Fall 2019 DATS6103 Final Project Report

Caroline Sklaver & Sayra Moore

The George Washington University

1. Introduction

In this project, we explore the influence that various temporal and meteorological features have on city-shared bike use in London from 2015 to 2017. Our goal is to create a model that best predicts the bike count of the total bikes being used when given different time, weather, and seasonal attributes.

The dataset was originally retrieved from the UK government site 'Transport For London', and was merged with weather data from freemeteo.com/i-weather.com by timestamp. This merged dataset was then made available on kaggle.com (5). The 10 columns of this data set include:

- *timestamp*
- *cnt* - the count of a new bike shares
- *t1* - real temperature in Celsius
- *t2* - temperature in Celsius "feels like"
- *hum* - humidity in percentage
- *wind_speed* - wind speed in km/h
- *weather_code* - category of weather: 1=clear, 2=scattered clouds, 3=broken clouds, 4=cloudy, 5=rainy, 10=thunderstorms, 26=snowfall, and 94=freezing fog
- *is_holiday* - Boolean: 1=holiday, 0=non-holiday
- *is_weekend* - Boolean, 1=weekend, 0=weekday
- *season* - category field meteorological seasons: 0:spring, 1=summer, 2=fall, and 3=winter

With this data, we will use Python3 to preprocess and create an application using PyQt5 that performs machine learning algorithms in order to predict London bike-share use in 2015.

2. Description of Data Mining and Algorithm

Linear regression is a simple, widely used algorithm that examines the linear relationship between two or more variables (3). Used with just one independent variable, a Simple Linear Regression depicts the relationship between a singular independent variable and the dependent variable represented by the equation:

$$Y = mX + c$$

In this equation, Y is the dependent variable, m is the slope of the regression line, X is the independent variable, and c is the intercept. The slope, m, depicts the effect of X on Y, meaning if X increases or decreases by 1 unit, Y increases by m units. The best fit line given the data is determined by using the Least Squares method to minimize the error. First, in minimizing the error, we must create a Loss function to measure the difference between the predicted value and the actual value (6)

$$L(x) = \sum_{i=1}^{n} (y_i - p_i)^2$$

This loss function must be minimized to find the best fit regression line. To do so, the partial derivative of L is equated to 0 and then solve for m and c (6):

$$m = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

$$c = \bar{y} - m\bar{x}$$

In our case, we have more than one independent variable. Thus the equation of the line becomes multi-dimensional:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots. + \beta_n X_n + \varepsilon$$

In this case, β_0 is the intercept, and β_n is the coefficient of each feature, X_n. The same logic and mathematical derivation for least squared error applies to both simple and multiple linear regressions.

3. Experimental Set-Up

3.1 Preprocessing

This dataset required a few preprocessing and cleaning techniques. After reading in the data from kaggle.com, a few column names were changed. Then, the "timestamp" column was broken down into "month", "day", and "hour" and added on to the data frame. Next, "weather_code" variable was converted to dummy variables ranging from 0-7. Lastly, the "count" dependent variable was standardized by taking the log() because it was highly right-skewed in the histogram. Other plots and visuals were then made for exploration and presentation purposes. This data frame was exported and used for the remainder of the analysis. This portion of the project was not extensive given that the dataset did not contain any missing values.

3.2 Implementing Regression Analysis

To run the linear regression models in Python, we use Sklearn package (7). First, we split the data into target and predictor variables, then use *train_test_split()* function to split the data into training and test sets. Next we use *LinearRegression()* and *fit()* functions to train our model. Lastly, we use *predict()* function to perform prediction of the test set. See appendix for specific python code. We can then evaluate our model by examining the difference between actual values and predicted values as discussed in the next section.

3.4 Evaluating Performance

Coefficients (Beta values), R2, mean-squared-error and residuals plots are tools we can use to evaluate the model. R2 is the coefficient of determination, which calculates how close the data is to the line, defined by the equation (8):

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

Where SS_{RES} is the sum of the residuals, which is calculated by the distance between the actual data points on the y-axis and their corresponding predicted values along the regression line, squared (8). SS_{TOT} is the sum of the distances between the actual data points on the y-axis and the mean of y, squared (8). The R2 value tells us how much of the variation in the dependent variable is explained by the predictor variable(s),

Mean-squared error is defined by the equation (4):

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2$$

This represents the average squared distance between the estimated values and actual values (2)..

With Sklearn package, we can obtain these three attributes with functions, *coef_*, *r2_score()*, and *mean_squared_error()*, respectively. Visualizing residual error is done by overlaying scatterplots of residual errors from training data and testing data. With the use of PyQT5, our GUI demonstrates each of these model evaluations with the ease of a button for differing features in the multiple linear regression models.

4. Results

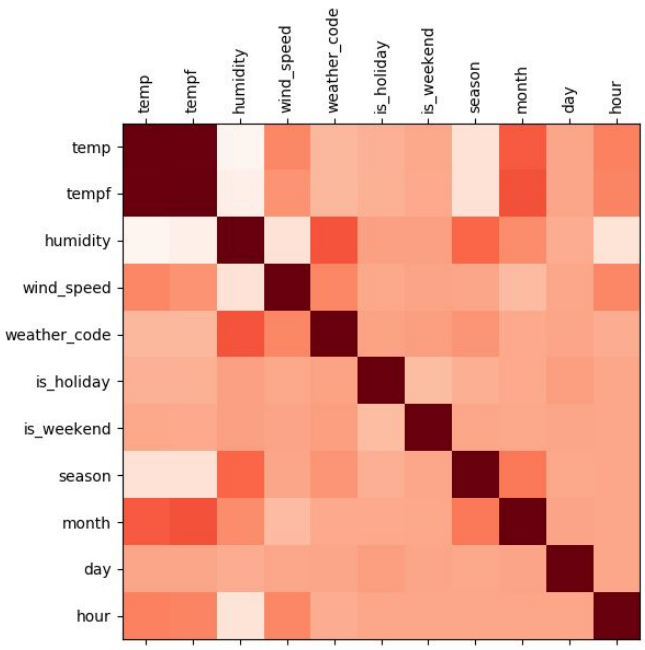

Figure 1: Correlation plot between all independent variables.

Our first visualization we created was a correlation matrix (Figure 1) that shows us the strength of the correlation between the independent variables using a heat map - darker colors represent stronger correlations. Clearly, in Figure 1 “temp” and “tempF” are highly correlated, which is inherent – the actual temperature is correlated with how the temperature “feels”. Next, “month” and both temperature measures are somewhat highly correlated, as well as “weather_code” and “season” with “humidity”. These results are quite intuitive.

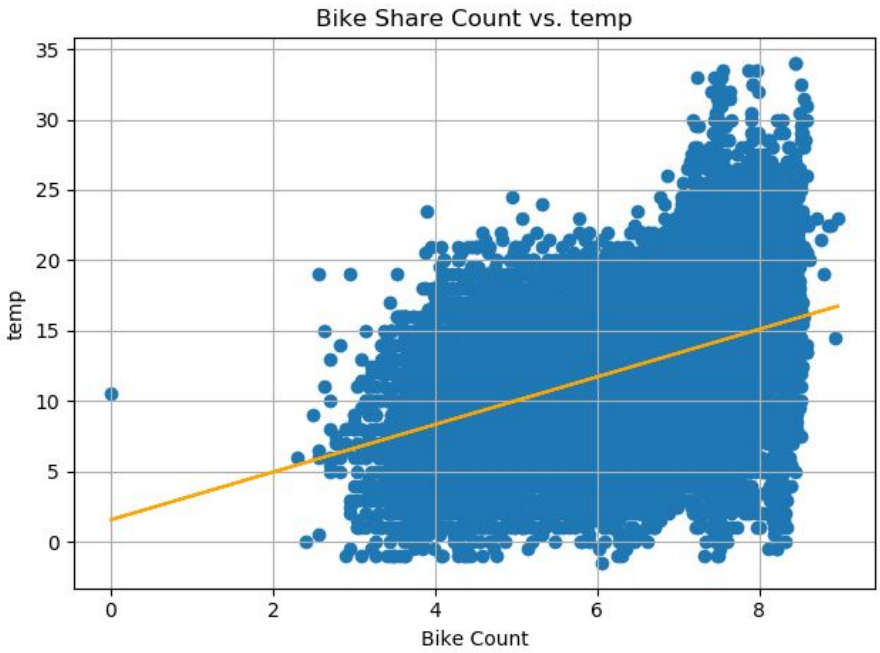

Figure 2: Regression Plot of temp vs. Bike Count

Next, we observed the relationship between the continuous independent variables and bike count. The first one we are looking at is the relationship between the temp and the bike count. We see that it has a direct relationship, meaning while the temp is increasing, so is the bike count.

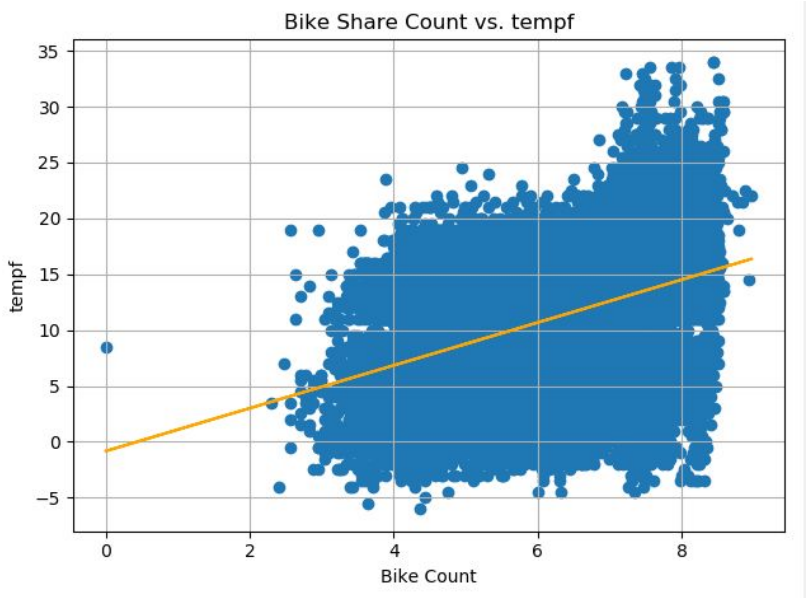

Figure 3: Regression Plot of tempf vs. Bike Count

The second relationship we are looking at is the effect of tempf and bike count. This also has a direct relationship, which is showing us that as the tempf is increasing, so is the bike count.

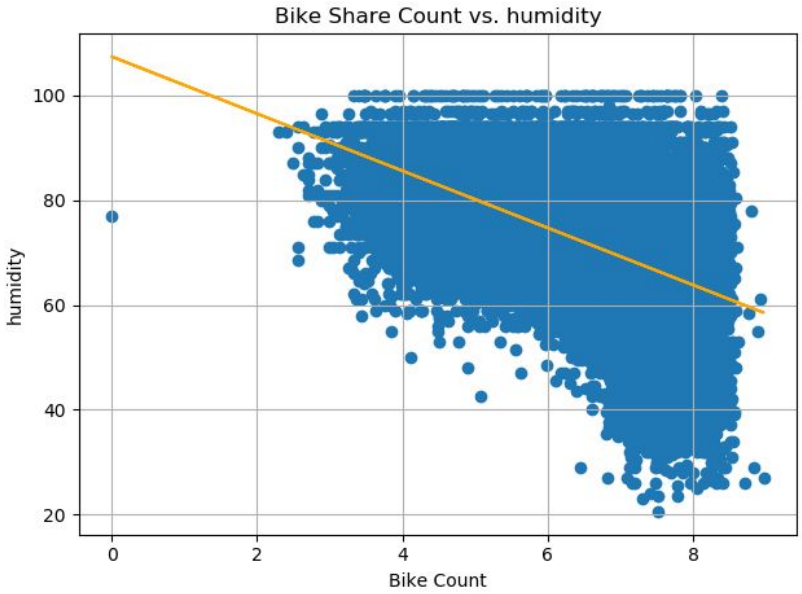

Figure 4: Regression Plot of humidity vs. Bike Count

The third relationship we have is between the level of humidity and the bike count. This relationship shows us that it is indirect, in which we see that as the humidity levels increase, the bike count decreases.

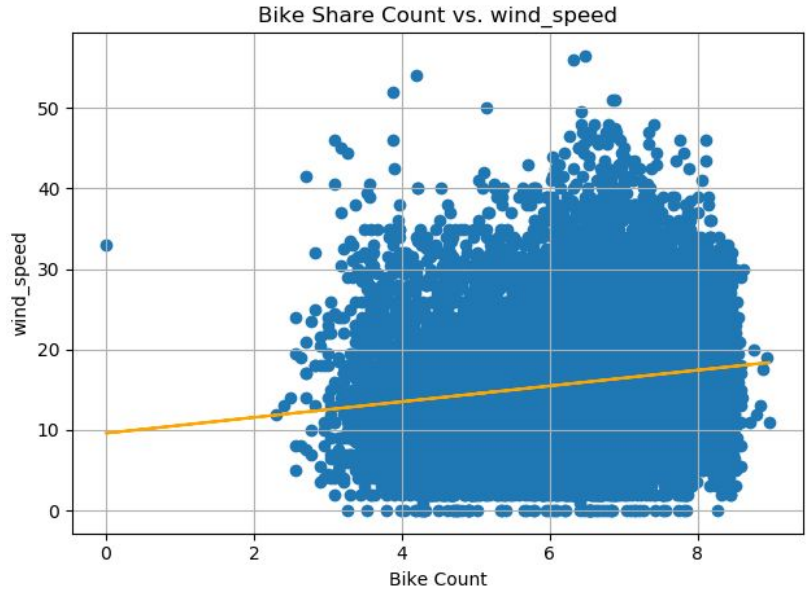

Figure 5: Regression Plot of wind_speed vs. Bike Count

Lastly, our fourth relationship is the wind_speed to the bike count. We see that as the wind_speed increases so does the bike count, therefore showing us a direct relationship. Although it is direct, we see that the increase is slow, especially by looking at how little rise the regression line has.

With our dataset, we were hoping to have the option to do Logistic Regression to build a model that would accurately predict if a bike share was being used or not. Unfortunately, our dataset only represented instances in which the bikes were being used, therefore eliminating the option for us to run many models such as Logistic Regression, KNN, Random Forest, and Decision Tree. Although we did have enough categorical variables to run decision trees, since we were not predicting if they did (1) or did not (0) purchase the bike share, then it would not have been logical, even if we had received some type of output.

After eliminating the other possible models to apply, we decided to apply Multiple Linear Regression.

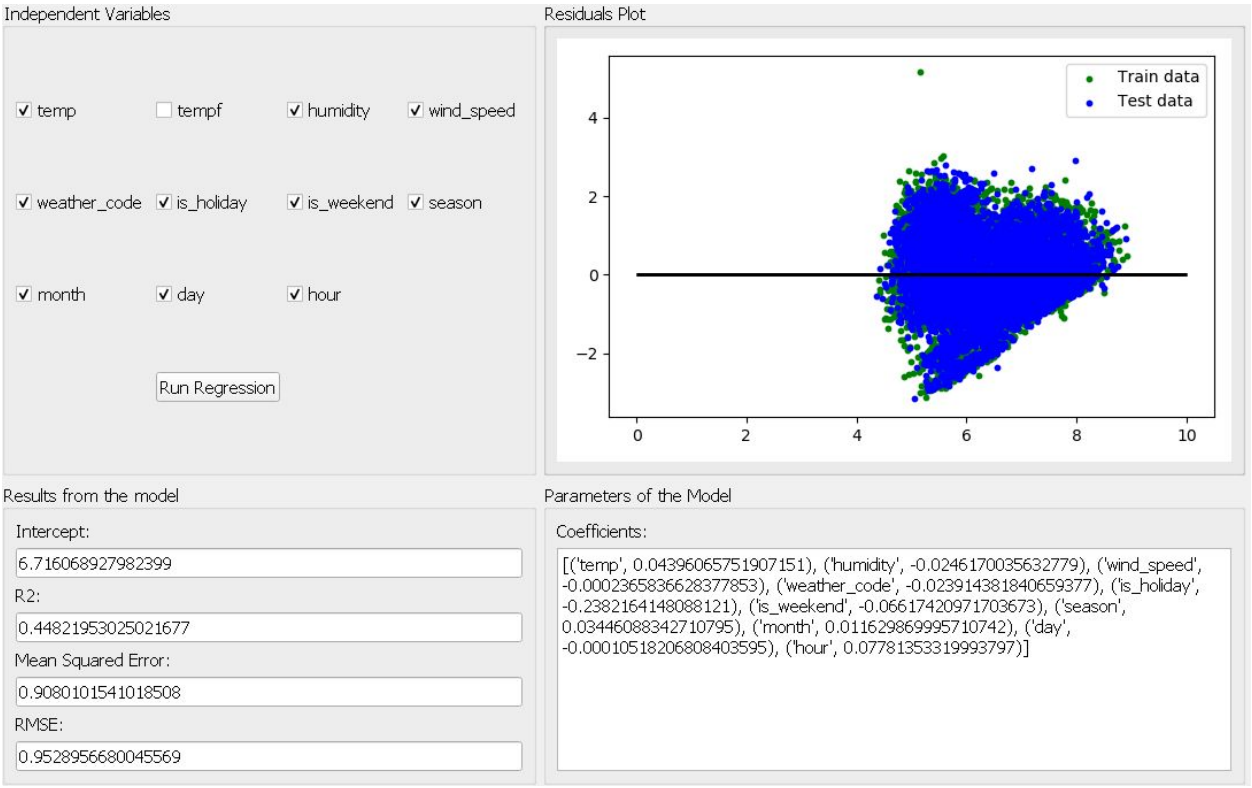

Figure 6: Multiple Linear Regression model output

After running the model, we saw that our R-squared was 0.448, which tells us that the model accounts for only 44.8% of the variation. Although a low R-squared isn't always bad. In our case with the dataset that we have it is because it is telling us that there is about 56% unexplained variation with the variables that we have in trying to determine the amount of bike shares being purchased.

As you can see in Figure 6, we used all the variables except "tempf". We chose to remove it because it was so highly correlated with "temp" that when it was ran with the rest, it only increase the R-squared by 0.0003, which means that only 0.03% more of the variation is explained by the model.

After calculation the Mean Squared Error, we got a value of .9080. To find the MSE, you simply take the average of the square of error. This value represents how close or how far the data points are from the regression line, or the line of best fit, also known as residuals. We interpret MSE as the amount

of error, the larger the number the more error and vice versa. Our model also shows us the Root Square Mean Error (RSME) which is just the square root of MSE.

Lastly, gathered our intercept and coefficients to create the Multiple Linear Regression equation. The equation is written as $\hat{Y}=b_0+b_1X_1+b_2X_2+...b_nX_n$. By using all of the coefficient outputs that we got and our intercept, our final equation would like like the following:

$$\hat{Y}=6.72 + 0.04(temp) - 0.02(humidity) - 0.00(wind\_speed) - 0.02(weather\_code) - 0.24(is\_holiday) - 0.07(is\_weekend) + 0.03(season) + 0.01(month) - 0.00(day) + 0.08(hour)$$

From here, we can simply take a number that we are given for each variable (for example, 5degrees C for temp, 45% humidity, 1 for summer, 6 for the month of June, etc), then we can plug those in accordingly then the final number would be the predicted number of bikes shares given the circumstances of the variables.

5. Conclusion

After doing our EDA and building the model, we saw that we didn't have a great model. I believe that if our target variable was binary (if they did or did not purchase a bike share) the model would have worked much better, even with the data that we have. But since we were simply predicting the number of instances (number of bike shares being purchased), with this specific dataset was not great, especially after seeing that the model only explained about 45% of the variation. If we were using the data for classification (if they did or did not purchase a bike share), then we would be able to run confusion matrix to determine the accuracy. Unfortunately, we could not use that method since our model was not used for classification. Overall, we would not recommend this model for trying to predict bike share purchases with our dataset.

References

1. Bogicevic, M. (2018, November 2). Multiple Linear Regression Using Python. Retrieved from https://medium.com/@manjabogicevic/multiple-linear-regression-using-python-b99754591ac0.
2. Columbia Mailman School of Public Health. (2019, June 10). Tutorial: Understanding Linear Regression and Regression Error Metrics. Retrieved from https://www.dataquest.io/blog/understanding-regression-error-metrics/.
3. Evaluating a Linear Regression Model. (n.d.). Retrieved from https://www.ritchieng.com/machine-learning-evaluate-linear-regression-model/.
4. freeCodeCamp.org. (2018, October 8). Machine learning: an introduction to mean squared error and regression lines. Retrieved from https://www.freecodecamp.org/news/machine-learning-mean-squared-error-regression-line-c7dde9a26b93/.
5. Mavrodiev, H. (2019, October 10). London bike sharing dataset. Retrieved from https://www.kaggle.com/hmavrodiev/london-bike-sharing-dataset.
6. Menon, A. (2018, September 13). Linear Regression Using Least Squares. Retrieved from https://towardsdatascience.com/linear-regression-using-least-squares-a4c3456e8570.
7. Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.
8. Stanford, M. (2017, December 20). Simple Linear Regression in Python. Retrieved from https://medium.com/@mjfstanford/simple-linear-regression-in-python-905b759ef0e6.