Project Rubric

Implement a Spark application, programmatically (using Python or your preferred language), the following:

- Using the 311-service data, use Apache Spark to load data on an AWS notebook (create your own cluster and link that notebook using your own AWS account or link to the cluster provided by instructor).
- Design a machine learning program that can predict the amount of time that it will take for an issue to go from “Open” to “Resolved” among “Resolved” tickets. Determine which features are useful to prediction.
- Provide the code that you have developed to run the proposed problem in Spark (AWS Notebook or python code, for example).
- Provide the results obtained and describe how you selected the features which are most important.

Input File Name: s3://pgarias-bucket-cloud/311_Service_Requests_from_2015_to_Present_head.csv

Input File Size: ~ 6GB

Input File Number of Records: 10420594 (~ 10 Million)

Notebook Name: rb1088_HW3_Notebook_Final

Programming Framework: PySpark

Execution Environment: AWS Notebook on Hadoop Cluster

Analysis

The PySpark application programmatically aimed to do the following:

1) Analyze NYC 311 Service Requests from 2010 to Present.

2) Apply machine learning and build a predictive model to try and predict the amount of time that it will take for an issue to go from “Open” to “Closed” among “Closed” tickets.

3) Determine which features are useful for prediction.

Preprocessing: Preprocessing the input file included the following steps:

- Dropping columns containing null values namely Vehicle Type, Taxi Company Borough, Taxi Pick Up Location, Bridge Highway Name, Bridge Highway Direction, Road Ramp, Bridge Highway Segment, Landmark.

- Dropping columns not suitable for using as predictor variables for machine learning program. This includes the following attributes:

- Unique Key: Unique identifier for the dataset.
- Descriptor: Text field identifying type of problem.
- Location Type: Categorical field identifying type of location
- Incident Zip: List of input codes (over 1000)
- Incident Address: Address text field
- Street Name: Street Name text field
- Cross Street 1: Text field
- Cross Street 2: Text field
- Intersection Street 1: Text field
- Intersection Street 2: Text field
- Address Type: Text field
- City: City details
- Facility Type: Facility type details
- Due Date: Date field
- Resolution Description: Text field description of resolution.
- Resolution Action Updated Date
- Community Board: Community board details
- BBL: Borough, Block, Lot details.
- X Coordinate (State Plane): Coordinate
- Y Coordinate (State Plane): Coordinate
- Park Facility Name: Parking facility name
- Park Borough: Parking borough name
- Latitude: Latitude details
- Longitude: Longitude details
- Location: Combination of latitude and longitude

- Dataset was filtered to obtain records with only **"Closed"** status.

- Label column **"Resolution_Time_Hours"** was created to extract delta time difference taken to close the ticket. **Outliers** were removed considering threshold of > 5 ~ 10 hours.

- Different types of noise and street complaint types were grouped under common header **"Noise"** and **"Street Issue"**.

- The input dataset was scanned to check the percentage of each complaint type present. Only those types were retained for which there were at least 5% records. Rest were removed. This was done to ensure better performance of the model to be built.

- Borough column data was cleaned and refined to remove erroneous entries.

- Open data channel type was cleaned and refined to remove erroneous entries.

- From the label column **"Resolution_Time_Hours"**, new column **"Complaint_Created_Day"** was created to identify the day of the month when the complaint was created.

- The remaining columns namely Created Date, Closed Date, Status etc. were dropped further before proceeding to the machine learning stage.

- The final model was built using the following set of attributes:

- Complaint Type (Categorical Input)
- Borough (Categorical Input)
- Open Data Channel Type (Categorical Input)
- Complaint_Created-Day (Numerical Input)
- Resolution_Time_Hours (Numerical Output Label)

The **feature selection** was finalized by trying several different combinations and calculating **feature importance score** for each predictor. The top 10 predictors with their corresponding scores are depicted below:

Name	Score
BoroughclassVec_MANHATTAN	0.376852
BoroughclassVec_BRONX	0.264449
Complaint_TypeclassVec_Noise	0.199831
BoroughclassVec_BROOKLYN	0.049985
Open_Data_Channel_TypeclassVec_PHONE	0.037339
BoroughclassVec_QUEENS	0.035270
Complaint_TypeclassVec_Illegal Parking	0.025610
Open_Data_Channel_TypeclassVec_ONLINE	0.007465
Complaint_Created_Day	0.003199

Modelling: Modelling was carried out using **RandomForestRegressor** and **GBTRegressor** supervised machine learning models which were used to analyze the highly non-linear input data and extradite patterns. Suitable metrics (RMSE) were noted.

Conclusion & Recommendation:

- The derived metrics indicated that the predictions were significantly off from the actual label.
- The entire process can be repeated with individual complaint types separately for estimation of output label rather than trying to build a generalized model encompassing whole list of available complaint types. This can give better targeted results.
- The exact threshold value of the output label which can be used to term the record as outlier needs to be determined appropriately.
- The input dataset consisted of a whole list of categorical variables which were mostly not used for modelling to restrict the number of categorical encoded inputs which would have introduced huge complexity in the model built.
- Once the range of label is determined for individual complaint types, suitable binning can be done and assigned appropriate labels. For a generalized model, this step was skipped because of lack of understanding regarding output label boundaries.