Course:

CIS 4130 Semester Project

Name:

Sherina Zheng

Email:

zhengsherina@gmail.com

Modeling NYC FHV (Uber/Lyft) Trip Data (2022-2023)

## Description of the Data Set

I propose to analyze and model the "NYC FHV (Uber/Lyft) Trip Data Expanded (01/2022-06/2023)" dataset, only a short period of time frame was picked because it’d take a long time to process if it were to be analyzed through databricks. This dataset contains comprehensive information about every For-Hire Vehicle (FHV) trip in New York City spanning the years 2022 to 2023. The dataset includes attributes such as:

0 hvfhs\_license\_num object

1 dispatching\_base\_num object

2 originating\_base\_num object

3 request\_datetime datetime64[ns]

4 on\_scene\_datetime datetime64[ns]

5 pickup\_datetime datetime64[ns]

6 dropoff\_datetime datetime64[ns]

7 PULocationID int64

8 DOLocationID int64

9 trip\_miles float64

10 trip\_time int64

11 base\_passenger\_fare float64

12 tolls float64

13 bcf float64

14 sales\_tax float64

15 congestion\_surcharge float64

16 airport\_fee float64

17 tips float64

18 driver\_pay float64

19 shared\_request\_flag object

20 shared\_match\_flag object

21 access\_a\_ride\_flag object

22 wav\_request\_flag object

23 wav\_match\_flag object

The dataset provides information about the total miles for the passenger trip, the total time in seconds for the passenger trip, the base passenger fare before tolls, tips, taxes, and fees, the total amount of all tolls paid in the trip, the total amount collected in the trip for the Black Car Fund, the total amount collected in the trip for NYS sales tax, the total amount collected in the trip for NYS congestion surcharge, and the airport fee of $2.50 for both drop off and pick up at LaGuardia, Newark, and John F. Kennedy airports.

Moreover, the dataset includes the total amount of tips received from the passenger, the total driver pay (not including tolls or tips and net of commission, surcharges, or taxes), the flag indicating whether the passenger agreed to a shared/pooled ride and whether the passenger shared the vehicle with another passenger who booked separately at any point during the trip.

## Intended Modeling Objective

My primary objective is to build a predictive model that can forecast demand for For-Hire Vehicles in different areas of New York City. This model will serve multiple purposes:

1) Demand Forecasting: We will use historical trip data to predict the tips received for FHV services in specific boroughs and neighborhoods at different times of the day and week.

2) Supply Optimization: By understanding demand patterns, we aim to help FHV service providers (e.g Uber, Lyft) optimize their supply of vehicles in high-demand areas during peak hours, reducing the “risk” of losing tips in the case where there are traffic jams, etc.

3) Pricing Strategy: The model will enable FHV companies to adjust pricing dynamically based on demand, improving both passenger experience and driver earnings.

4) Traffic and Route Optimization: Analyzing trip data will provide insights into traffic patterns and optimal routes, allowing drivers to minimize trip durations.

By modeling NYC FHV trip data, I aim to support the efficient operation of For-Hire Vehicle services in New York City, benefiting both passengers and service providers. This project aligns with the goal of improving urban mobility and transportation services in the city while enhancing the profitability of FHV companies.

\*\*\* URL/Location for downloading the data: \*\*\*

https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page

## Summary and Conclusions

Upon analysis of multiple data endpoints, I started to build a more foundational acknowledgement on my own project - The datasets used were found on an official website, where it seemed to have a clear-out outprint of their public data. But I realized it still needed a thorough data cleaning before it could be used, such as dropping the NULL and duplicate records.

Overall, I generalized 3 graphs with Pandas, Matplotlib, and Seaborn, each of which tells a different story. The first chart summarizes the total counts of orders made with 4 FHV companies, indicating that only HV0003: Uber and HV0005: Lyft, have recent records of orders whereas HV0002: Juno and HV0004: Via, none orders are found. This implies that the latters are less demanded in the public car service market, reflecting the fact that our day-to-day use of apps are mainly Uber or Lyft. The second chart compares the waiting time with the tips predicted, which aims to define the impacts of waiting duration upon customers’ tips. Expected relationship is a close linear regression where waiting time has a negative effect on tips, but there is no clear relationship as such, and most data points are squeezed at the very bottom left, with some outliers spanning at bottom right. I believe this was due to the lack of training on ample datasets, again, I plan to add more data when time is more flexible. The third chart compares between the actual tips given and the predicted tips. Prior to the given condition that tips below 15% of the base passenger fee, the chart is expected to display a positive relative with a slope approximately equal to 0.5. But it fails to predict accurate projections on actual tips.

In conclusion, this model is far from complete and placed into use. But it’s anticipated to be further trained with more datasets, ideally using Amazon EMR instead of mostly databricks (because it has been free) so larger data can be launched.

## 

## Github Handlings

This semester-long project has been posted onto my Github repository, which aims to grow my programming portfolio faced with the public. Everyone is welcomed to download and check this project, even though it still is in the immature phase where the data used to analyze and predict is within only a very short time frame, which is limited by the time we have been given to optimize it.

But I plan to expand the data scale in the future, and create a more sensible model in order to boost the prediction accuracy.

Github URL: https://github.com/Sherina-Zheng/Modeling-NYC-FHV-Uber-Lyft-Trip-Data-2022-2023-/tree/main

## 

## 

## Appendix I: Code Documented for Data Acquisition

# Get a list of files in a dataset

From the website of NYC Taxi & Limousine Commision

# Download the dataset using curl command

- FHV 6 Months in 2023:

curl -L -o fhv\_tripdata\_2023-06.parquet  https://d37ci6vzurychx.cloudfront.net/trip-data/fhv\_tripdata\_2023-06.p

# Copy the file to Amazon S3 storage (assuming you have a bucket named fhv-data-sz with /landing folder

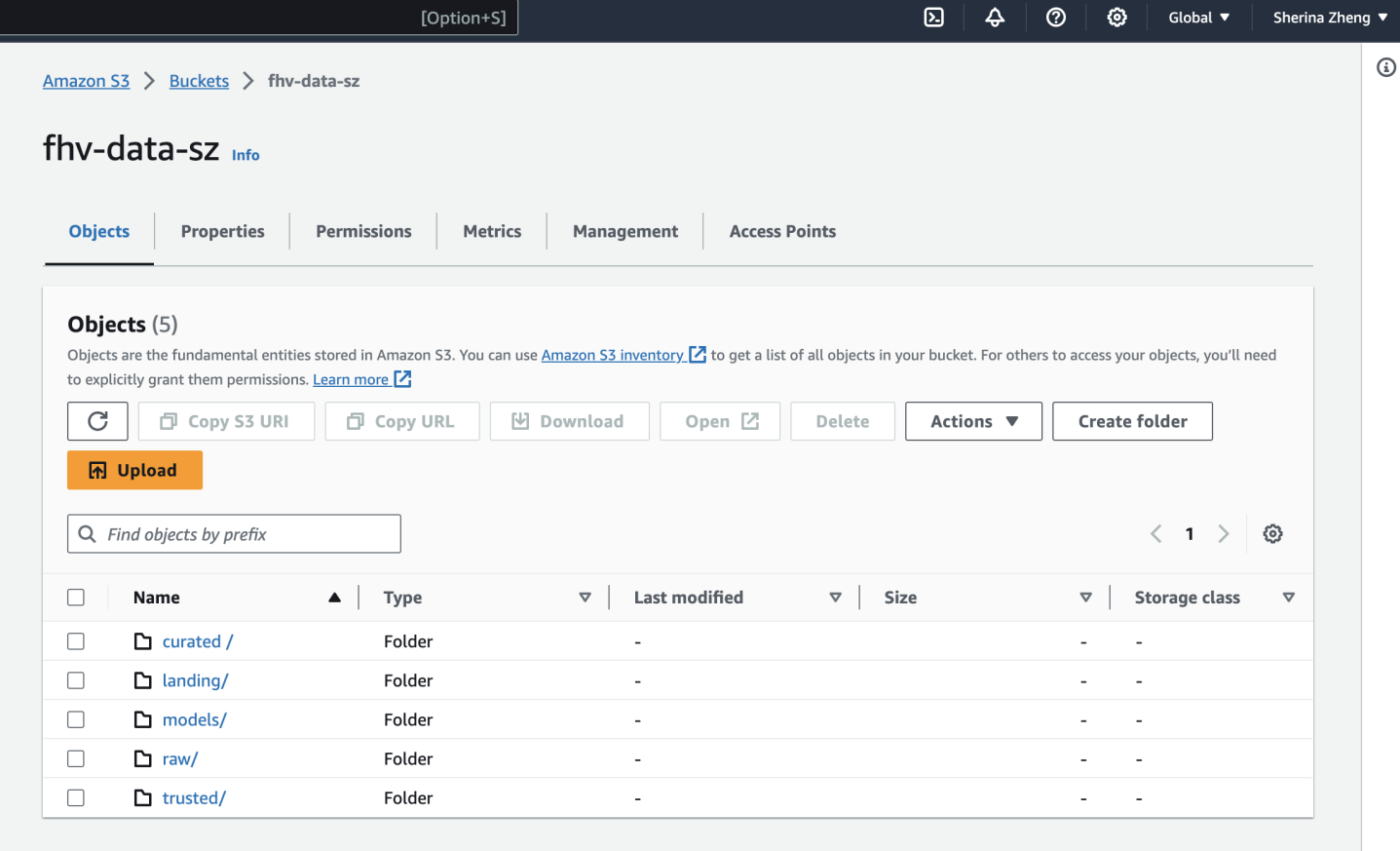
- FHV 6 Months in 2023:

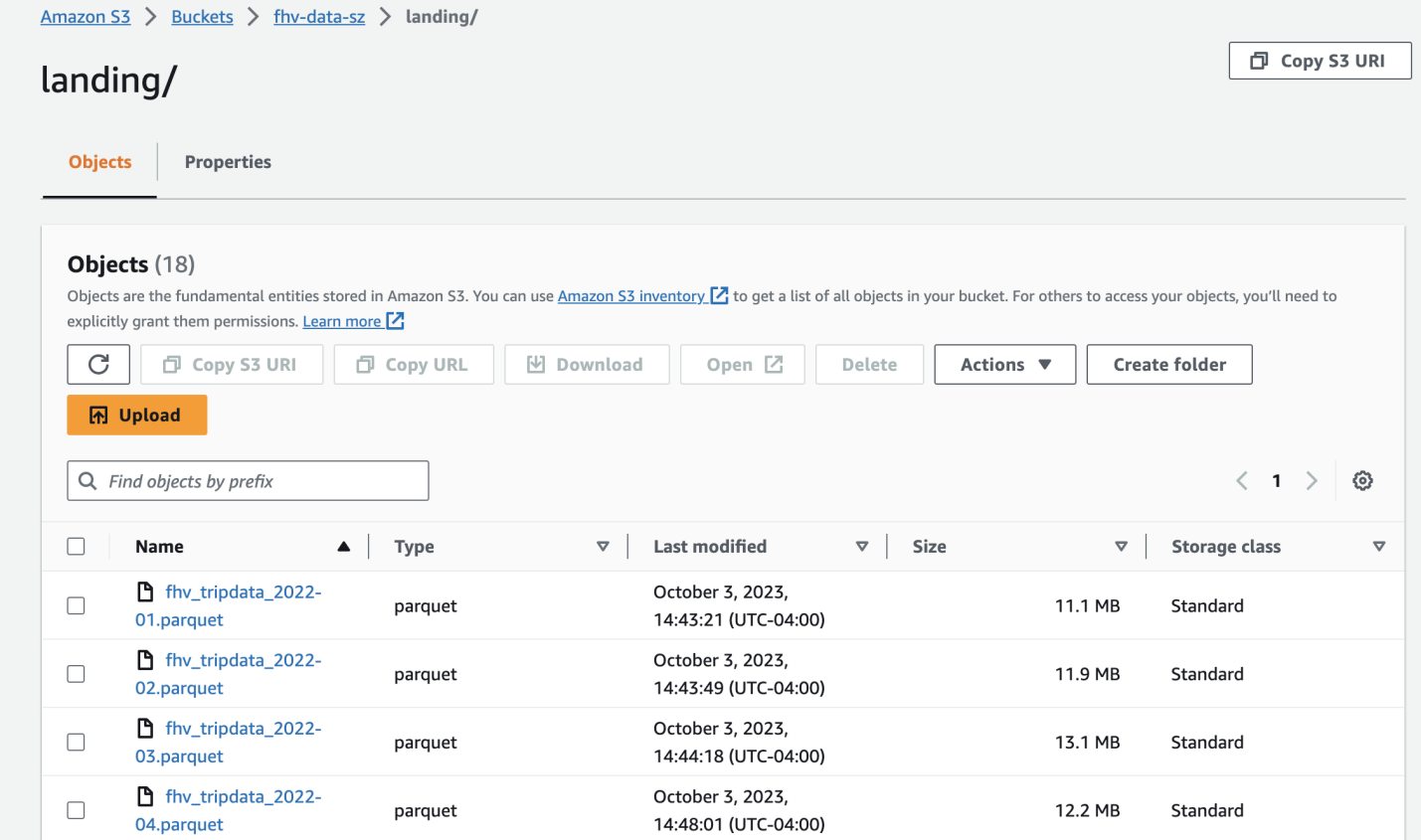
aws s3 cp  fhv\_tripdata\_2023-06.parquet    s3://fhv-data-sz/landing/fhv\_tripdata\_2023-06.parquet

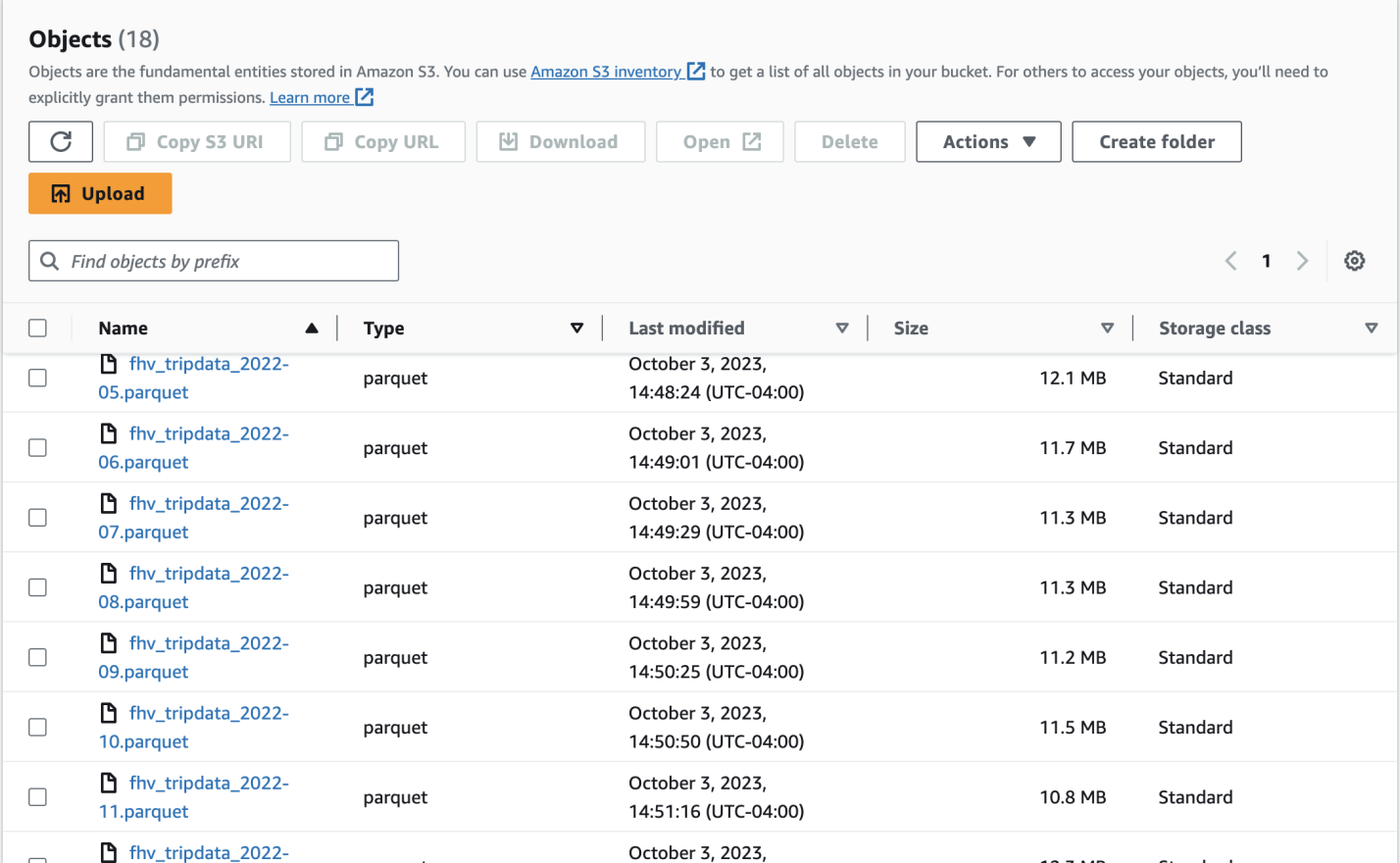
# Then remove the file from the local file system

rm fhv\_tripdata\_2023-06.parquet

# Screenshots For AWS S3 Storage files Saved on Buckets







## Appendix II: Code Documented for Data Exploratory

# Installing additional modules with %pip

%pip install pandas numpy fsspec s3fs boto3 seaborn

%pip install pyarrow fastparquet

%pip install fsspec s3fs

import pandas as pd

import boto3

import fastparquet

import pyarrow

# Set AWS credentials

aws\_access\_key = 'AKIAY5HJQS '

aws\_secret\_key = 'ed+ByKIfWnr0ES+9Kf1n '

# Create a Boto3 S3 client

s3 = boto3.client('s3', region\_name='us-east-2', aws\_access\_key\_id=aws\_access\_key, aws\_secret\_access\_key=aws\_secret\_key)

# Use s3 to access S3 bucket and retrieve the Parquet file

bucket\_name = 'fhv-data-sz'

object\_key = 'landing/fhv\_tripdata\_2023-01.parquet'

response = s3.get\_object(Bucket=bucket\_name, Key=object\_key)

# Read in s3 bucket file

fhv\_df = pd.read\_parquet("s3://fhv-data-sz/landing/fhv\_tripdata\_2022-01.parquet", sep='\t', on\_bad\_lines='skip')

# Get top 10 records

fhv\_df.head

# Show some statistical data

fhv\_df.info()

fhv\_df.describe()

# Which columns contain null values

print("Columns with null values")

print(fhv\_df.columns[fhv\_df.isnull().any()].tolist())

# reviews\_df.loc[:, reviews\_df.isnull().any()]

# How many records have null values?

print("Rows with null values:", fhv\_df.isnull().any(axis=1).sum())

# Drop records with nulls in certain columns

fhv\_df = fhv\_df.dropna(axis=0, subset=['star\_rating'])

# Find duplicate rows

fhv\_df.duplicated()

# Drop it if any

fhv\_df = fhv\_df.duplicated.drop\_dplicates()

# Import additional libraries needed to conduct visualization

import seaborn as sns

import matplotlib as plt

# Draw a boxplot to see if there are any outliers

sns.boxplot(fhv\_df)

# Plot a scatter diagram

fhv\_df.plot(kind='scatter', x='trip\_duration', y='trip\_fare')

plt.show

# Create a pairplot to see all linear relationships

sns.pairplot(data=fhv\_df)

## 

## Appendix III: Code Documented for Data Cleaning on Jupyter Notebook

# Drop duplicates if any

sdf = sdf.drop\_duplicates()

# Dropping any null values on 'on\_scene\_datetime' column where we will create the fearture

sdf = sdf.dropna(subset=['on\_scene\_datetime'])

## 

## Appendix IV: Code Documented for Feature Engineering

# Create an indexer for the three string-based columns

indexer = StringIndexer(inputCols=["hvfhs\_license\_num"], outputCols=["license\_index"])

sdf = indexer.fit(sdf).transform(sdf)

# Create an encoder for the three indexes

encoder = OneHotEncoder(inputCols=["license\_index", "PULocationID", "DOLocationID"],

outputCols=["Vendor", "PUVector", "DOVecotor"], dropLast=False)

sdf= encoder.fit(sdf).transform(sdf)

# Create Binarizers for needed columns

# This binarizer states if there was congestion or not, if yes, then the driver might have a higer possibility to be late, vice versa

congestion\_binarizer = Binarizer(threshold=0, inputCol="congestion\_surcharge", outputCol="if\_congestion")

sdf = congestion\_binarizer.transform(sdf)

# This binarizer tests if the driver was on scene on time as planned,

# by comparing the scheduled pickup\_datetime and the actual on\_scene\_datetime

sdf = sdf.withColumn("if\_on\_time",

when(col("on\_scene\_datetime") <= col("pickup\_datetime"), 1.0)

.otherwise(0.0))

# Calculate the time difference between request\_datetime and pickup\_datetime

# to see how long actually the customer had to wait for the driver to arrive

sdf = sdf.withColumn("waiting\_time\_seconds",

(unix\_timestamp(col("pickup\_datetime")) - unix\_timestamp(col("request\_datetime"))))

sdf = sdf.withColumn("waiting\_time\_minutes", col("waiting\_time\_seconds") / 60)

# Sett up a lable representing the percentage of the tips

sdf = sdf.withColumn("tipPercent", col("tips") / col("base\_passenger\_fare") )

# Determine if tips are considered "Good" or "Fair"

# Logistic Regression is good at predicting a binary output so we'll eliminate the third classifier

# sdf = sdf.withColumn("tipQuality",

# when(col("tipPercent") > 0.15, "Good")

# .when((col("tipPercent") >= 0.10) & (col("tipPercent") <= 0.15), "Fair")

# .otherwise("Poor"))

# Instead, use:

sdf = sdf.withColumn("tipQuality",

when(col("tipPercent") > 0.15, "Good")

.when(col("tipPercent") <= 0.15, "Fair"))

# Create an assembler for the individual feature vectors and the float/double columns

assembler = VectorAssembler(inputCols=['Vendor', 'PUVector', 'DOVecotor', 'tipPercent', 'if\_congestion', 'if\_on\_time', 'waiting\_time\_minutes'],

outputCol='features')

# Apply the assembler to the DataFrame

sdf = assembler.transform(sdf)

## Appendix V: Code Documented for Data Visualization

# pip install seaborn wordcloud

# Import additional libraries needed to conduct visualization

import seaborn as sns

import matplotlib.pyplot as plt

import matplotlib.pyplot as plt

# Transform Spark Dataframe into Pandas Dataframe

df = test\_results.select('tips', 'tipPercent', 'waiting\_time\_minutes','prediction').toPandas()

# Plotting histogram using Pandas and Matplotlib

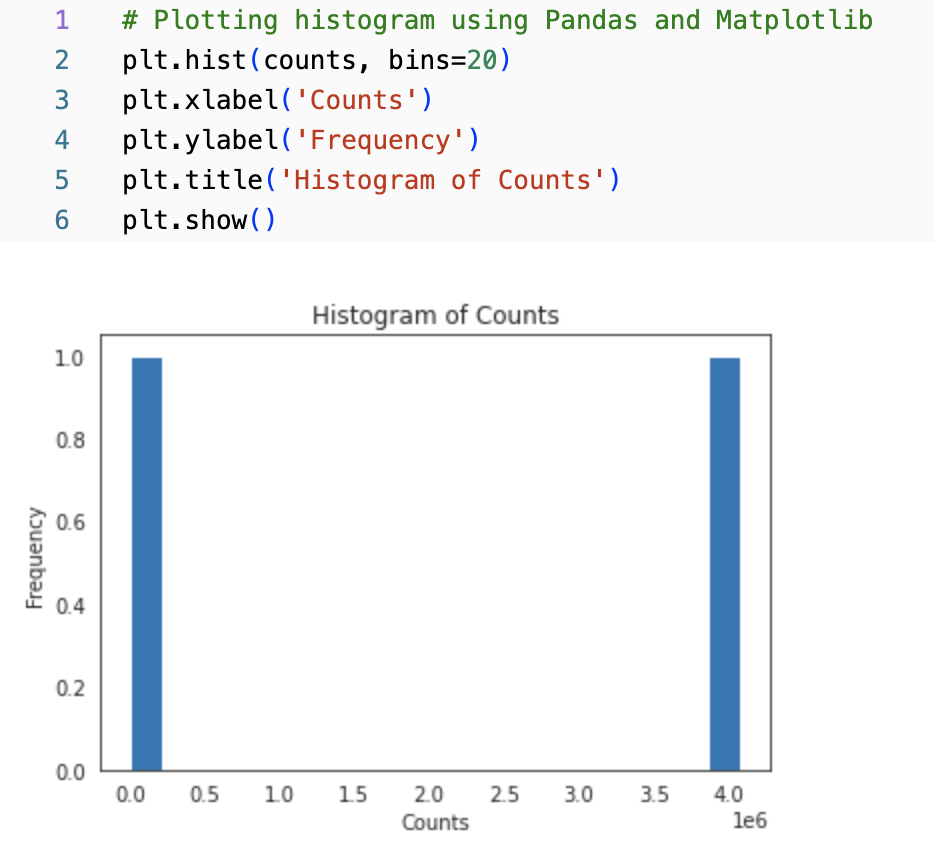
plt.hist(counts, bins=20)

plt.xlabel('Counts')

plt.ylabel('Frequency')

plt.title('Histogram of Counts')

plt.show()



# Draw a boxplot to see if the waiting time is a good factor of the tips

# Consider only 500 records

subset\_df = df.head(500)

# Scatter plot to visualize the relationship

plt.figure(figsize=(8, 6))

plt.scatter(subset\_df['waiting\_time\_minutes'], subset\_df['prediction'], alpha=0.5)

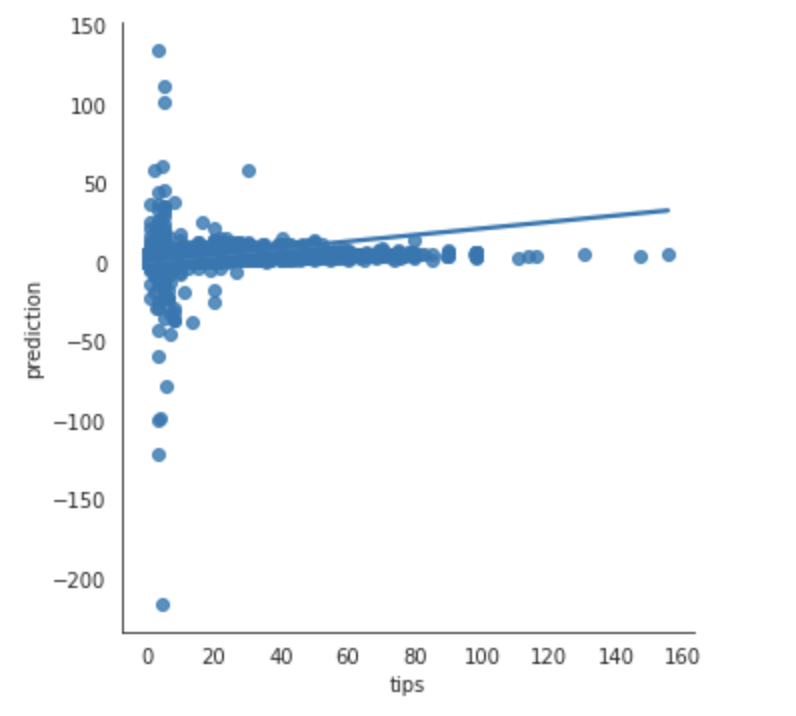
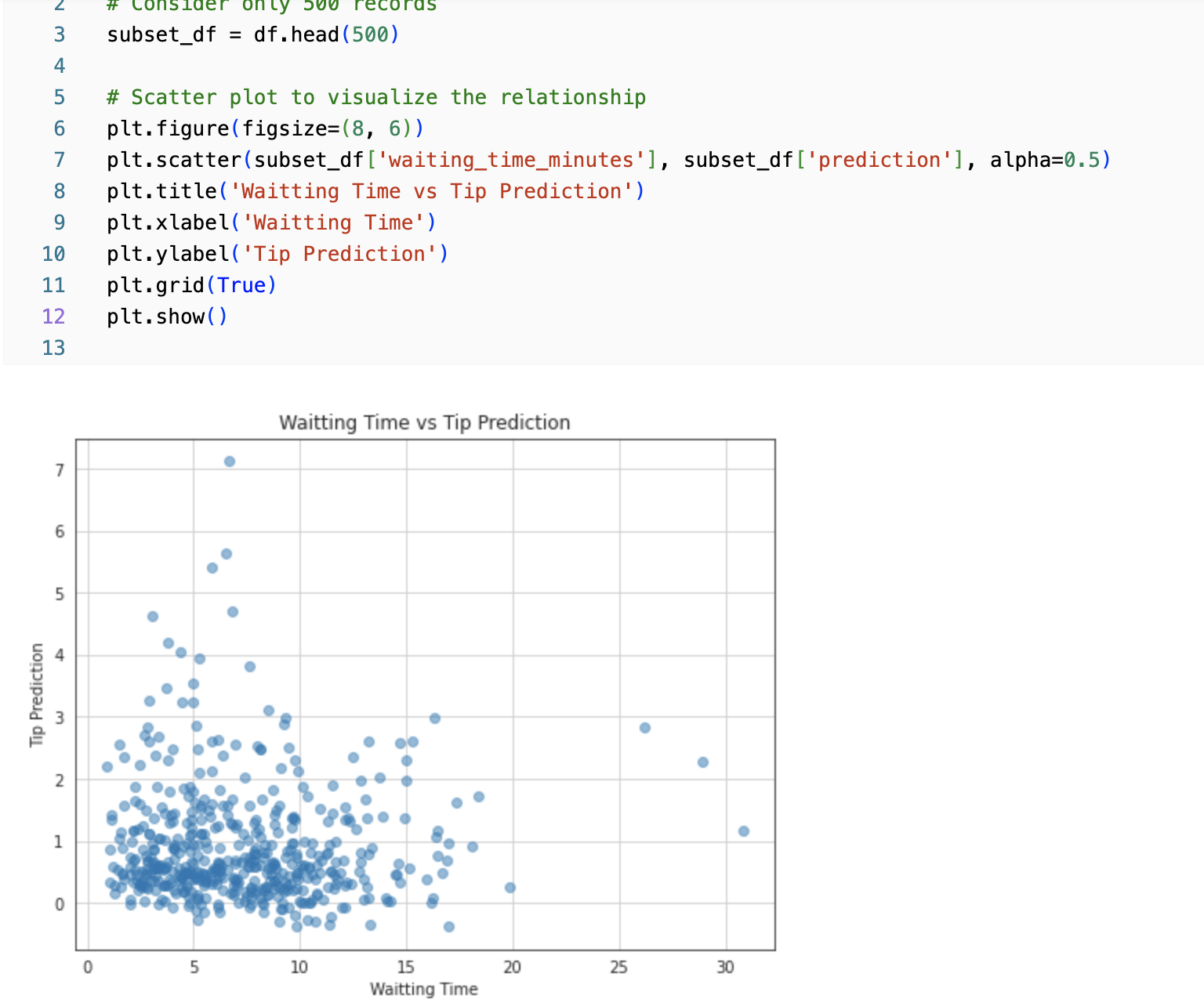
plt.title('Waiting Time vs Tip Prediction')

plt.xlabel('Waiting Time')

plt.ylabel('Tip Prediction')

plt.grid(True)

plt.show()



# Plot tips against predicted tips (predictions), see graph on top right

import seaborn as sns

# The Spark dataframe test\_results holds the original 'tips' as well as the 'prediction'

# Select and convert to a Pandas dataframe

df = test\_results.select('tips','prediction').toPandas()

# Set the style for Seaborn plots

sns.set\_style("white")

# Create a relationship plot between tip and prediction

sns.lmplot(x='tips', y='prediction', data=df)

# Set the style for Seaborn plots

sns.set\_style("white")