Uber/Lyft Dataset Analysis

Name Chris Myhre

ABSTRACT
This project is about analyzing a dataset of Uber and Lyft rides in Boston, MA from November and December 2018. This project seeks to analyze the dataset for potential patterns via python and other tools.

I. INTRODUCTION

The dataset used consists of Uber and Lyft rides in Boston, MA from November 2018 to December 2018. This dataset includes ride data like the type of cab, provider name, and weather data. The raw dataset has 57 columns and over 600,000 rows.

II. BACKGROUND

Boston, MA is a busy city with a population of approximately 700,000 residents. A very busy metropolitan with many events and businesses around. In a city like this, services like Uber and Lyft are popular and profitable. One question this project seeks to answer is that does the weather impact sales of ride services like these? During months like November and December, the weather is generally colder and rainy.

III. EXPLORATORY ANALYSIS

This dataset contains 57 columns and over 600,000 rows. The dataset contains information regarding ride ids, price, month, day and hour of ride, and weather data of the day and the immediate moment of the rides.

Table 1: Data Types

Variable Name	*Data Type*
Id	String (removed column)
Timestamp	String (removed column)
Hour	Integer
Day	Integer
Month	Integer
Datetime	String (removed column)
Timezone	String (removed column)
Destination	String
Cab_type	String
Product_id	String (removed column)
Name	String
Price	Float (dropped column)
Distance	Float
Surge_multiplier	Float
Latitude	Float (removed column)
Longitude	Float (removed column)
Temperature	Float
apparentTemperature	Float (removed column)
Short_summary (renamed weather)	String
Long_summary	String (removed column)
precipIntensity	Float
precipProbability	Float
Humidity	Float
windspeed	Float
windGust	Float
windGustTime	Integer (removed column)
Visibility	Float

temperatureHigh	Float
temperatureHighTime	Integer (removed column)
temperatureLow	Float
temperatureLowTime	Integer (removed column)
apparentTemperatureHigh	Float (removed column)
ApparentTemperatureHighTime	Integer (removed column)
apparentTemperatureLow	Float (removed column)
apparentTemperatureLowTime	Integer (removed column)
Icon	String (removed column)
dewpoint	Float
Pressure	Float
windBearing	Integer
cloudCover	Float
uvIndex	Integer
Visibility.1	Float (removed column)
Ozone	Float
sunriseTime	Integer (removed column)
sunsetTime	Integer (removed column)
moonPhase	Float
precipIntensityMax	Float
uvIndexTime	Integer (removed column)
temperatureMin	Float (removed column)
temperatureMinTime	Integer (removed column)
temperatureMax	Float (removed column)
temperatureMaxTime	Integer (removed column)
apparentTemperatureMin	Float (removed column)
apparentTemperatureMinTime	Integer (removed column)
apparentTemperatureMax	Float (removed column)
apparentTemperatureMaxTime	Integer (removed column)

IV. METHODS

Once the dataset was picked a copy of the dataset was made. In this copy, I deleted unnecessary rows that had irrelevant information or repeated information. Then the data set was uploaded to Google Drive and a .ipynb file was created on Google Drive and edited in Google Colab to write a EDA to analyze the data.

A. Data Preparation

Unnecessary columns were deleted to shrink the number of columns from 57 to 28. The categorical columns had binary dummy columns created to track each kind of value each categorical variable could be.

B. Experimental Design

At first a Logistic Regression model was trained on the data and got a accuracy of 96.92%. Then after realizing that the number of entries for each type of weather was imbalanced, SMOTE was applied to even it out. After applying SMOTE, the accuracy of the model remained unchanged. Must've not been that unbalanced.

C. Tools Used

The following tools were used for this analysis: Python v3.12.2 running on a Google Colab machine was used for all analysis and implementation. In addition to base Python, the following libraries were also used: Pandas 0.18.1, Numpy 1.11.3, Matplotlib 1.5.3, Seaborn 0.7.1, and SKLearn 0.18.1. These tools were selected to allow the data to be analyzed, graphed and used for training a Linear Regression model.

V. RESULTS

A. Classification Measures/ Accuracy measure

The model performed well with a 96.92% accuracy for no smote and with smote linear regression models.

B. Discussion of Results

Both models performed about the same for accuracy due to the dataset being balanced. In figure 1, the confusion matrix didn't pick up any correlations between price and weather. Figure 2 shows the confusion matrix of the model's accuracy which was a high 96.92%. Figure 3 shows the distribution of the number of rides ordered for each weather condition and surprisingly, rainy days only make up the minority of rides orders. Initially the opposite was expected but even the correlation matrix supports the fact that weather doesn't impact ride demand therefore not impacting ride price.

C. Problems Encountered

Some problems I ran into were mismatches in number of entries among variables. Google Gemini was used to correct one of the .ipynb notebook cells to make the model work correctly. Another struggle was time, it took Google Colab 25 minutes to train a model both times I trained a model.

D. Limitations of Implementation

Some limitations of my model would be the speed at which the models can be trained at and the RAM usage of the training, Google Colab had higher paid tiers that were unavailable to the project, so the highest free tier had to be enough. Because Google Colab hardware was used, the model was limited to ten thousand iterations which wasn't enough due to the size of the dataset, over 600,000 rows.

E. Improvements/Future Work

Some improvements to be made is to find a dataset that has more than just two months of data and to train the model on my local computer for faster training times. Using my own hardware would allow me to run more than ten thousand iterations of training to potentially increase the accuracy of the model slightly. 96.92% is already high so not much room for improvement there.

VI. CONCLUSION

Overall, the model did well predicting whether a ride price was low or not at 96.92% accuracy. Some improvements to the model would be a different dataset that included all twelve months of the year instead of the final two. Running the training locally instead of online would yield better results and faster training times.

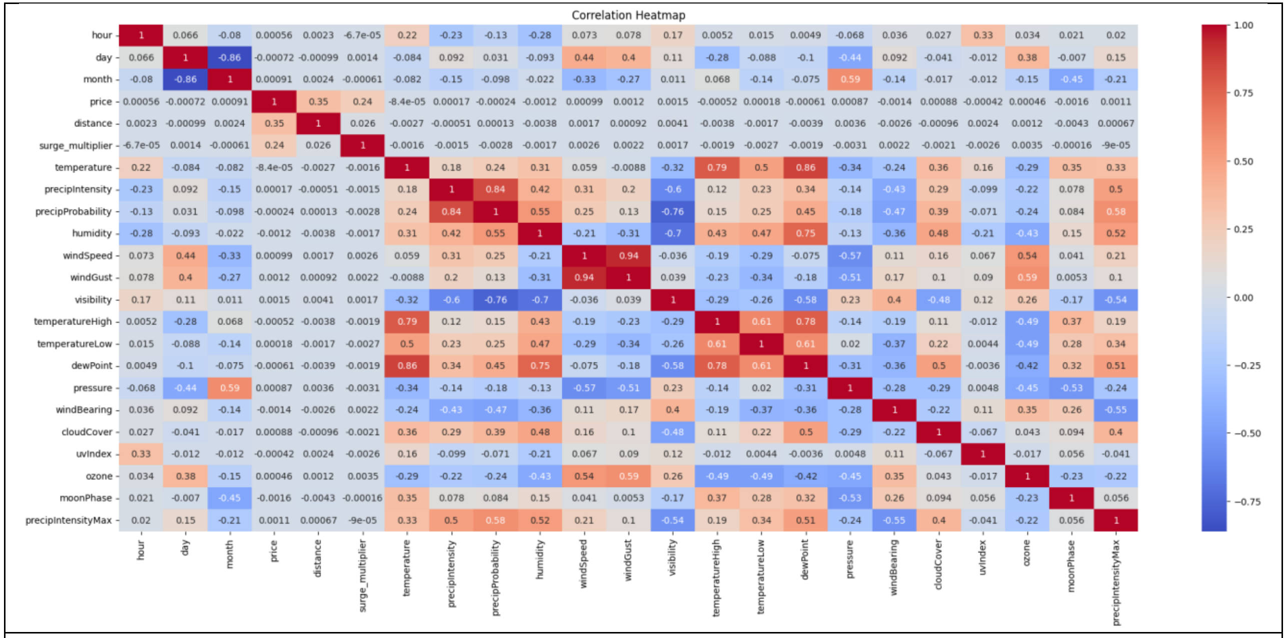

Figure 1: Python EDA Correlation Matrix

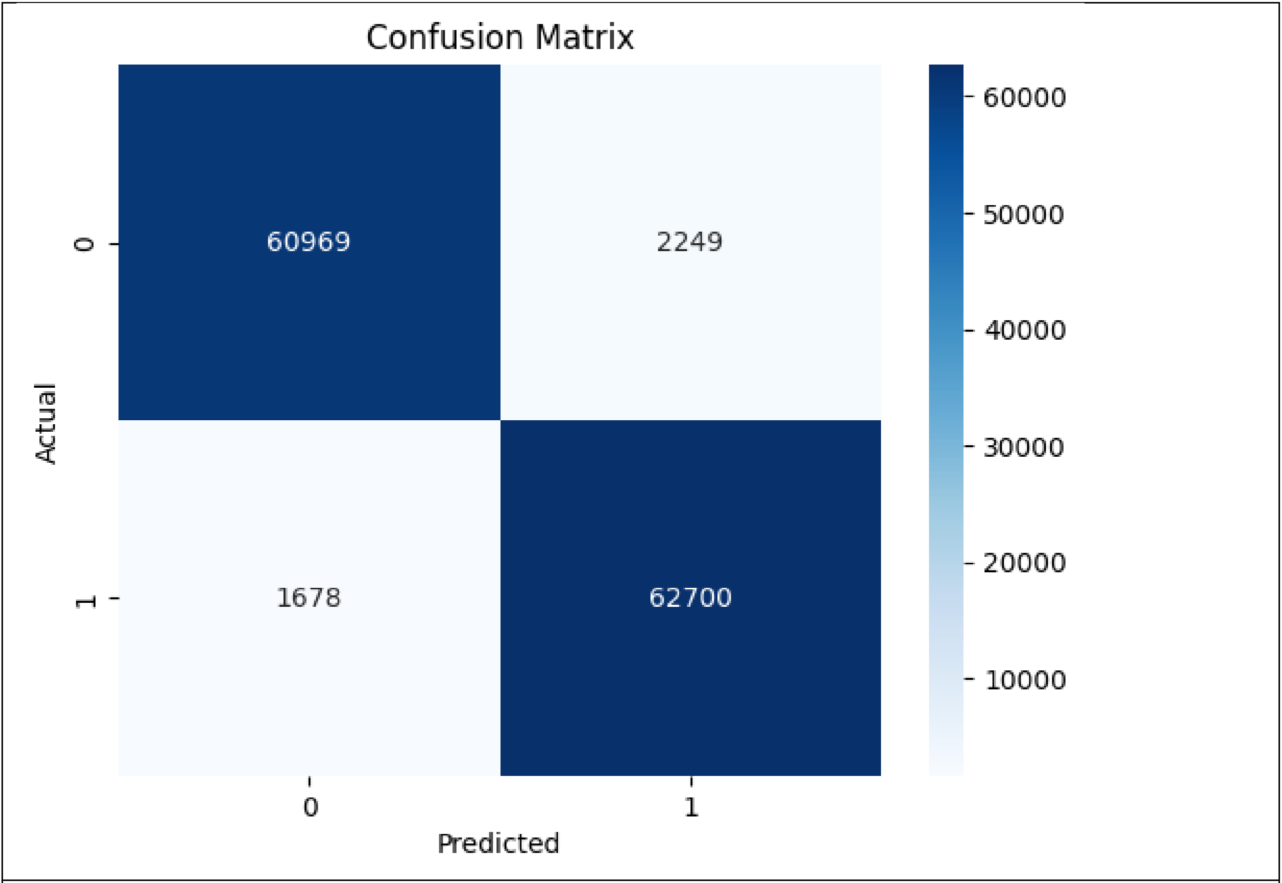

Figure 2: Python EDA Model Confusion Matrix of Accuracy

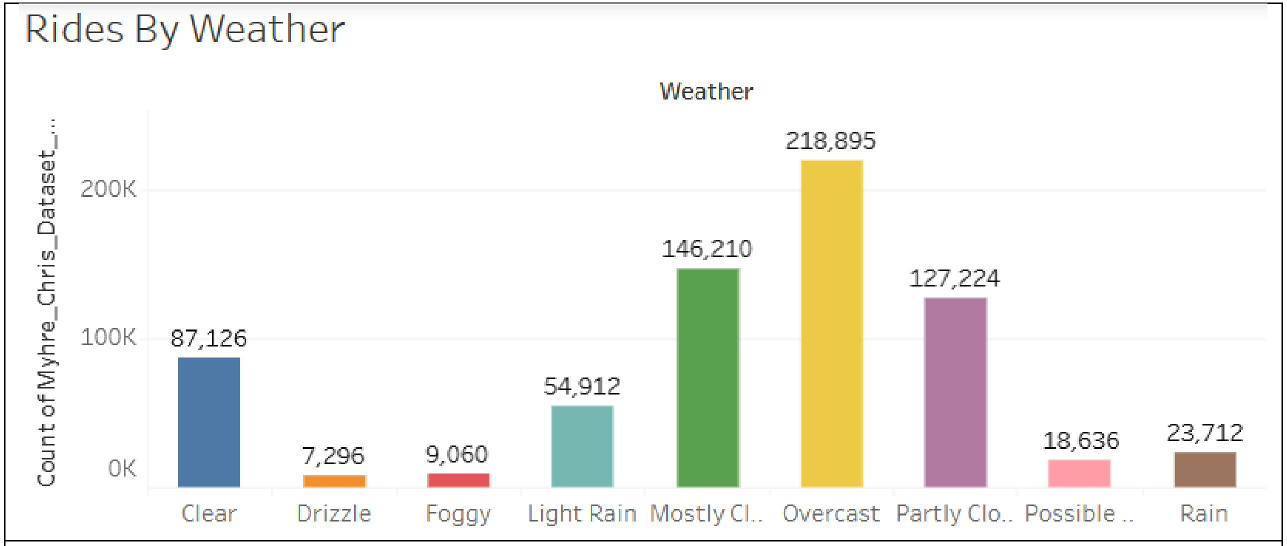

Figure 3: Tableau Visualization of the Distribution of Rides per Weather Condition