In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')

## Scikit Learn

from sklearn.model_selection import train_test_split , cross_val_score , GridSearchCV
from sklearn.metrics import *
from sklearn.linear_model import LinearRegression , Ridge , Lasso
from sklearn.metrics import mean_absolute_error , mean_squared_error , r2_score
from sklearn.preprocessing import OneHotEncoder , StandardScaler
from sklearn.tree import DecisionTreeRegressor
from sklearn.feature_selection import RFE
from sklearn.preprocessing import MinMaxScaler,StandardScaler
import joblib
import datetime
from sklearn.feature_selection import SelectFromModel
```

In [2]:
```
data=pd.read_csv(r"C:\Users\kunal perane\Downloads\archive.zip")
```

In [3]:
```
data.shape
```

Out[3]:
```
(200000, 9)
```

In [4]:
```
data.head(10)
```

Out[4]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_long
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.99
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.99
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.96
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.96
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.97
5	44470845	2011-02-12 02:27:09.0000006	4.9	2011-02-12 02:27:09 UTC	-73.969019	40.755910	-73.96
6	48725865	2014-10-12 07:04:00.0000002	24.5	2014-10-12 07:04:00 UTC	-73.961447	40.693965	-73.87
7	44195482	2012-12-11 13:52:00.00000029	2.5	2012-12-11 13:52:00 UTC	0.000000	0.000000	0.00
8	15822268	2012-02-17 09:32:00.00000043	9.7	2012-02-17 09:32:00 UTC	-73.975187	40.745767	-74.00
9	50611056	2012-03-29 19:06:00.000000273	12.5	2012-03-29 19:06:00 UTC	-74.001065	40.741787	-73.96

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 9 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   Unnamed: 0         200000 non-null  int64  
 1   key                200000 non-null  object 
 2   fare_amount        200000 non-null  float64
 3   pickup_datetime    200000 non-null  object 
 4   pickup_longitude   200000 non-null  float64
 5   pickup_latitude    200000 non-null  float64
 6   dropoff_longitude  199999 non-null  float64
 7   dropoff_latitude   199999 non-null  float64
 8   passenger_count    200000 non-null  int64  
dtypes: float64(5), int64(2), object(2)
memory usage: 13.7+ MB
```

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
Unnamed: 0           0
key                  0
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    1
dropoff_latitude     1
passenger_count      0
dtype: int64
```

In [7]:
```
data.dropna(axis=0,inplace=True)
data.isnull().sum()
```

Out[7]:
```
Unnamed: 0           0
key                  0
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      0
dtype: int64
```

In [8]:
```
def haversine(lon_1,lon_2,lat_1,lat_2):
    lon_1,lon_2,lat_1,lat_2=map(np.radians,[lon_1,lon_2,lat_1,lat_2])
    diff_lon=lon_2-lon_1
    diff_lat=lat_2-lat_1
    km=2*6371*np.arcsin(np.sqrt(np.sin(diff_lat/2.0)**2+np.cos(lat_1)*np.cos(lat_2)*np.s
    return km
```

In [9]:
```
data['Distance']=haversine(data['pickup_longitude'],data['dropoff_longitude'],data['pick
                           data['dropoff_latitude'])
```

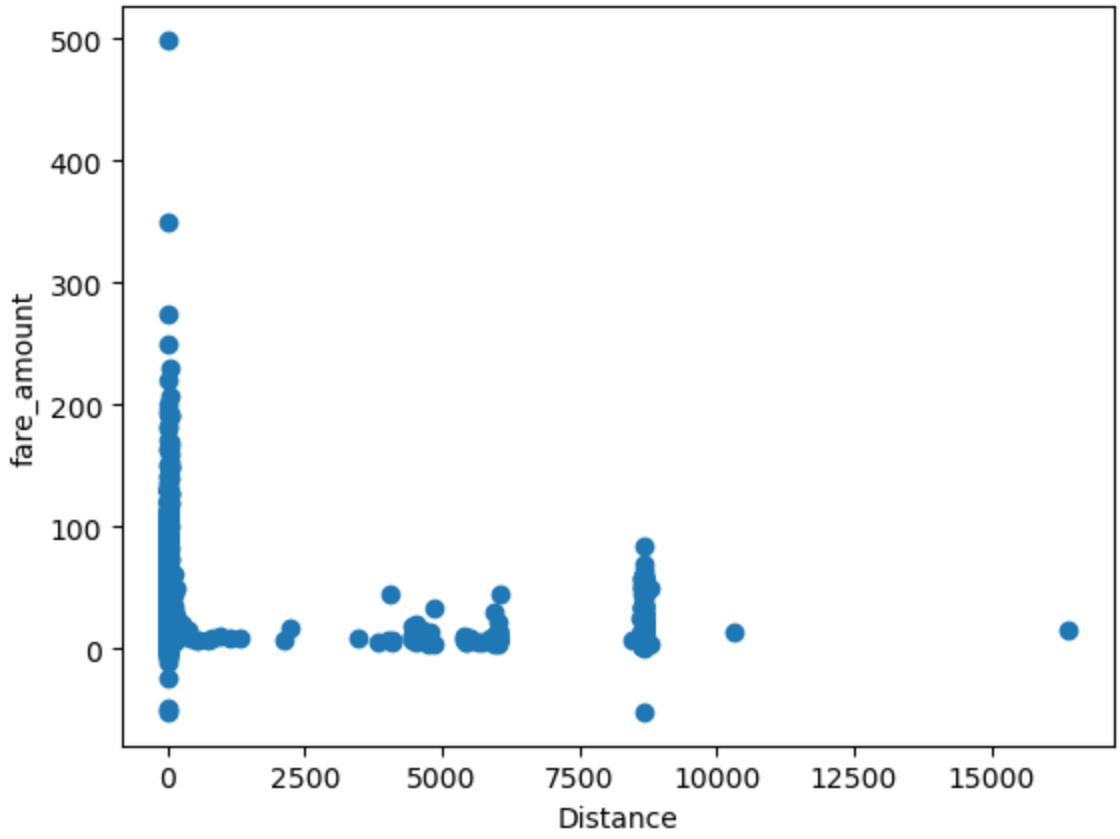
In [10]:
```
data['Distance']=data['Distance'].astype(float).round(2)
data.head()
```

Out[10]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_long
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.99
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.99
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.96
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.96
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.97

In [11]:
```
plt.scatter(data['Distance'],data['fare_amount'])
plt.xlabel('Distance')
plt.ylabel('fare_amount')
```

Out[11]:
```
Text(0, 0.5, 'fare_amount')
```



In [12]:
```
data.drop(data[data['Distance']>60].index,inplace=True)
data.drop(data[data['Distance']==0].index,inplace=True)
```

In [13]:
```
data.drop(data[data['fare_amount']==0].index,inplace=True)
data.drop(data[data['fare_amount']<0].index,inplace=True)
```

In [14]:
```
data.shape
```

Out[14]:
```
(193490, 10)
```

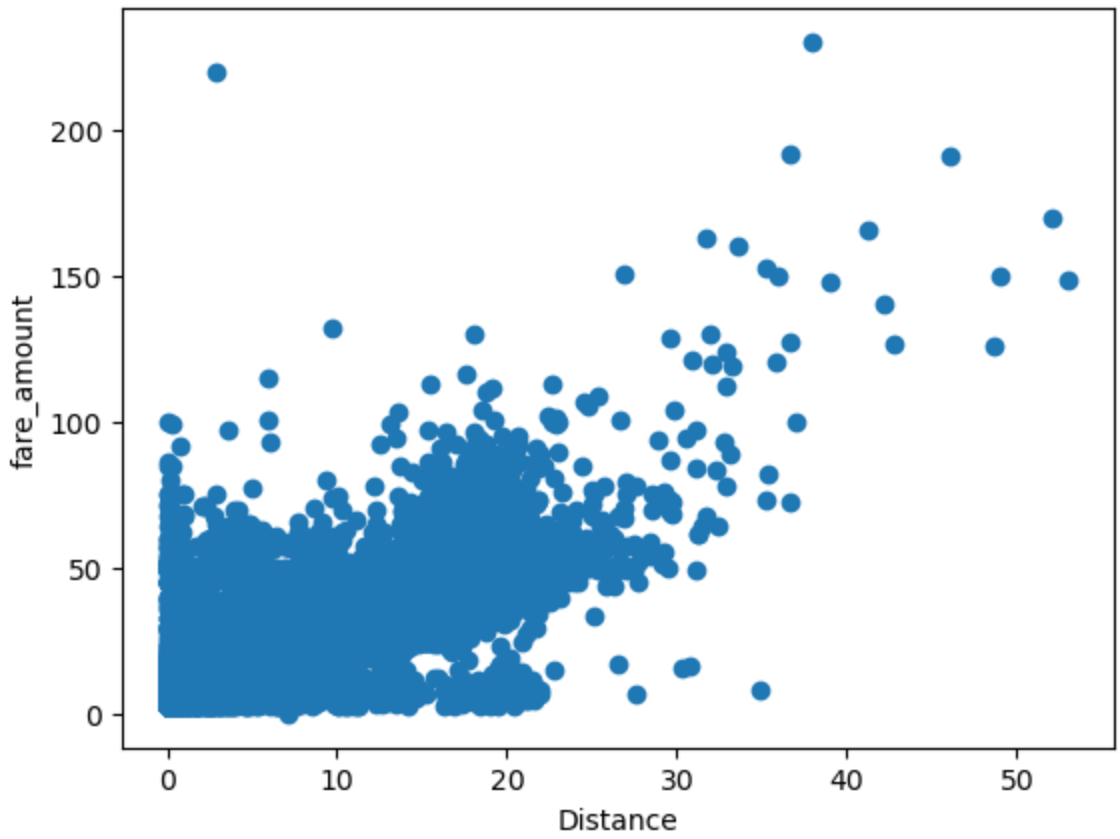
In [15]:
```
data.drop(data[(data['fare_amount']>100)&(data['Distance']<1)].index,inplace=True)
data.drop(data[(data['fare_amount']<100)&(data['Distance']>100)].index,inplace=True)
```

In [16]:
```
data.shape
```

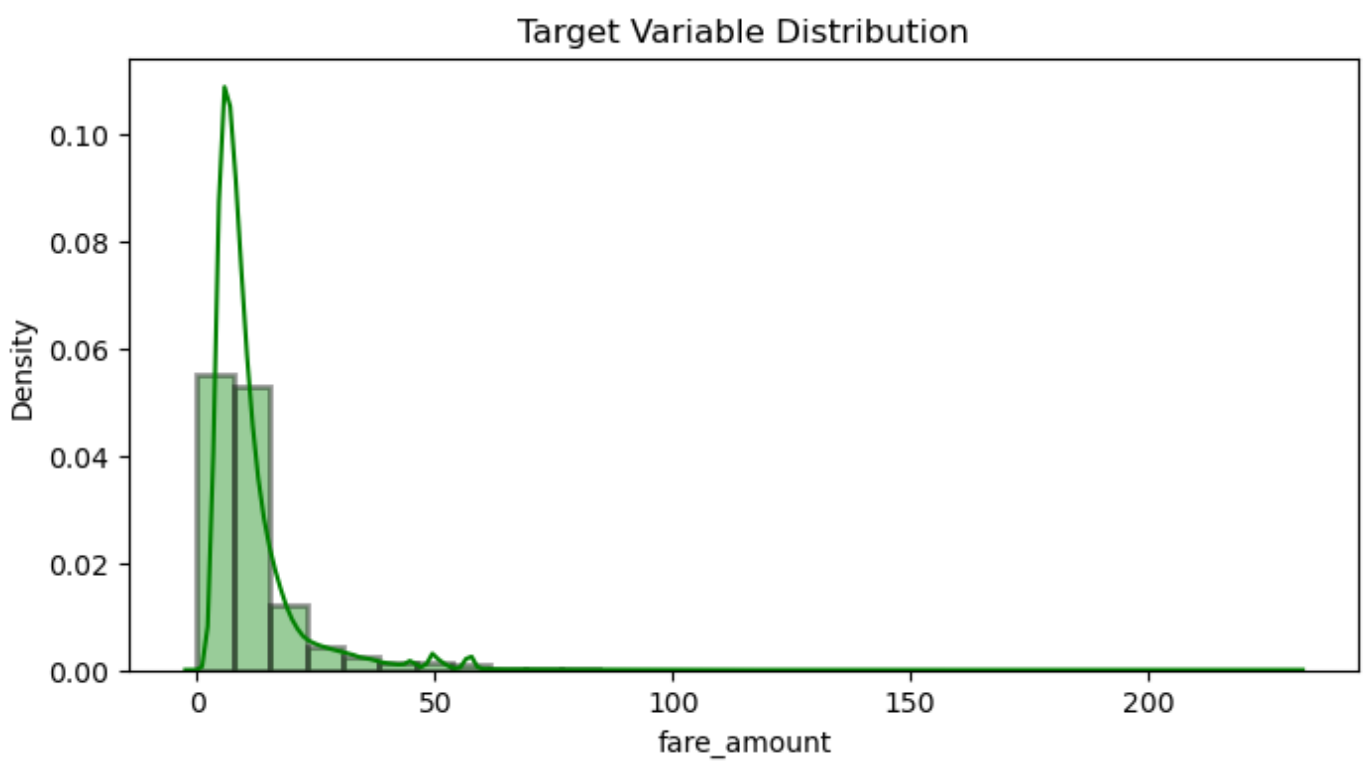
Out[16]:
```
(193481, 10)
```

In [17]:
```
plt.scatter(data['Distance'],data['fare_amount'])
plt.xlabel('Distance')
plt.ylabel('fare_amount')
```

Out[17]:
```
Text(0, 0.5, 'fare_amount')
```



In [18]:
```
plt.figure(figsize=[8,4])
sns.distplot(data['fare_amount'],color='g',hist_kws=dict(edgecolor='black',linewidth=2),
plt.title('Target Variable Distribution')
plt.show()
```



In [19]:
```
x=data['fare_amount']
y=data['Distance']
```

In [20]:
```
x=data['Distance'].values.reshape(-1,1)
y=data['fare_amount'].values.reshape(-1,1)
```

In [21]:
```
std=StandardScaler()
```

In [22]:
```
print(x)
```

```
[[ 1.68]
 [ 2.46]
 [ 5.04]
 ...
 [12.85]
 [ 3.54]
 [ 5.42]]
```

In [23]:
```
print(y)
```

```
[[ 7.5]
 [ 7.7]
 [12.9]
 ...
 [30.9]
 [14.5]
 [14.1]]
```

In [24]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state
```

In [25]:
```
print(x.shape,x_train.shape, x_test.shape)
```

```
(193481, 1) (154784, 1) (38697, 1)
```

In [26]:
```
y_std=std.fit_transform(y)
x_std=std.fit_transform(x)
```

In [27]:
```
reg=LinearRegression()
reg.fit(x_train,y_train)
```

Out[27]:
```
▼ LinearRegression
LinearRegression()
```

In [28]:
```
y_pred=reg.predict(x_test)
```

In [29]:
```
from sklearn import metrics
```

In [30]:
```
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test,y_pred))
print('Mean Squared Error:',metrics.mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error:',np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
Mean Absolute Error: 2.3163743495473246
Mean Squared Error: 18.841592163794555
Root Mean Squared Error: 4.340690286555187
```

In [31]:
```
print('R2:', r2_score(y_test, y_pred))
```

```
R2: 0.7968219124643158
```

In [32]:
```
reg= LinearRegression()
lr_scores = cross_val_score(reg, x_train , y_train , cv = 5)
```

In [33]:
```
ridge_model= Ridge(alpha=1.0)
ridge_scores = cross_val_score(ridge_model , x_train , y_train , cv = 5)
```

In [34]:
```
Lasso_model= Lasso(alpha=1.0)
Lasso_scores = cross_val_score(Lasso_model , x_train , y_train , cv = 5)
```

In [35]:
```
reg.fit(x_train , y_train)
lr_prediction = reg.predict(x_test)
lr_mae = mean_absolute_error(y_test , lr_prediction)
lr_mse = mean_squared_error(y_test ,lr_prediction)
lr_r2 = r2_score(y_test ,lr_prediction)
```

In [36]:
```
print('Linear MAE',lr_mae)
print('Linear MSE',lr_mse)
print('Linear R2',lr_r2)
```

```
Linear MAE 2.3163743495473246
Linear MSE 18.841592163794555
Linear R2 0.7968219124643158
```

In [37]:
```
Lasso_model.fit(x_train , y_train)
Lasso_prediction = Lasso_model.predict(x_test)
Lasso_mae = mean_absolute_error(y_test ,Lasso_prediction)
Lasso_mse = mean_squared_error(y_test ,Lasso_prediction)
Lasso_r2 = r2_score(y_test ,Lasso_prediction)
```

In [38]:
```
print('Lasso MAE',Lasso_mae)
print('Lasso MSE',Lasso_mse)
print('Lasso R2',Lasso_r2)
```

```
Lasso MAE 2.3342432358963325
Lasso MSE 18.956082522666897
Lasso R2 0.7955873070257342
```

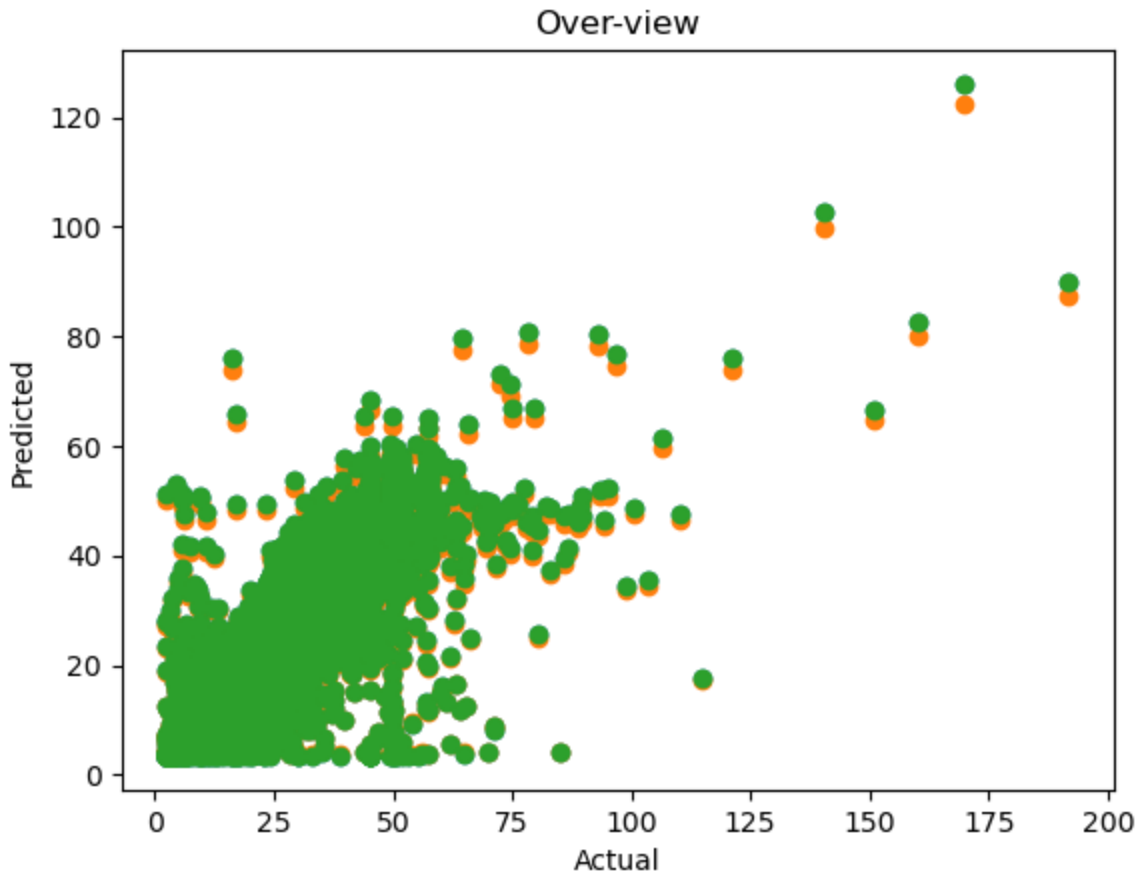
In [39]:
```
ridge_model.fit(x_train , y_train)
ridge_prediction = ridge_model.predict(x_test)
ridge_mae = mean_absolute_error(y_test ,ridge_prediction)
ridge_mse = mean_squared_error(y_test ,ridge_prediction)
ridge_r2 = r2_score(y_test ,ridge_prediction)
```

In [40]:
```
print('Lasso MAE',ridge_mae)
print('Lasso MSE',ridge_mse)
print('Lasso R2',ridge_r2)
```

```
Lasso MAE 2.3163745160114213
Lasso MSE 18.841592705604373
Lasso R2 0.7968219066217159
```

In [41]:
```
plt.scatter(y_test , lr_prediction , alpha=1.0 , label='Linear Regression')
plt.scatter(y_test , Lasso_prediction , alpha=1.0 , label='Lasso Regression')
plt.scatter(y_test , ridge_prediction , alpha=1.0 , label='ridge Regression')

plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Over-view')
plt.show()
```



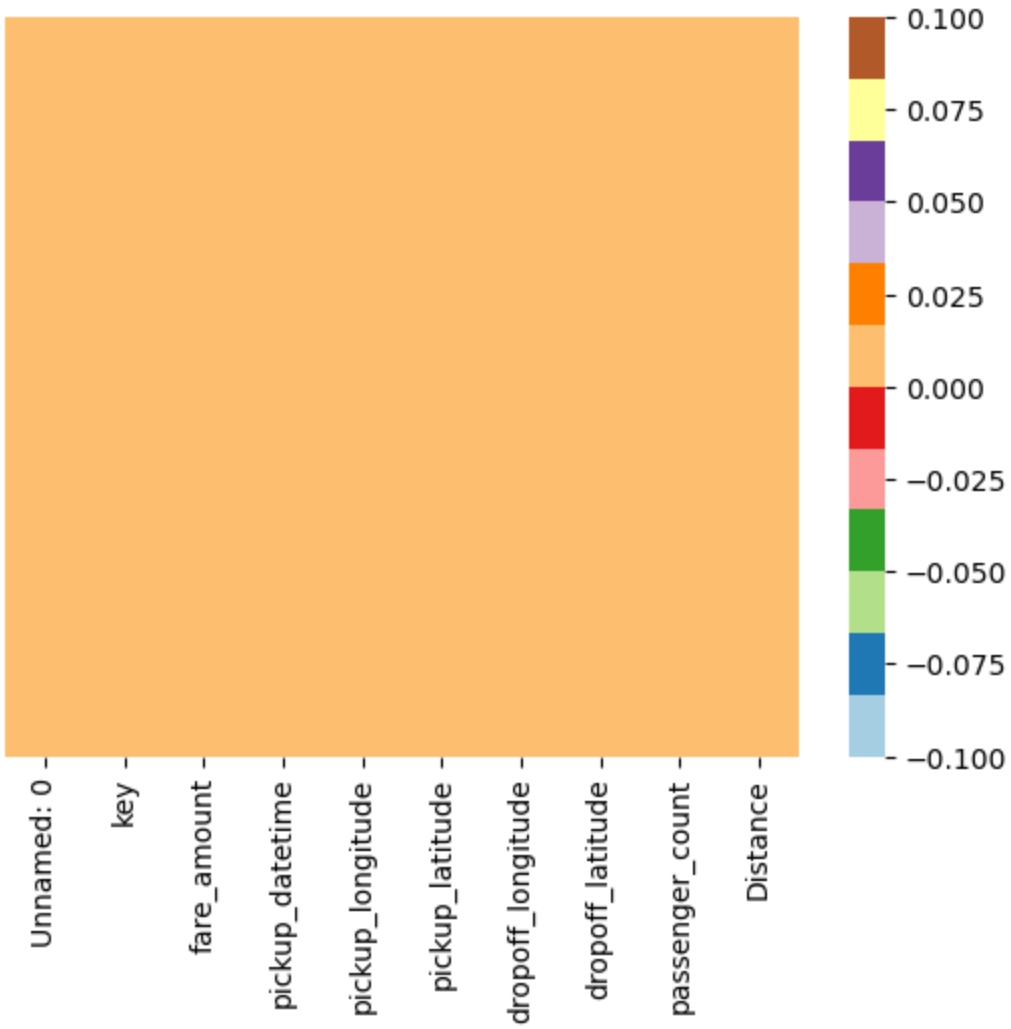
In [42]:
```
from sklearn.linear_model import HuberRegressor
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_scaled = scaler.fit_transform(x_test)
huber = HuberRegressor(epsilon=1.35)
huber.fit(x_scaled, y_test)
huber_prediction = huber.predict(x_scaled)
huber_mae =mean_absolute_error(y_test,huber_prediction)
huber_mse =mean_squared_error(y_test,huber_prediction)
huber_rmse = np.sqrt(huber_mse)
huber_r2 = r2_score(y_test,huber_prediction)
mae:',huber_mae)
```

```
print('huber mse:',huber_mse)
print('huber rmse:',huber_rmse)
print('huber r2:',huber_r2)
```

```
huber mae: 2.2482383524993073
huber mse: 19.247648299304625
huber rmse: 4.38721418434348
huber r2: 0.7924432109019499
```

In [43]:
```
sns.heatmap(data.isnull(),yticklabels=False,cmap="Paired")
```

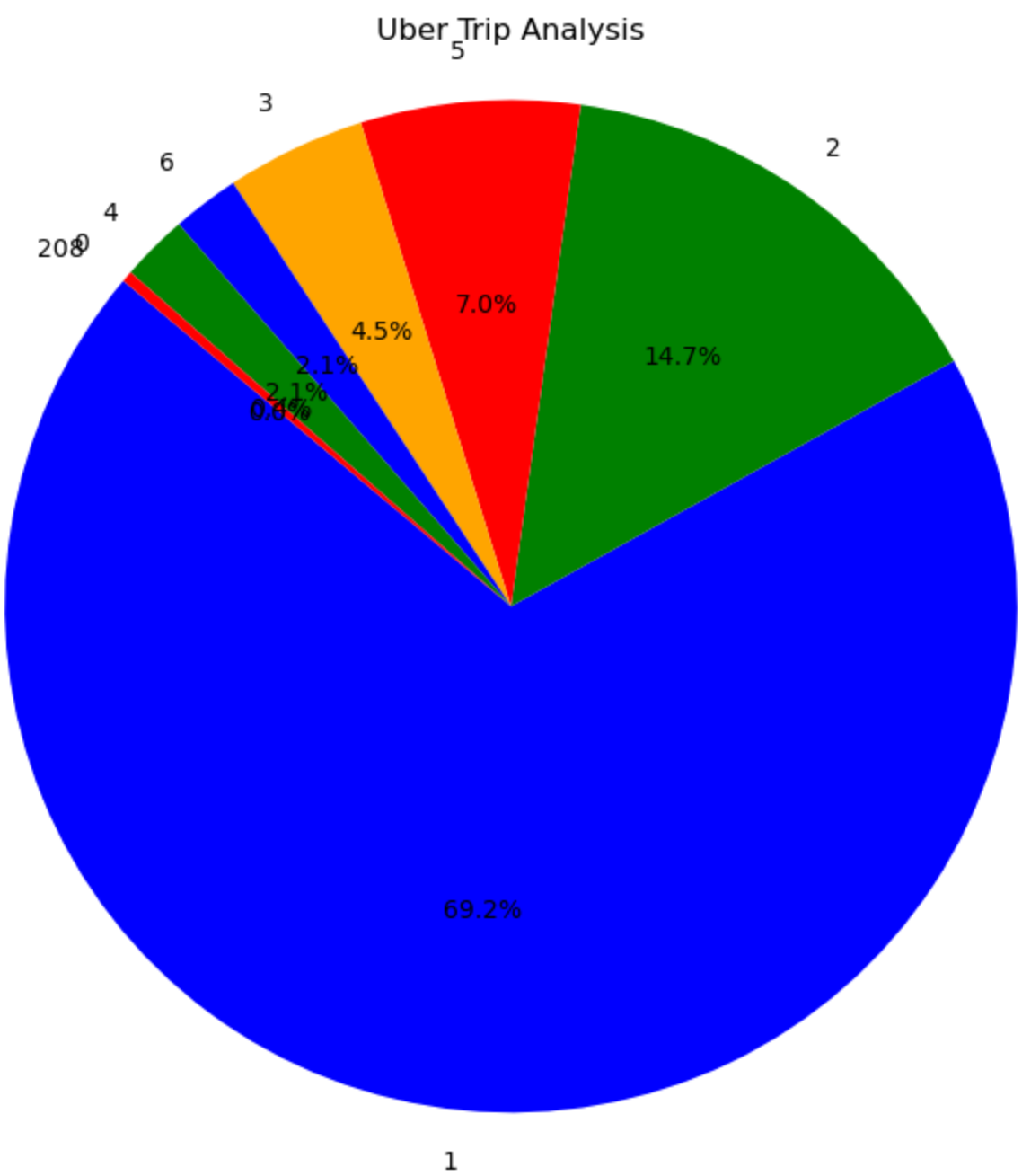
Out[43]:
```
<Axes: >
```



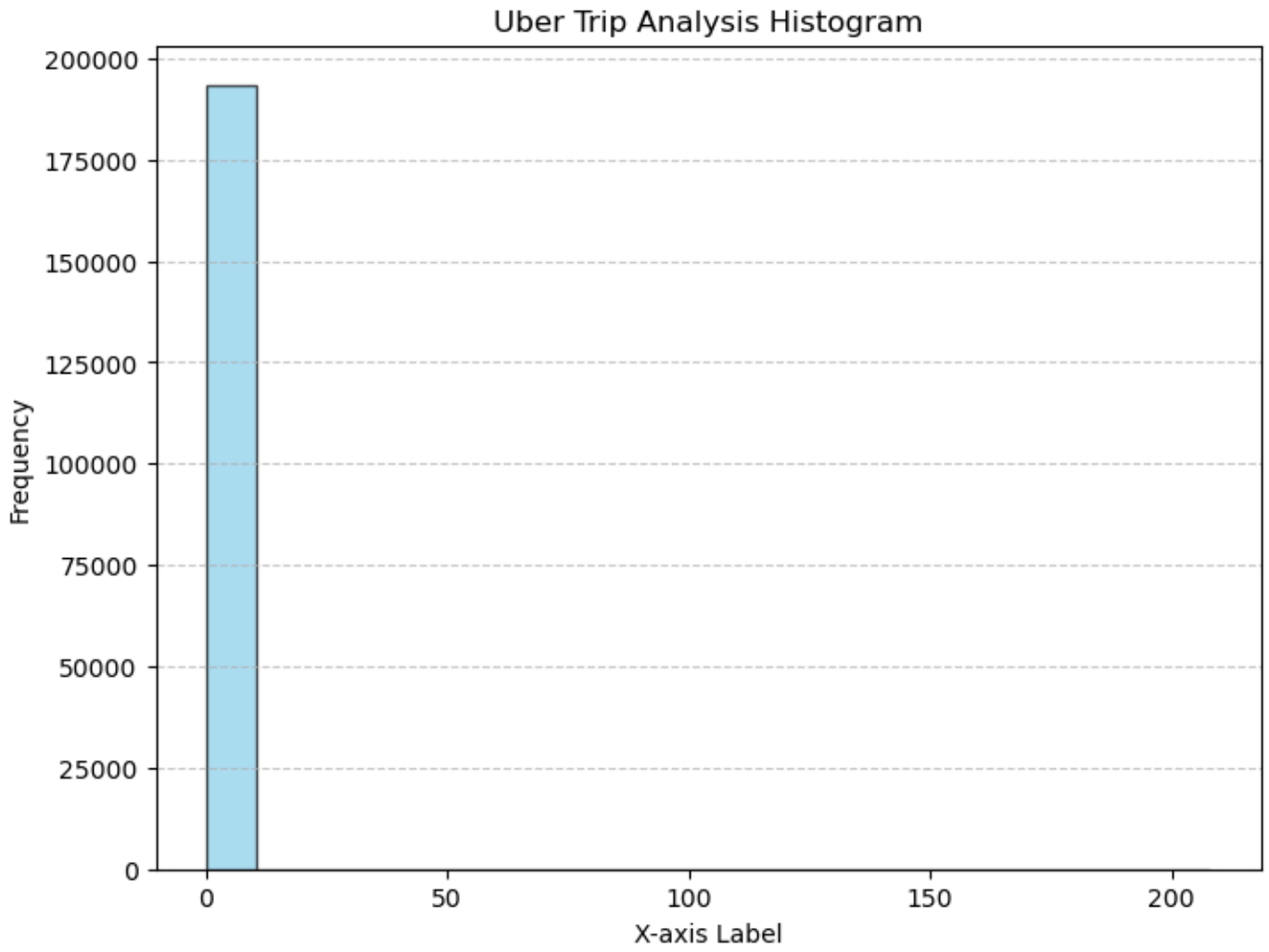
In [44]:
```
trip_counts = data['passenger_count'].value_counts()
```

In [45]:
```
labels = trip_counts.index
colors = ['blue', 'green', 'red', 'orange']
```

In [46]:
```
plt.figure(figsize=(8, 8))
plt.pie(trip_counts, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)
plt.title('Uber Trip Analysis')
plt.axis('equal')
plt.show()
```

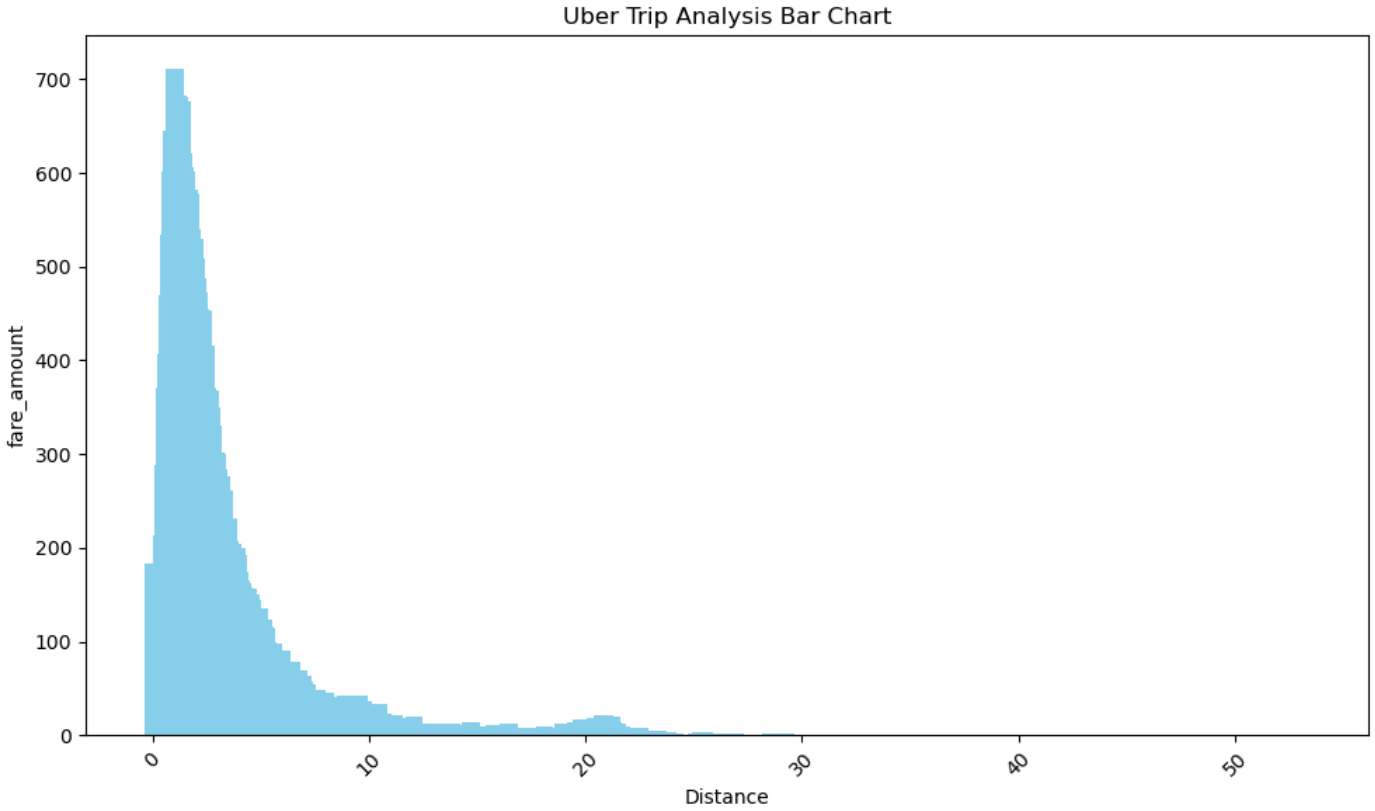


In [47]:
```
data_to_plot = data['passenger_count']
plt.figure(figsize=(8, 6))
plt.hist(data_to_plot, bins=20, color='skyblue', edgecolor='black', alpha=0.7)
plt.title('Uber Trip Analysis Histogram')
plt.xlabel('X-axis Label')
plt.ylabel('Frequency')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



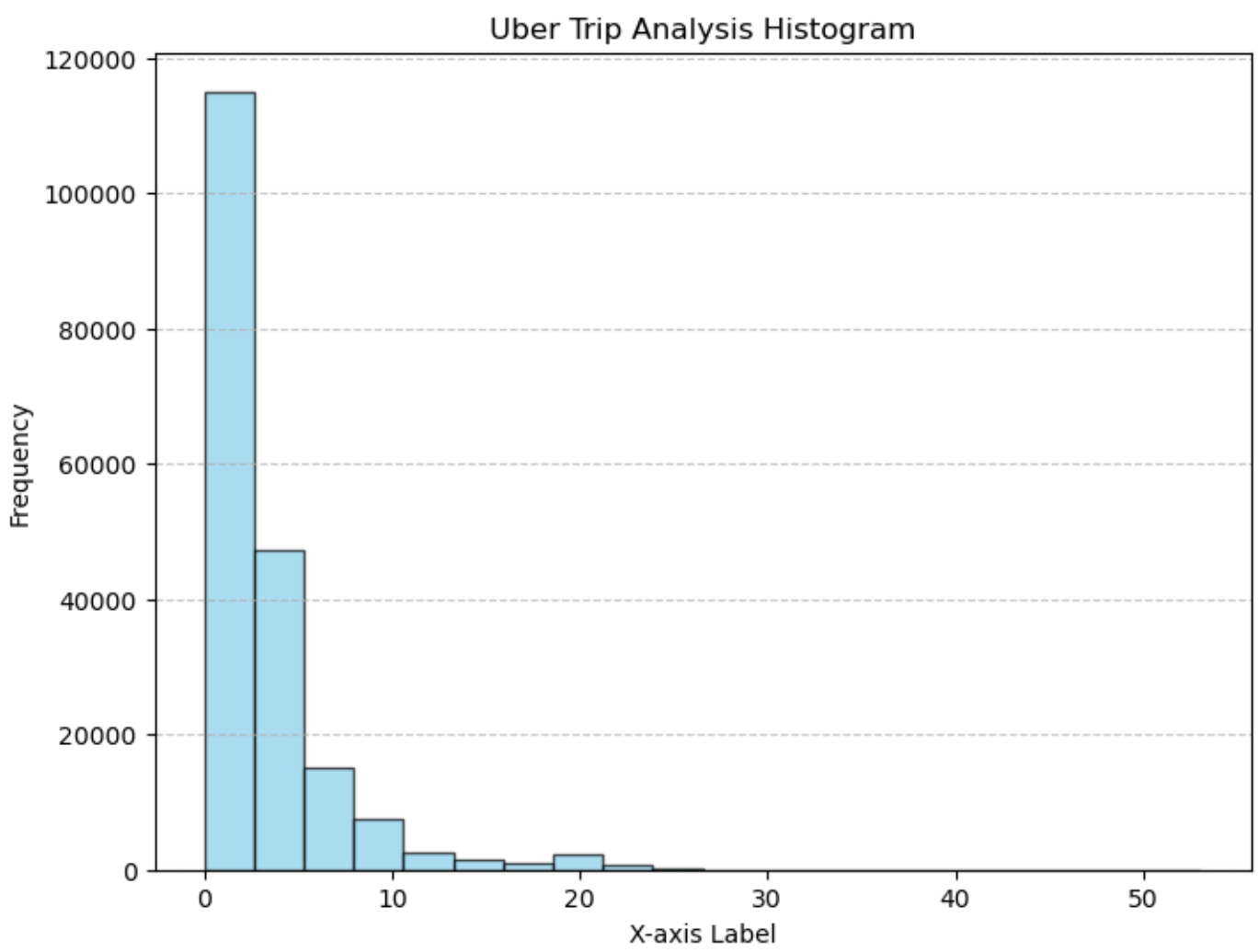
In [48]:
```
grouped_data = data.groupby('Distance')['fare_amount'].count().reset_index()
```

In [49]:
```
plt.figure(figsize=(10, 6))
plt.bar(grouped_data['Distance'], grouped_data['fare_amount'], color='skyblue')
plt.title('Uber Trip Analysis Bar Chart')
plt.xlabel('Distance')
plt.ylabel('fare_amount')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [50]:

```
data_to_plot = data['Distance']
plt.figure(figsize=(8, 6))
plt.hist(data_to_plot, bins=20, color='skyblue', edgecolor='black', alpha=0.7)
plt.title('Uber Trip Analysis Histogram')
plt.xlabel('X-axis Label')
plt.ylabel('Frequency')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



In [ ]: