Bike Sharing Predictions

An Application of Machine Learning

Patrick Thornton
Michigan State University

Abstract

As markets grow, the need to accurately predict future demand increases. With the field of machine learning growing at such a fast rate, accurate usage predictions are on the horizon. This paper will showcase one use case of machine learning in predicting the demand of rental bikes for a given hour using features relating to weather, time of day, and time of year.

1. INTRODUCTION

With the growing societal concern over our environment, many people have turned to alternative methods of transportation. One example of this is the bike sharing systems that are being introduced to large cities across the globe. With the growing market in bike sharing, the need to properly stock stations will increase. Using machine learning and regression models, we have the ability to use statistical data in order to predict the number of bikes that will be in operation within an hour.

Given a dataset of 6000 feature sets, each comprised of 10 categories, I will split the data into two datasets. One of these datasets I will use to train two models, a linear regression model and a random forest regression model. After training these two models I will use the remaining samples in my second set, henceforth denoted as my testing set, to evaluate the performance of my two models and compare the results. By the end of this document, I will have explained to you the process of model creation and its usefulness in the service industry.

2. DATASET OVERVIEW

2.1. DATASET CONSTRUCTION

In the field of machine learning, one of the most important aspects to developing any model is data. To create my bike sharing model I will be using a dataset that is the union of three separate datasets. The first dataset is the Capital Bikeshare system data[1]. This dataset provides the count of how many bikes were rented per hour and on which day. Next, I have my weather data gathered using i-weather's historical data[2] related to Washington D.C, which is where Capital Bikeshare's business operates. The final dataset provides holiday information[3]. For a date to be classified as a holiday, it must be recognized as a national or local holiday by the government of Washington D.C. After combining these three datasets, we are left with one larger dataset that has 10 features per sample and one label with values ranging 1 to 976 representing the number of bikes rented per hour.

2.2. DATASET FEATURES

The given dataset is split into two separate comma separated values (.csv) documents. The first of these documents is train.csv which is comprised of 5000 samples to be used in the training of my model. The second document is test.csv, which has 1000 samples I will use to validate the accuracy of my model. For each sample in my dataset, there are 11 columns that provide information for any given hour, of any given day relating to weather, ongoing holidays, or rental bike usage. The breakdown of these columns are as follows:

- dteday : date
- season : season (spring, summer, fall, winter)
- hr : hour (0 to 23
- holiday : weather day is holiday or not (Yes, No)
- weekday : day of the week (0 to 6)
- weathersit : weather (Clear, Light Snow, Mist + Cloudy)
- temp : Normalized temperature in Celsius.
    - The values are derived via (t-t_min)/(t_max-t_min), t_min=-8, t_max=+39 (only in hourly scale)
- atemp: Normalized feeling temperature in Celsius.
    - The values are derived via (t-t_min)/(t_max-t_min), t_min=-16, t_max=+50 (only in hourly scale)
- hum: Normalized humidity. The values are divided to 100 (max)
- windspeed: Normalized wind speed. The values are divided to 67 (max)
- cnt: count of total rental bikes

2.3. TASK

Using my given dataset, I will construct and compare two different regression models which will be evaluated based on their accuracy, correlation, and related trend direction between the predicted number of rental bikes shared in a given hour and the actual number of rental bikes shared in a given hour.

3. MACHINE LEARNING ALGORITHMS

3.1. CLASSIFICATION VS REGRESSION

When making a machine learning model, there are two ways a model can learn. A model can have supervised or unsupervised learning. The difference between the two is whether the true label for the testing data is known. If the true label is known, then it is considered supervised, and since we know the labels for the testing data, I will be using supervised learning for my model creation.

Within supervised learning, there are two subcategories, classification and regression. Classification models attempt to map a sample of features to a specific categorical representation of a label i.e. whether an email is considered spam or not. On the other hand, regression models are used to predict continuous variables that can have a large range of values i.e. the number of rental bikes shared in an hour.

As the previous paragraph hinted at, a regression model is a much better fit for my task. Of the regression models, I will be discussing two, linear regression and random forest regression. Even though I will be using regression for my models, it does not mean that there is not a classification approach to this problem as well. For a look at how one can use classification on a problem like this and the possible benefits to doing so, I will be discussing a classification application in the discussion section of this paper.

3.2. LINEAR REGRESSION

The goal of linear regression is to attempt to model the relationship of the features to the labels using a linear formula. For this project I used multivariate linear regression which is used in the case where there are multiple explanatory variables that the label could be dependent on.

Given that the predictor produced by linear regression is, by nature, linear, this method tends to be the best fit for a very niche type of dataset whose labels follow a linear trend. Being of linear form is not all bad, however. Even though linear regression has a very small realm of uses that return a high accuracy, the simplicity of it causes the model to be easy to implement and very efficient if the number of features is low. Linear regression has an overall time complexity of $O(p^2n+p^3)$ for training and $O(p)$ for predicting (where p is the number of weights, and n is the number of samples)[4]. Linear regression would be a significantly better fit for those that have a short list of features and care more for speed than accuracy.

3.3. RANDOM FOREST REGRESSION

Where linear regression is a relatively simple algorithm, random forest regression is a complex algorithm. Random forest algorithms work by creating many decisions trees that are each formed using different subsets of data in a method known as *Bagging*. Once all the training is completed and the decision tress are made, the model makes it's prediction by pushing the test sample down every decision tree and averaging the results between every tree to output a final, aggregated prediction.

Not only is the methodology of random forest creation more complex than that of linear regression, but so is the customization. With linear regression it is an extremely limited as to what parameters you can add to your model, only allowing for an intercept weight. With random forest, this list is significantly extended. Here you can choose the number of trees created, the max depth of each tree, the maximum features of a tree, and the minimum number of samples needed before a tree will split to make a new node, just to name a few. Changing all these parameters can help increase or decrease accuracy and speed.

When comparing the time complexity of random forest to that of linear regression, it is clear to see how much longer a random forest regressor can take. The time complexity of a random forest regressor is $O(n^2pn_{trees})$ for training and $O(pn_{trees})$ for predictions[4]. The first notable difference her is that the time it takes for a random forest regressor is much more dependent on the number of samples and number of tress in the model whereas linear regression was dependent on the number of features.

To decide between linear regression and random forest, one has to know what their priorities are and what their data looks like. To use a more complex model like random forest, time will almost always have to be sacrificed.

4. RESULTS

4.1. EVALUATION METRICS

With regression predictors, there are four primary ways to evaluate the accuracy of a model, and for this project I will be using all four.

4.1.1 ROOT MEAN SQUARE ERROR

Root mean square error (RMSE) is a very intuitive way to measure the accuracy of a model. RMSE is calculated by summing the square of the distance between two points, dividing the sum by the number of samples, and then taking the root of the quotient. The number outputted from the RMSE function is a real number that represents how far off, on average, the model's prediction was.

$$RMSE = \sqrt{\frac{\sum_{\mathrm{i=1}}^{\mathrm{M}}(\hat{y}_{\mathrm{i}} - \mathrm{y_i})^2}{\mathrm{M}}}$$

Where M is the length of the label vector, $\hat{y}$ is the predicted value and y is the actual values.

4.1.2 PEARSON CORRELATION

Pearson correlation is the measure of linear correlation between two values and is represented by a value between -1 and 1, with the two ends of the range representing a strong correlation. Another way to think of this is as the measure of how close each point is to the line of best fit. Pearson correlation, along with RMSE, are the two strongest representations of how accurate a model is that I will go over today. The next two evaluation metrics are primarily meant for ordinal variables where as Pearson and RMSE are used for *metric* variables. Pearson correlation is calculated as:

$$R_p = \frac{\sum_{i=1}^{M}(\hat{y}_i - \hat{\mu})(y_i - \mu)}{\sqrt{\sum_{i=1}^{M}(\hat{y}_i - \hat{\mu})^2}\sqrt{\sum_{i=1}^{M}(y_i - \mu)^2}}$$

Where M is the length of the label vector, ŷ is the predicted label, y is the actual label, and μ is the mean of the respective vector.

4.1.3 Spearman Correlation

Spearman correlation measures the monotonic relationship between two vectors, which in this case are the predicted labels and the actual labels. A monotonic relationship is a relationship where if one variable increases or decreases between two points in one vector, the same happens for the two points located in the same place in the other vector. To determine the monotonic relationships, each vector needs to be converted into a rank vector. A rank vector is a vector where each entry represents how many values in the same vector are larger than itself (i.e. [405,6,13,73] $\rightarrow$ [1,4,3,2], that is 13 is the 3rd largest value in the vector). Once the rank vector is established, Spearman correlation is calculated with the following formula:

$$R_s = \frac{\sum_{i=1}^{M}(\hat{r}_i - \hat{\mu})(r_i - \mu)}{\sqrt{\sum_{i=1}^{M}(\hat{r}_i - \hat{\mu})^2}\sqrt{\sum_{i=1}^{M}(r_i - \mu)^2}}$$

Where M is the length of the label vector, r hat is the predicted label, r is the actual label, and μ is the mean rank of the respective vector.

4.1.4 Kendall Tau

Kendall Tau Correlation measures the ordinal association between two vectors. The ordinal association between two vectors is the relationship between different variables of one vector and similar variables of a different vector. To use this metric, we first need to convert our predicted and actual label vectors to rank vectors. Once in rank form, we can compare the two vectors using Kendall Tau because ranks are *ordinal* variables. An ordinal variable are categorical variables that can be compared but mathematical operations do not work or have no purpose. Once in rank form, we must compare the *i-th* pair and *j-th* pair from each vector where *i* is some random integer between 0 and M-1, and *j* is *i+1*. When we compare these two pairs, we can put them into one of three categories:

- Concordant (P): if both $y_i > y_j$ and $\hat{y}_i > \hat{y}_j$ or both $y_i < y_j$ and $\hat{y}_i < \hat{y}_j$.
- Discordant (P): if both $y_i > y_j$ and $\hat{y}_i > \hat{y}_j$ or both $y_i < y_j$ and $\hat{y}_i > \hat{y}_j$.
- Tie ($Y_0/\hat{Y}_0$): $y_i = y_j$ or $\hat{y}_i = \hat{y}_j$

Once we have the number of concordant, discordant, and tied pairs, we calculate the Kendall Tau Correlation with the following function:

$$\tau_b = \frac{P\text{-}Q}{\sqrt{P + Q + Y_0}\sqrt{P + Q + \hat{Y}_0}}$$

Where P is the number of concordant pairs, Q is the number of discordant pairs, Y_0 is the number of ties in the actual label vector, and $\hat{Y}_0$ is the ties in the predicted label vector.

4.2. METHODOLOGY

An important part of developing any machine learning algorithm is optimization both speed and accuracy. Speed optimization comes from feature reduction and limiting how complex a model can get. Feature reduction can occur in a number of ways. If your feature set is comprised of only numerical data, you can use dimensional reduction strategies such as gaussian elimination to reduce the number of samples used for training, which would increase the speed of a random forest model. Feature correlation can also play a big part in choosing which features to eliminate before the model even runs. Using correlation, one can eliminate any features that have little to no correlation with the feature that the model is trying to predict. Eliminating a feature like this normally will still have some effect on the predicted value, but it will increase the speed of the model by a margin that likely outweighs the accuracy loss. Below is a heatmap correlation matrix between the feature set of my data and the count of rental bikes used.

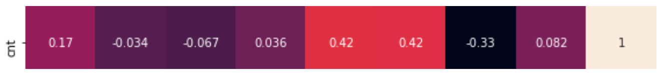

Figure 1: Heat Map for Cnt Correlation

The next data manipulation tactic to do before training would be to normalize your data. In the project I use a normalize function to normalize my feature data and given that my labels approximately followed a log normal distribution, I used a log transformation on my labels. Below you can see the before and after distributions of my labeling values:

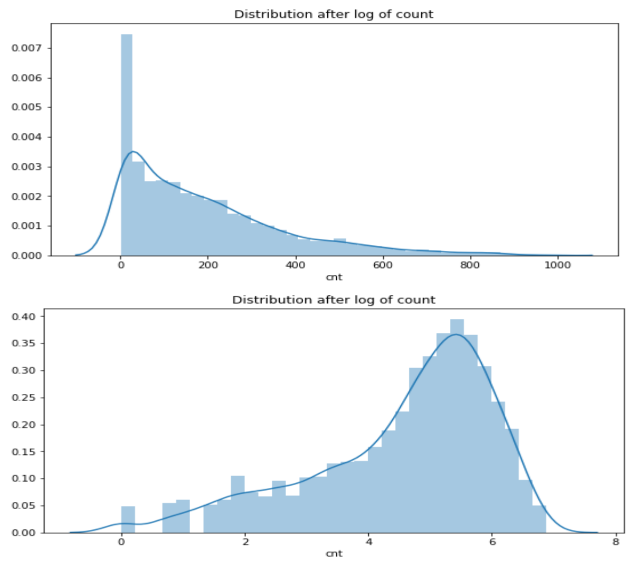

Figure 2: Log Transformation on Training Labels

Lastly, every model has feature weights which indicate how much effect a feature has on an output of a model. To lesser the complexity of a model, one can eliminate any features whose weights are smaller than a predetermined threshold for a small sacrifice of accuracy to improve the time. The 5 features with the highest weights in my linear regressor and my random forest regressor are:

Table 1: Feature importance for Random Forest

Feature	Importance
Hr	0.703
Temp	0.107
Weekday	0.083
Hum	0.035
Windspeed	0.022

The way to determine importance for linear regression is by looking at the weights associated with each feature. To find the weights of each feature, you can look at the predictor function "p(x)" that is produced by the model and match the variables to feature names. In this function the larger the magnitude of the weight, the more importance it had. Below you can see my predictor function as well as the 5 features with the highest weights:

p(x) = 2.76 + 0.10*x_1 + 1.77*x_2 + 1.07*x_3 + -1.54*x_4 + 0.17*x_5 + 0.01*x_6 + 0.19*x_7 + 0.52*x_8 + -0.21*x_9 + -0.39*x_10 + 0.29*x_11

Table 2: Feature weights for Linear Regression

Feature	Weight
Temp (x_2)	1.77
Atemp (x_3)	1.54
Winter (x_8)	0.85
Light Snow (x_10)	0.39
Mist+Cloudy (x_11)	0.29

For most regression models there are also hyperparameters that can be changed in order to improve the performance of a model. As mentioned in the Machine Learning Algorithms section of this document, my linear regression model had no such hyperparameters. However, in my random forest model, there are a few hyperparameters that can be optimized. For this project I looked at optimizing the number of trees (n_estimators) and the number of features. Below is a list of those hyper parameters and their respective OOB scores:

Table 3: Optimization table for number of trees

N_estimators	OOB_score
100	0.895
200	0.897
300	0.898
500	0.899
1000	0.899

Table 4: Optimization table for max features

Max_features	OOB_score
None	0.899
Sqrt	0.862
Log2	0.862

Out-of-bag (OOB) error is the measurement of the error between the actual values of an untested subset of data and the value that was predicted using a model. This score is similar to that of the accuracy score of a model, it is just determined using a different subset of data than that used for the validation set. For a good model, you want an OOB score close to 1, but the true value of an OOB score comes from comparing it to that of your model's accuracy score. Since the OOB score is determined by testing against a new subset, having an OOB score close to that of the model's accuracy score (determined by model.score using sklearn) shows that your model preforms consistently. The actual value of the model.score and the OOB score shows how accurate the model actually is. In terms of the n_estimators, you can see that the OOB score did not change between 500 and 1000, as such I saw no point in testing beyond 1000 and settled with 500 n_estimartors for the sake of time.

4.3. RESULTS AND DISCUSSION

4.3.1 LINEAR REGRESSION

Earlier in this document I talked about the pros and cons of a linear regression model. To reiterate, linear regression models tend to have poor accuracy for complex feature sets but have a quick runtime. Even though my dataset is not very complex compared to some other datasets, it still had 12 features, which is slightly higher than I would prefer when using linear regression. As a result, my linear regression model had a runtime of 0.003 second, which is very fast for a machine learning model on this scale. However, as one can image, this model's accuracy was poor in comparison to the random forest model. As talked about in the Evaluation Metrics section of I will use four different metrics: RMSE, Pearson Correlation, Spearman Correlation, and Kendall Tau Correlation. The evaluation metric results from linear regression can be seen in Table 5.

Table 5: Evaluation Metrics for Linear Regression

Evaluation Metric	Score
Root Mean Square Error	162.5652
Pearson Correlation	0.6848
Spearman Correlation	0.6766
Kendall Tau Correlation	0.4757

For a visual representation of how accurate my Linear regression model was, I have created a graph where the x-axis is the predicted values and the y-axis is the actual values. On this graph there is a line plotted where x=y, representing a perfect model where every predicted value is equal to the actual value.

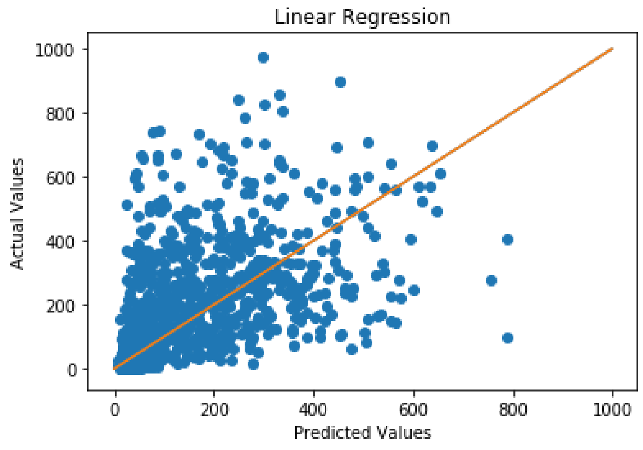

Figure 3: Linear Regression Model Visualization

4.3.2 Random Forest

Random forest regression was a much better fit for this project given the complexity and size of the dataset. With only 12 features, this dataset was still very simple as far as a random forest model is concerned. Having 5000 samples for training is not necessarily a large number of samples, but since the big-O of a random forest is determined in large part on the square of the number of samples, the runtime of this model was much longer than that of the linear regression model. The runtime of training and predicting labels using my random forest model was 5.24 seconds using 500 trees. This time can be reduced if I were to use less trees and would still achieve better results than the linear regression model, but 5 seconds is not long, so running my model like this is best. Clearly the runtime of this model is significantly longer than that of the linear regression model, nearly 18000 times as long, but the accuracy payoffs show. In Table 5 you can see the evaluation metrics for my random forest model.

Table 6: Evaluation Metrics for Random Forest Regression

Evaluation Metric	Score
Root Mean Square Error	76.5793
Pearson Correlation	0.9534
Spearman Correlation	0.9413
Kendall Tau Correlation	0.7888

The evaluation metrics for my random forest model are significantly better than that of my linear regression model. For a visual representation of my model, again I made a graph similar to that seen in the linear regression section.

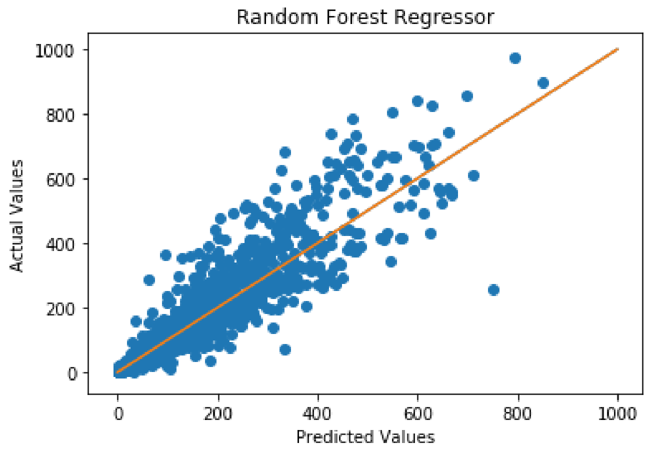

Figure 4: Random Forest Regression Model Visualization

4.3.3 Classifiers

Another approach to this machine learning problem would be to use a classifier instead of a regressor to predict the number of rental bikes used per hour. Now the primary issue with this is that classifiers quickly become less accurate the more classifications used. As one would imagine, using a classification model to try and predict the exact number bikes rented would be unrealistic considering there are over 700 possible classifications. So, how and why would you ever use a classifier for a dataset like this?

Consider the case where Capital Bikeshare needs to put out enough bikes every day so that they never run out, even if they have some not being used. If this were the case, they might not be concerned with having the exact number bikes in stock, so long as they rarely run out bikes. To use a classifier to solve this issue, I first need to decide on how many classifications I want and how to find separate my data into them. Earlier I stated that the less categories the better, so let's settle on four different categories. To find these categories lets first assume there will not be more than 999 bikes ever used in an hour. One solution would be to always put 1000 bikes out, but that a lot of work considering they hardly every need more than 700. With these assumptions I can split up by dividing the number of bikes used by 250 and rounding down, this would leave us with the following category breakdown:

Table 7: Possible category split for classification

Number of bikes (N)	Category
N < 250	Slow hour
250 < N < 500	Moderate hour
500 < N < 750	Busy hour
750 < N	Chaotic hour

Now that I only have a few categories I can use a classifier. I do not plan on walking through this in detail, but for those interested I thought I would include it. As a brief remark, if one used logistic regression with the categories found in Table 7, you would get an accuracy of 0.72, this accuracy grows to 0.84 if using three categories determined by dividing by 326 (minimum for three categories).

Cases like this appear decently often in the service industry. For example, if a company could predict the traffic they would have in any given day they would know how many employees they needed to work that day, which might be more valuable than trying to predict the exact amount of stock they would need using a less accurate model.

This section was not meant to build upon the project but more as an exploration of alternative ways to go about a similar problem.

5. FUTURE WORK

If this project were extended, there are two things I would like to do: extrapolate more data and try more regression models.

The dataset provided in this project had a column called "dteday" for my models I dropped this column due to specific date being noise for the vast majority of models. One way to turn this feature into a more helpful feature would be to extract the month from it. Unlike a specific day, a month has more consistent traits such as temperature, tourism, etc. Days can also share traits over the years, but unless the data goes back decades, the traits shared between two specific dates (i.e. July 11th, 2012 and July 11th, 2014) would be to volatile and have no chance of being normally distributed. Another way to extrapolate more information from this dataset would be to change how the holiday column works. Currently, the values in the column are either "Yes" or "No". If I could change this to contain the actual holiday taking place i.e. Christmas, 4th of July, etc., then this column might have a more meaningful impact.

Since I went with a regression model for my project, I was limited to my selection of models. The three models I could chose from were: Linear Regression, Random Forest, and Support Vector Machine (SVM). Given the complexity of making my own SVM model from scratch, I decided to focus my efforts elsewhere to make good Linear Regression and Random Forest models. With more time I would certainly attempt to make an SVM model for my dataset. Outside of my current selection of regression models, there are a lot more models that would be capable of handling this dataset. Just to get a glimpse of how other models would work, I investigated sklearn's other regression models and how accurately they preformed on my data. Just looking at the default parameters, here are some regression models in sklearn's library and how they preformed based sklearn's score method, including default random forest:

Table 8: Sklearn Regression Model Evaluations

Model	Score
Ridge	1.133
Huber	1.147
Elastic	1.146
Decision Tree	0.483
Extra Tree	0.268
Gradient Boost	0.301
Random Forest	0.258
Bagging Regressor	0.257

These scores are calculated by finding the cross-validation score using 10 subsets of the original data, and then averaging the scores. The scores are calculated using the coefficient of determination (R^2). The important thing to not here is the lower the R^2 score, the better the model. Looking at the different models, we see that Bagging regressor actually outperformed the Random Forest regressor, which would certainly be worth looking into in a future project or if this project were extended.

6. CONCLUSION

Using Capital Bikeshare's data, along with weather and holiday data for each day, I created two regression models to predict how many rental bikes would be in use for a given hour of a given day. The two types of models I created were Linear Regression and Random Forest regression. Between the two models, the random forest model outperformed the linear regression model in all four evaluation metrics. In particular, the RMSE of the two models indicated that the random forest model was on average closer to the exact results by 61 bikes used, ending with an error of 72 bikes on average. The largest take away from this project should be the difficulty in accurately predicting a continuous variable with such a large range of values using regression. With more features and further model tuning, I believe the accuracy of these models would significantly increase.

REFERENCES

1) Capital Bikeshare. (2019). *System Data | Capital Bikeshare*. [online] Available at: https://www.capitalbikeshare.com/system-data

2) *The Weather. Weather Forecasts. Current Weather. Severe Weather Warnings | i-weather.com*. (2019). *I-weather.com*. Retrieved 29 April 2019, from https://i-weather.com/weather/?language=english&country=us-united-states

3) *Holiday Schedules | dchr*. (2019). *Dchr.dc.gov*. Retrieved 29 April 2019, from https://dchr.dc.gov/page/holiday-schedules

4) *Computational complexity of machine learning algorithms*. (2018). *The Kernel Trip*. Retrieved 29 April 2019, from https://www.thekerneltrip.com/machine/learning/computational-complexity-learning-algorithms/

5) *Predicting bike sharing trends with Python*. (2018). *Medium*. Retrieved 29 April 2019, from https://medium.com/@wilamelima/analysing-bike-sharing-trends-with-python-a9f574c596b9

6) *Predicting Bike Rental Demand | Kaggle*. (2019). *Kaggle.com*. Retrieved 29 April 2019, from https://www.kaggle.com/thilakshasilva/predicting-bike-rental-demand

7) *Correlation*. (2018). *Python for Data Science*. Retrieved 29 April 2019, from https://pythonfordatascience.org/correlation-python/