Project 3, New York Ride-Share Price Prediction

Elliott Newman: Big Data Analytics: ISE:4172

Presentation Outline

- Dataset Description
- Data collection
- Data Preprocessing
- Summary Statistics
- Linear Regression
- Decision Tree Regression
- Limitations

Problem Statement

- Ryde is a New York City start up company hoping to develop a ride-share smart phone app. Ryde currently has an algorithm that determines a ride’s estimated fare but is hoping to make it more robust. There are many variables the algorithm that does not currently consider.
- I’m a Data Scientist hired to create a better way to estimate and predict prices for fares. My technique will be used in the next major app update.

Dataset Description

- Ryde is hoping to use New York Taxi and Limousine data, from recent years, to determine prices.
- Using the nyc.gov website, I gathered Yellow Taxi records from 2019 and 2020. Using sampling and reduction techniques, I merged all the data into a smaller dataset for analysis, and for usability in Google Colab.
- The next slides will review the data collection process and techniques I used to create my initial dataset.

Data Collection Process

The NYC.gov website has datasets on fare prices from January 2019- June 2020.

The data is mined/downloaded and saved in memory of the PySpark application.

Using PySpark, both 2019 and 2020 yellow taxi data are saved in a PySpark dataframe.

Main Data Mining Techniques for Data Collection

- Getting all of the 2019 and 2020 taxi fare data required loading in 18 csv files. Here is the programming procedure I used:

1. Created each website url as a string with a for loop ('https://s3.amazonaws.com/…/yellow_tripdata_2020-01.csv')
2. Using the '!wget' command, saved each dataset
3. Read in each .csv file as a PySpark dataframe
4. Used stratified random sampling to sampled each monthly dataframe, without replacement, and saved each sampled dataframe as its own variable.
5. Merged the sampled dataframes into one dataframe
6. Saved my training/test set to local machine in a .csv file to bypass rerunning web downloading steps again

Data Processing, Derived Variables

- After getting my data imported, I went through the variable selection and data cleaning process.
- I selected recommended variables for analysis. They needed to work within a regression model.
- Dropped null values
- Removed negative trip fares/travel distance values
- Converted strings to integers- Day of Week “1” through “7” to 1-7 for One Hot encoding
- Dummy variable creation
- Derived the taxi ride time from subtracting pickup date from drop-off date

- Packages/Methods used for data preparation:
- PySpark dateformat
- PySpark StringType
- PySpark IntegerType
- .between / .otherwise method
- Functools reduce
- Pyspark.Sql

Variables Used and Created

PICKUP TIME

YEAR

MONTH

DAY OF THE WEEK

AMERICAN HOLIDAY

RUSH HOUR

TAXI RIDE DURATION

TRAVEL DISTANCE

Categorical/Time Variable represented as dummy 0/1

Quantitative Variable

Dataset

This table shows the first 5 rows of the dataset, with dummy variables created for categorical variables. These are all examples of Data from Wednesday, January 1st, 2020 around midnight

```
df = df.select('2019','Month_dummy','Day_dummy','Morning', 'Noon', 'Afternoon', 'Evening', 'Late Night', 'Duration', 'Rush Hour','Holiday', 'trip_distance', 'fare_amount')
df.show(5)
```

```
+----+--------------+-------------+-------+----+---------+-------+----------+--------+---------+-------+-------------+-----------+
|2019|   Month_dummy|    Day_dummy|Morning|Noon|Afternoon|Evening|Late Night|Duration|Rush Hour|Holiday|trip_distance|fare_amount|
+----+--------------+-------------+-------+----+---------+-------+----------+--------+---------+-------+-------------+-----------+
|   0|(12,[1],[1.0])|(7,[3],[1.0])|      0|   0|        0|      0|         1|    1125|        0|      1|         2.42|       13.0|
|   0|(12,[1],[1.0])|(7,[3],[1.0])|      0|   0|        0|      0|         1|     611|        0|      1|         1.56|        8.5|
|   0|(12,[1],[1.0])|(7,[3],[1.0])|      0|   0|        0|      0|         1|    1982|        0|      1|         3.22|       20.5|
|   0|(12,[1],[1.0])|(7,[3],[1.0])|      0|   0|        0|      0|         1|     389|        0|      1|          1.1|        6.0|
|   0|(12,[1],[1.0])|(7,[3],[1.0])|      0|   0|        0|      0|         1|    2114|        0|      1|          3.4|       22.0|
+----+--------------+-------------+-------+----+---------+-------+----------+--------+---------+-------+-------------+-----------+
```

Notes:
- I used One-Hot encoding for Months and Day of the week
- Duration of ride is in seconds.
- Holidays include Christmas Day, New Years Day, 4th of July, Thanksgiving Day, Memorial Day, and Labor Day
- Rush hour is from 7:00 a.m. - 9:00 a.m., and 4:00 p.m. - 6:00 p.m.

Data Dimensions:
252,817 Rows
11 independent Columns
1 target variable

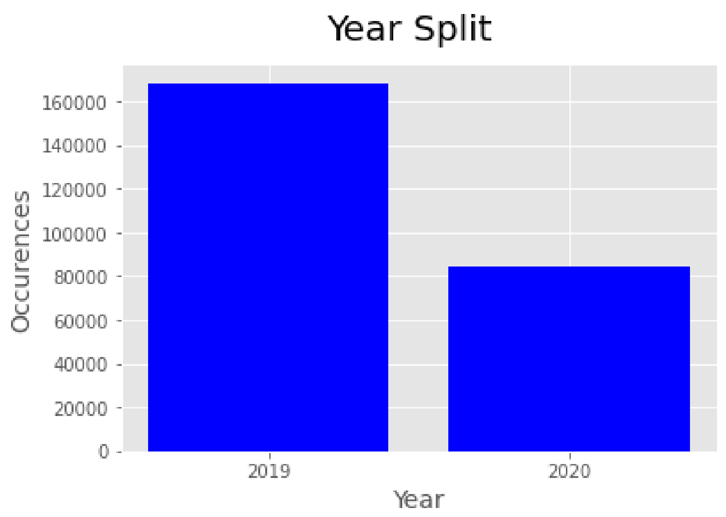
Summary Statistics-

summary	2019	Morning	Noon	Afternoon	Evening	Late Night
count	252817	252817	252817	252817	252817	252817
mean	0.6665216342255466	0.19935368270329923	0.10278976492878247	0.22072091671050 6	0.29953286369191945	0.17767001427910306
stddev	0.47145670481388074	0.3995152352936398	0.3036846949416809 3	0.4147334975454707	0.4580543168272847	0.38223547481778436
min	0	0	0	0	0	0
25%	0	0	0	0	0	0
50%	1	0	0	0	0	0
75%	1	0	0	0	1	0
max	1	1	1	1	1	1

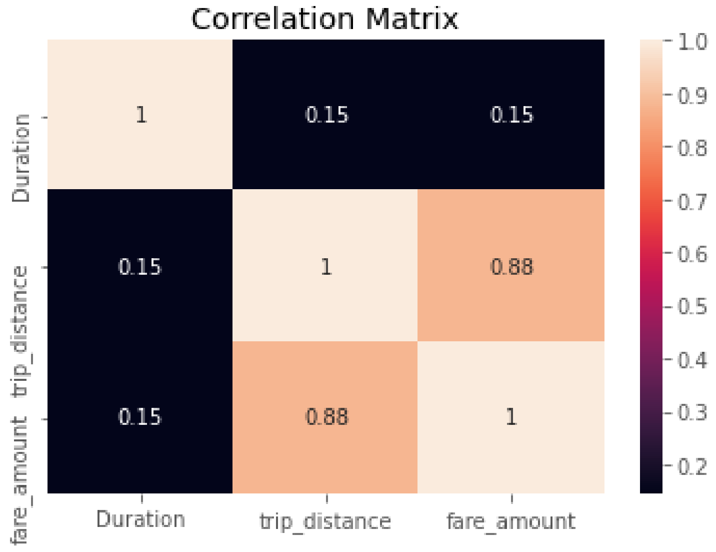
summary	Duration	Rush Hour	Holiday	trip_distance	fare_amount
count	252817	252817	252817	252817	252817
mean	1028.537361016071	0.19970176056198752	0.010058659030049403	2.971726901276407	13.141281678051705
stddev	3950.753947466595	0.3997769372457771	0.09978738294418403	3.8815553969798584	12.058597999178847
min	0	0	0	0.0	0.0
25%	394	0	0	0.98	6.5
50%	660	0	0	1.62	9.5
75%	1083	0	0	3.02	14.5
max	86390	1	1	181.53	917.0

Exploration of the Sample Set

I looked at some visualizations and other stats to get a better overview of the data I was working with, and noticeable trends



Most of the test set has data from 2019



There is a .88 correlation between miles traveled and fare price

Average fare amounts for Holidays and Days of the week. No noticeable outlier averages.

Holiday	fare_amount
0	13.133695
1	13.887959

DOW_Numb	fare_amount
1	13.344262
2	13.097132
3	13.173227
4	13.370718
5	13.268927
6	12.449302
7	13.308415

Linear Regression

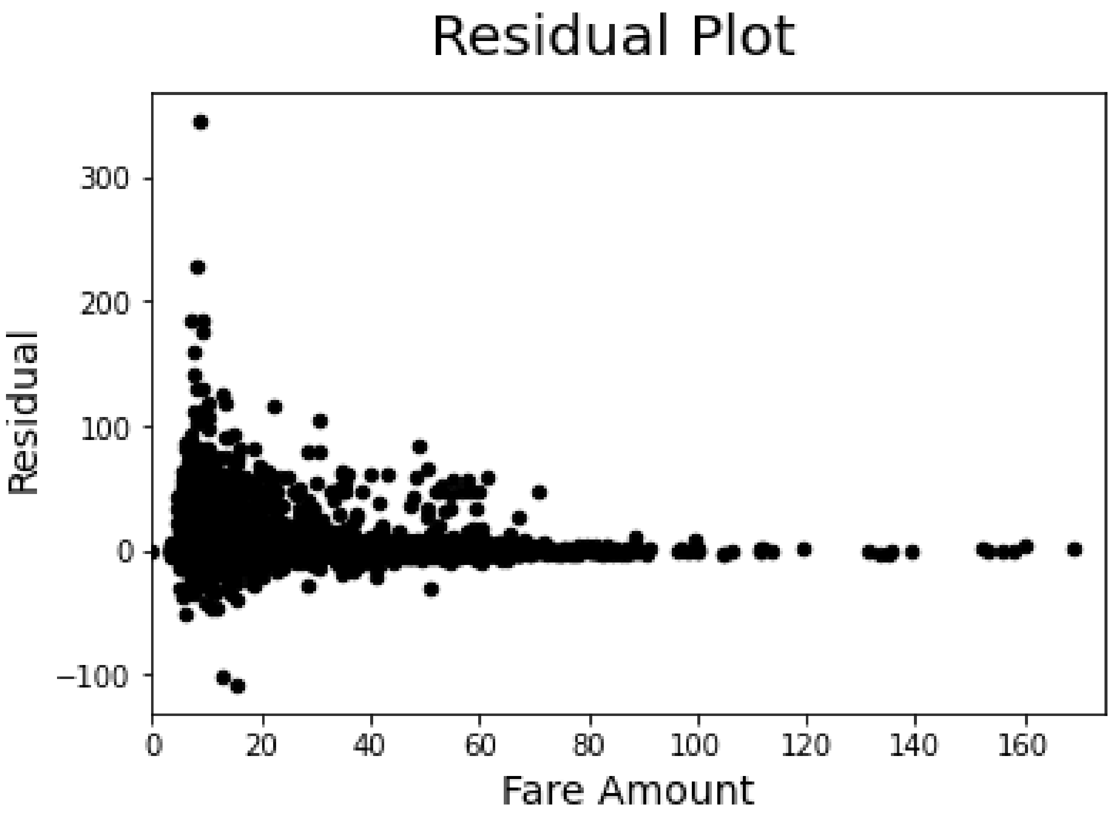
- To begin modeling, I ran a basic .7 / .3 split of a simple Linear Regression model in PySpark.
- Here are the Training and Testing results

```
--TRAINING RESULT METRICS--
rMSE: 5.925766740041358
MAE: 2.263170195450732
MSE: 35.11471145738039
R2: 0.7630959258446076
Adjusted R2: 0.7630585396853626
```

```
--TEST RESULT METRICS--
rMSE: 5.665148590383364
MAE: 2.2566045510773387
MSE: 32.09390855112262
R2: 0.7775819629482204
Adjusted R2: 0.7775000086774289
```

```
--INTERCEPT--
4.1979145617731675
```

Test Set Residuals and Coefficients




```
--COEFFICIENTS-
0.0 -0.766933098374347 -0.6077161280070266 -0.6680859571500605
-0.5517995611041213 -0.41281620391838514 -0.3126130528354665 -0.17734165496951595
-0.3732800889281817 -0.05338040603152217 0.01914699203804622 -0.05975468135459216
0.0 0.22824792416719483 0.524526650966548 0.662967722641735
0.7647777442519437 0.665448894588537 0.20587349589489454 0.6215984180729297
1.214492500798028 1.2417664562964559 0.5606208586243349 0.05952869843483893
4.674236537896589e-05 0.27755033193129736 -0.678220635757679 2.740025894038772
```

The residual plot is used to look at the dispersion of predicted fare values compared to residuals. A residual plot is “good” for your model if there aren’t clear patterns, they residuals symmetrically distributed, and near 0 on the y axis. I considered the plot in my final model evaluation slide.

Coefficients represent the effect and positive or negative direction of features on the predicted fare value.

Decision Tree Regression

- I also ran a Decision Tree Regressor. Decision trees are classification or regression algorithms that find a predicted value using a tree structure. Documentation is available in Spark for decision trees. Here are the results of the model.

```
--TEST RESULT METRICS--
RMSE: 5.34134
MSE: 28.5299
R^2: 0.79817
MAE: 1.76543
```

```
DT Model depth: 5
DT Model Numb nodes: 63
```

```
dt_model.featureImportances
```

```
SparseVector(29, {3: 0.0003, 7: 0.0, 8: 0.0005, 9: 0.0002, 24: 0.0017, 25: 0.1444, 28: 0.8529})
```

Features 25 and 28 are Duration and Trip Distance. They have high importance values for features of the model.

Final Model Comparison

- Linear Regression
  - In our case, the model had a Lower R^2 and higher error values
  - This model has a dispersed pattern in the residuals, though some large residuals are clustered near smaller fare values.
  - Easily interpretable for business audiences (XW + b, intercept + coefficients)
  - Can use backwards Linear Regression to “improve” the model and look for statistically significant features
  - Works well for linear data
  - Training Time Complexity: $O((p^2)n)+p^3)$

- Decision Tree Regression
  - Higher R^2 Value, lower errors
  - Metrics for training set error evaluation not as readily available in the DecisionTreeRegressor() function as Linear Regression
  - Can describe feature Importance
  - Tunable parameters to improve model- such as tree depth
  - Can work well for non-linear data
  - Training Time Complexity: $O((n^2)p)$

Final Evaluation

Overall, there are benefits of both models approaches. The decision tree has a higher R^2 and lower test error metrics. Based on this alone, it would qualify as the "better" model.

References

- Blog.americansafetycouncil.com
- Classmates
- Dad and aunt, Brooklyn natives
- Matplot Documentation
- Mygeodata.cloud
- Nytimes.com
- Spark Documentation
- Stackoverflow
- Towardsdatascience.com