Tax Trip Statistical Data Analysis Documentation

Question: 1 What libraries are used in the notebook for data analysis and visualization?

Answer: The notebook uses the following libraries:

1. pandas for data manipulation.
2. matplotlib.pyplot for basic plotting.
3. seaborn for enhanced visualizations.
4. statsmodels.api for statistical models (like the Q-Q plot).
5. scipy.stats for statistical tests (like the t-test).
6. warnings to manage warning messages.

Question: 2 What preprocessing steps are applied to the duration column?

Answer:

1. The tpep_pickup_datetime and tpep_dropoff_datetime columns are converted to datetime objects.
2. The duration is calculated by subtracting tpep_pickup_datetime from tpep_dropoff_datetime.
3. The duration is converted to minutes.
4. Outliers are removed from the duration column.

Question: 3 How does the notebook handle missing values?

Answer:

The notebook handles missing values by using the `dropna()` method to remove rows containing missing values.

Question: 4 What kind of plot is used to visualize the distribution of fare_amount for different payment_type values?

Answer:

The notebook uses a histogram to visualize the distribution of `fare_amount` for 'Card' and 'Cash' payment types.

Question: 5 What statistical test is performed to compare the fare_amount between 'Card' and 'Cash' payment types? What does the result indicate?

Answer:

1. The notebook performs an independent samples t-test (using scipy.stats.ttest_ind) to compare the fare_amount for 'Card' and 'Cash' payment types.

2. The result will show a t-statistic and a p-value. The p-value indicates the statistical significance of the difference in fare amounts between the two payment types. If the p-value is below a chosen significance level (e.g., 0.05), it suggests a statistically significant difference.

Import Libraries:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import scipy.stats as st
import warnings
warnings.filterwarnings('ignore')
```

Load Data:

```
df = pd.read_csv("/content/2023_Yellow_Taxi_Trip_dataset.xls")
df.head(10)
```

Exploratory Data Analysis (EDA):

```
df.shape    # Get the number of rows and columns
df.dtypes    # Check the data types of columns
```

Data Type Conversion and Feature Engineering:

```
df['tpep_pickup_datetime'] = pd.to_datetime(df['tpep_pickup_datetime'],
errors='coerce')
df['tpep_dropoff_datetime'] = pd.to_datetime(df['tpep_dropoff_datetime'],
errors='coerce')
df['duration'] = df['tpep_dropoff_datetime'] - df['tpep_pickup_datetime']
df = df[['passenger_count','payment_type','fare_amount','trip_distance','duration']]
```

Handling Missing Values and Duplicates:

```
df.isnull().sum() # Count missing values per column
df.dropna(inplace=True) # Remove rows with missing values
df.drop_duplicates(inplace=True) # Remove duplicate rows
```

Data Filtering and Cleaning:

```
df = df[df['payment_type']<3]
df    = df[(df['passenger_count']>0)&(df['passenger_count']<6)]
df['payment_type'].replace([1,2],['Card','Cash'],inplace=True)
df = df[
      (df['fare_amount'] > 0) &
      (df['trip_distance'] > 0) &
      (df['duration'].dt.total_seconds() > 0)
```

```
]
for col in ['fare_amount','trip_distance','duration']:
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    IQR = q3-q1
    lower_bound = q1-1.5*IQR
    upper_bound = q3+1.5*IQR
df = df[(df[col]>=lower_bound) & (df[col]<=upper_bound)]
```

Data Visualization:

```
plt.figure(figsize=(10, 5))
plt.hist(df[df['payment_type'] == 'Card']['fare_amount'],
         histtype='barstacked', bins=20, edgecolor='k',
         color='#FA643F', label='Card')
plt.hist(df[df['payment_type'] == 'Cash']['fare_amount'],
         histtype='barstacked', bins=20, edgecolor='k',
         color='#FFBCAB', label='Cash')
plt.title('Fare Amount Distribution by Payment Type')
plt.xlabel('Fare Amount')
plt.ylabel('Frequency')
plt.legend()
plt.show()

plt.title('Preference of Payment Type')
plt.pie(df['payment_type'].value_counts(normalize=True),
        labels=df['payment_type'].value_counts().index,
        startangle=90,shadow=True,autopct='%1.1f%%',
        colors=['#FA643F', '#FFBCAB'])
plt.show()

ax = df.plot(
    x='payment_type',
    kind='barh',
    stacked=True,
    color=['#FA643F', '#FFBCAB', '#CBB2B2', '#F1F1F1', '#A3C9A8'])
for p in ax.patches:
    width = p.get_width()
    if width > 0:
        x, y = p.get_xy()
        ax.text(
            x + width / 3,
            y + p.get_height() / 3,
            '{:.0f}%'.format(width),
            ha='center',
            va='center'
        )
```

Statistical Analysis:

```
df['fare_amount'] = pd.to_numeric(df['fare_amount'], errors='coerce')
clean_fare = df['fare_amount'].dropna()

sm.qqplot(clean_fare, line='45')
plt.title('Q-Q Plot of Fare Amount')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')
plt.grid(True)
plt.show()

card_sample = df[df['payment_type']=='Card']['fare_amount']
cash_sample = df[df['payment_type']=='Cash']['fare_amount']

t_stats, p_value = st.ttest_ind(a = card_sample, b = cash_sample, equal_var = False)
print('T statistic', t_stats, 'P value', p_value)
```