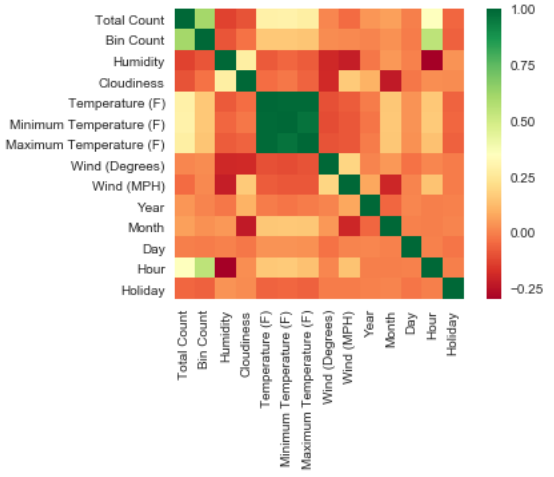

Fig. 1: Heatmap showing the correlation between features.

A subsequent heatmap showing the correlation between the remaining features was generated and is illustrated in Fig. 2.

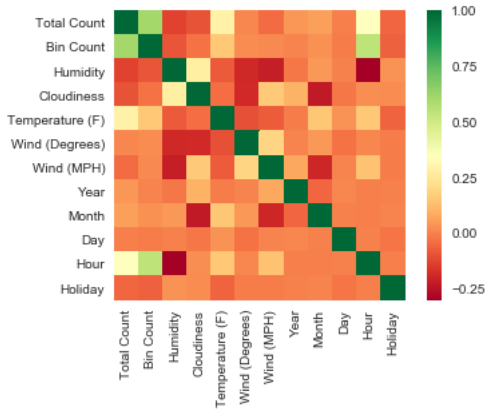

Fig. 2: Heatmap showing the correlation between remaining features.

Lastly, in order to ensure cyclical features, such as 'Hour' and 'Month', remain cyclical in the analysis, the features were replaced with their sine and cosine transformations and relabeled as 'Time' and 'Month', respectively. The cyclical nature of the features is illustrated in Fig. 3 and Fig 4.

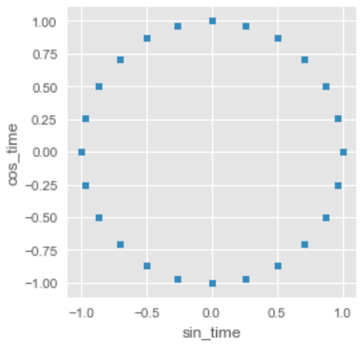

Fig. 3: Scatterplot showing the cyclical feature of 'Time'.

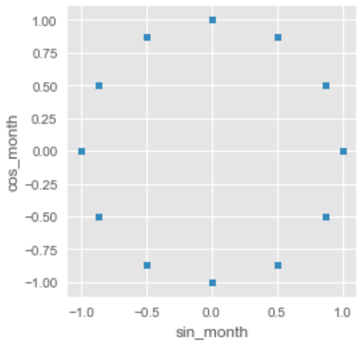

Fig. 4: Scatterplot showing the cyclical feature of 'Month'.

V. Choosing a Learning Algorithm

Now that we have a set of features to test on, it is time to pick a learning algorithm for regression and classification, respectively: k-nearest neighbors, logistic regression, multiple linear regression, and naïve Bayes classification.

In choosing a learning algorithm, we first defined our feature and response variables. The remaining eight (8) features: 'Humidity', 'Wind (Degrees)', 'Year', 'Month', 'Day', 'Time', 'Temperature (F)' and 'Wind (MPH)' were used as our predictor variables and 'Total Count' and 'Bin Count', converted to reflect 'Low' as '0' and 'High' as '1', were used as our response variables, each respectively. Second, we split our data into a training set and testing set using a ratio of 75:25. For our logistic regression and naïve Bayes algorithms, we also stratified the split to ensure equal representation of the response variable. Lastly, to prevent features on far larger scales from unduly influencing our algorithms, we scaled our training set and testing set features, respectively, such that the range is between 0 and 1.

The k-nearest neighbors' algorithm, or KNN, essentially predicts the label of a data point by looking at the 'k' closest data points (by way of a distance function) and taking a majority vote.

Prior to running our algorithm, we first determine what value of 'k' will generate our most accurate model. To determine 'k', we used two methods: 1) a model complexity curve and 2) an exhaustive and computationally expensive k-fold cross-validation. The model complexity curve, as illustrated in Fig. 1, estimated our most accurate prediction to be at 'k' = 1. The result of our k-fold cross-validation (number of folds = 5) was also estimated to be at ‘k’ = 1.

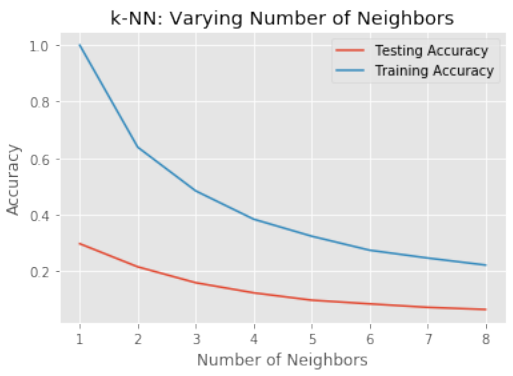

Fig. 5: Model complexity curve for k-nearest neighbors.

With ‘k’ = 1, using the Minkowski distance measure, we ran our k-nearest neighbors’ algorithm on our training set and tested the response on our testing set, using ‘Total Count’ as our response variable. Our model resulted in an accuracy score of 0.30, as denoted in Table I.

The logistic regression algorithm, despite its name, is used in classification problems, not regression problems. Essentially, given a feature, logistic regression will output a probability with respect to a binary response variable. If the probability is greater than 0.5, the algorithm labels the data as ‘1’. If the probability is less than 0.5, the algorithm labels the data as ‘0’.

Prior to running our algorithm, we first determine what inverse regularization strength value will generate our most accurate model. To determine the inverse regularization strength, we used an exhaustive and computationally expensive k-fold cross-validation. The result of our k-fold cross-validation (number of folds = 5) was estimated to be 0.52.

With an inverse regularization strength value equal to 0.52, we ran our logistic regression algorithm on our training set and tested the response on our testing set, using ‘Bin Count’ as our response variable. Our model resulted in an accuracy score of 0.89, as denoted in Table I. We also illustrated the receiver operating characteristic curve (ROC) to reflect the set of points for all possible thresholds of the model resulting in an area under the ROC equal to 0.94. An area under the curve score was also computed using a 5-fold cross validation resulting in: 0.90, 0.95, 0.96, 0.94, and 0.94. It appears the model is most accurate between a true positive rate of 0.86 and 0.96.

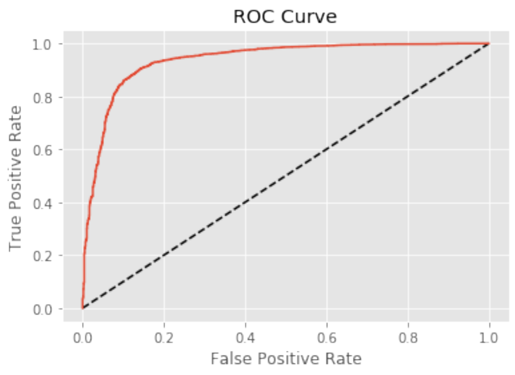

Fig. 6: ROC curve for logistic regression.

The multiple linear regression algorithm tries to establish a linear relationship between our continuous response variable, ‘Total Count’, and our features (or predictor variables).

As no additional pre-processing was needed, we ran our multiple linear regression algorithm on our training set and tested the response on our testing set, using ‘Total Count’ as our response variable. Our model resulted in an accuracy score (R^2) of 0.43, as denoted in Table I.

The Naïve Bayes classification algorithm is a categorial classification technique based on Bayes’ Theorem with an assumption of independence among features. In simple terms, the algorithm assumes the presence of a particular feature in a class is unrelated to the presence of any other feature.

As no additional pre-processing was needed, we ran our naïve Bayes classification algorithm on our training set and tested the response on our testing set, using ‘Bin Count’ as our response variable. Our model resulted in an accuracy score of 0.86, as denoted in Table I.

TABLE I. COMPARISON OF ACCURACIES

Algorithm	**Accuracy**
K-Nearest Neighbors (K = 1)	0.30
Logistic Regression	0.89
Multiple Linear Regression	0.43
Naïve Bayes Classification	0.86

Table I: While several algorithms perform reasonably well, it is clear that k-nearest neighbors and multiple linear regression are not good candidates for a learning algorithm.

Table I. shows the results. It is clear the logistic regression algorithm performed the best of all the algorithms, but does that make it the most appropriate to use? The same holds for the multiple linear regression algorithm, the best of the regression algorithms. To answer this question, we include an analysis of all four (4) algorithms in the feature search section, expecting to increase accuracy with tuned features.

VI. Feature Search

We have seen decent performance across several different algorithms and would now like to get the best performance over each particular algorithm.

For our k-nearest neighbors' algorithm, we used sequential backward selection (SBS), an algorithm that removes features sequentially from the full feature subset until the new feature subspace contains the desired number of features.

Of the eight (8) features originally selected for the k-nearest neighbors' algorithm, it seems 'Month', 'Day', 'Time', 'Temperature (F)', and 'Wind (MPH)' are the most important and maximum number of features for accuracy when predicting the 'Total Count' or total number of bikes rented per hour. Thus, the k-nearest neighbors' algorithm was re-run on the five (5) aforementioned features and resulted in no significant change to the accuracy of the model.

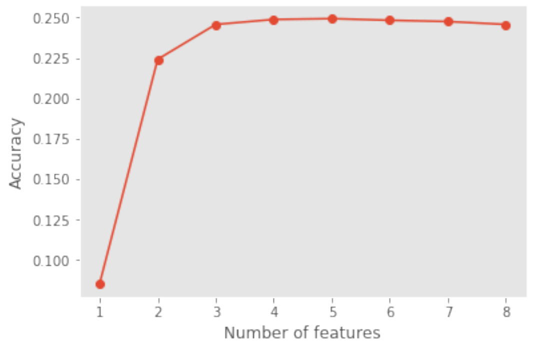

Fig. 7: Plotted k-nearest neighbors SBS feature selection.

For our logistic regression algorithm, we used recursive feature elimination (RFE) to select the top three (3) features contributing most to predicting the 'Bin Count' or classification of bike usage as 'High' or 'Low'. RFE works by recursively removing features and building a model on the features that remain.

Of the eight (8) features originally selected for the logistic regression algorithm, it seems 'Wind (Degrees)', 'Time', and 'Temperature (F)' are the most important features when predicting the 'Bin Count' or the classification of bike usage as 'High' or 'Low' – with the second and third most important features being 'Month' and 'Wind (MPH)'. Thus, the logistic regression algorithm was re-run the on the five (5) aforementioned features and resulted in no significant change to the accuracy of the model.

For our multiple linear regression algorithm, we used LASSO, a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the model. The LASSO procedure performs regularization by adding to the loss function a penalty term of the absolute value of each coefficient multiplied by an alpha of 0.4.

Of the eight (8) features originally selected for the multiple linear regression algorithm, it seems 'Time' (coefficient = -167.33) and 'Temperature (F)' (coefficient = 2.86) are the most important features when predicting the 'Total Count' or total number of bikes rented per hour. Excluding 'Time' and 'Temperature (F)', it seems 'Month' (-34.22) is the second set of most important features when predicting the 'Total Count' or total number of bikes rented per hour. Thus, the multiple linear regression algorithm was re-run on the three (3) aforementioned features and resulted in no significant change to the accuracy of the model.

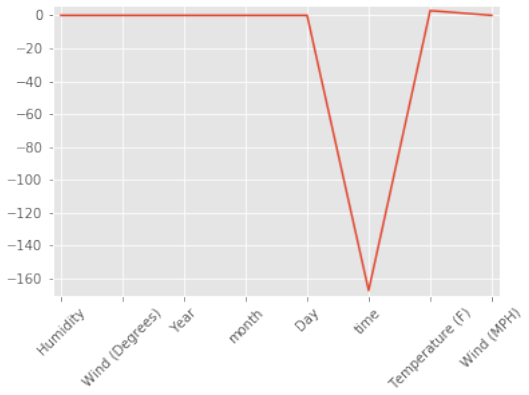

Fig. 8: Plotted multiple linear regression LASSO procedure coefficients (R^2: 0.43).

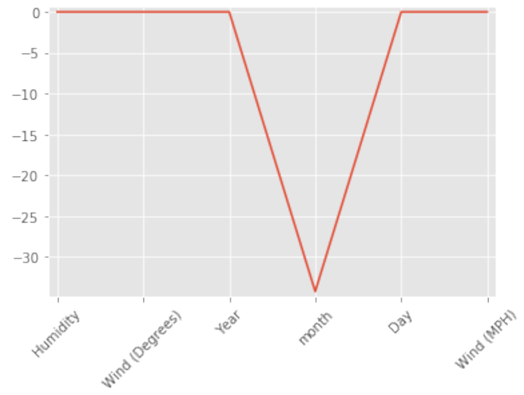

Fig. 9: Plotted multiple linear regression second LASSO procedure coefficients (R^2: 0.079).

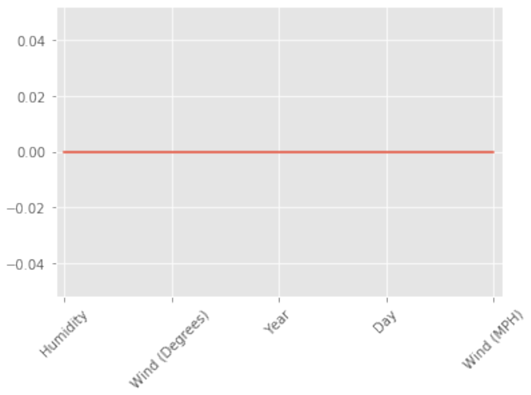

Fig. 10: Plotted multiple linear regression third LASSO procedure coefficients (R^2: 0.023).

For our naïve Bayes classification algorithm, we were unable to find an appropriate feature selection method specific to the algorithm. As the algorithm is a classification algorithm, the eight (8) features originally selected for naïve Bayes classification were replaced by features identified by way of RFE for the logistic regression algorithm: 'Wind (Degrees)', 'Time', 'Temperature (F)', 'Month', and 'Wind (MPH)'. Thus, the naïve Bayes classification algorithm was re-run on the five (5) aforementioned features and resulted in no significant change to the accuracy of the model.

The optimal feature space for each algorithm varied, but there were three features each algorithm shared for optimization: 'Month', 'Time', and 'Temperature (F)'. With that said, it is surprising to not see higher accuracy after tuning the eight (8) original features.

TABLE II. RE-COMPARISON OF ACCURACIES

Algorithm	Accuracy
K-Nearest Neighbors (K = 1)	0.30
Logistic Regression	0.89
Multiple Linear Regression (R^2)	0.43
Naïve Bayes Classification	0.86

Table II: After individually tuning using algorithm specific feature searches, the optimized performance of our four (4) algorithms remains the same.

VII. MISCLASSIFICATIONS

As cross-validation can be used to estimate the test error associated with a learning method in order to evaluate its performance. We performed a 10-fold cross validation to determine the misclassification error for our k-nearest neighbors' algorithm. Using a value of 'k' = 1, we observed a misclassification error of 0.62.

The confusion matrix for the logistic regression algorithm with an inverse regularization strength of 0.52 is shown in Table II with the precision and recall statistics. Most of the contents of the matrix lie on the diagonal, indicating a correct classification.

TABLE III. LOGISTIC REGRESSION METRICS

		Predicted class		
		Low	*High*	*Total*
True class	*Low*	1303	246	**1549**
	High	314	3331	**3645**
	Total	**1617**	**3577**	**5194**

	Precision	*Recall*
Low	0.81	0.84
High	0.93	0.92

Table III: The confusion matrix for the logistic regression algorithm and associated precision and recall statistics.

We observe two types of error in the logistic regression cross validation, with the first and largest source of error, 309, being between a predicted class of 'Low' and a true class of 'High'. And, the second error, 245, being between a predicted class of 'High' and a true class of 'Low'.

The root-mean-squared error (RMSE) for the multiple linear regression algorithm, measuring the amount of error between the testing set and the training set, resulted in approximately 286 units of error. That is, the measure of the differences between 'Total Count' values predicted by our model and the 'Total Count' values observed is 286.

The confusion matrix for the naïve Bayes classification algorithm is shown in Table III with the precision and recall statistics. Similar to the logistic regression algorithm, most of the contents of the matrix lie on the diagonal, indicating a correction classification.

TABLE IV. NAÏVE BAYES CLASSIFICATION METRICS

		Predicted class		
		Low	*High*	*Total*
True class	*Low*	1380	169	**1549**
	High	475	3170	**3645**
	Total	**1855**	**3339**	**5194**

	Precision	*Recall*
Low	0.74	0.89
High	0.95	0.87

Table IV: The confusion matrix for the naïve Bayes algorithm and associated precision and recall statistics.

Similar to the logistic regression, we observe two types of error in the naïve Bayes cross validation, with the first and largest source of error, 485, being between a predicted class of 'Low' and a true class of 'High'. And, the second error, 199, being between a predicted class of 'High' and a true class of 'Low'.

Thus, we observed the opportunity for each model to misclassify to be consistent with the accuracy of our models.

VIII. CONCLUSIONS AND LIMITATIONS

Of the models tested for regression, using a multiple linear regression algorithm, we can semi-successfully predict the number of Capital Bikeshare bikes rented in (metro) Washington, D.C with an accuracy of 0.43. And, of the models tested for classification, using a logistic regression algorithm, we can successfully classify the number of Capital Bikeshare bikes rented in (metro) Washington, D.C. into categories of volume: 'Low' and 'High' with an accuracy of 0.89.

Optimal conditions for maximum and/or minimum bike usage, according to our models, are not significantly different. The two features that can best distinguish optimal conditions are 'Time', or hour of day the bikes are used, and 'Temperature (F)'. For maximum bike usage, optimal conditions appear to be early afternoon when the temperature is in the 60's (degrees F). For minimum bike usage, optimal conditions appear to be early morning when the temperate is in the 50's (degrees F).

As we are currently limited in our knowledge and understanding, in a future analysis, we would like to learn about how the class labels were being decided in each of the four algorithms. Such an analysis *may* lead to a more accurate model, but, for now, we conclude.

Acknowledgment

Thanks to Eric Dorata and Mark McComiskey who provided helpful advice throughout the course of this analysis.

References

[1] Capital Bike Share, 2016. *Index of bucket "capitalbikeshare-data"*. [Online] Available at: https://s3.amazonaws.com/capitalbikeshare-data/index.html

[2] Capital Bike Share, 2017. *Index of bucket "capitalbikeshare-data"*. [Online] Available at: https://s3.amazonaws.com/capitalbikeshare-data/index.html

[3] Müller A., 2018. *Supervised Learning with scikit-learn*. [Online]. Available at: https://www.datacamp.com/courses/supervised-learning-with-scikit-learn