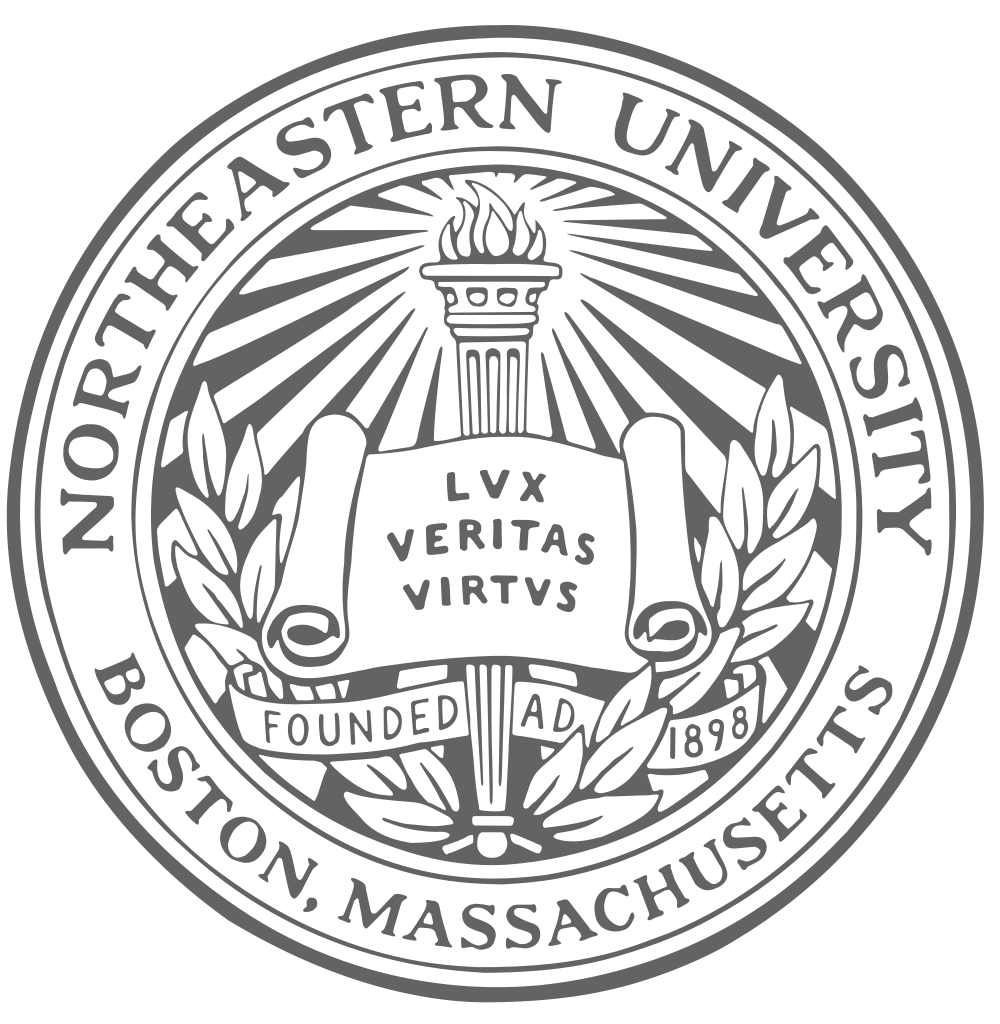
****

Final Project Paperwork

**New York Taxi Trip Duration**

Northeastern University

**ALY6110: Data Management & Big Data**

**CRN:** 7120

Professor Valeriy Shevchenko

**Submission Date:** October 23, 2019

**Submitted By:**

Pragya Bhandari (bhandari.p@husky.neu.edu)

**SUMMARY**

A majority of New Yorkers heavily rely on public transportation or livery services, of which New York City Taxis comprise of a significant part. It is interesting to realize that only about 22% of the residents in Manhattan own a car compared to the US national average of 91% of households owning at least one car. As a result, the livery services in the city has grown from 63,000 to more than 100,000. New York City has been rated as the third most congested city in the world and the second worst in the US, in terms of traffic. The number of taxi rides in New York City amount to approximately 200 million annually. With a large amount of taxi trips data being generated, an in-depth analysis on a regular basis can provide extremely valuable insights for various stakeholders such as city administration, riders, taxi drivers and app developers. An evidence-based approach can help streamline and organize traffic which will further encourage the city planners in creating a better urban infrastructure. Prediction of trip durations, fares, and geographical distribution of traffic can help passengers plan their commute, and better inform the drivers about peak times that can generate maximum profit. Therefore, an analysis of the taxi trip duration in New York City is crucial for the population that is largely affected by it on a daily basis.

**Business Problem**

Understanding the geographical distribution of Taxi traffic and prediction of total ride duration of trips in New York City.

**About the Dataset**

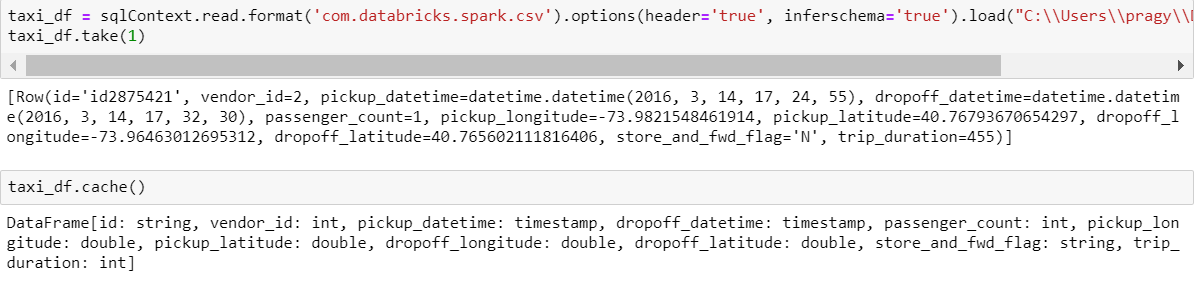
The New York City Taxi Trip Duration data set has been taken from the data repository of Kaggle. It was originally released the NYC Taxi and Limousine Commission (TLC), which consists of pickup time, geo-coordinates, number of passengers, and several other relevant variables. The data files comprise of 1.5 million training and 630k test observations. There are 11 variables that describe the data set and are listed below:

1. **id** – unique identifier for each trip
2. **vendor\_id** – identifies the provider associated with each trip record
3. **pickup\_datetime** – date and time when the meter was engaged
4. **dropoff\_datetime** – date and time when the meter was disengaged
5. **passenger\_count** – number of passengers in the vehicle
6. **pickup\_longitude** – longitude where the meter was engaged
7. **pickup\_latitutde** – latitude where the meter was engaged
8. **dropoff\_longitude** – longitude where the meter was disengaged
9. **dropoff\_latitutde** – latitude where the meter was disengaged
10. **store\_and\_fwd\_flag** – indicates whether the trip record was held in vehicle memory before sending to the vendor because the vehicle did not have a connection to the server – Y = store and forward; N = not a store and forward trip
11. **trip\_duration** – duration of the trip in seconds

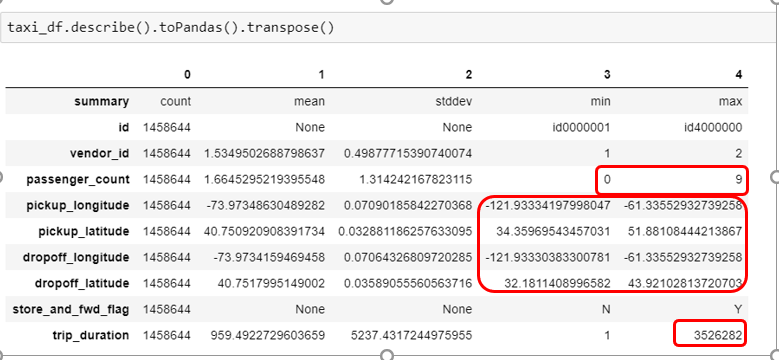
**CONTENT/ANALYSIS**

**Data Preprocessing**

We begin by loading the data using the sqlContext package of PySpark and then store it in the cache memory as also shown in the screenshot below. The .cache() function stores the RDD in the cache memory such that Spark doesn’t have to compute it every time we run our business queries.



After we have loaded our training data, we explore it’s structure using the .describe() command as shown below:



From the output of the descriptive statistics performed above, we can notice that there exist a few outliers in our data, mentioned below and also marked in the output table above:

1. **passenger\_count**

The passenger\_count variable has a minimum value of 0 which is irrelevant for our business case. Another important observation is that the maximum value of passenger\_count is 9, which is highly unlikely for a taxi cab. There may be a situation in which children are accompanied by adults, however, it is a rare occurrence to have that many children in a cab so we choose to consider it as an outlier as it could distort our analysis.

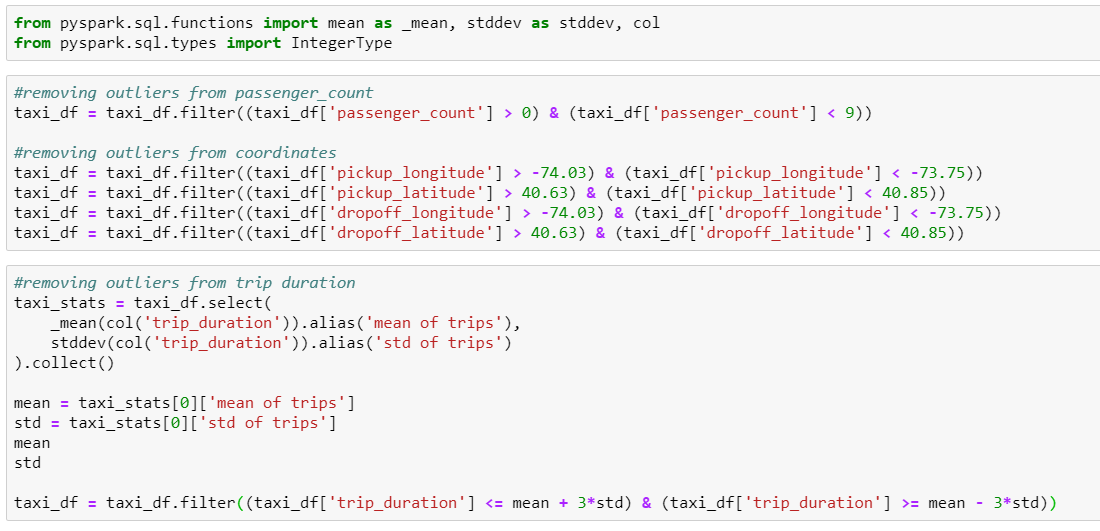
1. **Location coordinates – Longitude and Latitude**

The Latitude of NYC lies within 40.7128 and 40.748, whereas the longitude is within the range of -74.0059 and -73.968. However, the Maximum and minimum values lie way outside our range, hence, we remove them as outliers as we are only interested in analyzing the taxi trip durations within NYC.

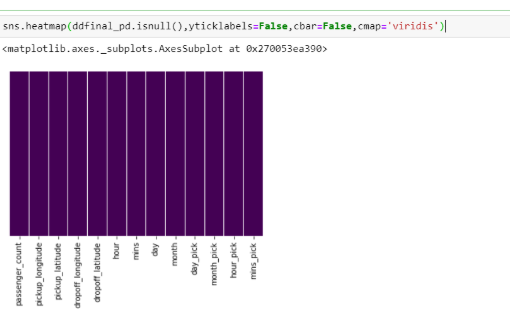
1. **trip\_duration**

The descriptive statistics summary also shows that our dependent variable ‘trip\_duration’ has a maximum value of 3526282 i.e. 980 hours approx., which is an unrealistic trip time so we choose to consider as an outlier and remove it from our variable in the step ahead. We take the trip duration values 3 standard deviations within the mean and eliminate the ones that lie outside our range.

After we have gained an understanding of the outliers in our data set, we can run the following commands using the sqlContext in Pyspark to get rid of them:



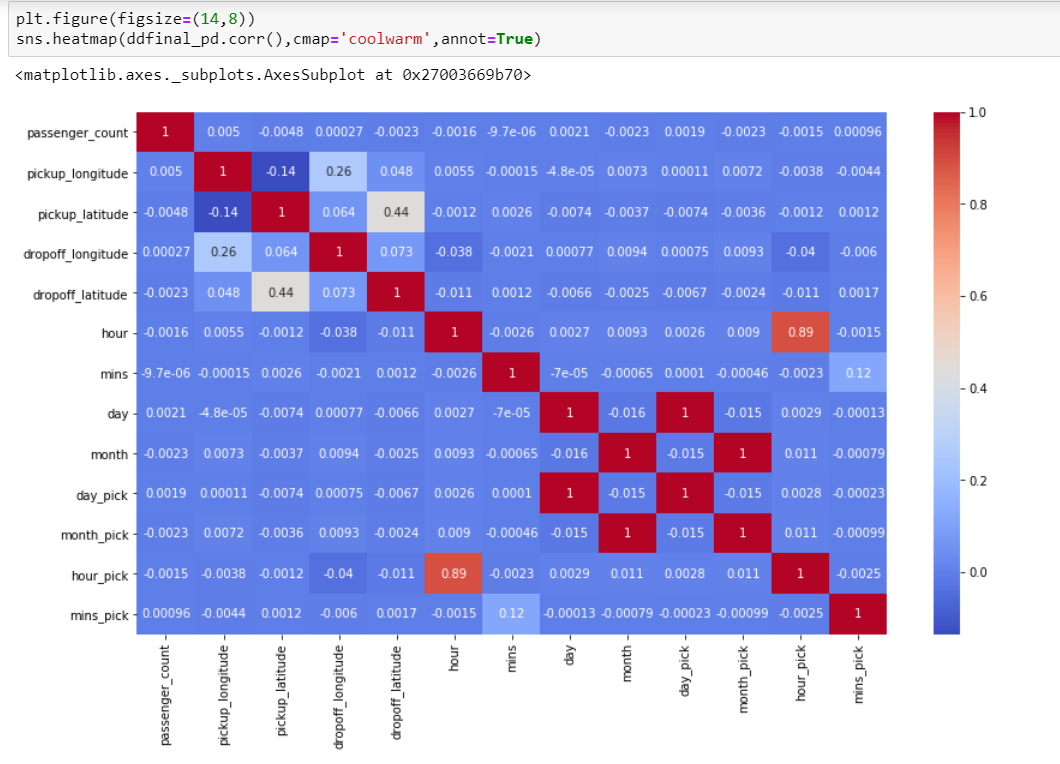
We also checked for missing values using the following code and plot:



In the plot above, each column is represented in a bar. Since the color is same throughout the bar, we can infer that there are no missing values.

**EDA**

We have plotted a heatmap of the independent variables that affect the taxi trip duration in NYC. As we can see from the plot that most features depict a low level of correlation, which also satisfies the assumption of No Multicollinearity before we build a linear regression model. However, there seems to be a high correlation between a few variables such as hour\_pick & hour, day\_pick & day, month\_pick & month and mins\_pick & mins. These can also be seen having a shade of red which depicts a high level of correlation. Variables such as hour\_pick & hour, day\_pick & day, month\_pick & month and mins\_pick & mins represent pickup date&time whereas Hour, minutes, day and month variables represent dropoff datetime. Since, dropoff and pickup datetime have almost similar values, there’s a high correlation between these variables.

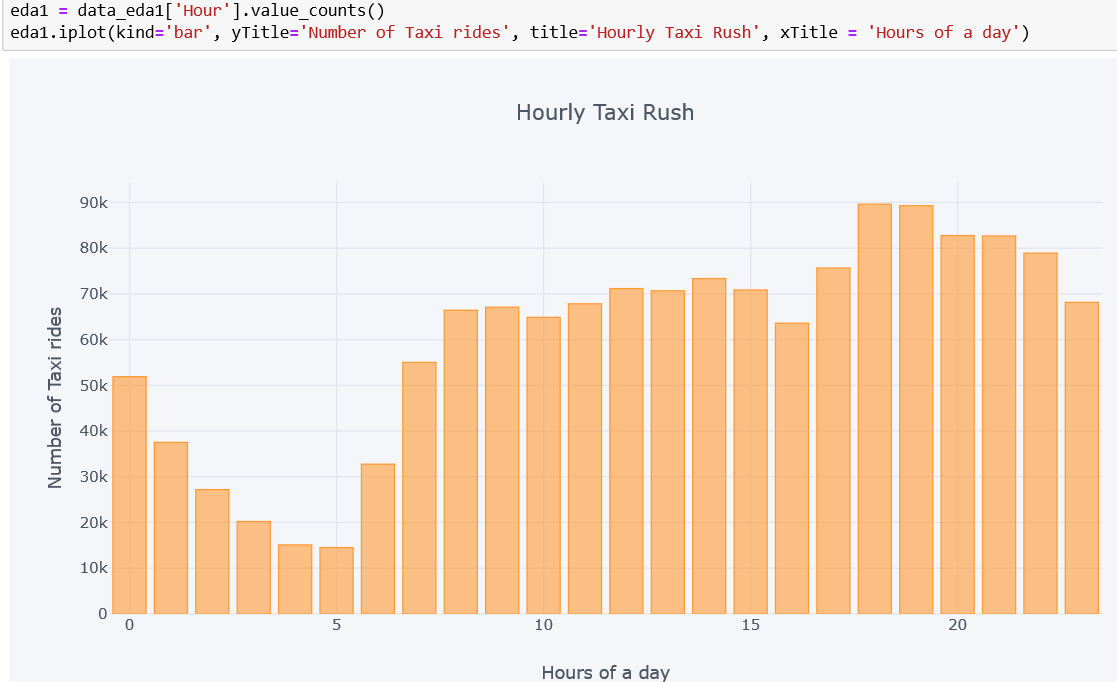


Next, we will dig deep into the dataset and try to find some valuable insights. First, we will find the day of the week that had maximum taxi trips. The output and the code for the same can be seen in the screenshots below:

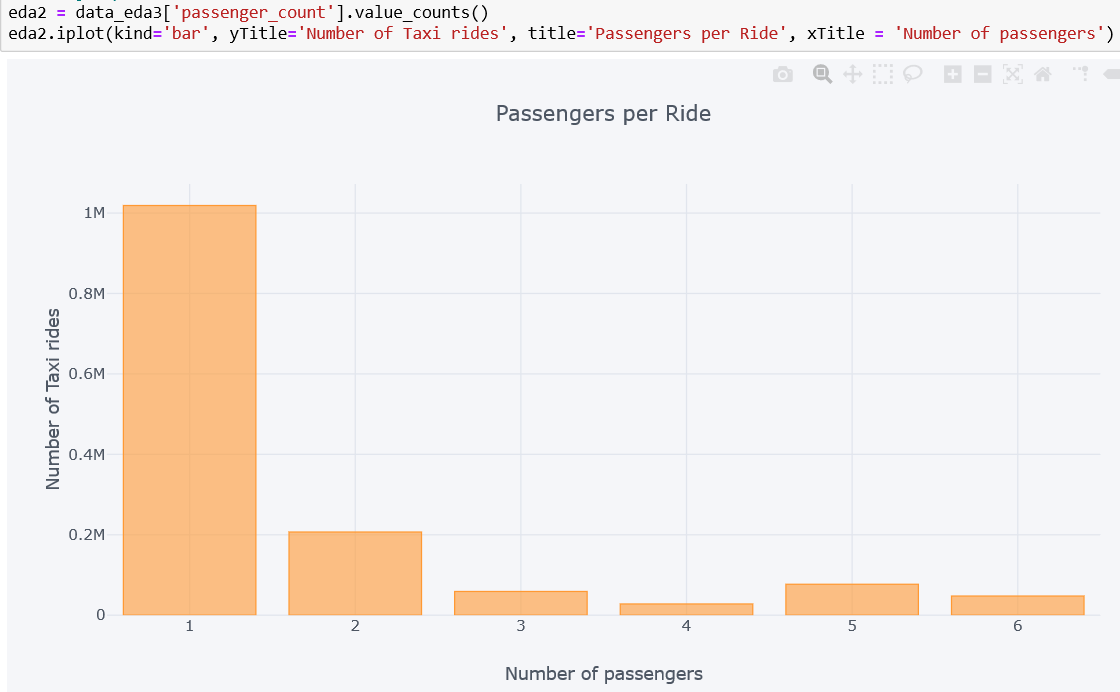


From the above screenshot, we can infer that Friday reported maximum number of trips followed by Saturday and Thursday. Hence, we can conclude that based on this data, trips on weekends were more than on the weekdays as people go out on the weekdays and they prefer taking a taxi.

Next, we will find the busy times of taxi rides in New York City. Below is the code and screenshot of the output:



From the above screenshot, we can infer that the busy times were in the evening from 6 PM to 10 PM as maximum taxi rides were reporting during this period. Next, through this data, we will find the trend of number of passengers in a taxi ride. Below is the code and output for the same:



From the screenshot above, we can infer that most of the rides had one passengers in them followed by 2 passengers.

Next, we will perform Principal Component Analysis on our dataset.

**Principal Component Analysis (PCA)**

**Purpose**

We have also performed a Principal Component Analysis as the number of features in the data set is huge and each variable is measured in a different unit than the other. Therefore, PCA helps when we are dealing with a Multivariate Analysis which can create the problem of overfitting. So PCA essentially, reduces the number of variables to only the important ones that account for the maximum variation in the outcome variable. In addition, the process of orthogonal transformation produces uncorrelated variables that helps in overcoming the problem of multicollinearity. However, PCA is done at the cost of producing less interpretable variables.

**What PCA does?**

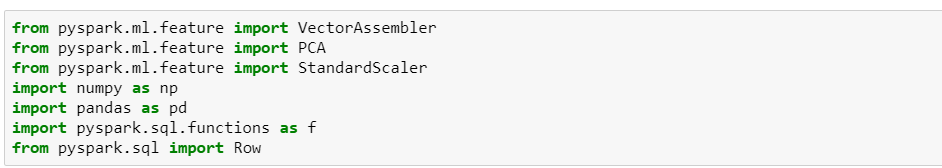
• PCA performs dimension reduction i.e. reduces a large set of variables to a small set that contains features that are most relevant for predicting the outcome variable

• It is performed through the process of Feature Extraction where direction and magnitude of variables decide the principal components that explain the most variation. Eigenvectors determine the direction of the features, whereas eigenvalues determine the magnitude of the independent variables

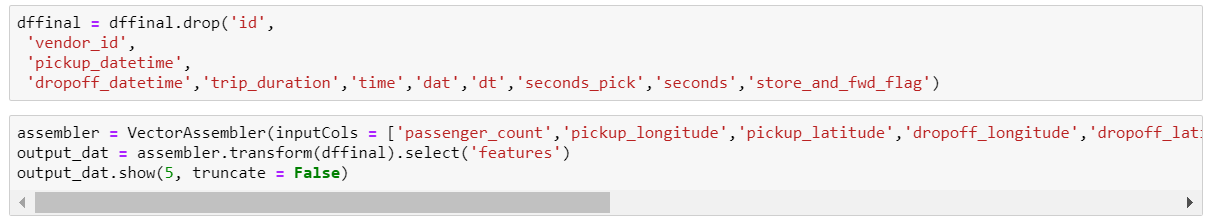
• We can also choose the number of Principal Components on the basis of a scree plot or proportion of variance explained

**PCA on NYC Taxi Trip duration data**

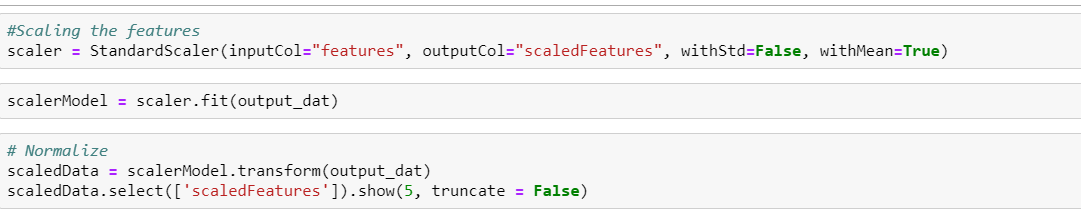
In our Taxi Trip data set, we have used pyspark libraries such as VectorAssembler and pyspark.ml.feature for conducting a Principal Component Analysis also shown in the code below:



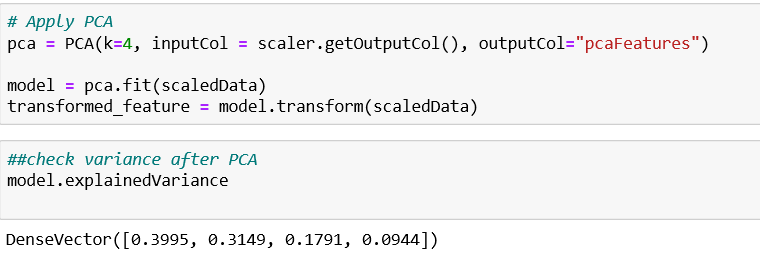
Before we begin the PCA process, we drop the string type variables from our data set and then use the Vector Assembler package to define inputs and output columns.



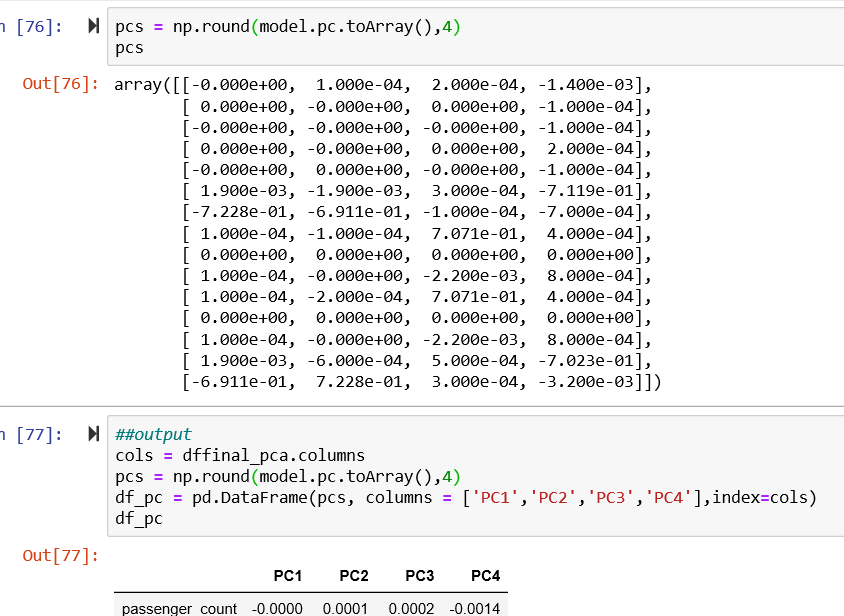
After defining our variables, we scale the features using Standard Scaler as all variables are measured in different units that are further normalized, which also reduces the variation within the features.

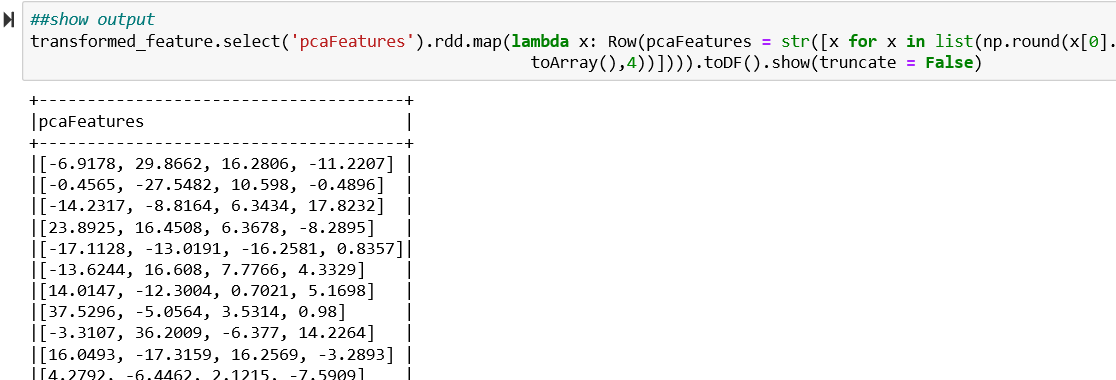


Once we have scaled and normalized our set of input variables, we finally apply PCA as shown in the screenshot below and also compute the **explained** variance with 4 Principal Components i.e. **k = 4**:

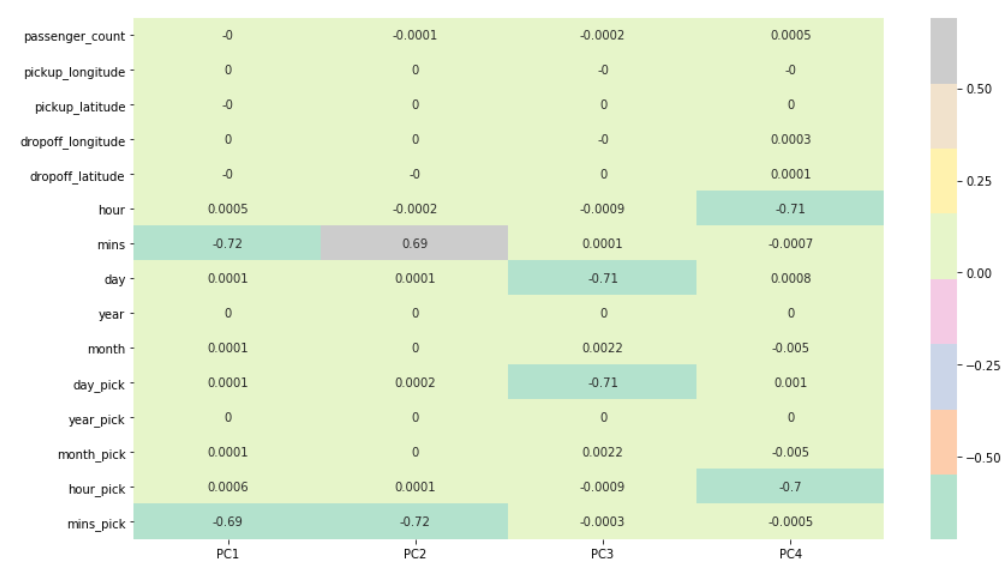


The above output shows that choosing 4 Principal Components results in a total 98.89% explained variation in our dependent variable i.e. taxi trip duration. Adding more Principal Components beyond 4 will add negligible explained variance and, hence, account for noise in our model. Hence, we have selected 4 Principal components and converted them into the Pyspark data frame from Pandas using the rdd.map() function as shown in the code below:





We can also visualize the PCA that we have performed above using a heatmap. This Heatmap shows the correlation between Principal Components and features or independent variables. The most important features are the ones with higher correlation represented by the corresponding color scale. As we go towards the gray color, the correlation increases.



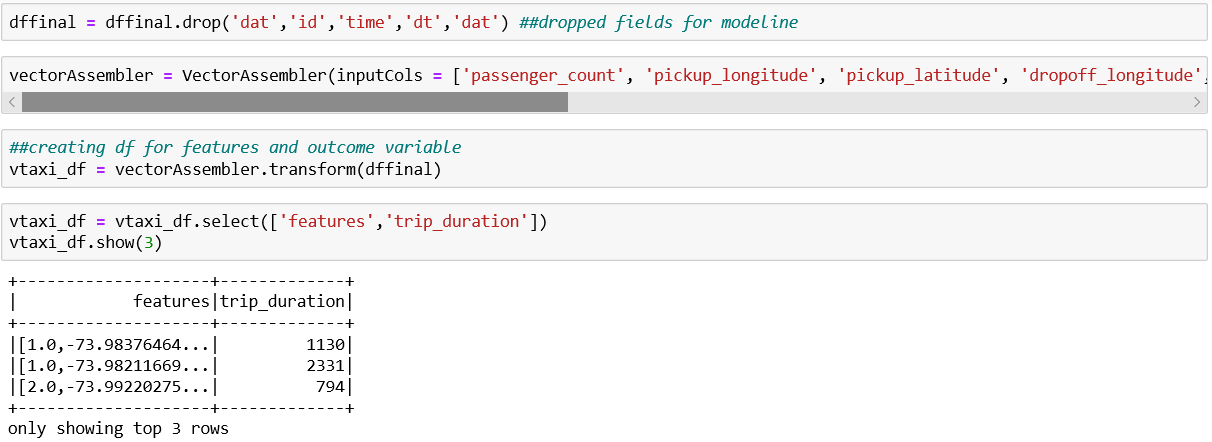
Next, we will perform regression analysis to predict trip durations of taxi rides in New York City.

**Regression Analysis**

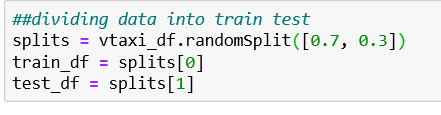
In this section, we will apply three regression algorithms namely, Linear Regression, Decision Tree Regression and Random Forest Regression. After that, we will compare these 3 models on the basis of R squared value and root mean squared error.

**Linear Regression**

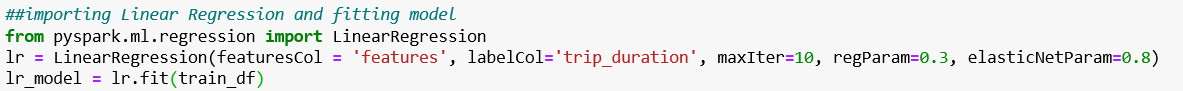
First, we will prepare our data before we start the with the modeling. We will start by removing the string fields which are not required in the model. Then, using the vector assembler, we will create a data frame of all our features and the outcome variable. The code for all these steps is as follows:



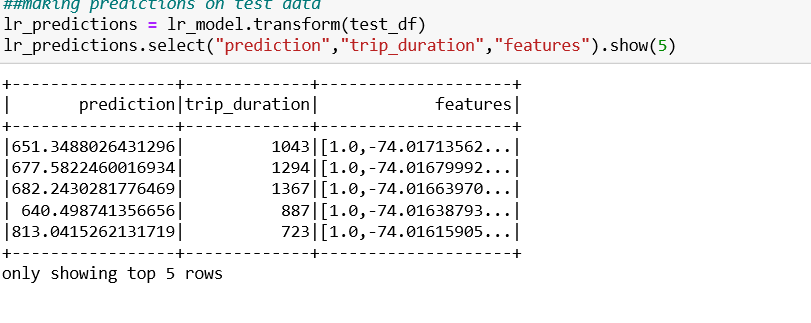
Next, we will split our dataset into training and test based on the 70:30 ratio. We allocate 70% data to our training dataset and 30% data to our testing dataset with the help of the following code:



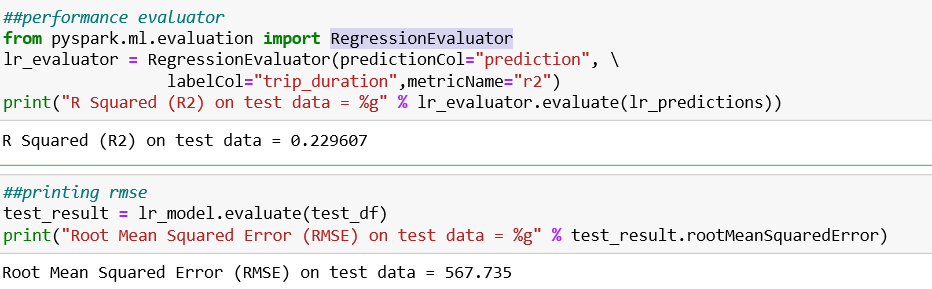
Now, we will import the LinearRegression function from pyspark.ml.regression package and then we will train our model using the training dataset. The code for these steps is as follows:



Next, we will test our model using the testing data. Below is the screenshot for the same:

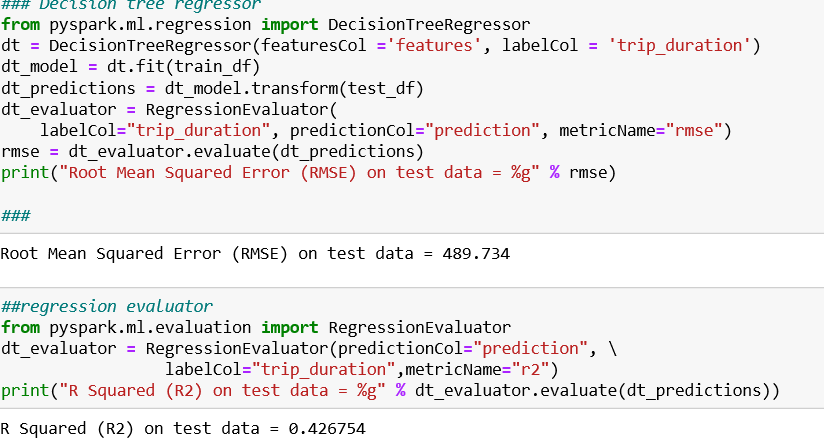


The final step is to evaluate the performance of our model. For this purpose, we will import the RegressionEvaluator function from pyspark.ml.evaluation package. We will use the R squared value and the root mean squared error value to evaluate our model’s performance. Below is the screenshot for the same:



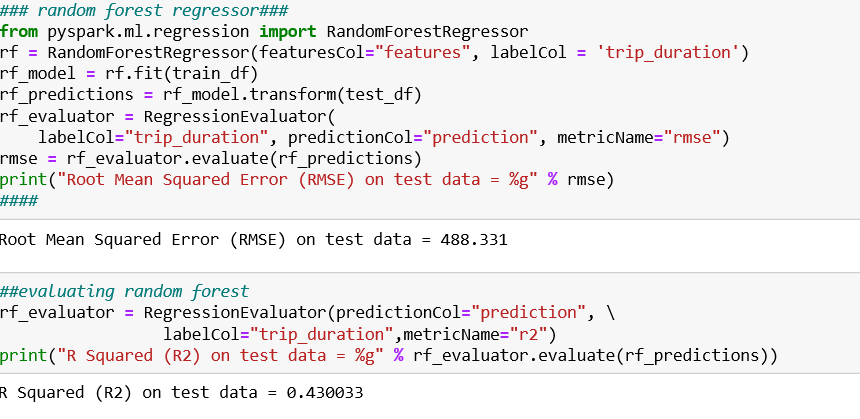
As we can see from the screenshot above, the R Squared value is 0.229. This means that 22.9% of variations in trip duration is explained by our predictor variables. In simple linear models the variance is low, and bias is high. From the above screenshot, we can also see that the value of Root Mean Squared Error is 567.735. This means that on an average, the predicted values of trip durations are 567.7 seconds away from the actual regression line. Next, we will perform a decision tree regression using the same steps.

Below is the screenshot of the code for decision tree regression:



As we can see from the screenshot above, the R squared value increased as compared to the linear regression model. The R Squared value comes out to be 0.426 which means that 42.6% of variations in the trip duration is explained by our predictor variables. Also, we noticed that Root Means Squared Error decreased to 489.734 as compared to linear regression. This means that on an average, the predicted values of trip duration are 489.7 seconds away from the actual regression line. Next, we will perform random forest regression using the same steps.

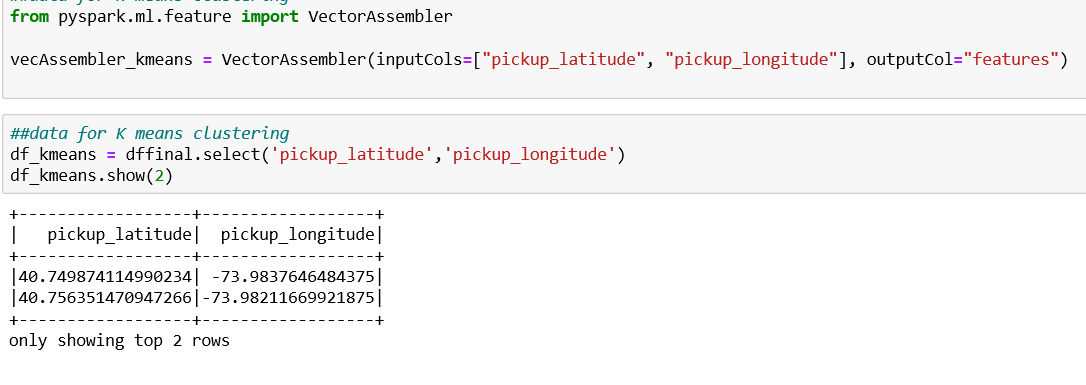
Below are the screenshots for the code of random forest regression:



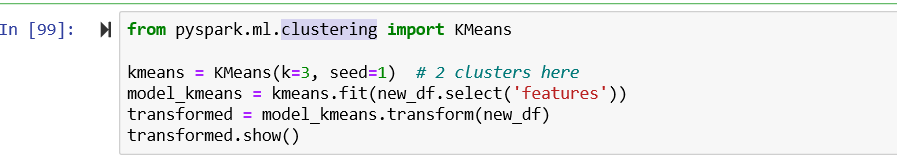
As we can see from the screenshot above, the random forest regressor performed slightly better as compared to the Decision Tree Regressor as it gives us a better R Squared value and a lower Root Mean Squared Error. The R Squared value comes out to be 0.4300 which means that 43% of variations in trip durations are explained by our predictor variables. The value of Root Mean Squared Error is 488.331 which means that on an average, the predicted values of trip duration are 488 seconds away from the actual regression line. Hence, from the regression analysis we can conclude that Random Forest Regressor gave us the best possible predictions.

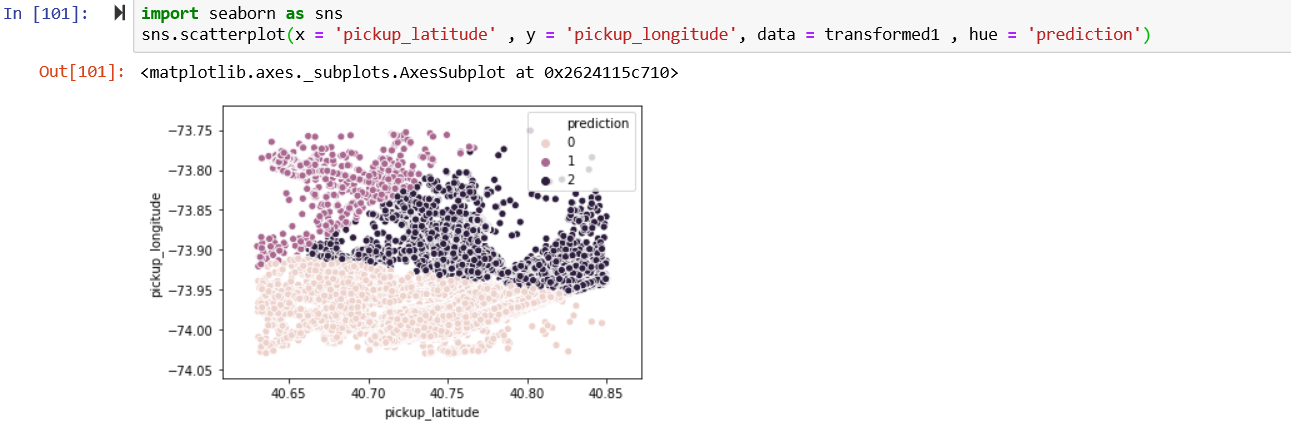
**Clustering Analysis:**

In this section, we have performed K-Means clustering on the pickup locations to highlight the in-demand areas of New York City. First, we create a data frame in pyspark using the code below:



Once our data frame is ready, we will import the Kmeans function from pyspark.ml.clustering. After that, we will apply Kmeans clustering for 3 clusters. The code and output for the same is as follows:





From the plot above, we can infer that classes 0 and 2 have the greatest number of rides. Hence, this type of clustering can be used by taxi companies such as Uber and Lyft and they can direct their drivers to these location so as to increase more rides and thereby increasing profits.

**COMMENTS**

In this project, we used pyspark on our standalone system. We noticed that spark is very fast, and it is very easy to analyze Big Data through spark. The data that might take hours on other platforms is handled with ease using spark. Another big advantage of Hadoop is that it can save lot of costs. However, we faced some challenges as pyspark is a little different from normal python language and it was completely new for us. Also, the way in which data is stored in pyspark data frame is entirely different from how we store in python. Hence, it took us some time to learn the way pyspark works so that we can analyze our data using pyspark.

**CONCLUSION**

After thoroughly analyzing the New York Taxi Trips dataset, we found that the maximum number of trips were reported on Fridays followed by Saturday and Thursday. We also found that the most frequent times during which the trips were taken was between 6 PM to 10 PM. Another interesting insight we drew from the dataset was that most of the trips were taken by a single passenger. We also predicted trip durations using Linear Regression, Decision Tree Regression and Random Forest Regression and we concluded that Random Forest Regression gave us the best prediction out of the three regression models we performed. This model can help taxi companies in predicting the trip durations when they provide an estimate to their customer. The clustering model we created can help the taxi companies as they can divert their drivers in the busy pickup areas.

**REFERENCES**

* Kaggle.com. (2019). *New York City Taxi Trip Duration | Kaggle*. [online] Available at: <https://www.kaggle.com/c/nyc-taxi-trip-duration>
* Spark.apache.org. (2019). *Welcome to Spark Python API Docs! — PySpark 2.4.4 documentation*.[online] <https://spark.apache.org/docs/latest/api/python/index.html>
* Data.cityofnewyork.us. Available at:<https://data.cityofnewyork.us/Transportation/2018-Yellow-Taxi-Trip-Data/t29m-gskq>
* Lee, C. (2017). How to Install and Run PySpark in Jupyter Notebook on Windows. Chang Hsin Lee. Retrieved <https://changhsinlee.com/install-pyspark-windows-jupyter/>
* TLC Trip Record Data. (n.d.). Retrieved from <https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page>
* Spark MLLib — Computational Statistics in Python. (2019). People.duke.edu. Retrieved from <http://people.duke.edu/~ccc14/sta-663-2016/21D_Spark_MLib.html>
* Random Forest as a Regressor: A Spark-based Solution - DZone Big Data. (2019). dzone.com. Retrieved from <https://dzone.com/articles/random-forest-as-a-regressor-a-spark-based-solution>
* Machine Learning with Spark and Python. (2018). Epigno Blog. Retrieved 24 October 2019, from <https://blog.epigno.systems/2018/02/18/machine-learning-with-pyspark-linear-regression/>