Final Term Project

CS5764 Information Visualization

Prepared by:
Manim Tirkey
mtirkey@vt.edu

Instructor:
Reza Jafari

Date of Submission:
12/06/2023

Table of Contents

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

Table of Figures and Tables

Tables

Figures

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

8	Statistics of cleaned features
9	Multivariate Kernel Density Estimate between Numerical Features
10	Distribution Plot of Fare Amount
11	Count plot of Passenger Count
12	Filled KDE Plot of Tip Amount
13	Stacked Barplot of Payment Type by Vendor ID
14	Grouped Bar Plot of Total Amount by Rate Code ID and Vendor ID
15	Boxen plot of Total Amount vs Payment Type
16	Regression Plot between Total Amount and Trip Duration
17	Violin Plot of Total Amount per Day of Week
18	Rug Plot for Pickup Hour
19	Hexbin plot of Tip Amount vs Total Amount
20	Line Plot of Trip Distance vs Trip Duration
21	Heatmap of Fare Amount - Passenger Count vs Pickup Day
22	Strip Plot of Tip Amount by Day of Week
23	Cluster Map of Total Amount by Payment Type and Pickup Day
24	Swarm Plot of Trip Distance vs Payment Type
25	Histogram with KDE of Trip Duration
26	Joint plot of Congestion Surcharge vs Trip Duration

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

27	QQ Plot of Target Variable - Total Amount
28	Pair plot between Trip Duration, Trip Distance and Total Amount grouped by Rate Code ID
29	Bivariate Kernel Density Plot between Trip Duration and Pickup Hour
30	3D plot (with Contour Plot) of Trip Distance, Total Tip Amount and Total Fare Amount aggregated by pickup location
31	Subplot of Count plots of Number of Pickups and Drop offs made on each day of week
32	Subplot - Pie Plot of Top 10 Most Frequent Pickup and Drop off locations
33	Subplot - Comparison between Pickup and Drop off Hours
34	Subplot - Contour plot comparison of various features
35	Dashboard - Dataset Description
36	Dashboard - Dataframe Cleaner
37	Dashboard - Cleaned Dataframe
38	Dashboard - PCA Analysis
39	Dashboard - PCA Analysis tracing component with >95% Explained Variance
40	Dashboard - Outlier Detector
41	Dashboard - Generating Boxplot after we remove Outliers

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

42	Dashboard - Normality Tester Dashboard - Selecting Different Normality Tests
43	Dashboard - Plotting Histograms for Features which are not Normal and transforming them using Box-Cox transformation
44	Dashboard - Pearson Heatmap
45	Dashboard - Selecting multiple features to dynamically calculate the correlation heatmap
46	Dashboard - Pickup Hour Graph
47	Dashboard - Line plot of Aggregated Total Amount Data by Weekday
48	Dashboard - NYC Borough Analysis
49	Dashboard - Area Plots for most frequented pickup and drop off locations
50	Dashboard - Count Plots for Pickups and Dropoffs made at EWR Borough
51	Dashboard - Taxi Price Estimator
52	Dashboard - About Me Section

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

Abstract

This project delves into the extensive dataset of New York City's yellow taxis, presenting a rich resource for data scientists and urban planners alike. With a comprehensive record of approximately 3 million taxi trips over the span of one month in January 2023, this dataset captures intricate details of urban mobility. It includes temporal data such as pickup and drop-off hours, spatial dimensions involving location IDs, and quantitative metrics like trip distance, fare, payment methods, passenger counts etc.

Supplied under the TPEP/LPEP initiatives by tech companies, this dataset represents a goldmine for analyzing urban transportation trends, understanding fare dynamics, and exploring driver-passenger behavior. In this analysis, the total amount paid by riders serves as the potential target variable, while remaining features, outlined in the provided dictionary, are considered independent features for Principal Component Analysis (PCA). We will also perform various Normality Tests to check if the data is Normal or not. We will subsequently perform Singular Value Decomposition as well.

A thorough Exploratory Data Analysis (EDA) has been conducted to uncover various patterns and trends within the yellow taxi framework of NYC. The findings of this analysis are expected to inform urban planning strategies and contribute valuable insights into the dynamics of urban transportation in one of the world's most dynamic cities.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

1. Introduction

The energetic metropolis of New York City with its intricate infrastructure of rented vehicles and the symbolic luxury of the Yellow Taxis has long been a symbol of Urban Dynamism. In the realm of Data Science, this cityscape reveals a valuable asset in the form of an extensive dataset featuring its yellow taxis. This dataset is a comprehensive record of Urban Mobility, recording the details of each taxi ride and providing a profound insight into the rhythm of the city's transit dynamic.

The dataset contains temporal information about the pickups and dropoffs, and label encoded spatial data for which we refer to another location based dictionary. Other features of the dataset are congestion charges, airport fees, MTA tax details which are proven to be redundant data and which contribute very little to the actual analysis. We are also provided with label encoded categorical details about type of payment, taxi vendor information, rate code ID according to which the final rate has been decided. We have continuous numerical details like fare amount, trip distance, total amount and tipping details out of which our potential target is Total Amount. The dataset contains label encoded values by default so while preprocessing we do not have to worry about type conversion and label encoding of multi class data.

For the Term Project we will first start with the cleaning of the dataset. For cleaning the dataset we check what kind of values are missing and what rows contain NaN values. The findings show that only those values are missing which were user input by the driver of that particular ride. The reason that we draw for those missing values is that, firstly, it's a manual input column and another reason is that sometimes it is left unfilled because the driver might have considered it a default input (we will cover this reason briefly in the Pre-Processing of Dataset Section). We apply a few methods in order to clean the dataset like ‘fillna’.

After cleaning is done we will create new columns from the temporal data given by the dataset for pickup and dropoff timings. We create new columns such as pickup hour, trip duration (in minutes), pickup and dropoff day (days of week) to get better insight from the temporal data.

After that we check for outliers. We will see that there exist a huge amount of outliers inside the data. By applying IQR method we will firstly find the outliers and remove most of them from the dataset. We don’t remove all kinds of outliers as some outliers might give us insight about certain patterns like tipping patterns during weekends etc.

Outlier removal is highly useful for performing Statistical Analysis. As before outlier removal, PCA analysis gave absurd results, so it proved to be beneficial for these analyses.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

For performing the PCA we use the cleaned data and find that for explaining 95% of the whole data we only need a few features. We also perform feature reduction using PCA and find the condition number of both the untransformed and transformed dataset and observe the drastic difference between both the datasets.

A brief Normality test is also performed on all the quantifiable features of our dataset and we also record a number of observations for those tests. We have to deliberately undersample the dataset to 5000 observations only because certain Normality tests are sensitive towards the number of observations. We don't have to normalize our data even if we get non-normal data as we do not wish to remove certain observations for our EDA.

We also calculate the Heatmap of Pearson Correlation Coefficient in our Dash App to check the correlation between all the features. We observe some observations to be NaN and that is mostly due to the fact that the whole column is sparse (despite undersampling).

The next part of our report will highlight the process of EDA and showcase various findings inside our data using various in-built plots libraries. We make very interesting plots like strip plots, area plots, line plots, hexbin plots and other plots to find a lot of exciting observations. A lot of storytelling is done using a few subplots as well. We also analyze our data using tables as well.

The last and most exciting of our project will be our Dash App. Our Dash App is a user-friendly interface to analyze the taxi data. We have integrated various tests like PCA analysis, Normality test, Outlier Detectors and Pearson Heatmaps to thoroughly study the data. Furthermore, we will also see some graphical analysis of our data and also a ML model predicting our estimated total amount of fare based on the inputs like pickup location, time of day etc.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

2. Description of Dataset

The dataset of our interest is the NYC taxi dataset (specifically for the time period January 2023). Yellow and green taxi trip records include fields capturing pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts. The data used in the attached datasets were collected and provided to the NYC Taxi and Limousine Commission (TLC) by technology providers authorized under the Taxicab & Livery Passenger Enhancement Programs (TPEP/LPEP).

2.1 Feature Description:

- VendorID - A code indicating the TPEP provider that provided the record. 1= Creative Mobile Technologies, LLC; 2= VeriFone Inc.
- tpep_pickup_datetime - The date and time when the meter was engaged.
- tpep_dropoff_datetime - The date and time when the meter was disengaged.
- Passenger_count - The number of passengers in the vehicle.
- Trip_distance - The elapsed trip distance in miles reported by the taximeter.
- PULocationID - TLC Taxi Zone in which the taximeter was engaged
- DOLocationID - TLC Taxi Zone in which the taximeter was disengaged
- RateCodeID - The final rate code in effect at the end of the trip. 1= Standard rate 2=JF 3=Newark 4=Nassau or Westchester 5=Negotiated fare 6=Group ride
- Store_and_fwd_flag - This flag indicates whether the trip record was held in vehicle memory before sending it to the vendor, aka “store and forward,” because the vehicle did not have a connection to the server.
- Payment_type - A numeric code signifying how the passenger paid for the trip. 1= Credit card 2= Cash 3= No charge 4= Dispute 5= Unknown 6= Voided trip
- Fare_amount - The time-and-distance fare calculated by the meter.
- Extra - Miscellaneous extras and surcharges. Currently, this only includes the $0.50 and $1 rush hour and overnight charges.
- MTA_tax - $0.50 MTA tax that is automatically triggered based on the metered rate in use.
- Improvement_surcharge - $0.30 improvement surcharge assessed trips at the flag drop. The improvement surcharge began being levied in 2015.
- Tip_amount - Tip amount – This field is automatically populated for credit card tips. Cash tips are not included.
- Tolls_amount - Total amount of all tolls paid on a trip.
- Total_amount - The total amount charged to passengers. Does not include cash tips.
- Congestion_Surcharge - Total amount collected in trip for NYS congestion surcharge.
- Airport_fee - $1.25 for pick up only at LaGuardia and John F. Kennedy Airports

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

This dataset has been selected as it contains both categorical and numerical data to explain each and every ride. Our preferred target is ‘Total Amount’, which is the amount of money paid by the rider to the driver. We will for all the upcoming discussions treat ‘Total Amount’ as the dependent variable and other remaining features as the independent variables.

This dataset is important for a few reasons. The analysis of this dataset can help urban planners about transportation planning and also give them a lead about traffic pattern analysis. We can also gain economic insights just by aggregating the data and can further analyze how much each driver and the association is earning and how they impact the local economy. We can also draw inferences from the traffic data for the environmental consequences. So, it is a very important dataset to analyze such patterns.

3. Pre-processing of Dataset

We firstly proceed to check the amount of missing data. We discover that only five columns have missing values. Passenger Count, Rate Code ID, Store and Forward Flag, Congestion Surcharge and Airport Fee. The amount of missing data is huge (71743 rows to be precise).

VendorID	0
tpep_pickup_datetime	0
tpep_dropoff_datetime	0
passenger_count	71743
trip_distance	0
RatecodeID	71743
store_and_fwd_flag	71743
PULocationID	0
DOLocationID	0
payment_type	0
fare_amount	0
extra	0
mta_tax	0
tip_amount	0
tolls_amount	0
improvement_surcharge	0
total_amount	0
congestion_surcharge	71743
airport_fee	71743

Fig 1. Count of Empty Rows per feature

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

We do not want to lose data as it might disrupt our further analyses so we choose to 'fillna' our data according to certain constraints.

```
passenger_mode = df['passenger_count'].mode()[0]
df['passenger_count'].fillna(passenger_mode, inplace=True)
ratecodeID_mode = df['RatecodeID'].mode()[0]
df['RatecodeID'].fillna(ratecodeID_mode, inplace=True)
store_and_fwd_flag_mode = df['store_and_fwd_flag'].mode()[0]
df['store_and_fwd_flag'].fillna(store_and_fwd_flag_mode, inplace=True)
df['congestion_surcharge'].fillna(0.0, inplace=True)
df['airport_fee'].fillna(0.00, inplace=True)
```

For passenger count, Rate code ID and Store and Forward Flag we are filling the empty rows with the mode. For other continuous variables we are assuming that the driver has not entered the data as there was neither a pickup from the airport nor was there any congestion in traffic so we fill the empty rows with 0.

After replacing the NaN data we get the following output.

```
VendorID                  0
tpep_pickup_datetime      0
tpep_dropoff_datetime     0
passenger_count           0
trip_distance             0
RatecodeID                0
store_and_fwd_flag        0
PULocationID              0
DOLocationID              0
payment_type              0
fare_amount               0
extra                     0
mta_tax                   0
tip_amount                0
tolls_amount              0
improvement_surcharge     0
total_amount              0
congestion_surcharge      0
airport_fee               0
```

Fig 2. Count of Empty Rows per feature after cleaning

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

The next step of our preprocessing was to label encode the only string type column of our dataset i.e. Store and Forward Flag. We simply encoded 'Y' with 1 and 'N' with 0.

```
category_mapping_fwd_flag = {'N': 0, 'Y': 1}
```

The next step was to extract meaningful data from the temporal data provided in the dataset. The data is labeled as '`tpep_dropoff_datetime`' and '`tpep_pickup_datetime`'. We use this datetime format data to extract new features for our dataset.

```
df['trip_duration'] = (df['tpep_dropoff_datetime'] - df['tpep_pickup_datetime']).dt.total_seconds() / 60
df['pickup_day'] = df['tpep_pickup_datetime'].dt.day_name()
df['dropoff_day'] = df['tpep_dropoff_datetime'].dt.day_name()
df['pickup_hour'] = df['tpep_pickup_datetime'].dt.hour
df['dropoff_hour'] = df['tpep_dropoff_datetime'].dt.hour
```

The new features that we have extracted are trip duration (in minutes), pickup and dropoff day, pickup and dropoff hour

trip_duration	pickup_day	dropoff_day	pickup_hour
8.43	Sunday	Sunday	0
6.32	Sunday	Sunday	0
12.75	Sunday	Sunday	0
9.62	Sunday	Sunday	0
10.83	Sunday	Sunday	0

Fig 3. Newly added features using Temporal Data

We also label encode the pickup and dropoff day but we only label encode it as per our convenience.

Subsequently, we also aggregate our data to find out some specific patterns in our data but we will thoroughly discuss that in coming sections. Also, we will clean the dataset in our interactive dashboard.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

4. Outlier Detection and Removal

Outliers were a big problem for the analysis. Outliers directly skewed the data and gave absurd observations for the PCA analysis. Outliers also skewed out plots and removed any scope for EDA. So, it was very important for us to remove outliers so as to get any meaningful observations from our plots.

The method we apply for detecting and removing the outliers is the IQR method. We calculate the 25th and 75th quartile for our data and then relax our quartile ranges a little bit and remove the observations beyond that range.
It is very important to note that we did not remove all the outliers. We have only removed the outliers from the theoretical range.

```
def QQ_calc_2(temp):
    for column_name in temp.columns:
        if pd.api.types.is_numeric_dtype(temp[column_name]):
            Q1 = temp[column_name].quantile(0.25)
            Q3 = temp[column_name].quantile(0.75)
            IQR = Q3 - Q1

            lower_bound = Q1 - 1.5 * IQR
            upper_bound = Q3 + 1.5 * IQR

            temp = temp[(temp[column_name] >= lower_bound) & (temp[column_name] <= upper_bound)]

    return temp
```

We plotted a before and after boxplot of our target variable to check if the outliers have been removed.

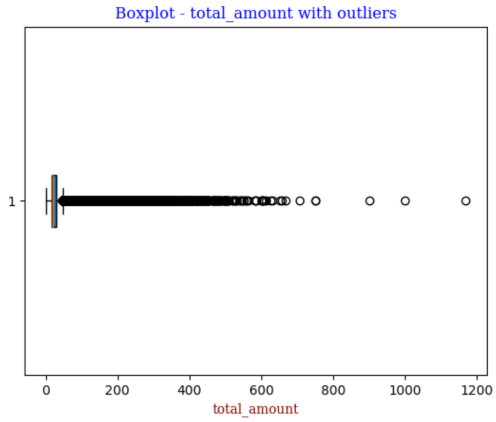


Fig 4. Boxplot - Total Amount with Outliers

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

We can easily notice a dense amount of outliers existing throughout our data. This data could've distorted our plots.

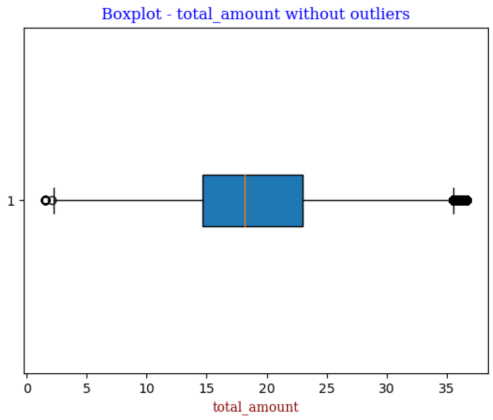


Fig 5. Boxplot - Total Amount without Outliers

After removal of outliers we can see that the data is very clean but some of us might notice that we still have some outliers. The reason for this non-removal of outliers is for data representation. We want to study some plots in which these outliers might come handy in recognizing patterns. That being said, we have also opted to not remove outliers from columns like Passenger Count, Payment Type and Congestion Surcharge because of the fact that most of the data is filled with a single type of observation for each column and the IQR method assumes the remaining values as outliers.

We will further analyze outliers in our interactive dashboard.

5. Principal Component Analysis (PCA)

We performed PCA in our cleaned and label encoded dataset. We need to remember that PCA only tells us how many components are important but does not give an exact description of which component to remove. So, if we want to get a hint about which component to remove we also need to investigate the correlation and covariance matrices as well.

For performing PCA we had to first standardize our dataset. We standardized our whole dataset irrespective of their categorical or numerical status. We also need to keep in mind that we have undersampled the data to 10,000 observations from 3 million observations because the processing time for this will be huge.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

```
df_sampled2 = df_temp.sample(n=10000, random_state=5764)
df_sampled2.reset_index(drop=True, inplace=True)
```

The PCA is performed on the Independent Features only, meaning that we will not include 'Total Amount' in our analysis.

```
X_PCA = df_sampled2.drop(['total_amount'], axis=1)
y_PCA = df_sampled['total_amount']
```

```
scalar = StandardScaler()
X_std = scalar.fit_transform(X_PCA)
X_std = pd.DataFrame(X_std, columns=X_PCA.columns)
```

The total number of columns after dropping our dependent component is 20 so we ran a PCA for 20 components.

```
pca = PCA(n_components=20, svd_solver='full')
pca.fit(X_std)
```

The following observation has been made for the Cumulative Explained Variance Ratio. We can easily notice that not all of the components are required to describe the whole data. We can easily remove 9 features from our data to reduce the dimension of our data.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

PCA Analysis	
Component	Cumulative Explained Variance Ratio
1	0.2
2	0.34
3	0.48
4	0.58
5	0.68
6	0.75
7	0.82
8	0.88
9	0.94
10	0.96
11	0.98
12	1.0
13	1.0
14	1.0
15	1.0
16	1.0
17	1.0
18	1.0
19	1.0
20	1.0

Table 1 - PCA Analysis

An obvious observation that can be made from the above table is that only about 9 components are required to explain almost 95% of the whole data. So, we ran another PCA but this time we kept n_components = 9. This is done to calculate the Singular Value Decomposition and Condition Number of both the raw and transformed feature space.

```
pca2 = PCA(n_components=9, svd_solver='full')
pca2.fit(X_std)
X_PCA2 = pca2.transform(X_std)
```


App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

Singular Values	
Raw Condition Number = 5178127073296003986478598864240 64.00	Transformed Condition Number = 1.84
173.36	173.36
146.46	146.46
140.83	140.83
128.04	128.04
117.99	117.99
107.60	107.60
100.53	100.53
95.27	95.27
94.40	94.40
56.66	-
51.84	-
45.52	-
22.50	-
14.08	-
0.13	-
0.00	-
0.00	-
0.00	-
0.00	-
0.00	-

Table 2 - Singular Values and Condition Number of Raw and Reduced Feature Space

As evident from the above table, we can see a drastic drop in the Condition Number from raw to reduced feature space implying loss of collinearity in the transformed feature space. Singular Values indicate that we just need 9 components to explain approximately 95% of the whole dataset and we can use this transformed feature space for Machine Learning as well.

We have also integrated the PCA analysis in our interactive dashboard and following is our observation.

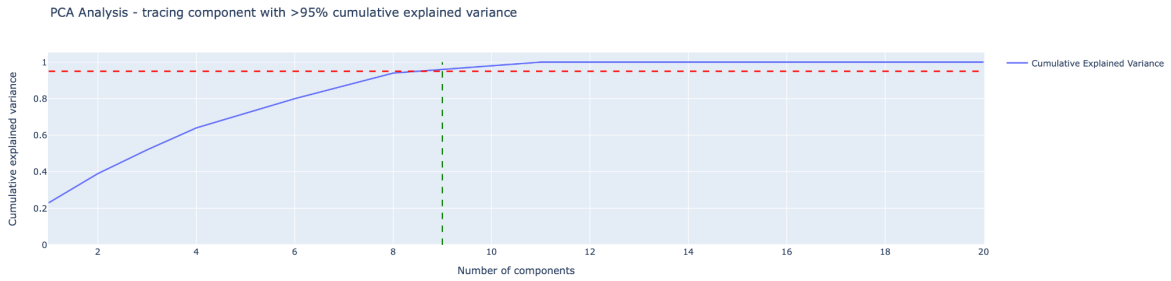


Fig 5. PCA Analysis - Tracing component with >95% cumulative explained variance

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

As observed before, we see only 9 features are required to explain 95% of the whole dataset. We have traced the exact moment at which the cumulative explained variance crosses our threshold of 95%.

6. Normality Tests

We have performed various Normality tests in our data as well. We need to keep in our minds that these Normality tests give wrong observations in case there are more than 10,000 observations, so we will only test this in an undersampled dataset of 5000 rows.

```
df_sampled3 = df.sample(n=5000, random_state=5805)
df_sampled3.reset_index(drop=True, inplace=True)
```

The tests that we selected for our analysis are:

- Shapiro Test
- D'Agostino's K^2 test
- Kolmogorov–Smirnov test

We also need to keep in mind that we are running these tests on label encoded categorical features as well. The minimum p-value that we will consider for a data to be normal is 0.01.
The feature space is purely numerical and we do not standardize any of the features to skew our actual observation. Also we have removed features with sparse data as they tend to give false verdicts.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

Normality Test				
Feature	Da_k_squared Test	K_S test	Shapiro Test	Verdict
VendorID	Stats = 2238.25, p = 0.00	Stats = 0.46, p = 0.00	Stats = 0.56, p = 0.00	Not Normal
passenger_count	Stats = 3062.74, p = 0.00	Stats = 0.43, p = 0.00	Stats = 0.52, p = 0.00	Not Normal
trip_distance	Stats = 853.50, p = 0.00	Stats = 0.10, p = 0.00	Stats = 0.92, p = 0.00	Not Normal
RatecodeID	Stats = 13246.15, p = 0.00	Stats = 0.51, p = 0.00	Stats = 0.01, p = 0.00	Not Normal
store_and_fwd_flag	Stats = 8796.02, p = 0.00	Stats = 0.52, p = 0.00	Stats = 0.05, p = 0.00	Not Normal
PULocationID	Stats = 915.42, p = 0.00	Stats = 0.19, p = 0.00	Stats = 0.92, p = 0.00	Not Normal
DOLocationID	Stats = 952.87, p = 0.00	Stats = 0.20, p = 0.00	Stats = 0.92, p = 0.00	Not Normal
payment_type	Stats = 4266.15, p = 0.00	Stats = 0.44, p = 0.00	Stats = 0.40, p = 0.00	Not Normal
fare_amount	Stats = 321.05, p = 0.00	Stats = 0.09, p = 0.00	Stats = 0.96, p = 0.00	Not Normal
extra	Stats = 405.19, p = 0.00	Stats = 0.23, p = 0.00	Stats = 0.82, p = 0.00	Not Normal
tip_amount	Stats = 303.95, p = 0.00	Stats = 0.14, p = 0.00	Stats = 0.95, p = 0.00	Not Normal
total_amount	Stats = 227.67, p = 0.00	Stats = 0.06, p = 0.00	Stats = 0.97, p = 0.00	Not Normal
congestion_surcharge	Stats = 4311.71, p = 0.00	Stats = 0.55, p = 0.00	Stats = 0.21, p = 0.00	Not Normal
trip_duration	Stats = 261.32, p = 0.00	Stats = 0.06, p = 0.00	Stats = 0.97, p = 0.00	Not Normal
pickup_day	Stats = 8794.75, p = 0.00	Stats = 0.14, p = 0.00	Stats = 0.91, p = 0.00	Not Normal
dropoff_day	Stats = 9027.89, p = 0.00	Stats = 0.14, p = 0.00	Stats = 0.91, p = 0.00	Not Normal
pickup_hour	Stats = 259.48, p = 0.00	Stats = 0.09, p = 0.00	Stats = 0.96, p = 0.00	Not Normal
dropoff_hour	Stats = 282.64, p = 0.00	Stats = 0.10, p = 0.00	Stats = 0.95, p = 0.00	Not Normal

Table 3 - Normality Test

The result is unsurprising but very obvious that all of the features are ‘Not Normal’. Most of the features are Not normal because of the fact that we receive a very low amount of p-value upon analysis. But, some of the statistics value indicate that had the p-value been more than 0.01 the feature would have been considered normal. (Rejecting the null hypothesis)

For the Shapiro Test, when the statistics is not close to 1 it is considered to be deviated from Normal Distribution. For most features that is the exact case. But, trip distance, fare amount, total amount, Pickup location, Drop off location, pickup and drop off hour, trip duration show a good value of statistics i.e. close to 1.

For the Kolmogorov–Smirnov test, the statistics display the theoretical distance from actual observation. So, the more it is close to the theoretical distribution the more it is considered Normal. Trip distance, fare amount, total amount, Pickup location, Drop off location, pickup and drop off hour, trip duration show a good value of statistics for that purpose.

For the D'Agostino's K^2 test, the larger the value of statistics the more it is considered to be deviated from the Normal Distribution. Trip distance, fare amount, total amount, Pickup location, Drop off location, pickup and drop off hour, trip duration show smaller values of statistics compared to other remaining features.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

We will also analyze the distributions and perform a Box-Cox transformation of the 'Not Normal' feature in our interactive dashboard.

7. Heatmap and Pearson Correlation Coefficient

We calculated the Pearson Correlation Coefficient Matrix between all the features and plotted a heatmap to analyze what features are closely related to each other and show any other pattern.

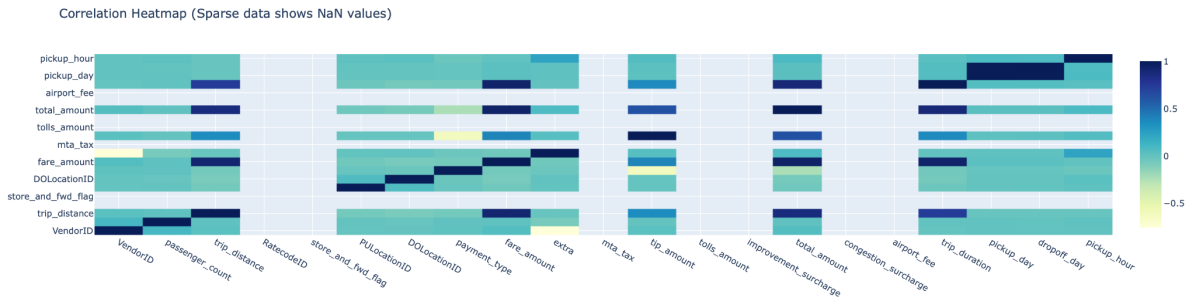


Fig 6. Heatmap - Pearson Correlation Coefficient

Right after we plotted the heatmap the first thing that we observed was the empty spaces inside the heatmap. We had a question as well: how is it even possible? The answer lies in the fact that the data which showed such behavior was sparse data. It was not showing any correlation whatsoever. These features had the same value throughout the rows. e.g. mta_tax contains only two values either $0 or $2.5 but most of the feature space is mostly filled with $0 and it shows no variation throughout the data. So, because of this observation only we see that some features produce NaN values inside the heatmap.

Apart from this observation we can clearly see that most of data is uncorrelated except a few like

- Trip Distance and Total Amount (0.86)
- Trip Distance and Fare Amount (0.91)
- Pickup day and Drop off day (0.99)
- Extra and Vendor ID (-0.76)

Let's discuss these correlations and why they must be highly correlated.

1. Trip Distance and Total Amount are highly correlated for the obvious reason that the more the miles a driver travels the more amount they earn.
2. Trip Distance and Fare Amount are highly correlated for the same reason as above.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

3. Pickup Day and Drop off Day are highly correlated because of the fact that almost all of the pickups and dropoffs have been made on the same day. Later on further analysis we will also observe that since the trip duration has been under 1 hour for most trips we see that these observations make more sense.
4. Extra and Vendor ID are negatively correlated which might indicate that for some particular Vendors (Verifone) the Extra fees are NOT taken at all.

8. Statistics

In the previous few sections we have already run a thorough statistical analysis. Let us first summarize them.

- Around 2% of our data contained null values which were cleaned subsequently.
- The data contained a high density of outliers according to the boundaries set by the IQR method and we removed almost all of the outliers by relaxing the lower and upper bound of each feature but we need to remember that some of the features have been excluded so as to retain patterns in the data.
- We then ran a PCA analysis on our data by standardizing our feature space. We concluded that the total number of components required to explain almost 95% of data is 9. The condition number of raw feature space was 5178127073296003986478598864240 64 and the condition number of the transformed feature space dropped drastically due to loss of collinearity inside the matrix to 1.84.
- We ran various normality tests and found that the feature space is entirely 'Not Normal' based on the fact that the p-values of none of the features exceeded 0.01 despite having good values for statistics of the various tests.
- Lastly we ran Pearson Correlation Coefficient Analysis which showed us that trip distance, fare amount and total amount are highly correlated. Also, pickup day and drop off day are highly correlated and extra and vendor ID are highly negatively correlated.

8.1 General Observations

Now lets see the statistics of each feature present in our cleaned dataset.

	VendorID	passenger_count	trip_distance	PULocationID	DOLocationID	fare_amount	extra	mta_tax	tip_amount	tolls_amount
count	2458729.00	2458729.00	2458729.00	2458729.00	2458729.00	2458729.00	2458729.00	2458729.0	2458729.00	2458729.0
mean	1.72	1.35	1.75	170.19	168.30	12.10	1.37	0.5	2.42	0.0
std	0.45	0.89	1.08	65.19	67.99	5.02	1.45	0.0	1.70	0.0
min	1.00	0.00	0.00	1.00	1.00	0.00	0.00	0.5	0.00	0.0
25%	1.00	1.00	0.97	137.00	137.00	7.90	0.00	0.5	1.00	0.0
50%	2.00	1.00	1.50	163.00	163.00	11.40	1.00	0.5	2.52	0.0
75%	2.00	1.00	2.30	236.00	236.00	14.90	2.50	0.5	3.58	0.0
max	2.00	6.00	6.73	265.00	265.00	27.85	6.25	0.5	7.67	0.0

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

improvement_surcharge	total_amount	congestion_surcharge	airport_fee	trip_duration	pickup_hour	dropoff_hour
2458729.0	2458729.00	2458729.00	2458729.0	2458729.00	2458729.00	2458729.00
1.0	19.18	2.38	0.0	10.71	14.14	14.19
0.0	6.02	0.54	0.0	5.48	5.72	5.78
1.0	1.50	0.00	0.0	0.00	0.00	0.00
1.0	14.64	2.50	0.0	6.45	10.00	11.00
1.0	18.20	2.50	0.0	9.88	15.00	15.00
1.0	23.00	2.50	0.0	14.27	19.00	19.00
1.0	36.70	2.50	0.0	26.50	23.00	23.00

Fig 7. Statistics of cleaned features

Some observations:

1. PULocationId and DOLocationId have the highest deviation which is followed by total_amount.
2. A few fare amounts can be seen to have 0 value which implies that some of the trips might have been void trips or no charge trips. But we also see that total amount to NOT have any 0 charged trips which says that even if fare was not taken the rider decided to pay a small amount.
3. Airport fee is so sparse that it does not have any standard deviation and mean is essentially 0.
4. Average pickup and drop off hours are around 2 pm which shows most trips occur in that range. We will verify this data graphically in the coming sections.

8.2 Multivariate Kernel Density Estimate

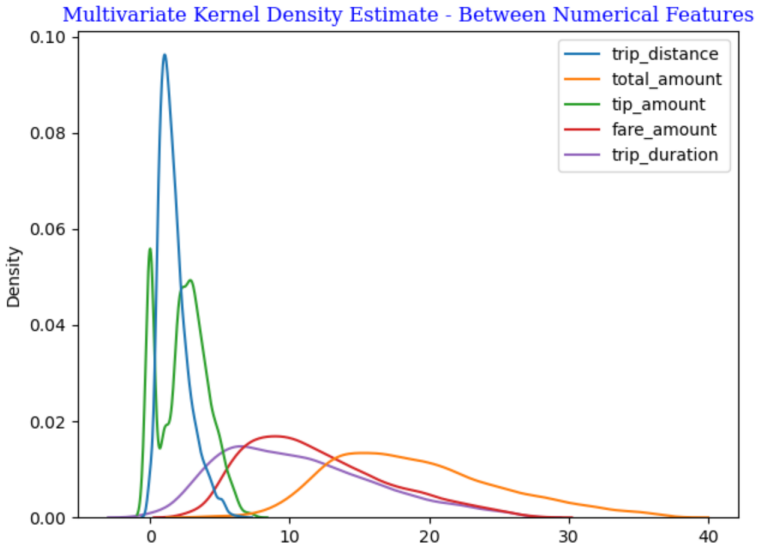


Fig 8. Multivariate Kernel Density Estimate between Numerical Features

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

Some observations on Multivariate Kernel Density Estimate:

1. All the features are right skewed.
2. Tip amount is bimodal and we can see the reason behind that is the tipping is either zero (for most of the trips) or around $3-5.
3. Trip distance is highly right skewed and it is because of the fact that most trips have been made between 1-5 miles.
4. Trip Duration and Fare Amount are equally distributed between 0 to 30 units
5. Total amount is right skewed as well as right shifted from the Fare Amount but it has almost the same distribution with varying ranges.

9. Data Visualization

Now let us visualize various static plots to analyze trends in our data and draw inferences from the plots.

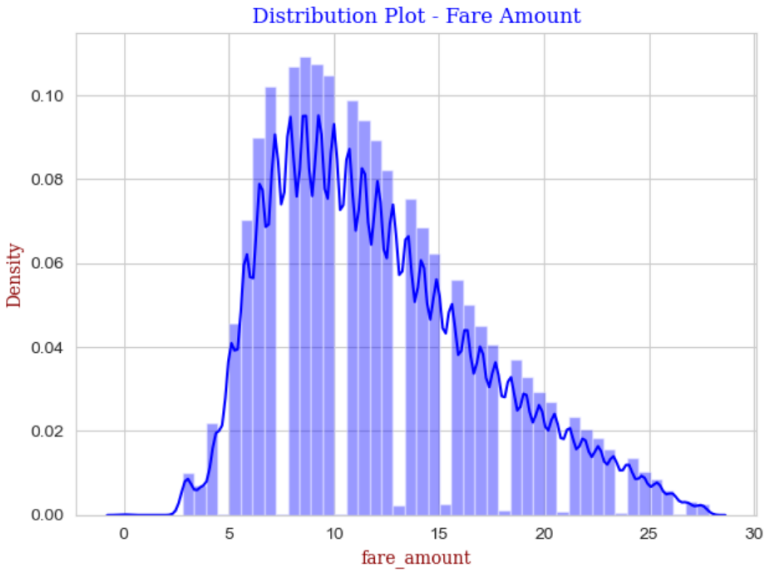


Fig 9. Distribution Plot of Fare Amount

The Distribution is right skewed and most fares amount are around $8-10 The total range of fares is around $4-28. The gaps in the distribution is mostly due to the choice of binwidth (which is the default selection).

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

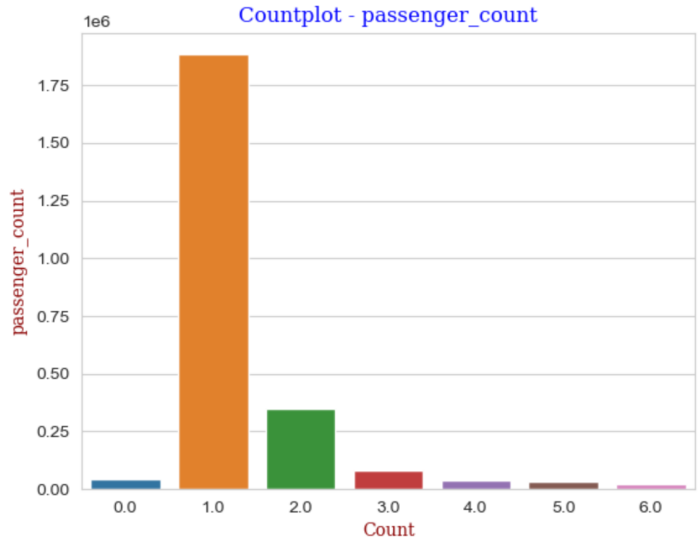


Fig 10. Count plot of Passenger Count

Most of the rides have only 1 passenger, nextly the second highest number of passengers is 2. Interestingly, a few passenger counts are 0 which might mean that either the input is erroneous or some drivers were just delivering physical goods.

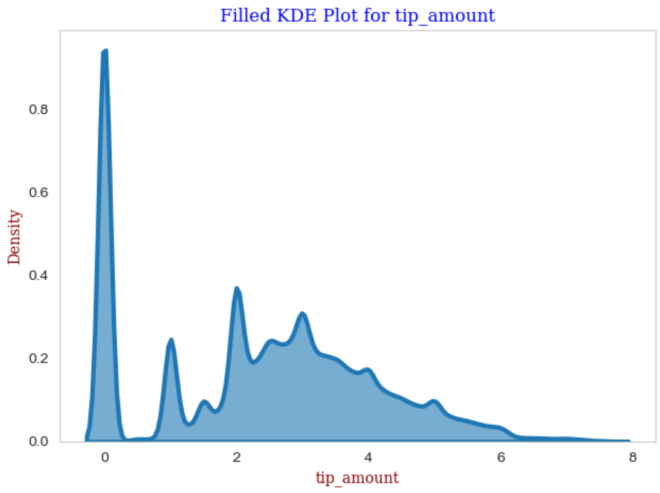


Fig 11. Filled KDE Plot of Tip Amount

A keen observation that can be made is that a great deal of passengers are NOT TIPPING at all. If tipping is being done most of the passengers are tipping around $2 - $4. Some anomalies can

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

be seen and we can see some people tipping $8 as well. So in a sense this graph is bimodal in nature (but it is not strictly bimodal).

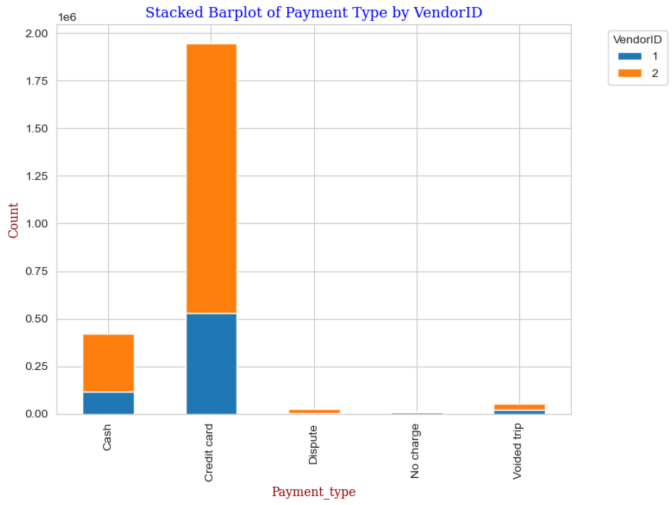


Fig 12. Stacked Barplot of Payment Type by Vendor ID

Most of the passengers prefer to pay using Credit CardSecond most preferred payment type is Cash. We can also see that Verifone (Vendor ID 2) has the most number of riders for each payment type. Creative Mobile Technology (Vendor ID 1) has a fair share of customers for each payment type as well.

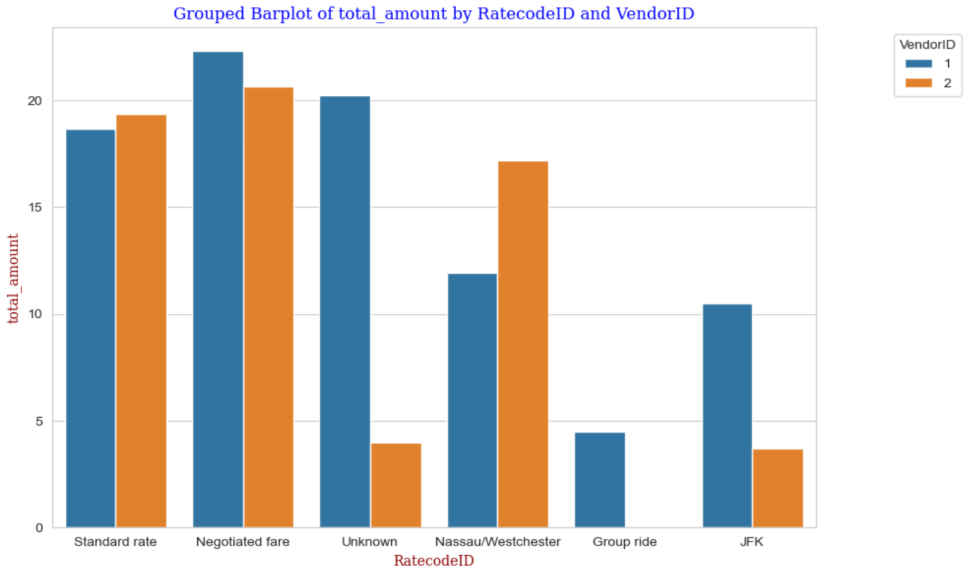


Fig 13. Grouped Bar Plot of Total Amount by Rate Code ID and Vendor ID

For Verifone vendors the drivers we can see a high volume of drivers taking negotiated fares. Second most preferred effective rate code is Standard rate (meter rate). Verifone vendors don't

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

prefer Group Rides at all. Creative Mobile Technology drivers can also be seen having a high volume of negotiated fares but second most preferred rate code is unknown. At Westchester airport location we see Verifone drivers earn better and the opposite can be said for JFK for Creative Mobile Technology.

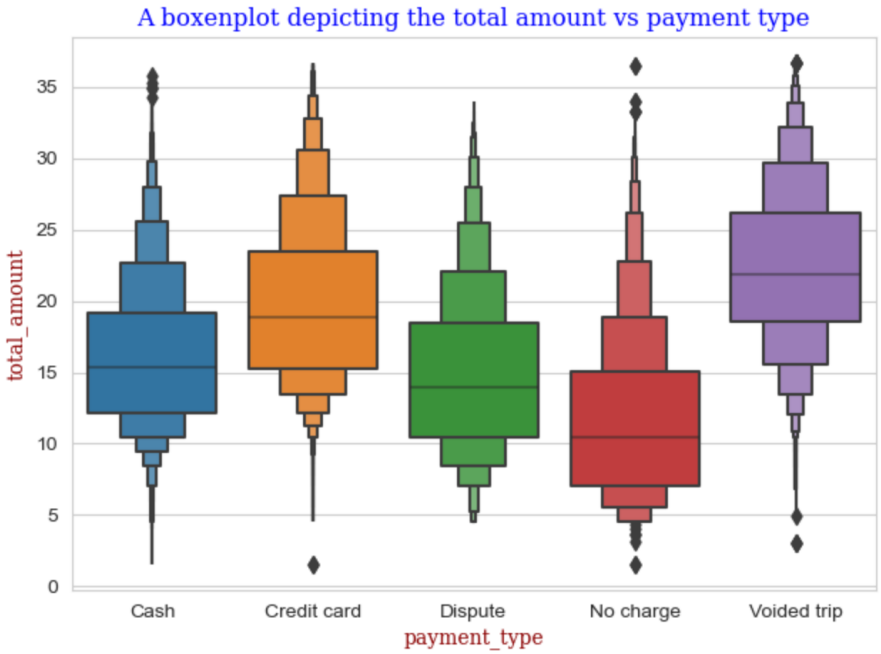


Fig 14. Boxen plot of Total Amount vs Payment Type

This boxen plot describes the volume of total amount given by each particular payment type. It DOES NOT mean that Voided Trip had the highest total amount paid. It means that the average total amount given during voided trips had a higher volume at around $23. Furthermore, For credit cards we can see the median is around $18, for cash it is around $16. We must also notice that a few outliers exist in these boxenplots as well.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

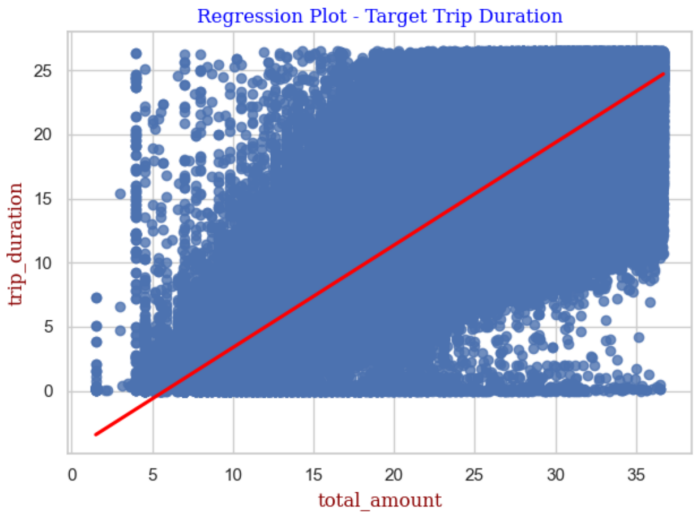


Fig 15. Regression Plot between Total Amount and Trip Duration

This observation is quite obvious, that we get a positive correlation. The slope of the regression line is positive. But as seen clearly it is not a linear relationship. The scatterplot is densely populated so we cannot draw direct inferences for this relationship.

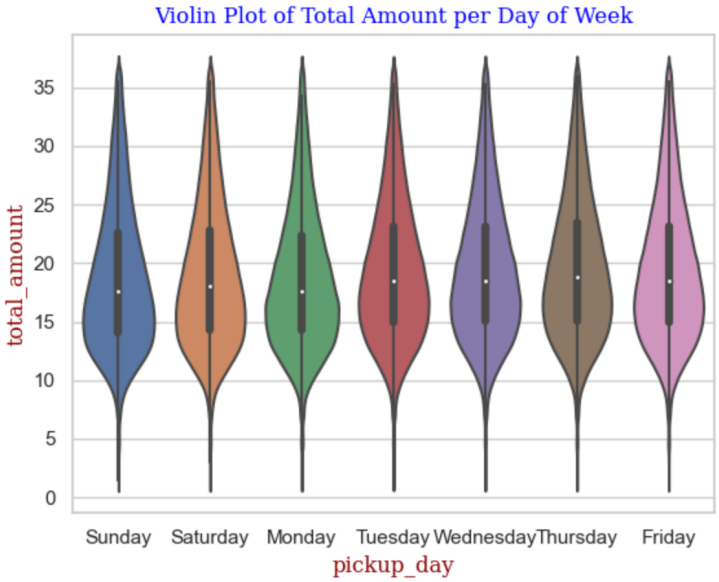


Fig 16. Violin Plot of Total Amount per Day of Week

For each weekday we see identical probabilities of getting a total amount near the median. The probabilities show a drastic drop below $5 total amount. If we compare the median's highest median for total amount is for Tuesday at around $17.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

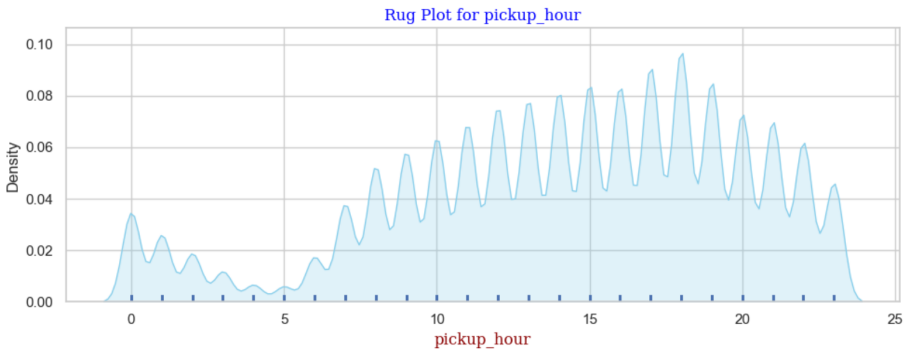


Fig 17. Rug Plot for Pickup Hour

This rug plot helps us determine which pickup hours are highly dense. We see an obvious observation that the rush hour (3pm - 8pm) has the highest density. Another observation that can be made is that a few rides are also taken just after midnight but it slowly fades after we reach 5am.

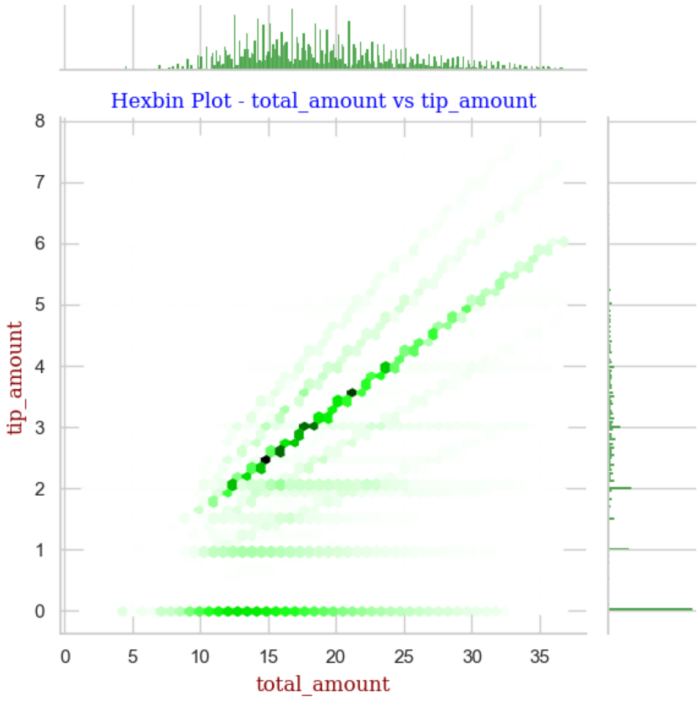


Fig 18. Hexbin plot of Tip Amount vs Total Amount

As established earlier a lot of passengers opted not to tip and it is evident from the green line at the bottom. But other than that we see that there is a positive correlation between them. Some hexbins can be seen to have a darker color implying a high amount of tipping done in the range of $2-4.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

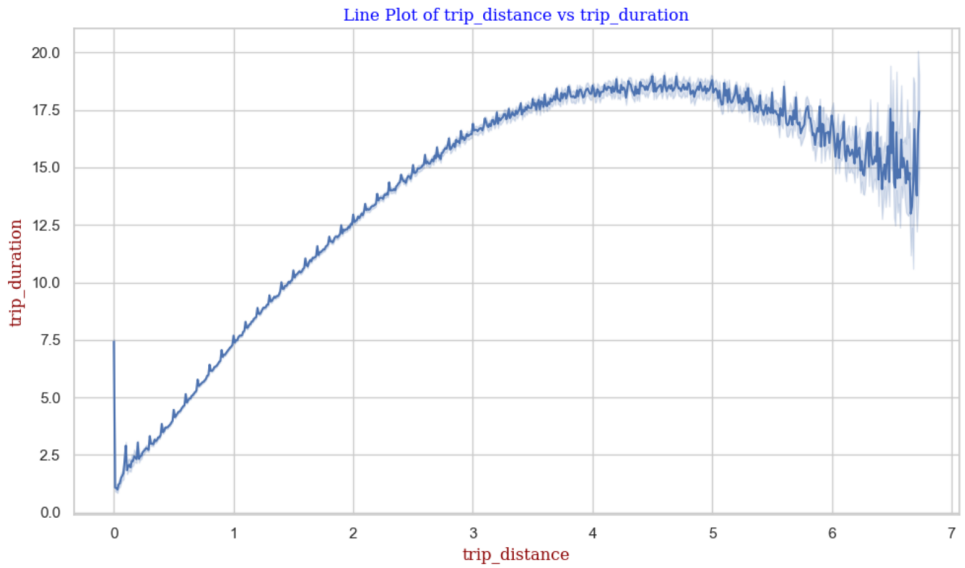


Fig 19. Line Plot of Trip Distance vs Trip Duration

We might expect that a linear relationship might exist between Trip Duration and Trip Distance. But it is NOT linear rather (almost) logarithmic. But it is needless to say that trip distance increases with trip duration but it definitely shows a drop when distance increases.

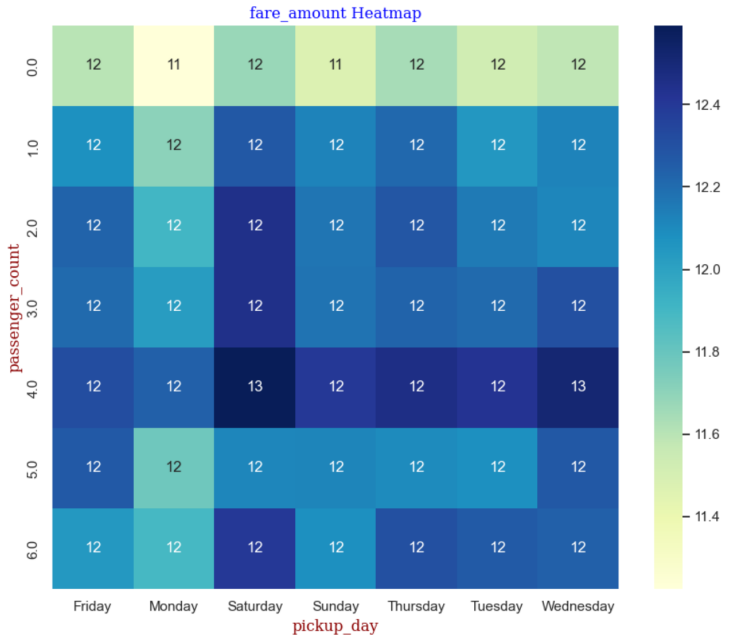


Fig 20. Heatmap of Fare Amount - Passenger Count vs Pickup Day

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

Each block shows the Fare Amount Density given each Pickup Day given the number of passengers. We see that a higher amount of fare is given for passenger count of 4 during Saturdays and Wednesdays. Also least fare is given for 0 passengers for all the Pickup Days

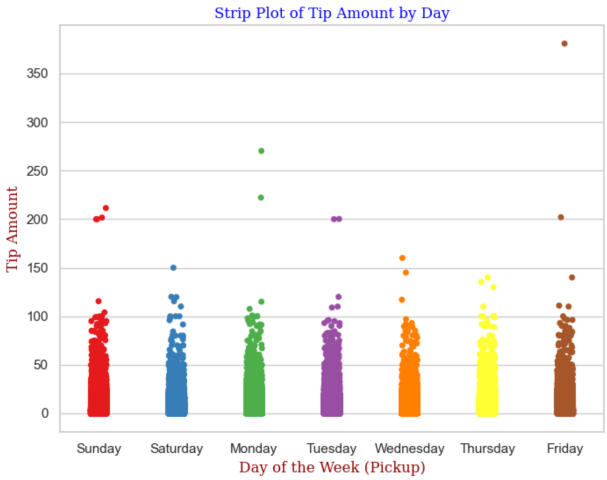


Fig 21. Strip Plot of Tip Amount by Day of Week

This plot is made on the data with outliers. It shows a one dimensional scatter plot of tips for each day of week. We can observe that the anomalies exist on Friday, Sunday and surprisingly Monday.

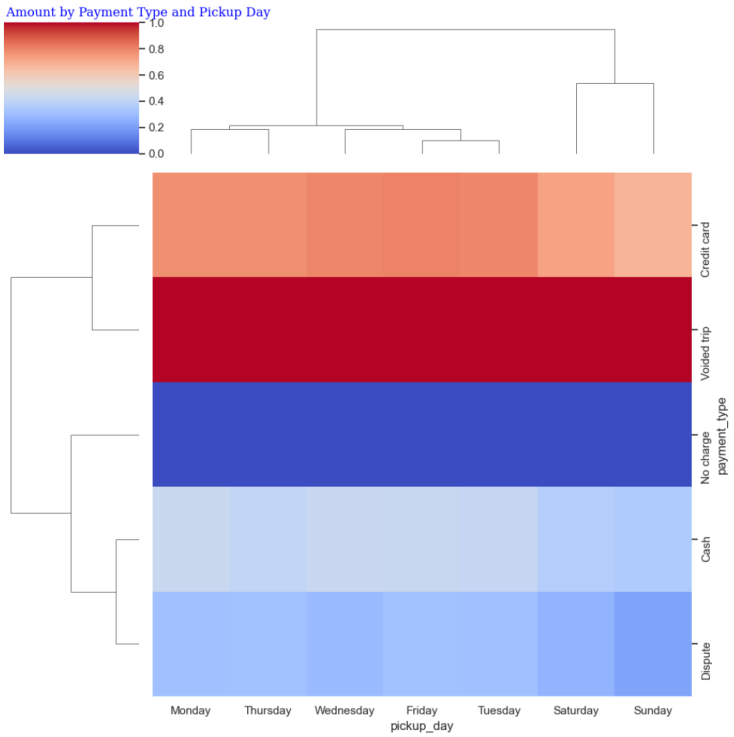


Fig 22. Cluster Map of Total Amount by Payment Type and Pickup Day

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

The data might be misleading as it shows a high density of Total Amount for a Voided Trip. It just means that none of the Voided trips were left uncharged. Our cluster map shows 'Clusters' of most similar categories. Dispute and Cash categories are closely related and similarly Voided and Credit Cards are closely related. We can also see that Monday, Thursday and Friday, Tuesday and Saturday, Sunday show similar patterns thus the clusters

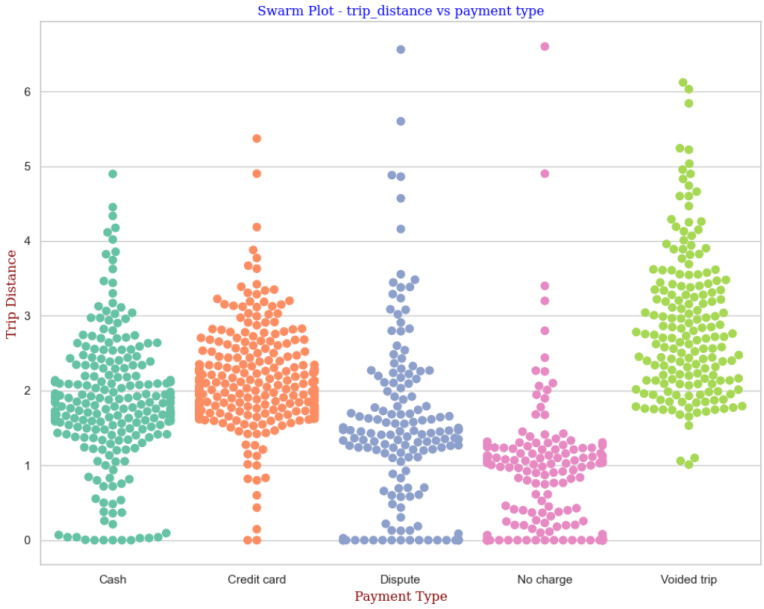


Fig 23. Swarm Plot of Trip Distance vs Payment Type

This plot shows at which intervals we get the highest average trip distance. For payments in Cash we see that it is highly dense in the range 1.5 - 2.1 Miles. For Credit Cards the range is 1.6-2.2 Miles. For Dispute we see a less dense graph at 1.2-1.5 Miles. For No Charge the distribution shifts to a lower range of around 1 Mile. For Voided Trip we see the distribution to be lightly spread around 1.9 - 3.5 Miles.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

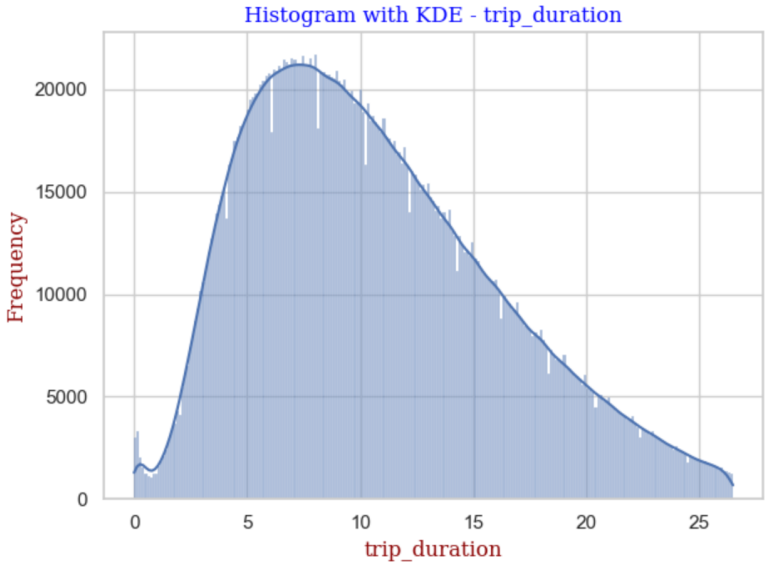


Fig 24. Histogram with KDE of Trip Duration

We can see the trips are all in the range of 3-25 minutes (all under an hour). Data is right skewed and most trips are complete between 5-10 minutes.

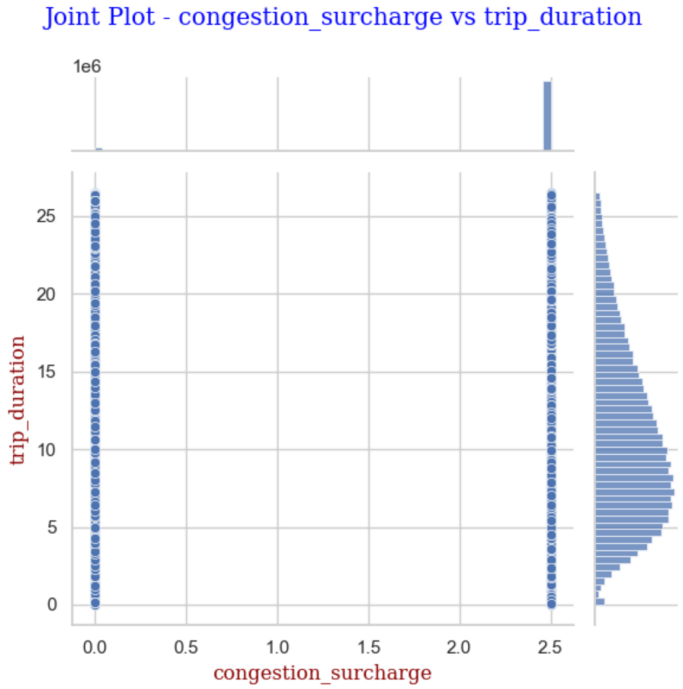


Fig 25. Joint plot of Congestion Surcharge vs Trip Duration

At one glance we can see that there is no particular correlation between congestion surcharge and trip duration. We can see that most trips have a congestion surcharge but that doesn't imply that the trip duration was high because of that meaning congestion surcharge doesn't depend on time rather the actual traffic.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

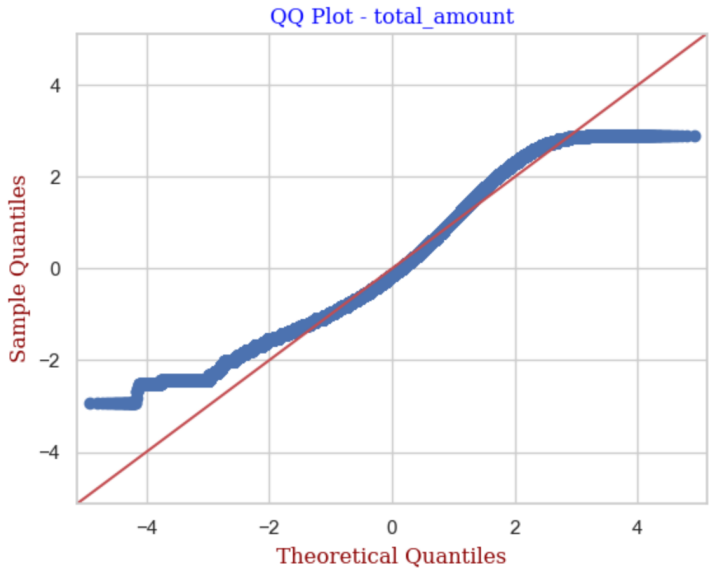


Fig 26. QQ Plot of Target Variable - Total Amount

The QQ-plot for total amount might suggest a few things. The S-shape of this might suggest that it has skewness. This shape might also indicate that Normal distribution might not be an appropriate model for total amount.

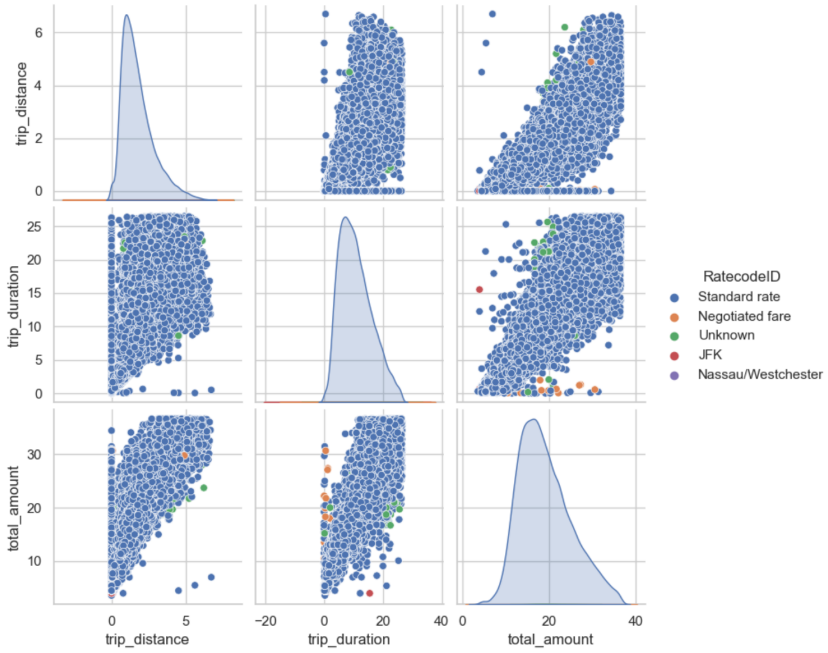


Fig 27. Pair plot between Trip Duration, Trip Distance and Total Amount grouped by Rate Code ID

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

We took undersampled data (50000 observations) to plot this pairplot. Diagonals show distributions of each feature (which is mostly right skewed). Trip duration vs distance can be seen to have almost positive correlation. Total amount vs Distance can be seen to have a positive correlation and so does the Trip duration vs Total amount. Almost all the trips are standard rate.

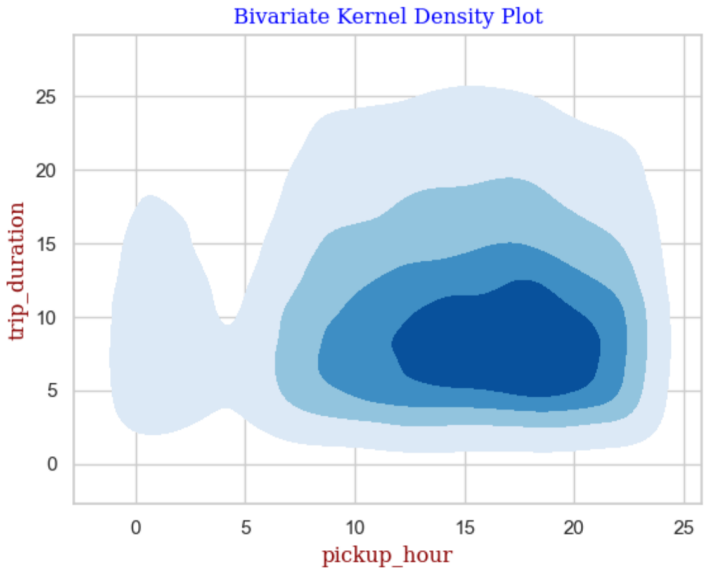


Fig 28. Bivariate Kernel Density Plot between Trip Duration and Pickup Hour

Referencing earlier observations we can see Pickup hour is highly dense at the rush hour (dark blue). Same can be said for Trip Duration which is mostly between 5-10 mins. Anomaly in pickup hour is the pickups occurring at midnight hours.

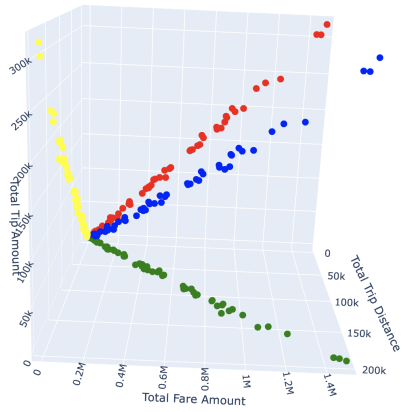


Fig 29. 3D plot (with Contour Plot) of Trip Distance, Total Tip Amount and Total Fare Amount aggregated by pickup location

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

It is very clear that for pickup locations with higher fare amount we have higher tip amount and higher trip distance. This shows clear positive correlation among all three features.

10. Subplots

For exploring other aspects of our Dataset we have also plotted some Subplots which explain the relationship between features and some common trends as well.

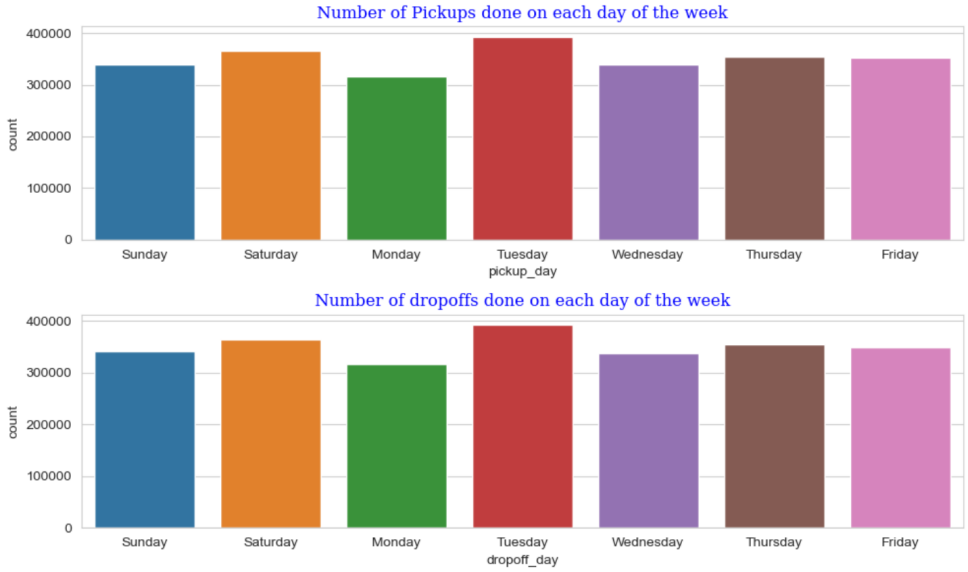


Fig 30. Subplot of Count plots of Number of Pickups and Drop offs made on each day of week

We can easily see that both the plots are almost identical. Interestingly, Tuesdays have the highest volume of both pickups and dropdowns. Least amount of pickups/dropdowns are being done on Monday. During weekends we can see that Highest volume is on Saturday suggesting that a lot of people take rides on Saturdays and the sudden drop on Sundays mean that people might be taking an off during Sundays and not moving out that much.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

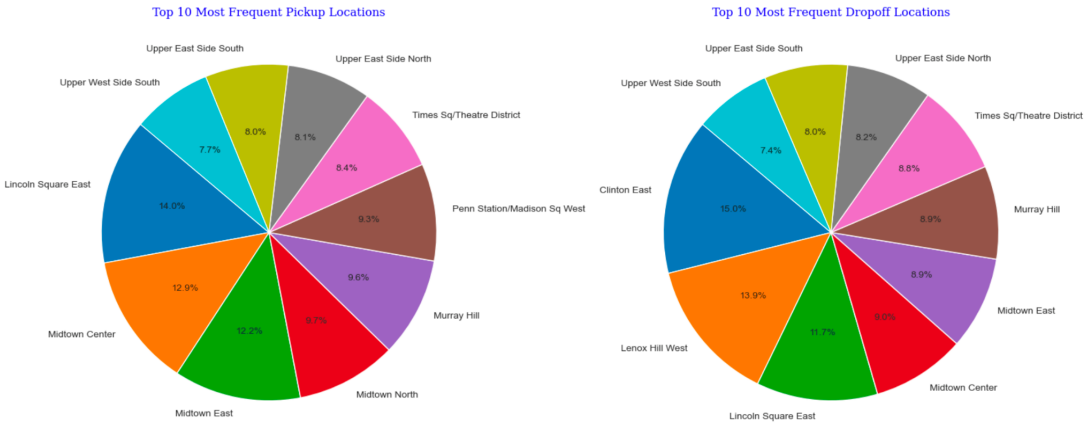


Fig 31. Subplot - Pie Plot of Top 10 Most Frequent Pickup and Drop off locations

This subplot draws the comparison between pickup and dropoff locations. Most pickups are being done on Lincoln Square East and most drop offs occur at Clinton East. This subplot also suggests that among the top 10 pickup locations the least amount of pickups happen at Upper West Side South and least amount of Drop Offs occur at the same place.

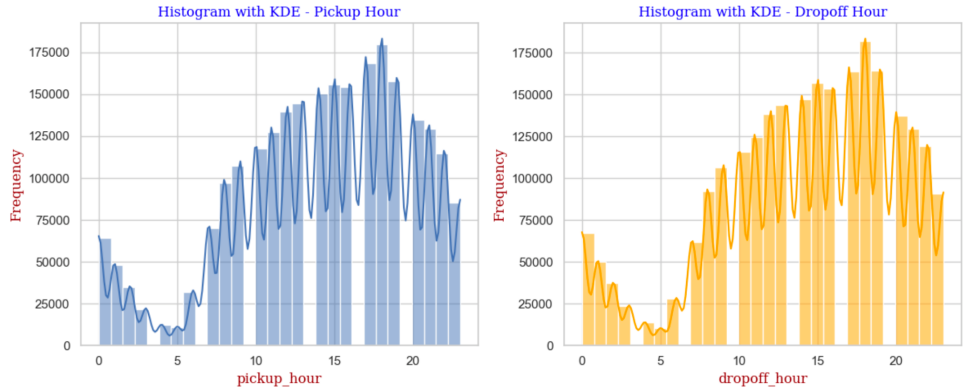


Fig 32. Subplot - Comparison between Pickup and Drop off Hours

The histograms are almost identical to each other showing that most trips have been made under an hour. The shape of the histogram is the same as the KDE plot of the rug plot. Some plot gaps are due to the fact that the bin width is the default binwidth. This subplot also shows that peak hours of pickup and drop offs are also during the rush hour.

We also made a trace of the 3D plot mentioned in the previous section. This plot was made to make a better contrast of the actual observations made in the 3D plot.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

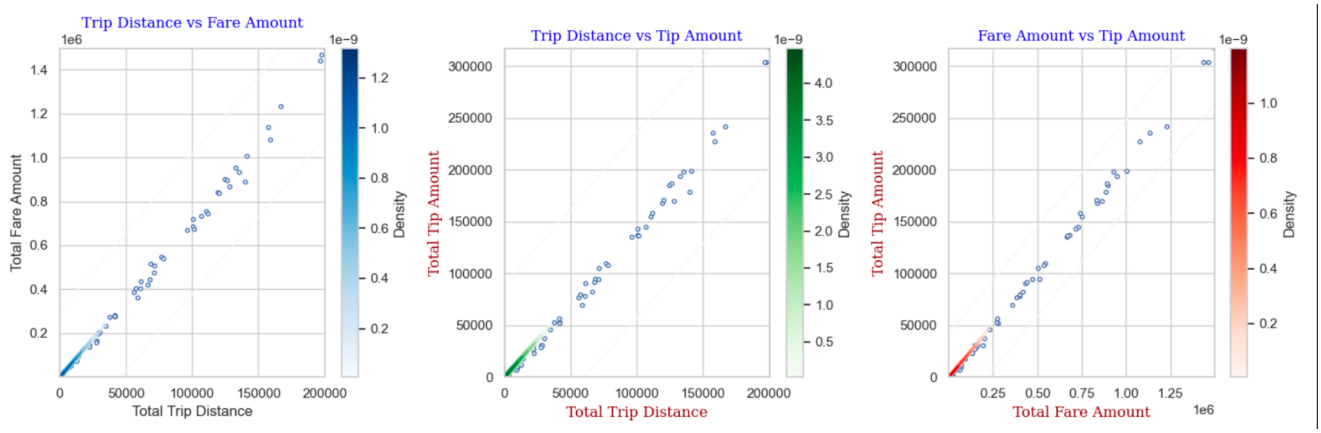


Fig 33. Subplot - Contour plot comparison of various features

To see a clear trend we traced contour plots of our previous 3D plot. All three plots show positive correlation. Total Fare vs Tip amount at later stages show a little drop from the trend but it follows the general trend.

11. Tables

The following is a reference table by which we make further observations on various Boroughs of NYC.

Taxi Zone Dictionary			
LocationID	Borough	Zone	service_zone
1	EWR	Newark Airport	EWR
2	Queens	Jamaica Bay	Boro Zone
3	Bronx	Allerton/Pelham Gardens	Boro Zone
4	Manhattan	Alphabet City	Yellow Zone
5	Staten Island	Arden Heights	Boro Zone
6	Staten Island	Arrochar/Fort Wadsworth	Boro Zone
7	Queens	Astoria	Boro Zone
8	Queens	Astoria Park	Boro Zone
9	Queens	Auburndale	Boro Zone
10	Queens	Baisley Park	Boro Zone
11	Brooklyn	Bath Beach	Boro Zone
12	Manhattan	Battery Park	Yellow Zone
13	Manhattan	Battery Park City	Yellow Zone
14	Brooklyn	Bay Ridge	Boro Zone
15	Queens	Bay Terrace/Fort Totten	Boro Zone
16	Queens	Bayside	Boro Zone
17	Brooklyn	Bedford	Boro Zone
18	Bronx	Bedford Park	Boro Zone
19	Queens	Bellerose	Boro Zone
20	Bronx	Belmont	Boro Zone
21	Brooklyn	Bensonhurst East	Boro Zone
22	Brooklyn	Bensonhurst West	Boro Zone

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

23	Staten Island	Bloomfield/Emerson Hill	Boro Zone
24	Manhattan	Bloomingdale	Yellow Zone
25	Brooklyn	Boerum Hill	Boro Zone
26	Brooklyn	Borough Park	Boro Zone
27	Queens	Breezy Point/Fort Tilden/Riis Beach	Boro Zone
28	Queens	Briarwood/Jamaica Hills	Boro Zone
29	Brooklyn	Brighton Beach	Boro Zone
30	Queens	Broad Channel	Boro Zone
31	Bronx	Bronx Park	Boro Zone
32	Bronx	Bronxdale	Boro Zone
33	Brooklyn	Brooklyn Heights	Boro Zone
34	Brooklyn	Brooklyn Navy Yard	Boro Zone
35	Brooklyn	Brownsville	Boro Zone
36	Brooklyn	Bushwick North	Boro Zone
37	Brooklyn	Bushwick South	Boro Zone
38	Queens	Cambria Heights	Boro Zone
39	Brooklyn	Canarsie	Boro Zone
40	Brooklyn	Carroll Gardens	Boro Zone
41	Manhattan	Central Harlem	Boro Zone
42	Manhattan	Central Harlem North	Boro Zone
43	Manhattan	Central Park	Yellow Zone
44	Staten Island	Charleston/Tottenville	Boro Zone
45	Manhattan	Chinatown	Yellow Zone
46	Bronx	City Island	Boro Zone
47	Bronx	Claremont/Bathgate	Boro Zone
48	Manhattan	Clinton East	Yellow Zone
49	Brooklyn	Clinton Hill	Boro Zone
50	Manhattan	Clinton West	Yellow Zone
51	Bronx	Co-Op City	Boro Zone
52	Brooklyn	Cobble Hill	Boro Zone
53	Queens	College Point	Boro Zone
54	Brooklyn	Columbia Street	Boro Zone
55	Brooklyn	Coney Island	Boro Zone
56	Queens	Corona	Boro Zone
57	Queens	Corona	Boro Zone
58	Bronx	Country Club	Boro Zone
59	Bronx	Crotona Park	Boro Zone
60	Bronx	Crotona Park East	Boro Zone
61	Brooklyn	Crown Heights North	Boro Zone
62	Brooklyn	Crown Heights South	Boro Zone
63	Brooklyn	Cypress Hills	Boro Zone
64	Queens	Douglaston	Boro Zone
65	Brooklyn	Downtown Brooklyn/MetroTech	Boro Zone
66	Brooklyn	DUMBO/Vinegar Hill	Boro Zone
67	Brooklyn	Dyker Heights	Boro Zone
68	Manhattan	East Chelsea	Yellow Zone
69	Bronx	East Concourse/Concourse Village	Boro Zone
70	Queens	East Elmhurst	Boro Zone
71	Brooklyn	East Flatbush/Farragut	Boro Zone
72	Brooklyn	East Flatbush/Remsen Village	Boro Zone
73	Queens	East Flushing	Boro Zone
74	Manhattan	East Harlem North	Boro Zone
75	Manhattan	East Harlem South	Boro Zone
76	Brooklyn	East New York	Boro Zone
77	Brooklyn	East New York/Pennsylvania Avenue	Boro Zone
78	Bronx	East Tremont	Boro Zone
79	Manhattan	East Village	Yellow Zone
80	Brooklyn	East Williamsburg	Boro Zone
81	Bronx	Eastchester	Boro Zone
82	Queens	Elmhurst	Boro Zone
83	Queens	Elmhurst/Maspeth	Boro Zone
84	Staten Island	Eltingville/Annadale/Prince's Bay	Boro Zone
85	Brooklyn	Erasmus	Boro Zone
86	Queens	Far Rockaway	Boro Zone
87	Manhattan	Financial District North	Yellow Zone
88	Manhattan	Financial District South	Yellow Zone

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

89	Brooklyn	Flatbush/Ditmas Park	Boro Zone
90	Manhattan	Flatiron	Yellow Zone
91	Brooklyn	Flatlands	Boro Zone
92	Queens	Flushing	Boro Zone
93	Queens	Flushing Meadows-Corona Park	Boro Zone
94	Bronx	Fordham South	Boro Zone
95	Queens	Forest Hills	Boro Zone
96	Queens	Forest Park/Highland Park	Boro Zone
97	Brooklyn	Fort Greene	Boro Zone
98	Queens	Fresh Meadows	Boro Zone
99	Staten Island	Freshkills Park	Boro Zone
100	Manhattan	Garment District	Yellow Zone
101	Queens	Glen Oaks	Boro Zone
102	Queens	Glendale	Boro Zone
103	Manhattan	Governor's Island/Ellis Island/Liberty Island	Yellow Zone
104	Manhattan	Governor's Island/Ellis Island/Liberty Island	Yellow Zone
105	Manhattan	Governor's Island/Ellis Island/Liberty Island	Yellow Zone
106	Brooklyn	Gowanus	Boro Zone
107	Manhattan	Gramercy	Yellow Zone
108	Brooklyn	Gravesend	Boro Zone
109	Staten Island	Great Kills	Boro Zone
110	Staten Island	Great Kills Park	Boro Zone
111	Brooklyn	Green-Wood Cemetery	Boro Zone
112	Brooklyn	Greenpoint	Boro Zone
113	Manhattan	Greenwich Village North	Yellow Zone
114	Manhattan	Greenwich Village South	Yellow Zone
115	Staten Island	Grymes Hill/Clifton	Boro Zone
116	Manhattan	Hamilton Heights	Boro Zone
117	Queens	Hammels/Arverne	Boro Zone
118	Staten Island	Heartland Village/Todt Hill	Boro Zone
119	Bronx	Highbridge	Boro Zone
120	Manhattan	Highbridge Park	Boro Zone
121	Queens	Hillcrest/Pomonok	Boro Zone
122	Queens	Hollis	Boro Zone
123	Brooklyn	Homecrest	Boro Zone
124	Queens	Howard Beach	Boro Zone
125	Manhattan	Hudson Sq	Yellow Zone
126	Bronx	Hunts Point	Boro Zone
127	Manhattan	Inwood	Boro Zone
128	Manhattan	Inwood Hill Park	Boro Zone
129	Queens	Jackson Heights	Boro Zone
130	Queens	Jamaica	Boro Zone
131	Queens	Jamaica Estates	Boro Zone
132	Queens	JFK Airport	Airports
133	Brooklyn	Kensington	Boro Zone
134	Queens	Kew Gardens	Boro Zone
135	Queens	Kew Gardens Hills	Boro Zone
136	Bronx	Kingsbridge Heights	Boro Zone
137	Manhattan	Kips Bay	Yellow Zone
138	Queens	LaGuardia Airport	Airports
139	Queens	Laurelton	Boro Zone
140	Manhattan	Lenox Hill East	Yellow Zone
141	Manhattan	Lenox Hill West	Yellow Zone
142	Manhattan	Lincoln Square East	Yellow Zone
143	Manhattan	Lincoln Square West	Yellow Zone
144	Manhattan	Little Italy/NoLiTa	Yellow Zone
145	Queens	Long Island City/Hunters Point	Boro Zone
146	Queens	Long Island City/Queens Plaza	Boro Zone
147	Bronx	Longwood	Boro Zone
148	Manhattan	Lower East Side	Yellow Zone
149	Brooklyn	Madison	Boro Zone
150	Brooklyn	Manhattan Beach	Boro Zone
151	Manhattan	Manhattan Valley	Yellow Zone
152	Manhattan	Manhattanville	Boro Zone
153	Manhattan	Marble Hill	Boro Zone
154	Brooklyn	Marine Park/Floyd Bennett Field	Boro Zone

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

155	Brooklyn	Marine Park/Mill Basin	Boro Zone
156	Staten Island	Mariners Harbor	Boro Zone
157	Queens	Maspeth	Boro Zone
158	Manhattan	Meatpacking/West Village West	Yellow Zone
159	Bronx	Melrose South	Boro Zone
160	Queens	Middle Village	Boro Zone
161	Manhattan	Midtown Center	Yellow Zone
162	Manhattan	Midtown East	Yellow Zone
163	Manhattan	Midtown North	Yellow Zone
164	Manhattan	Midtown South	Yellow Zone
165	Brooklyn	Midwood	Boro Zone
166	Manhattan	Morningside Heights	Boro Zone
167	Bronx	Morrisania/Melrose	Boro Zone
168	Bronx	Mott Haven/Port Morris	Boro Zone
169	Bronx	Mount Hope	Boro Zone
170	Manhattan	Murray Hill	Yellow Zone
171	Queens	Murray Hill-Queens	Boro Zone
172	Staten Island	New Dorp/Midland Beach	Boro Zone
173	Queens	North Corona	Boro Zone
174	Bronx	Norwood	Boro Zone
175	Queens	Oakland Gardens	Boro Zone
176	Staten Island	Oakwood	Boro Zone
177	Brooklyn	Ocean Hill	Boro Zone
178	Brooklyn	Ocean Parkway South	Boro Zone
179	Queens	Old Astoria	Boro Zone
180	Queens	Ozone Park	Boro Zone
181	Brooklyn	Park Slope	Boro Zone
182	Bronx	Parkchester	Boro Zone
183	Bronx	Pelham Bay	Boro Zone
184	Bronx	Pelham Bay Park	Boro Zone
185	Bronx	Pelham Parkway	Boro Zone
186	Manhattan	Penn Station/Madison Sq West	Yellow Zone
187	Staten Island	Port Richmond	Boro Zone
188	Brooklyn	Prospect-Lefferts Gardens	Boro Zone
189	Brooklyn	Prospect Heights	Boro Zone
190	Brooklyn	Prospect Park	Boro Zone
191	Queens	Queens Village	Boro Zone
192	Queens	Queensboro Hill	Boro Zone
193	Queens	Queensbridge/Ravenswood	Boro Zone
194	Manhattan	Randalls Island	Yellow Zone
195	Brooklyn	Red Hook	Boro Zone
196	Queens	Rego Park	Boro Zone
197	Queens	Richmond Hill	Boro Zone
198	Queens	Ridgewood	Boro Zone
199	Bronx	Rikers Island	Boro Zone
200	Bronx	Riverdale/North Riverdale/Fieldston	Boro Zone
201	Queens	Rockaway Park	Boro Zone
202	Manhattan	Roosevelt Island	Boro Zone
203	Queens	Rosedale	Boro Zone
204	Staten Island	Rossville/Woodrow	Boro Zone
205	Queens	Saint Albans	Boro Zone
206	Staten Island	Saint George/New Brighton	Boro Zone
207	Queens	Saint Michaels Cemetery/Woodside	Boro Zone
208	Bronx	Schuylerville/Edgewater Park	Boro Zone
209	Manhattan	Seaport	Yellow Zone
210	Brooklyn	Sheepshead Bay	Boro Zone
211	Manhattan	SoHo	Yellow Zone
212	Bronx	Soundview/Bruckner	Boro Zone
213	Bronx	Soundview/Castle Hill	Boro Zone
214	Staten Island	South Beach/Dongan Hills	Boro Zone
215	Queens	South Jamaica	Boro Zone
216	Queens	South Ozone Park	Boro Zone
217	Brooklyn	South Williamsburg	Boro Zone
218	Queens	Springfield Gardens North	Boro Zone
219	Queens	Springfield Gardens South	Boro Zone
220	Bronx	Spuyten Duyvil/Kingsbridge	Boro Zone

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

221	Staten Island	Stapleton	Boro Zone
222	Brooklyn	Starrett City	Boro Zone
223	Queens	Steinway	Boro Zone
224	Manhattan	Stuy Town/Peter Cooper Village	Yellow Zone
225	Brooklyn	Stuyvesant Heights	Boro Zone
226	Queens	Sunnyside	Boro Zone
227	Brooklyn	Sunset Park East	Boro Zone
228	Brooklyn	Sunset Park West	Boro Zone
229	Manhattan	Sutton Place/Turtle Bay North	Yellow Zone
230	Manhattan	Times Sq/Theatre District	Yellow Zone
231	Manhattan	TriBeCa/Civic Center	Yellow Zone
232	Manhattan	Two Bridges/Seward Park	Yellow Zone
233	Manhattan	UN/Turtle Bay South	Yellow Zone
234	Manhattan	Union Sq	Yellow Zone
235	Bronx	University Heights/Morris Heights	Boro Zone
236	Manhattan	Upper East Side North	Yellow Zone
237	Manhattan	Upper East Side South	Yellow Zone
238	Manhattan	Upper West Side North	Yellow Zone
239	Manhattan	Upper West Side South	Yellow Zone
240	Bronx	Van Cortlandt Park	Boro Zone
241	Bronx	Van Cortlandt Village	Boro Zone
242	Bronx	Van Nest/Morris Park	Boro Zone
243	Manhattan	Washington Heights North	Boro Zone
244	Manhattan	Washington Heights South	Boro Zone
245	Staten Island	West Brighton	Boro Zone
246	Manhattan	West Chelsea/Hudson Yards	Yellow Zone
247	Bronx	West Concourse	Boro Zone
248	Bronx	West Farms/Bronx River	Boro Zone
249	Manhattan	West Village	Yellow Zone
250	Bronx	Westchester Village/Unionport	Boro Zone
251	Staten Island	Westerleigh	Boro Zone
252	Queens	Whitestone	Boro Zone
253	Queens	Willets Point	Boro Zone
254	Bronx	Williamsbridge/Olinville	Boro Zone
255	Brooklyn	Williamsburg (North Side)	Boro Zone
256	Brooklyn	Williamsburg (South Side)	Boro Zone
257	Brooklyn	Windsor Terrace	Boro Zone
258	Queens	Woodhaven	Boro Zone
259	Bronx	Woodlawn/Wakefield	Boro Zone
260	Queens	Woodside	Boro Zone
261	Manhattan	World Trade Center	Yellow Zone
262	Manhattan	Yorkville East	Yellow Zone
263	Manhattan	Yorkville West	Yellow Zone
264	Unknown	NV	nan
265	Unknown	nan	nan

Table 4 - Taxi Zone Dictionary

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

12. Dashboard

In this section we will discuss our Dash Application and its features. Also, we would like to showcase some of the observations which we believe can be better visualized dynamically.
(Click here to Access)

12.1 Dataset Description

Fig 34. Dashboard - Dataset Description

The application opens with a brief description of our Dataset and it also cites the sources from which the data has been referenced. We can navigate through various tabs to access different functionalities of our application.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

12.2 Dataframe Cleaner

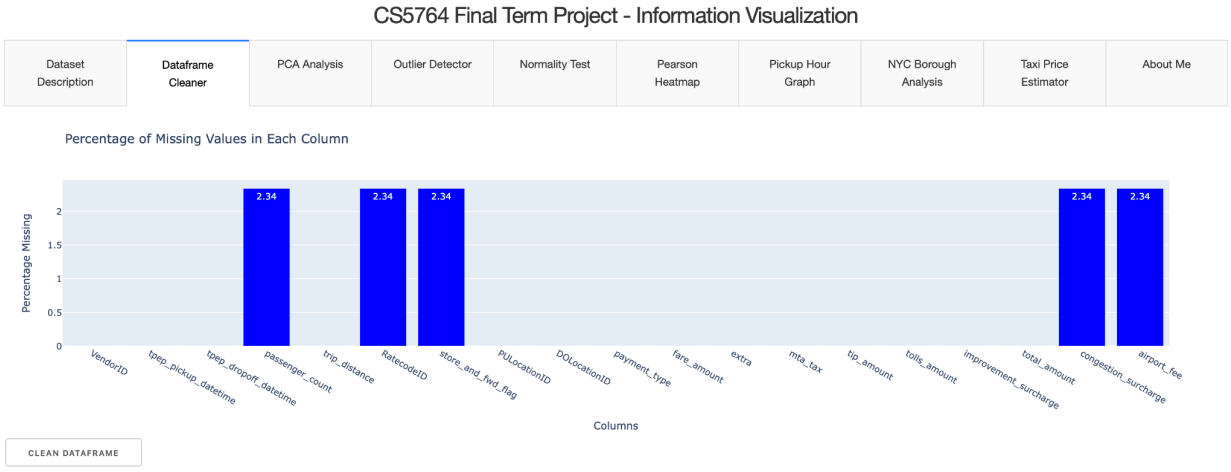
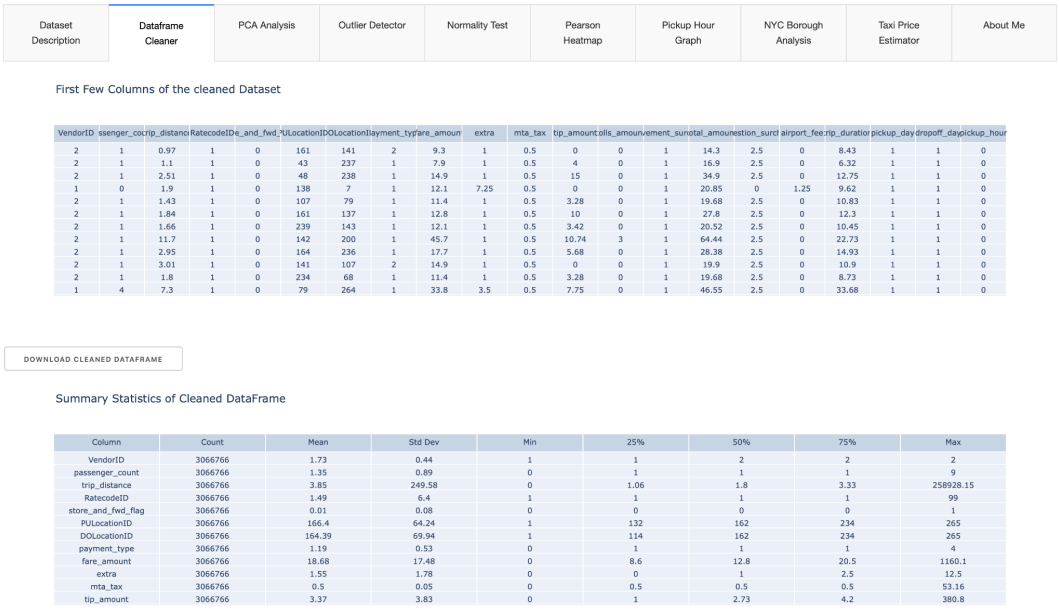


Fig 35. Dashboard - Dataframe Cleaner

The first feature of the application is a 'Dataframe Cleaner'. Upon clicking on the Dataframe Cleaner tab we first see a bar plot showcasing percentage of missing observations in the unclean dataset. As explored earlier we can clearly see Passenger Count, Rate Code ID, Store and Forward Flag, Congestion Surcharge and Airport fee to have empty values.

The next thing that we can do is - by the click of a button we can clean the whole dataset and download the cleaned dataset.



VendorID	ssenger_co	rip_distanc	RatecodeID	e_and_fwd_	ULocationID	DOLocationI	ayment_typ	fare_amoun	extra	mta_tax	tip_amount	olls_amoun	ement_sur	otal_amoun	stion_surc	airport_fee	rip_duratio	pickup_day	dropoff_day	pickup_hour
2	1	0.97	1	0	161	141	2	9.3	1	0.5	0	0	1	14.3	2.5	0	8.43	1	1	0
2	1	1.1	1	0	43	237	1	7.9	1	0.5	4	0	1	16.9	2.5	0	6.32	1	1	0
2	1	2.51	1	0	48	238	1	14.9	1	0.5	15	0	1	34.9	2.5	0	12.75	1	1	0
1	0	1.9	1	0	138	7	1	12.1	7.25	0.5	0	0	1	20.85	0	1.25	9.62	1	1	0
2	1	1.43	1	0	107	79	1	11.4	1	0.5	3.28	0	1	19.68	2.5	0	10.83	1	1	0
2	1	1.84	1	0	161	137	1	12.8	1	0.5	10	0	1	27.8	2.5	0	12.3	1	1	0
2	1	1.66	1	0	239	143	1	12.1	1	0.5	3.42	0	1	20.52	2.5	0	10.45	1	1	0
2	1	11.7	1	0	142	200	1	45.7	1	0.5	10.74	3	1	64.44	2.5	0	22.73	1	1	0
2	1	2.95	1	0	164	236	1	17.7	1	0.5	5.68	0	1	28.38	2.5	0	14.93	1	1	0
2	1	3.01	1	0	141	107	2	14.9	1	0.5	0	0	1	19.9	2.5	0	10.9	1	1	0
2	1	1.8	1	0	234	68	1	11.4	1	0.5	3.28	0	1	19.68	2.5	0	8.73	1	1	0
1	4	7.3	1	0	79	264	1	33.8	3.5	0.5	7.75	0	1	46.55	2.5	0	33.68	1	1	0

Column	Count	Mean	Std Dev	Min	25%	50%	75%	Max
VendorID	3066766	1.73	0.44	1	1	2	2	2
passenger_count	3066766	1.35	0.89	0	1	1	1	9
trip_distance	3066766	3.85	249.58	0	1.06	1.8	3.33	258928.15
RatecodeID	3066766	1.49	6.4	1	1	1	1	99
store_and_fwd_flag	3066766	0.01	0.08	0	0	0	0	1
PULocationID	3066766	166.4	64.24	1	132	162	234	265
DOLocationID	3066766	164.39	69.94	1	114	162	234	265
payment_type	3066766	1.19	0.53	0	1	1	1	4
fare_amount	3066766	18.68	17.48	0	8.6	12.8	20.5	1160.1
extra	3066766	1.55	1.78	0	0	1	2.5	12.5
mta_tax	3066766	0.5	0.05	0	0.5	0.5	0.5	53.16
tip_amount	3066766	3.37	3.83	0	1	2.73	4.2	380.8

Fig 36. Dashboard - Cleaned Dataframe

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

The final interface shows us a few details about our dataset which include the first few columns of our dataset and how they look after being cleaned. And also the summary of our cleaned dataframe.

12.3 PCA Analysis

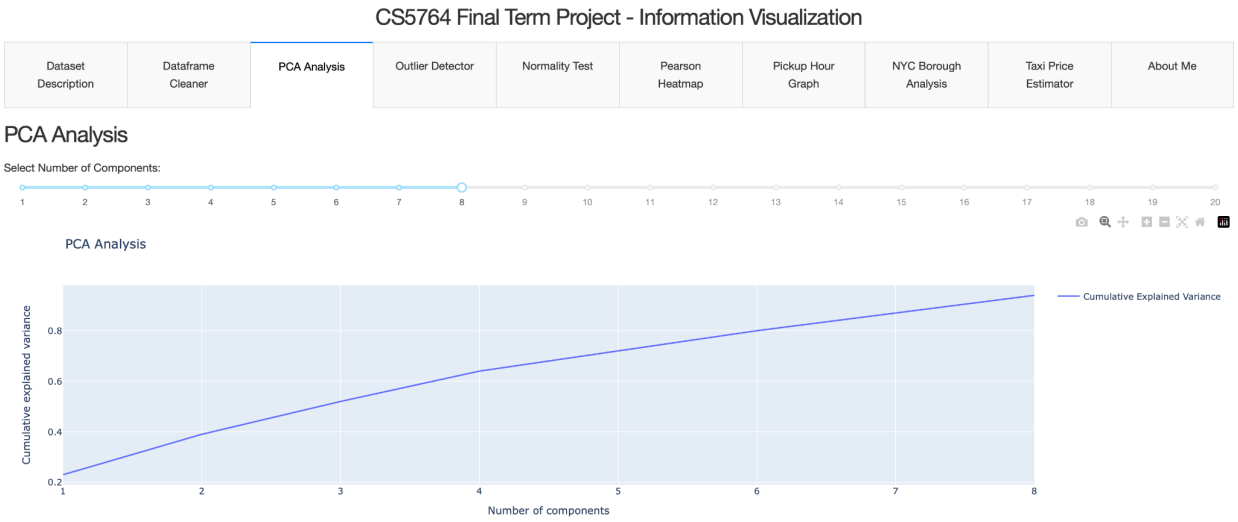


Fig 37. Dashboard - PCA Analysis

The next tab is PCA Analysis is responsible for showcasing the PCA Analysis of our Dataset. Here we have a slider to choose the number of components and the moment we cross the 95% threshold of cumulative explained variance it traces that particular component.

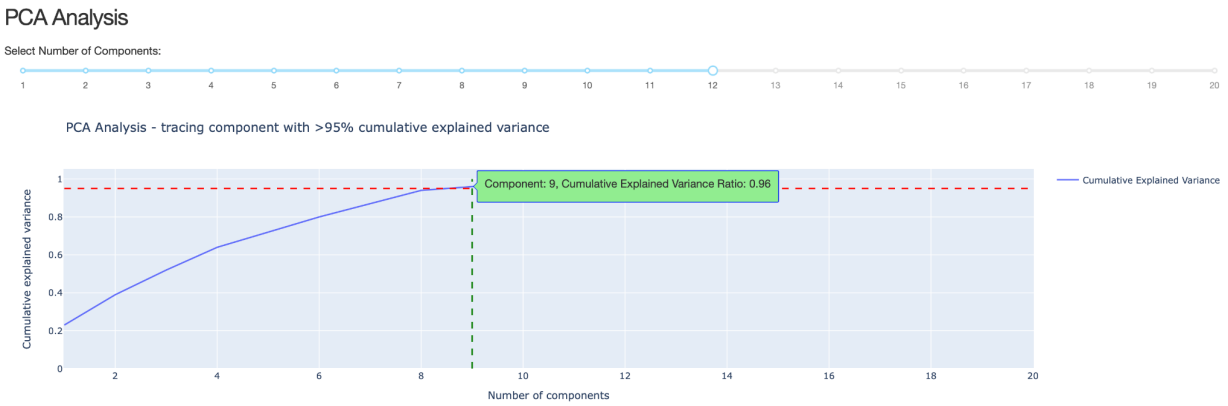


Fig 38. Dashboard - PCA Analysis tracing component with >95% Explained Variance

A tooltip is also designed to showcase which number of components has what cumulative explained ratio. We get the same output of 9 components as seen in previous analysis.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

12.4 Outlier Detector

The next tab is an Outlier Detector. This tab is responsible for plotting boxplots of the features which contain outliers.

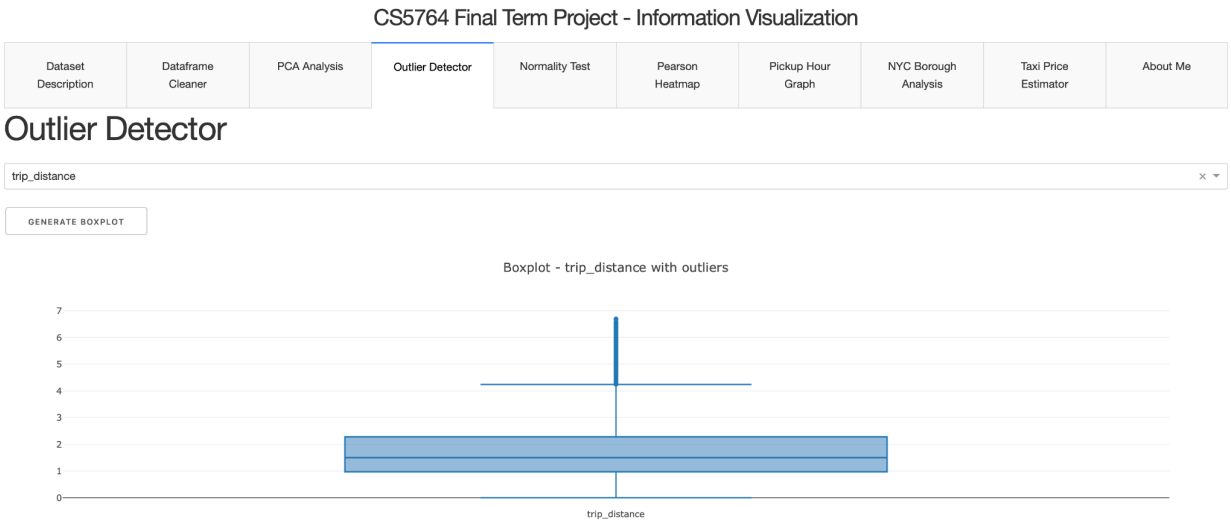


Fig 39. Dashboard - Outlier Detector

The interface is fairly simple. We choose the feature in which we want to detect outliers. After selecting that feature we can simply click on ‘Generate Boxplot’ to generate a boxplot with and without outliers.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

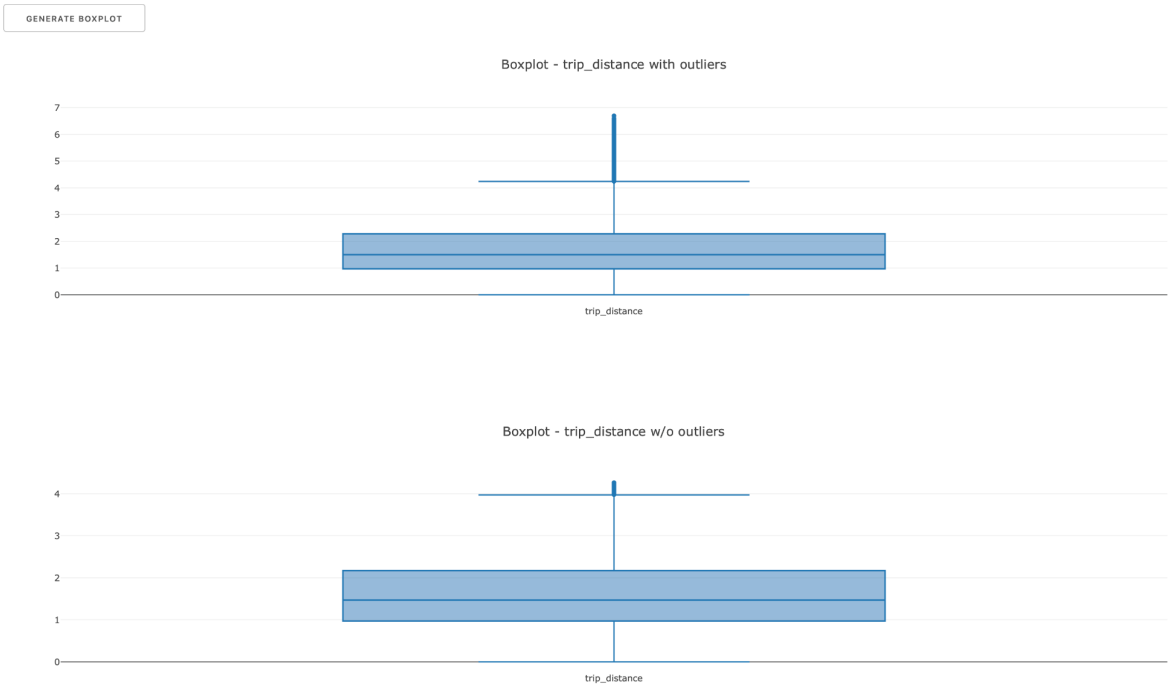


Fig 40. Dashboard - Generating Boxplot after we remove Outliers

12.5 Normality Test

The next tab is responsible for performing normality tests on the selected feature.

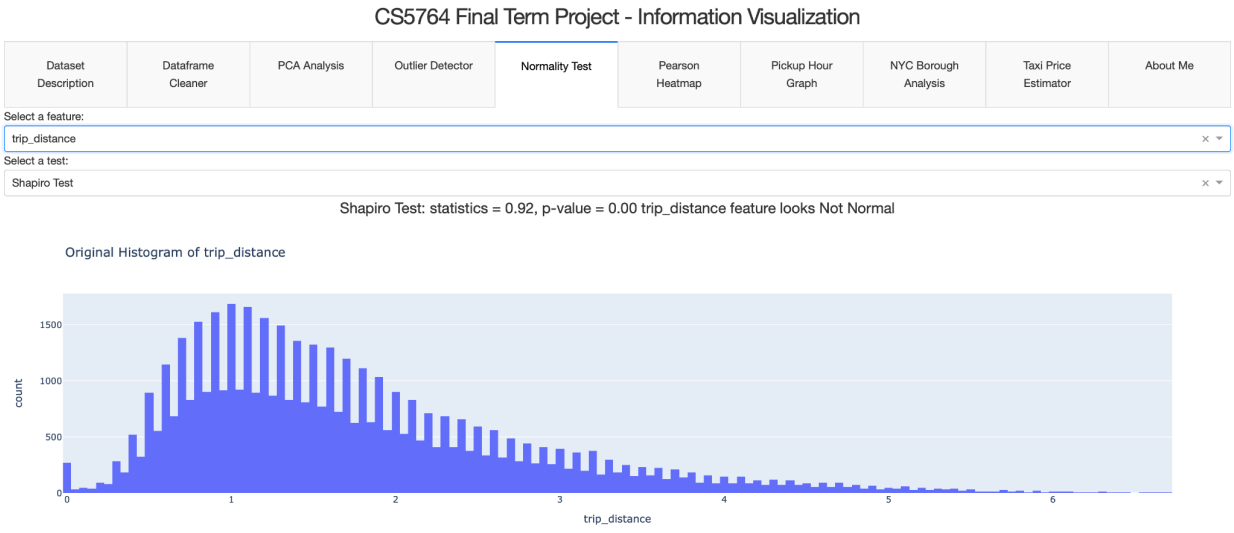


Fig 41. Dashboard - Normality Tester

The moment we select a feature we are also asked to pick a test of our own.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

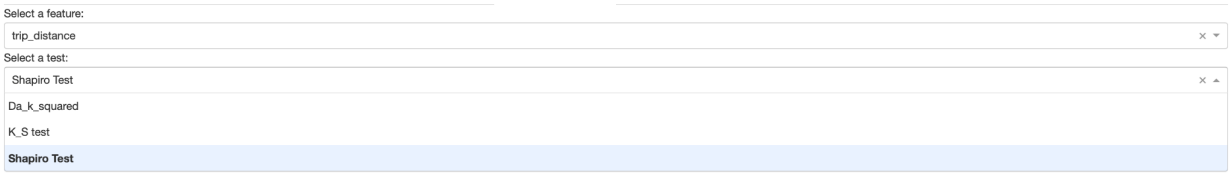


Fig 42. Dashboard - Selecting Different Normality Tests

The moment we select a feature and test can see two things

1. Test Statistics and the information of that feature being Normal or Not.
2. If the feature is not Normal, we perform a Box-Cox transformation on that feature and display another histogram for that.

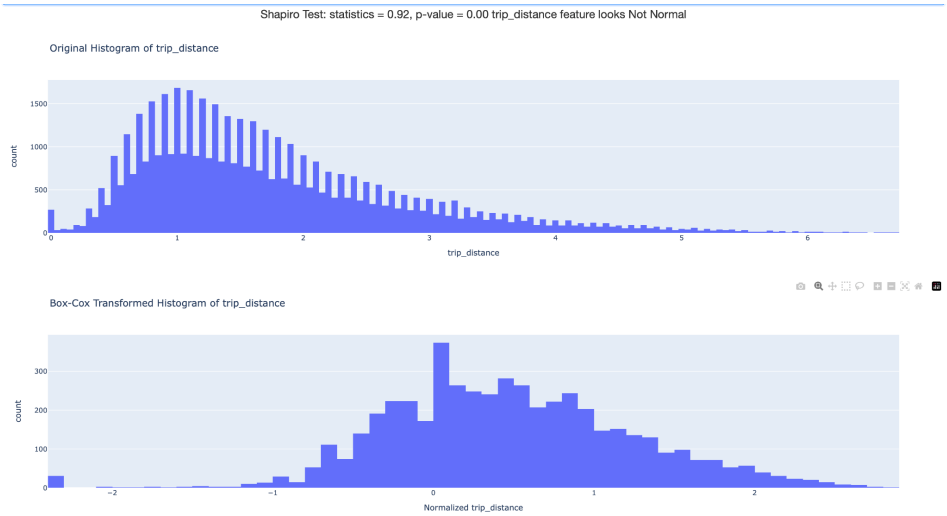


Fig 43. Dashboard - Plotting Histograms for Features which are not Normal and transforming them using Box-Cox transformation

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

12.6 Pearson Heatmap

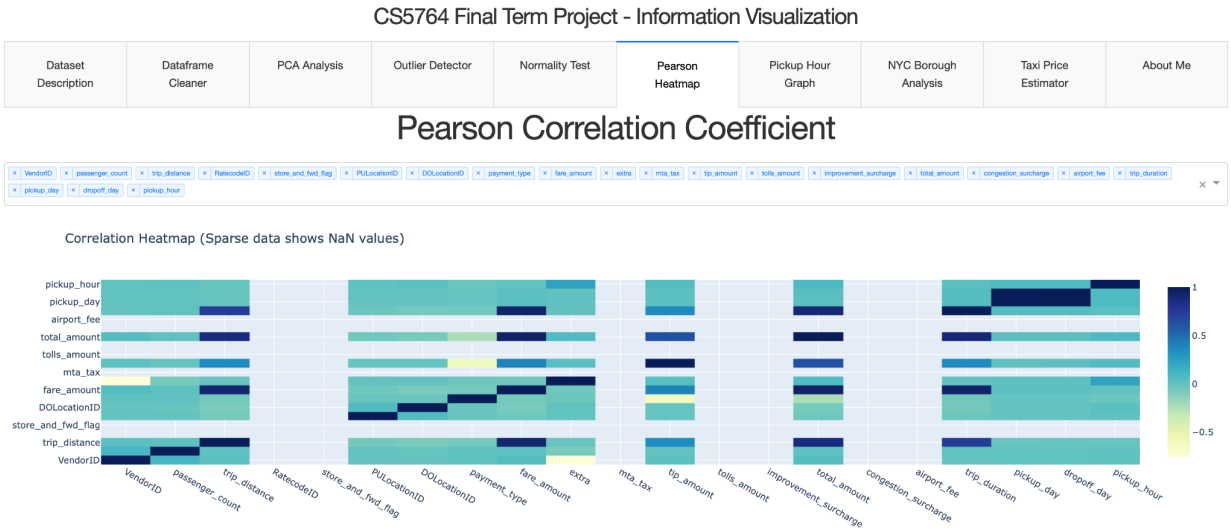


Fig 44. Dashboard - Pearson Heatmap

This tab in our application is responsible for a statistical analysis of finding correlation coefficients between all the features. We can select multiple features from the dropdown and fit it inside the heatmap.

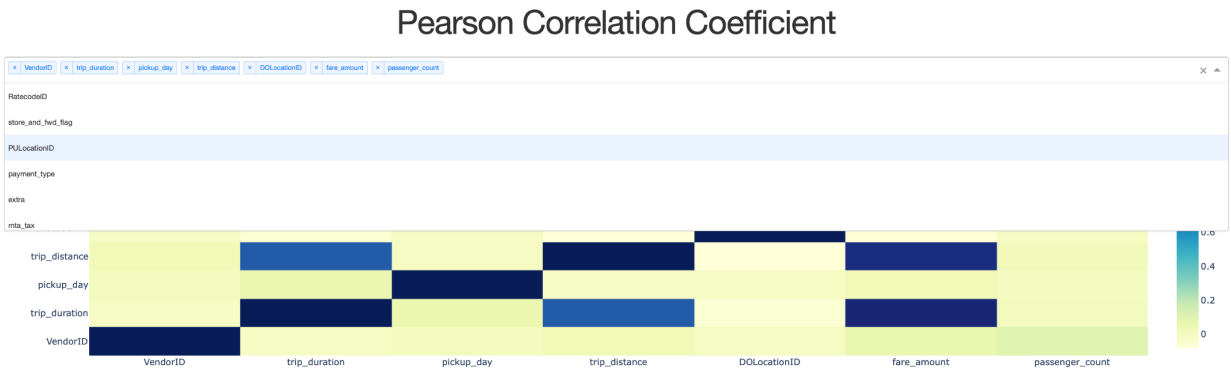


Fig 45. Dashboard - Selecting multiple features to dynamically calculate the correlation heatmap

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

12.7 Pickup Hour Graph

This tab in our application is responsible for generating a line graph showing the trends of Average Total Amount vs Pickup Hour.

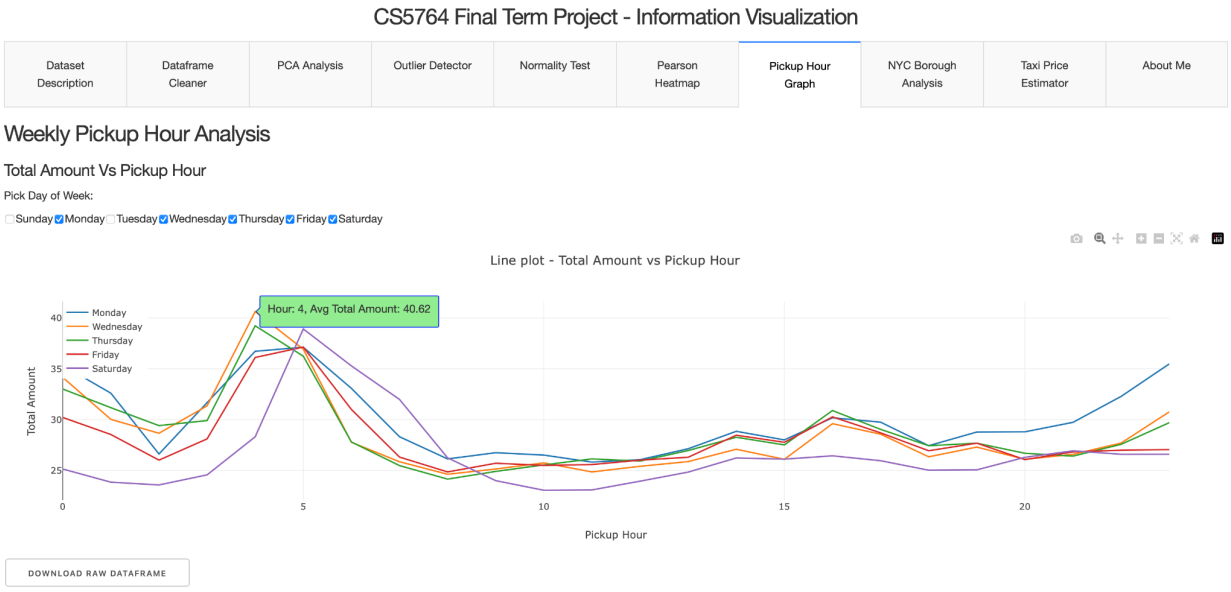


Fig 46. Dashboard - Pickup Hour Graph

In our interface we have a checklist which picks for which day you want to plot the graph for. We also have another download button to download only the particular day data. So, if you just want to download data for Friday, Saturday and Sunday, you just select the days and click on download to get yourself a .csv file. We also have a tooltip showing which hour has what average of Total Amount.

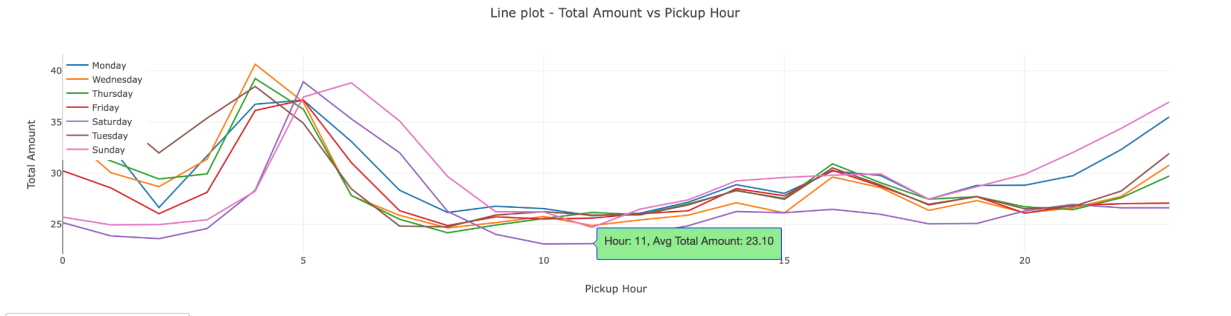


Fig 47. Dashboard - Line plot of Aggregated Total Amount Data by Weekday

Lets see a few observations from this line plot as well. Contrary to our previous plots we see that the average total amount is very high during midnight and it is near the global mean during

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

the rush hour. This looks counter intuitive but the actual reason for such behavior is that during Midnight hours we can see that drivers charge an 'Extra' charge because of which we see such a pattern. Another observation which can be made is that for the Month of January 2023, The highest averages have been around Wednesdays.

12.8 NYC Borough Analysis

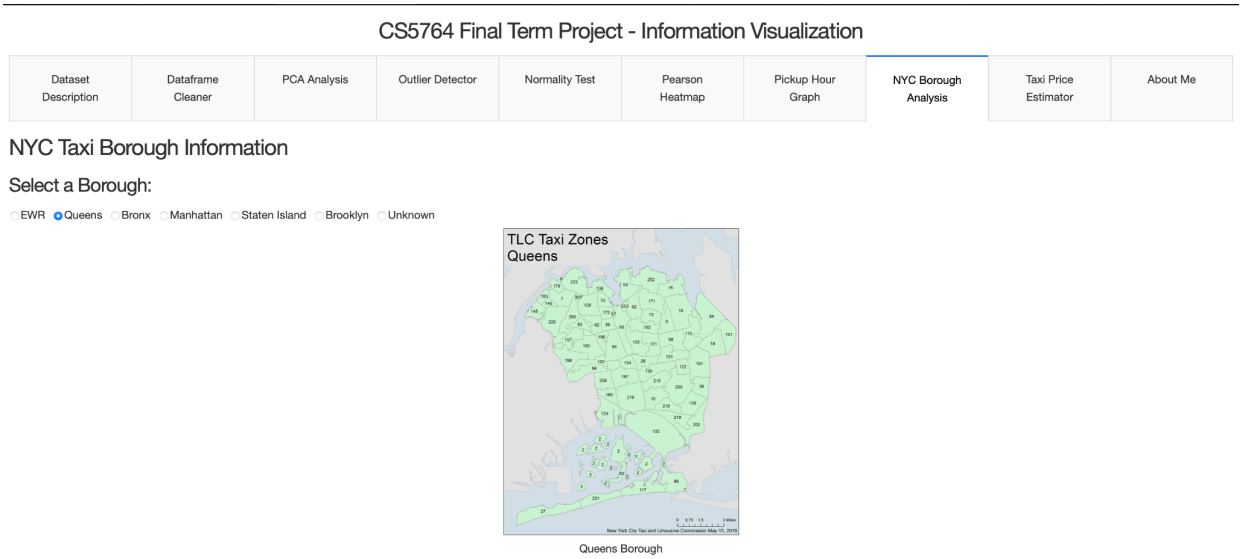


Fig 48. Dashboard - NYC Borough Analysis

The next tab we see is a thorough analysis of all the Boroughs of NYC. We plot **Area Plots** of top 10 most frequented locations by the drivers for both pickups and drop offs.

Our interface allows us to select various Boroughs available in the Dataset via a Radio Item button.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

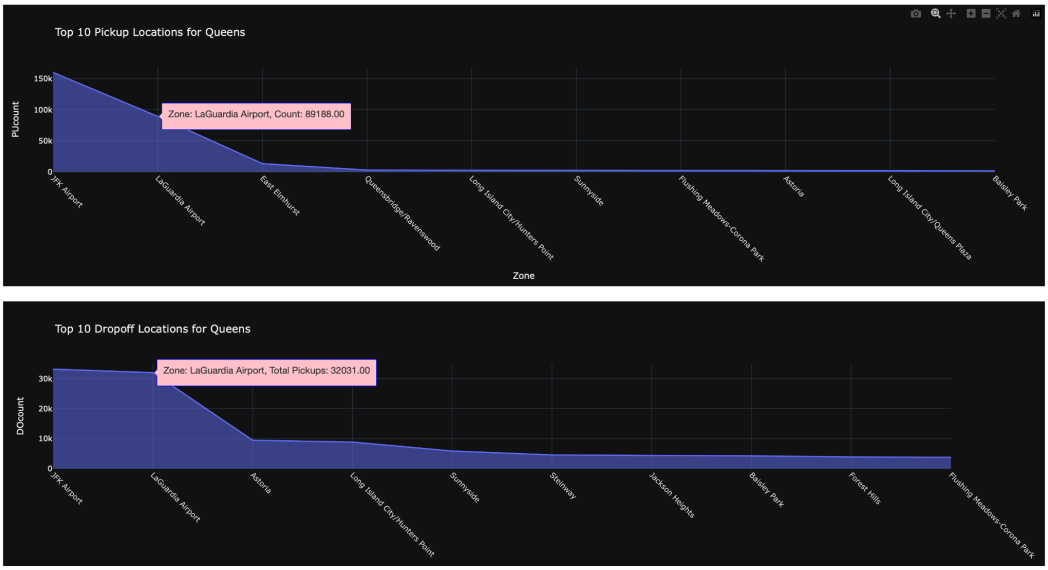


Fig 49. Dashboard - Area Plots for most frequented pickup and drop off locations

We also have a tooltip printing the zone and the amount of total pickups and drop offs made for that zone.

For some boroughs there are only single location of pickup and drop off i.e. EWR and Unknown (which are the unregistered trips). For those boroughs we just plot the count plots.

Fig 50. Dashboard - Count Plots for Pickups and Dropoffs made at EWR Borough

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

12.9 Taxi Price Estimator

The final tab in our application is a Taxi Price Estimator trained on a Machine Learning Model (Linear Regression). We get a lot of options in our inputs using which we can find a range and highly probable taxi fare.

Fig 51. Dashboard - Taxi Price Estimator

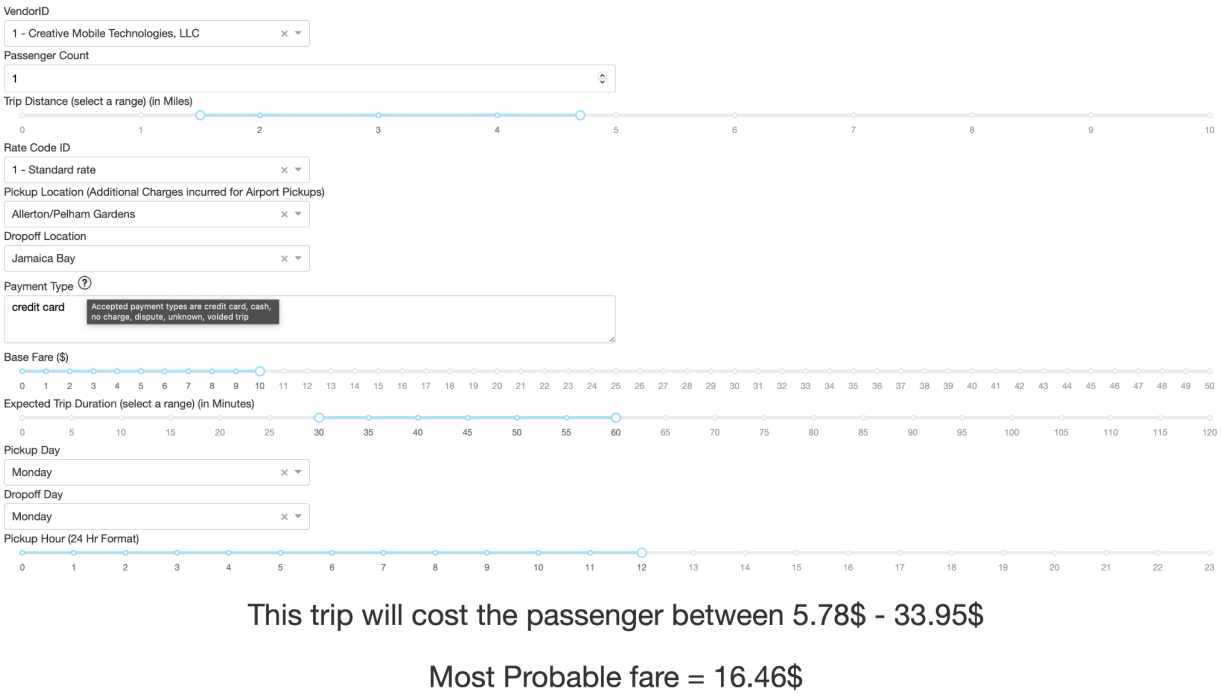
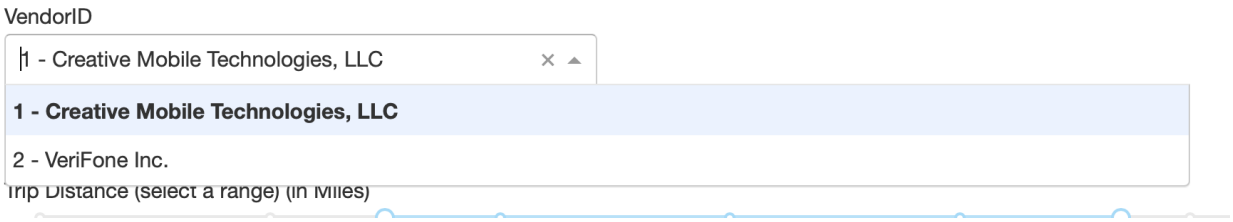


Fig 52. Dashboard - Various Inputs in Estimator for predicting taxi fare price

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

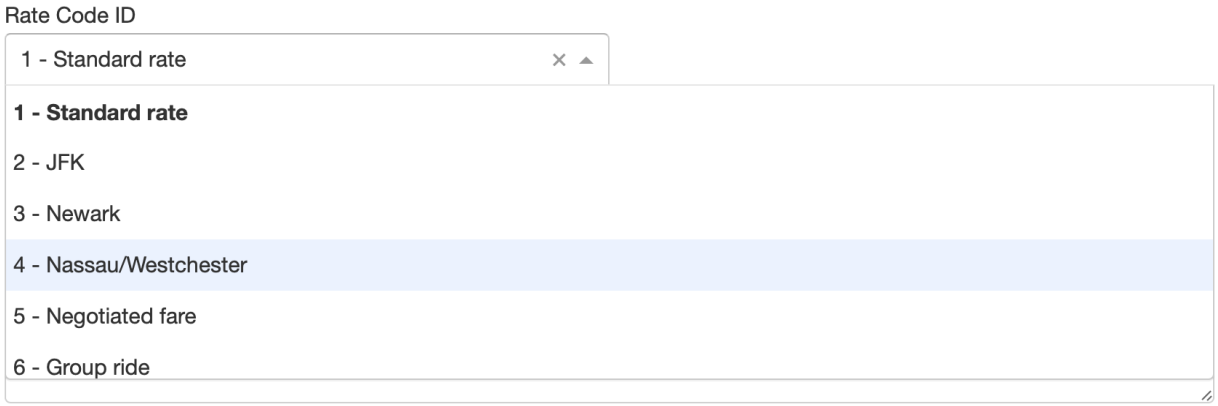
The first input is a Vendor ID selector. It is a simple dropdown which asks us which vendor the current driver belongs to.



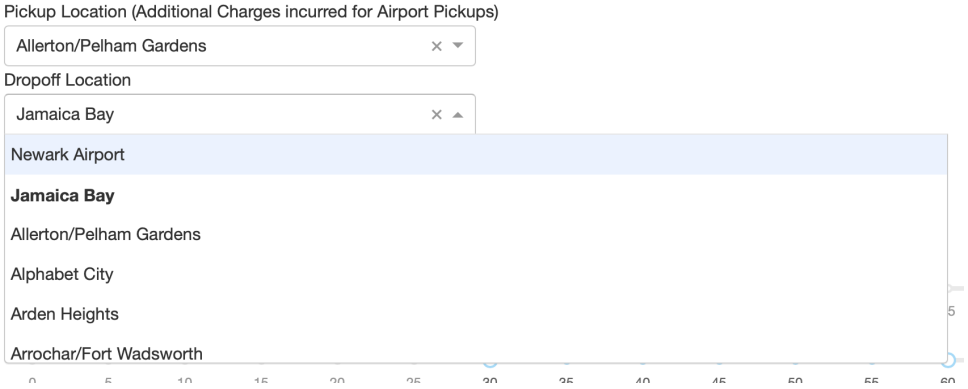
The second input is the number of passengers during that trip.

Third input is a range slider for approximate trip distance.

Fourth input is a simple dropdown asking the user to input the Rate Code.

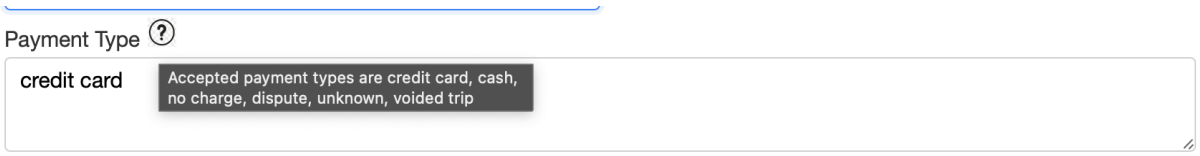


Fourth and Fifth inputs are our pickup and drop off locations.



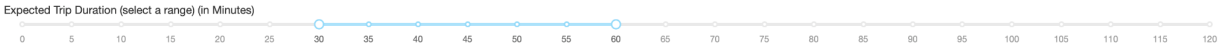
App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

Sixth Input is a Text Area Input. It asks what type of payment the customer is going to make. We are also showing a small hint on what kind of inputs are valid.

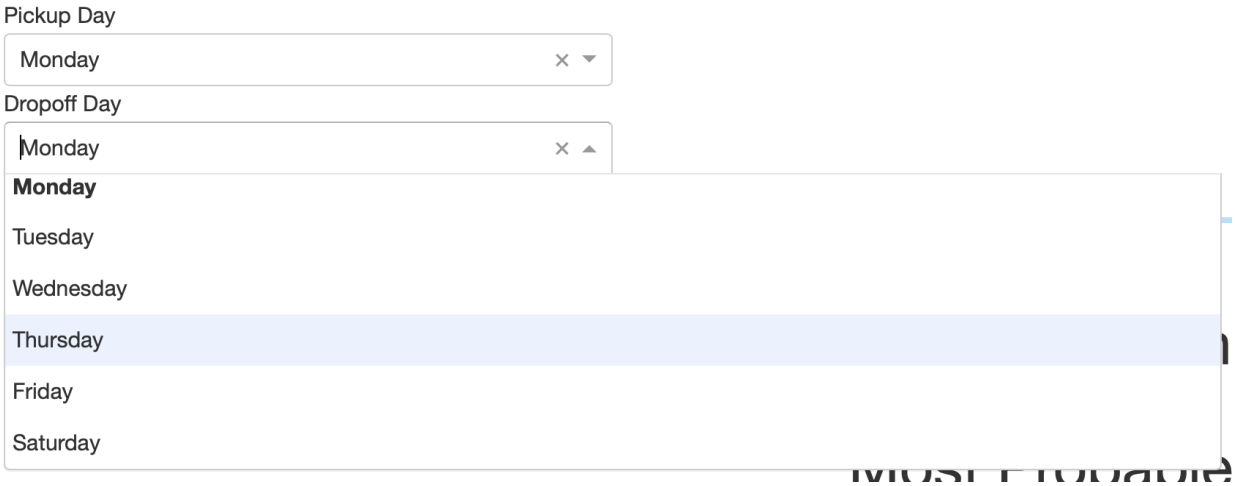


Seventh Input is a simple slider which sets the base fare ($)

Eighth Input is again a range slider asking for a expected trip duration



Ninth and Tenth Inputs are simple dropdowns asking for pickup and drop off day



Last Input is again a slider asking for the time of day (in 24 Hour Format)

This trip will cost the passenger between 9.07$ - 33.95$

Most Probable fare = 16.46$

The final output is a range of fare amount and also the most probable fare. The range is calculated based on the Pickup Location. We have calculated the standard deviations of total amount of each pickup location and adjust that accordingly with our estimated price to give this range as an output.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

12.10 About Me

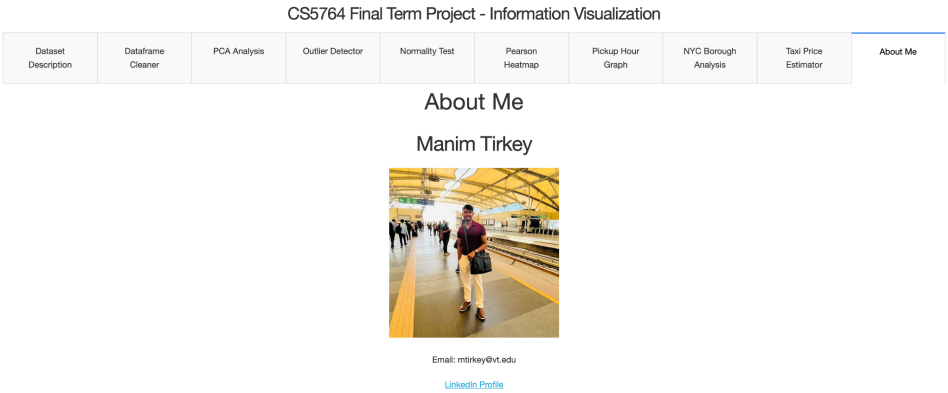


Fig 52. Dashboard - About Me Section

The last tab is an ‘About Me’ Tab printing the Author information and his LinkedIn ID as well.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

13. Conclusion

The NYC taxi dataset is an extensive real world dataset. With the limited information contained inside the dataset as columns we can expand it to a different set of columns to explain our data in different ways. The data comes with temporal information out of which we extracted pickup hour, drop off hour, trip duration and other temporal attributes which helped us determine the general trends of time. We also aggregated our data in various ways to understand which particular regions in our list of zones have the most pickups, high paid trips and highly tipped drivers. We also created an application for dynamic analysis of various statistical tests and also dynamically plotted various plots for noticing any patterns in our data.

Let us briefly summarize what observations we have made.

13.1 Understanding Plots

We created many plots to understand the general behavior of fare given along with the amount of tips given to the driver. We understood that during the month of January 2023, a few anomalies in the tips were found during the weekends. We concluded that most trips contained only one passenger. We saw various distributions of fare amount, total amount and tip amount which made us understand that it is strictly not correlated. Here I would like to refer to the official response of TLC services which said - “It is impossible to pre-calculate a fare, because the meter rate depends on traffic, construction, weather, and route to the destination.” Our analysis conformed with this statement and we couldn’t find any general trend among these. But some obvious observations were made such as high trip duration means higher fare, high fare amount meant higher total amount as well. We also saw that Verifone vendors have a better market capture given that they had more volume of taxis roaming around NYC and getting their trips done. We also saw rug plot and histograms to see that most pickups are being made during the rush hour and after that we see most pickups being made after midnight. We also saw that credit cards are the most favored mode of payment indicating a lot of people have now switched to this mode of payment, cash being the second most preferred mode of payment.

13.2 Dashboard Setup

We created a dashboard by using the Dash framework. We found that it is a very convenient way of making a web application but it still requires optimization in order to load faster for users. We created our app based on our NYC taxi dataset. This app helped us to perform various statistical analysis which included PCA analysis, Normality tests, Outlier Detection and Pearson Correlation Coefficient. These tests helped us understand that real life data is NOT perfect and we need a lot of transformations (including cleaning of data) to understand the actual behavior of that data. For experimentations with various html and dash components we also made a taxi

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

price estimator which uses Linear Regression to estimate the taxi fare. We also analyzed various boroughs of NYC and the pickup hour trend (which turned out to be counter intuitive at first glance).

13.3 Dashboard Functionality

The dashboard is user-friendly. We do not have any complicated access methods to run the application. We do not need any prior knowledge in order to operate the application. It is a functional application which also helps us estimate fare prices based on pickup location.

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

Appendix

PCA Analysis, Normality Tests and Singular Value Decomposition

```
# =============
# PCA
# =============
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

df_sampled = df.sample(n=50000, random_state=5764)
df_sampled.reset_index(drop=True, inplace=True)

df_sampled3 = df.sample(n=5000, random_state=5805)
df_sampled3.reset_index(drop=True, inplace=True)

df_sampled3['store_and_fwd_flag']                                    =
df_sampled3['store_and_fwd_flag'].replace(category_mapping_fwd_flag)
df_sampled3['payment_type']                                          =
df_sampled3['payment_type'].replace(reversed_mapping)
df_sampled3['pickup_day'] = df_sampled3['pickup_day'].replace(weekday_mapping)
df_sampled3['dropoff_day']                                           =
df_sampled3['dropoff_day'].replace(weekday_mapping)
df_sampled3['RatecodeID']                                            =
df_sampled3['RatecodeID'].replace(reverse_category_mapping_ratecode)

df_sampled3        =        df_sampled3.drop(['mta_tax',        'tolls_amount',
'improvement_surcharge', 'airport_fee'], axis=1)


def normal_test(x, selected_test):
   if selected_test == 'Shapiro Test':
       stats, p = shapiro(x)
       return stats, p

   elif selected_test == 'K_S test':
       np.random.seed(5764)

       mean = np.mean(x)
       std = np.std(x)
       dist = np.random.normal(mean, std, len(x))
       stats, p = kstest(x, dist)

       return stats, p

   elif selected_test == 'Da_k_squared':
       stats, p = normaltest(x)
```


App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

```
        return stats, p


stats_table = PrettyTable()

stats_table.field_names = ['Feature', 'Da_k_squared Test', 'K_S test', 'Shapiro
Test', 'Verdict']

for col in df_sampled3.columns:
    stats1, p1 = normal_test(df_sampled3[col], 'Da_k_squared')
    stats2, p2 = normal_test(df_sampled3[col], 'K_S test')
    stats3, p3 = normal_test(df_sampled3[col], 'Shapiro Test')

    verdict = 'Normal'
    if p1 < 0.01 and p2 < 0.01 and p3 < 0.01:
        verdict = 'Not Normal'

    stats_table.add_row([col, f'Stats = {stats1: .2f}, p = {p1 :.2f}', f'Stats =
{stats2: .2f}, p = {p2 :.2f}',
                         f'Stats = {stats3: .2f}, p = {p3: .2f}', verdict])

print(stats_table.get_string(title='Normality Test'))

df_sampled2 = df_temp.sample(n=10000, random_state=5764)
df_sampled2.reset_index(drop=True, inplace=True)

X_PCA = df_sampled2.drop(['total_amount'], axis=1)
y_PCA = df_sampled['total_amount']

scalar = StandardScaler()
X_std = scalar.fit_transform(X_PCA)
X_std = pd.DataFrame(X_std, columns=X_PCA.columns)

pca = PCA(n_components=20, svd_solver='full')
pca.fit(X_std)


exp = np.cumsum(pca.explained_variance_ratio_).round(2)

pca_p = PrettyTable()

pca_p.field_names = ['Component', 'Cumulative Explained Variance Ratio']

for i in range(len(exp)):
    pca_p.add_row([i + 1, exp[i]])

print(pca_p.get_string(title='PCA Analysis'))
```


App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

```
pca2 = PCA(n_components=9, svd_solver='full')
pca2.fit(X_std)
X_PCA2 = pca2.transform(X_std)

_, d_raw, _ = np.linalg.svd(X_std)
_, d_pca, _ = np.linalg.svd(X_PCA2)

pq = PrettyTable()
pq.field_names = [f'Raw Condition Number = {np.linalg.cond(X_std): .2f}',
                  f'Transformed Condition Number = {np.linalg.cond(X_PCA2):
.2f}']

for i, (raw_val, pca_val) in enumerate(zip(d_raw, d_pca)):
   pq.add_row([f'{raw_val :.2f}', f'{pca_val :.2f}'])

for i in range(9, 20):
   pq.add_row([f'{d_raw[i]: .2f}', '-'])

print(pq.get_string(title='Singular Values'))
```

Statistical Analysis

```
# ===========
# Statistics
# ===========
print(df.describe().round(2).to_string())

sns.kdeplot(data=df_sampled3[['trip_distance', 'total_amount', 'tip_amount',
                              'fare_amount', 'trip_duration']])
plt.title("Multivariate Kernel Density Estimate - Between Numerical Features",
fontdict=title_font)
plt.tight_layout()
plt.show()
```

Snippet for Training the Model

```
# =============
# Linear model for total_amount
# =============

X = df_sampled.drop(['total_amount'], axis=1)
y = df_sampled['total_amount']
```


App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

```
X_train, X_test, y_train, y_test = train_test_split(X, y, shuffle=True,
random_state=5705, test_size=0.2)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

comparison = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})

# print(comparison.head().round(2).to_string())

# plt.scatter(y_test, y_pred)
#     plt.plot([min(y_test),     max(y_test)],     [min(y_test),     max(y_test)],
linestyle='--', color='red', linewidth=2)
# plt.xlabel('Actual Values')
# plt.ylabel('Predicted Values')
# plt.title('Actual vs Predicted Values')
# plt.show()

mse_model = mean_squared_error(y_test, y_pred)
# print(f'The MSE of the linear regression prediction is = {mse_model: .3f}')

coefficients = model.coef_
intercept = model.intercept_
feature_names = X.columns

# equation = f"The linear regression equation: y = {intercept:.2f} "
# for i, (coef, feature_name) in enumerate(zip(coefficients, feature_names)):
#     equation += f"+ {coef:.2f} * {feature_name} "
# print(equation)
```


App Access Link - https://dashapp-t5b726vjza-nn.a.run.app

References

- *TLC Trip Record Data - TLC*. (n.d.). https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page
- *Passenger Frequently Asked Questions - TLC*. (n.d.). https://www.nyc.gov/site/tlc/passengers/passenger-frequently-asked-questions.page
- *Kolmogorov–Smirnov test*. (2023, November 8). Wikipedia. https://en.wikipedia.org/wiki/Kolmogorov–Smirnov_test
- *D'Agostino's K-squared test*. (2023, April 3). Wikipedia. https://en.wikipedia.org/wiki/D%27Agostino%27s_K-squared_test
- *Shapiro–Wilk test*. (2023, July 29). Wikipedia. https://en.wikipedia.org/wiki/Shapiro–Wilk_test
- *Dash Documentation & User Guide | Plotly*. (n.d.). https://dash.plotly.com/?_gl=1*kzijju*_ga*ODY4NzE4ODI3LjE2OTQyMDk4MzQ.*_ga_6G7EE0JNSC*MTcwMTkxMTY5MS4yOS4wLjE3MDE5MTE2OTEuNjAuMC4w
- *seaborn: statistical data visualization — seaborn 0.13.0 documentation*. (n.d.). https://seaborn.pydata.org

App Access Link - https://dashapp-t5b726vjza-nn.a.run.app