CAB DRIVERS TIPS PREDICTION

DSCI 6003-01 MACHINE LEARNING FINAL PROJECT

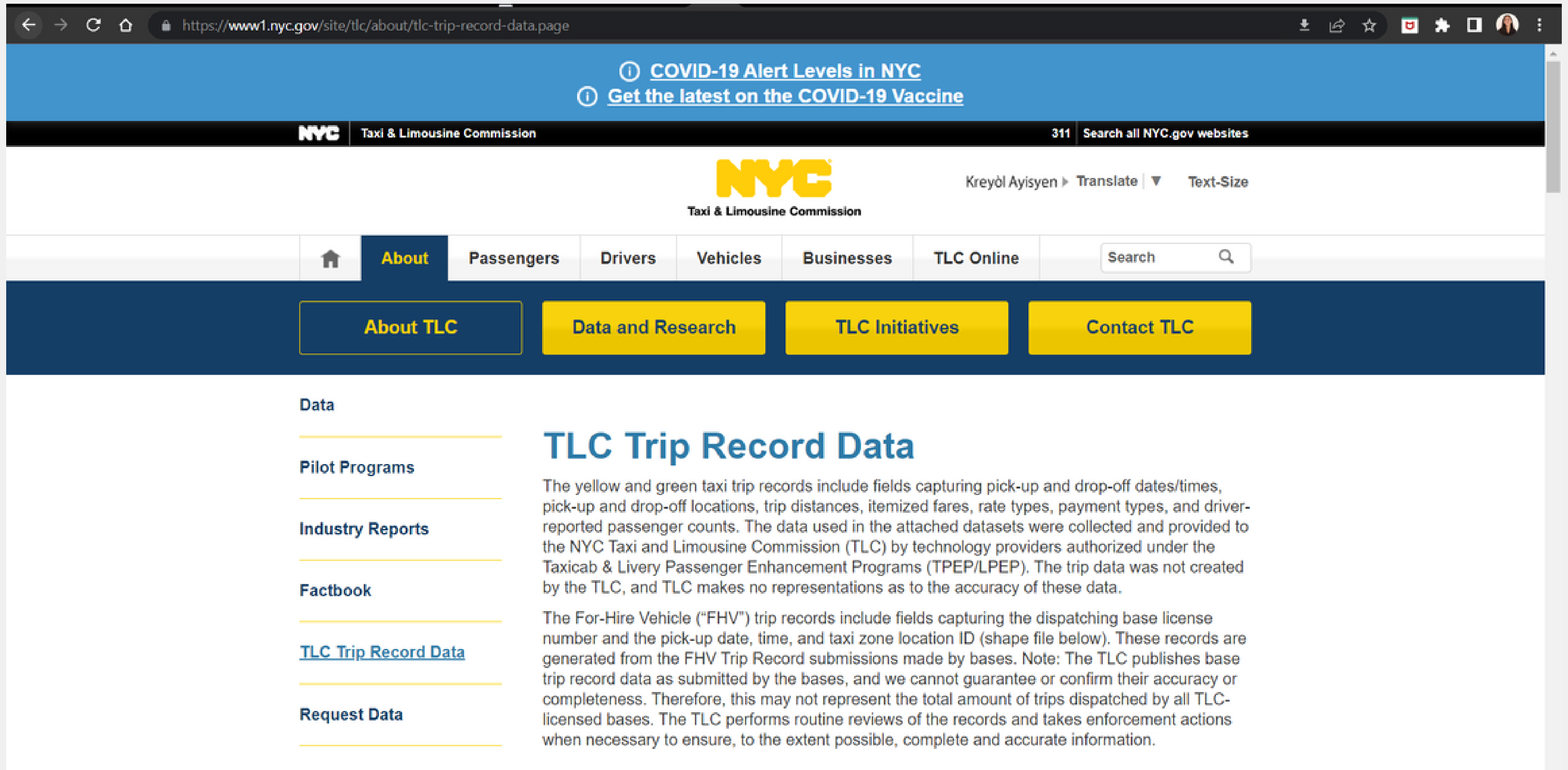
Tulika M. Kotian

GOAL

Cab drivers are often tipped after a ride apart from the base fare amount for the trip. A tip is essentially the additional fare given to a driver, post the trip. For the purpose of this project for a good data source, yellow taxis of NYC were taken as reference, and the nature of tipping, and to predict the tips based on various features were performed.

DATASET

As mentioned, the NYC Yellow Taxi dataset was used for this project. The official NYC website has yellow taxi and green taxi records for every month for the past 15 years, and just one month's dataset has about 1 million rows!



Source Link:
https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

DATASET FEATURES

The original dataset has 19 columns including the tip_amount column, and datasets of January 2021 and June 2021 were taken to create a comparative analysis between the tipping behavior during holidays and non holidays season, and during the climatic seasons(Winter, Summer)

FEATURES LIST

VendorID,
Pickup timestamp,
Dropoff timestamp,
Passenger count,
Trip distance,
Pickup location,
Dropoff location,
fare_amount,
tip_amount etc.

Out[50]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID
0	1.0	2021-01-01 00:30:10	2021-01-01 00:36:12	1.0	2.10	1.0	N	142	43
1	1.0	2021-01-01 00:51:20	2021-01-01 00:52:19	1.0	0.20	1.0	N	238	151
2	1.0	2021-01-01 00:43:30	2021-01-01 01:11:06	1.0	14.70	1.0	N	132	165
3	1.0	2021-01-01 00:15:48	2021-01-01 00:31:01	0.0	10.60	1.0	N	138	132
4	2.0	2021-01-01 00:31:49	2021-01-01 00:48:21	1.0	4.94	1.0	N	68	33

Out[50]:

DOLocationID	payment_type	fare_amount	extra	mta_tax	tip_amount	tolls_amount	improvement_surcharge	total_amount	congestion_surcharge
43	2.0	8.0	3.0	0.5	0.00	0.0	0.3	11.80	2.5
151	2.0	3.0	0.5	0.5	0.00	0.0	0.3	4.30	0.0
165	1.0	42.0	0.5	0.5	8.65	0.0	0.3	51.95	0.0
132	1.0	29.0	0.5	0.5	6.05	0.0	0.3	36.35	0.0
33	1.0	16.5	0.5	0.5	4.06	0.0	0.3	24.36	2.5

METHODOLOGY

01 Importing seasons(winter, summer) for Jan and Jul datasets

02 Calculating trip duration from pickup and dropoff times

03 Checking for holidays wrt the US Calendar

04 Narrowing down and filtering the data for more accurate predictions and lesser outliers

05 Feature selection by checking correlation between the target (tip_amount) and other features

06 Making sample dataset to be passed to the models

07 Implementing 1) Multiple Linear Regression, and 2) Gradient Boosting Regressor

08 Predictions & Performance Check

Seasons

In [2]:
```
jan = pd.read_csv('jan21_part1.csv')
jul = pd.read_csv('jul21_part1.csv')
```

```
C:\Users\tulik\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3063: DtypeWarning: Columns (6) have mixed type
s.Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [3]:
```
jan['season'] = 'Winter'
jul['season'] = 'Summer'
```

In [4]:
```
df = pd.concat([jan, jul]).reset_index(drop=True)
df.head()
```

Out[4]:

DOLocationID	payment_type	fare_amount	extra	mta_tax	tip_amount	tolls_amount	improvement_surcharge	total_amount	congestion_surcharge	season
43	2.0	8.0	3.0	0.5	0.00	0.0	0.3	11.80	2.5	Winter
151	2.0	3.0	0.5	0.5	0.00	0.0	0.3	4.30	0.0	Winter
165	1.0	42.0	0.5	0.5	8.65	0.0	0.3	51.95	0.0	Winter
132	1.0	29.0	0.5	0.5	6.05	0.0	0.3	36.35	0.0	Winter
33	1.0	16.5	0.5	0.5	4.06	0.0	0.3	24.36	2.5	Winter

Trip Duration & Holidays

In [5]:

```
df['tpep_pickup_datetime'] = pd.to_datetime(df['tpep_pickup_datetime'])
df['tpep_dropoff_datetime'] = pd.to_datetime(df['tpep_dropoff_datetime'])
df['date'] = df['tpep_pickup_datetime'].dt.normalize()
df['time'] = df['tpep_pickup_datetime'].dt.hour
df['weekday'] = df['date'].dt.day_name()
df['duration'] = df['tpep_dropoff_datetime'] - df['tpep_pickup_datetime']
df['duration'] = df['duration'] / np.timedelta64(1, 'm')
```

In [7]:

```
from pandas.tseries.holiday import USFederalHolidayCalendar as calendar
cal = calendar()
holidays = cal.holidays(start='2021-01-01', end='2021-12-31')
df['holiday'] = df['date'].isin(holidays)
df['week'] = df['date'].dt.dayofweek
df.loc[df['week'] >= 5, 'day_type'] = "weekend"
df.loc[df['week'] < 5, 'day_type'] = "workday"
df.loc[df['holiday'] == True, 'day_type'] = "holiday"
df = df.drop(['holiday', 'week'], axis=1)
```

In [51]:

```
df.head()
```

Out[51]:

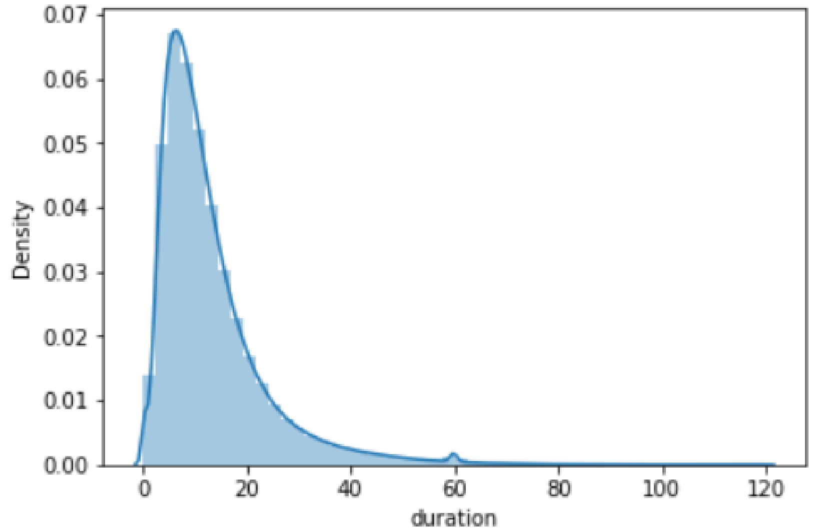
pe	fare_amount	tip_amount	tolls_amount	total_amount	congestion_surcharge	season	date	time	weekday	duration	day_type	tip_percent	time_desc
.0	8.0	0.00	0.0	11.80	2.5	Winter	2021-01-01	0	Friday	6.033333	holiday	0.000000	Late Night
.0	3.0	0.00	0.0	4.30	0.0	Winter	2021-01-01	0	Friday	0.983333	holiday	0.000000	Late Night
.0	42.0	8.65	0.0	51.95	0.0	Winter	2021-01-01	0	Friday	27.600000	holiday	16.650626	Late Night
.0	16.5	4.06	0.0	24.36	2.5	Winter	2021-01-01	0	Friday	16.533333	holiday	16.666667	Late Night
.0	8.0	2.35	0.0	14.15	2.5	Winter	2021-01-01	0	Friday	8.016667	holiday	16.607774	Late Night

Filtering dataset

In [14]:
```
sns.distplot(df[(df['duration'] > 0) & (df['duration'] < 120)]['duration'])
```

```
C:\Users\tulik\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated funct:
and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with sim:
r flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

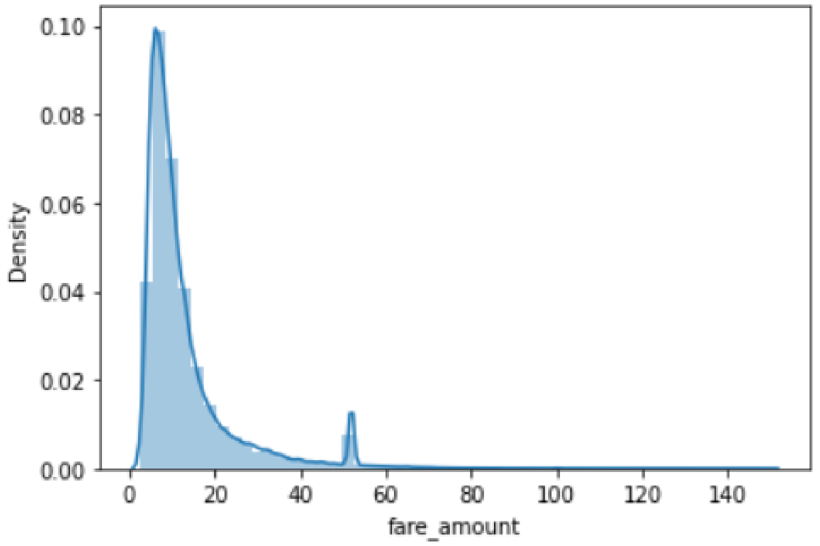
Out[14]: <AxesSubplot:xlabel='duration', ylabel='Density'>



In [15]:
```
sns.distplot(df[(df['fare_amount'] >= 2.5)&(df['fare_amount'] <= 150)]['fare_amount'])
```

```
C:\Users\tulik\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `di
and will be removed in a future version. Please adapt your code to use either `displot` (a f
r flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[15]: <AxesSubplot:xlabel='fare_amount', ylabel='Density'>

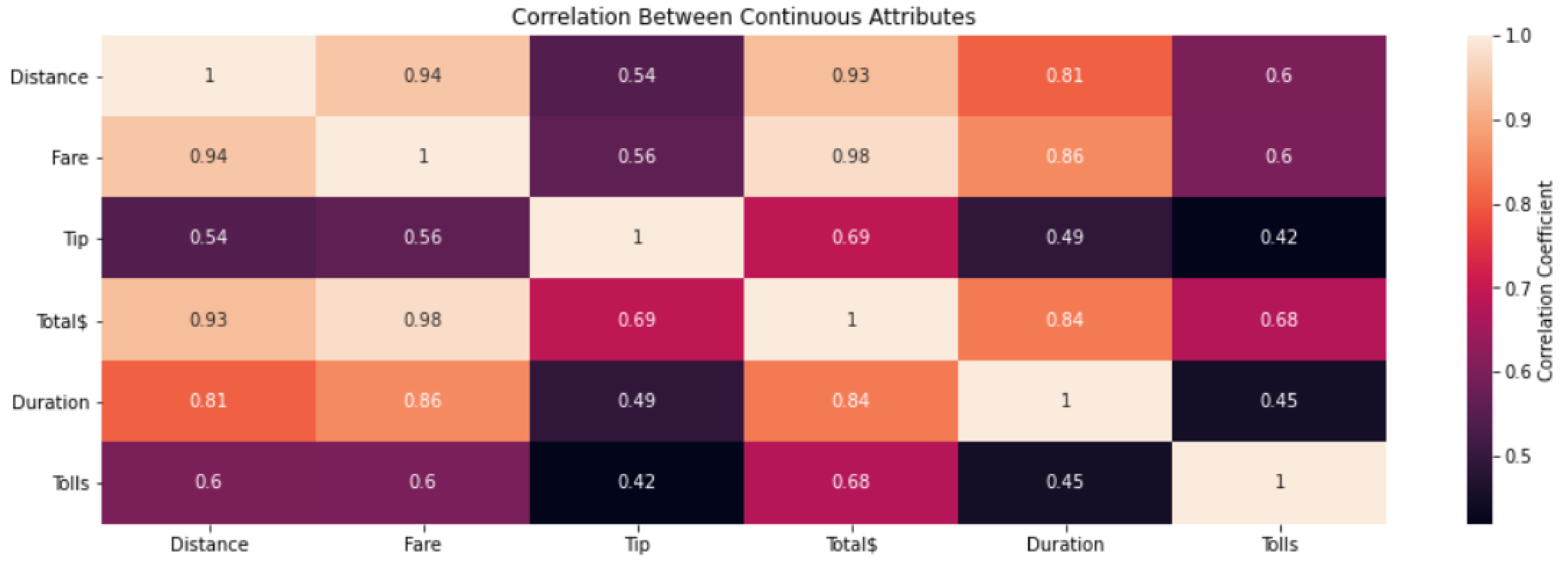


In [17]:
```
df = df[(df['passenger_count'] > 0) & (df['passenger_count'] < 7)]

# trip_distance <= 0
df = df[(df['trip_distance'] > 0) & (df['trip_distance'] <= 100)]

# exclude duration longer than 180 mins
df = df[(df['duration'] > 0) & (df['duration'] <= 180)]

# payment method other than cash and credit card
df = df[(df['payment_type'] != 3) & (df['payment_type'] != 4) & (df['payment_type'] != 5) & (df['payment_type'] != 6)]

# exclude instances with fare amount less than 2.5
df = df[(df['fare_amount'] >= 2.5) & (df['fare_amount'] <= 250)]

# remove trips with tip precentage over 50%
df = df[(df['tip_percent'] >= 0) & (df['tip_percent'] <= 50)]
```

Correlation

In [23]:

```python
col = ['trip_distance', 'fare_amount', 'tip_amount', 'total_amount', 'duration', 'tolls_amount']
label = ['Distance', 'Fare', 'Tip', 'Total$', 'Duration', 'Tolls']
import matplotlib.pyplot as plt
# plot correlation plot
fig, ax = plt.subplots(figsize=(16, 5))
corr_matrix = df[col].corr()
sns.heatmap(corr_matrix, annot=True, ax=ax, cbar_kws={'label': 'Correlation Coefficient'})
ax.set_xticklabels(label)
ax.set_yticklabels(label)
ax.set_title("Correlation Between Continuous Attributes")
plt.show()
```



Final Dataset

In [24]:
```python
sample = df[df['payment_type'] == 1].sample(frac=0.05, replace=True, random_state=30034).reset_index(drop=True)
```

In [25]:
```python
COL = ['passenger_count', 'fare_amount', 'tip_amount', 'tolls_amount', 'season', 'day_type', 'time_desc']
sample_filtered = sample.loc[:, COL].reset_index(drop=True)
```

In [26]:
```python
y = sample_filtered['tip_amount']
X_train, X_test, y_train, y_test = train_test_split(sample_filtered, y, test_size=0.3, random_state=0)
```

In [27]:
```python
xCOLS = ['fare_amount', 'tolls_amount']

scaler = StandardScaler()
X_train[xCOLS] = scaler.fit_transform(X_train[xCOLS])
X_test[xCOLS] = scaler.transform(X_test[xCOLS])
```

In [28]:
```python
baseline = ols(formula='tip_amount ~ 1', data=X_train).fit()
print(baseline.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             tip_amount   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                  0.000
Method:                 Least Squares   F-statistic:                       nan
Date:                Wed, 27 Apr 2022   Prob (F-statistic):                nan
Time:                        14:59:30   Log-Likelihood:            -2.2969e+05
No. Observations:               99917   AIC:                         4.594e+05
Df Residuals:                   99916   BIC:                         4.594e+05
Df Model:                           0
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      2.9379      0.008    385.248      0.000       2.923       2.953
==============================================================================
Omnibus:                    77901.861   Durbin-Watson:                   2.010
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          2495000.717
Skew:                           3.512   Prob(JB):                         0.00
Kurtosis:                      26.451   Cond. No.                         1.00
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Multiple Linear Regression

In [29]:

```python
base_train_pred = baseline.predict(X_train)
base_test_pred = baseline.predict(X_test)
train_rmse = mean_squared_error(y_train, base_train_pred, squared=False)
test_rmse = mean_squared_error(y_test, base_test_pred, squared=False)

print("Train RMSE:", train_rmse)
print("Test RMSE:", test_rmse)
```

```
Train RMSE: 2.4105607505095015
Test RMSE: 2.4214646830412057
```

In [31]:

```python
full_model = ols(formula='tip_amount ~ C(passenger_count) + fare_amount + tolls_amount + season + time_desc + day_type',
print(full_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             tip_amount   R-squared:                       0.569
Model:                            OLS   Adj. R-squared:                  0.569
Method:                 Least Squares   F-statistic:                 1.318e+04
Date:                Wed, 27 Apr 2022   Prob (F-statistic):               0.00
Time:                        14:59:52   Log-Likelihood:            -1.8767e+05
No. Observations:               99917   AIC:                         3.754e+05
Df Residuals:                   99906   BIC:                         3.755e+05
Df Model:                          10
Covariance Type:            nonrobust
===================================================================================================
                                      coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------------------
Intercept                           2.8665      0.038     75.140      0.000       2.792       2.941
C(passenger_count)[T.medium]       -0.0202      0.032     -0.635      0.526      -0.082       0.042
C(passenger_count)[T.small]        -0.0120      0.024     -0.495      0.621      -0.060       0.036
season[T.Winter]                   -0.0777      0.011     -7.074      0.000      -0.099      -0.056
time_desc[T.Evening]                0.0439      0.013      3.483      0.000       0.019       0.069
time_desc[T.Late Night]            -0.0129      0.017     -0.762      0.446      -0.046       0.020
time_desc[T.Morning]               -0.0630      0.014     -4.664      0.000      -0.090      -0.037
day_type[T.weekend]                 0.0830      0.031      2.692      0.007       0.023       0.143
day_type[T.workday]                 0.1260      0.030      4.222      0.000       0.068       0.185
fare_amount                         1.5771      0.006    247.685      0.000       1.565       1.590
tolls_amount                        0.3517      0.006     55.440      0.000       0.339       0.364
==============================================================================
Omnibus:                    42180.873   Durbin-Watson:                   2.011
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          9270686.228
Skew:                          -0.878   Prob(JB):                         0.00
Kurtosis:                      50.156   Cond. No.                         18.1
==============================================================================
```

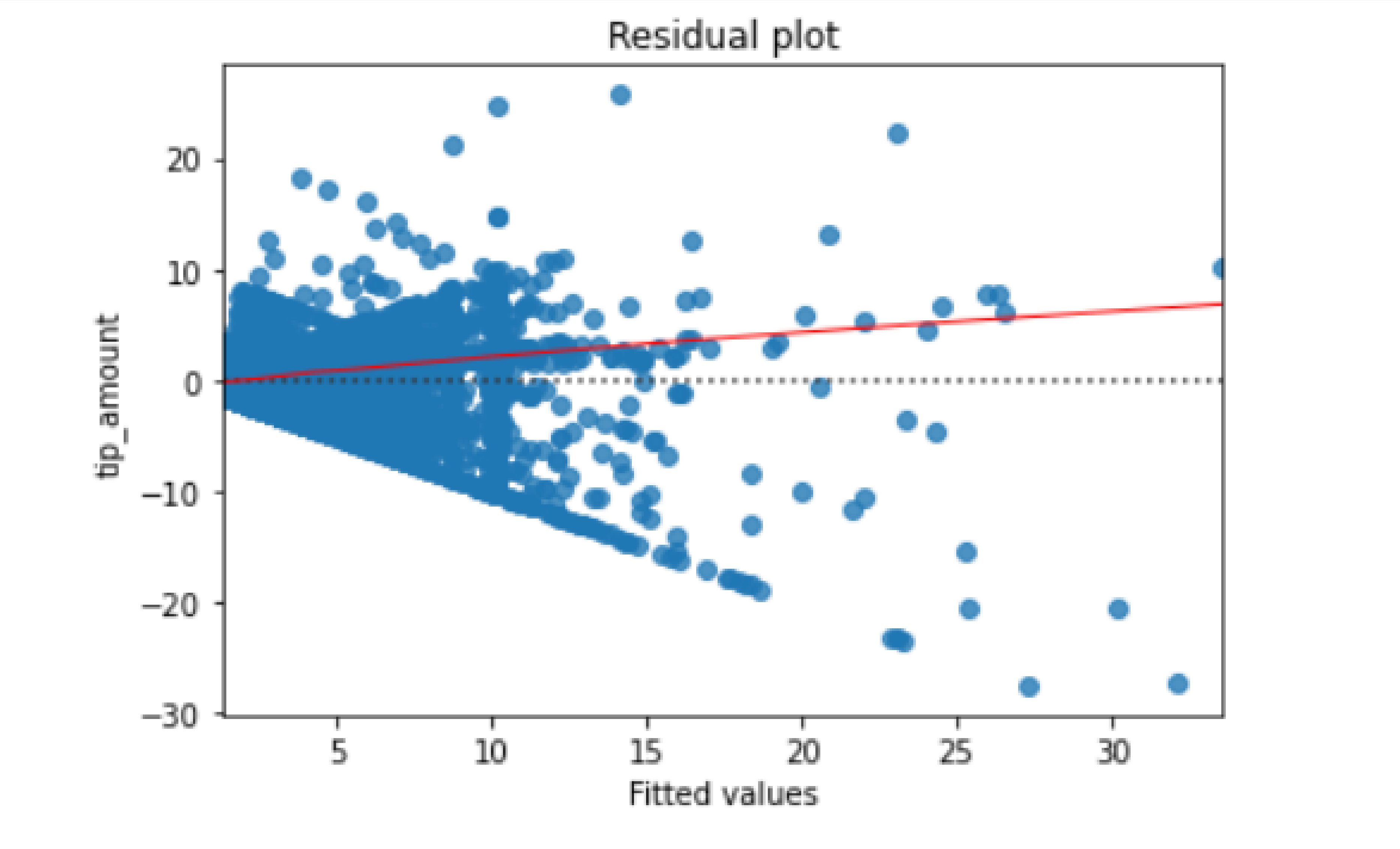
Gradient Boosting Regressor

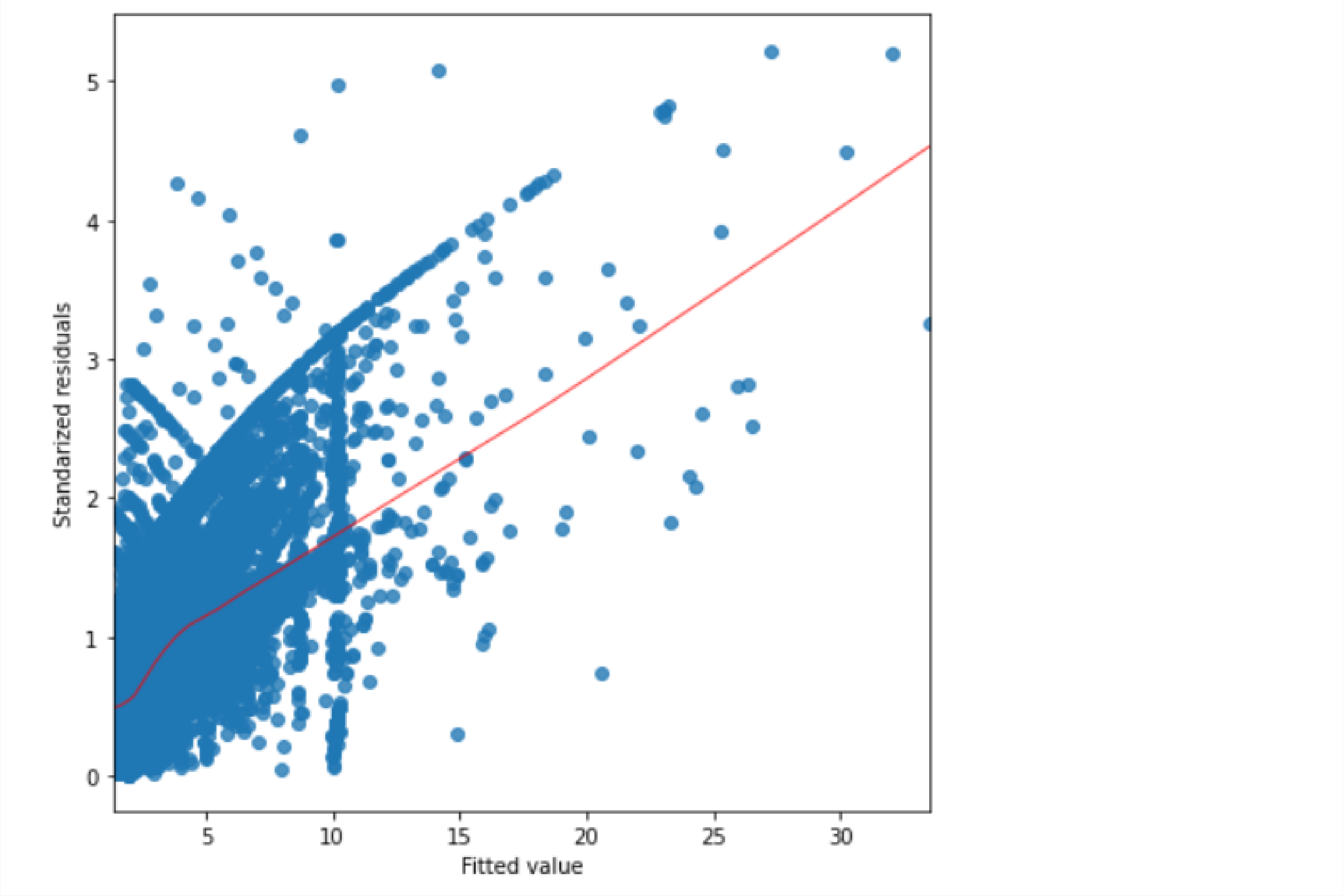
In [38]:
```
reg = GradientBoostingRegressor(random_state=0)
reg.fit(X_train, y_train)
```
Out[38]:
```
GradientBoostingRegressor(random_state=0)
```

In [39]:
```
reg.feature_importances_
```
Out[39]:
```
array([9.57566027e-01, 3.24405065e-02, 7.10823171e-04, 1.13328955e-03,
       4.63688705e-03, 4.46109301e-04, 6.31666959e-04, 1.01293270e-03,
       6.80036651e-04, 7.41721197e-04])
```

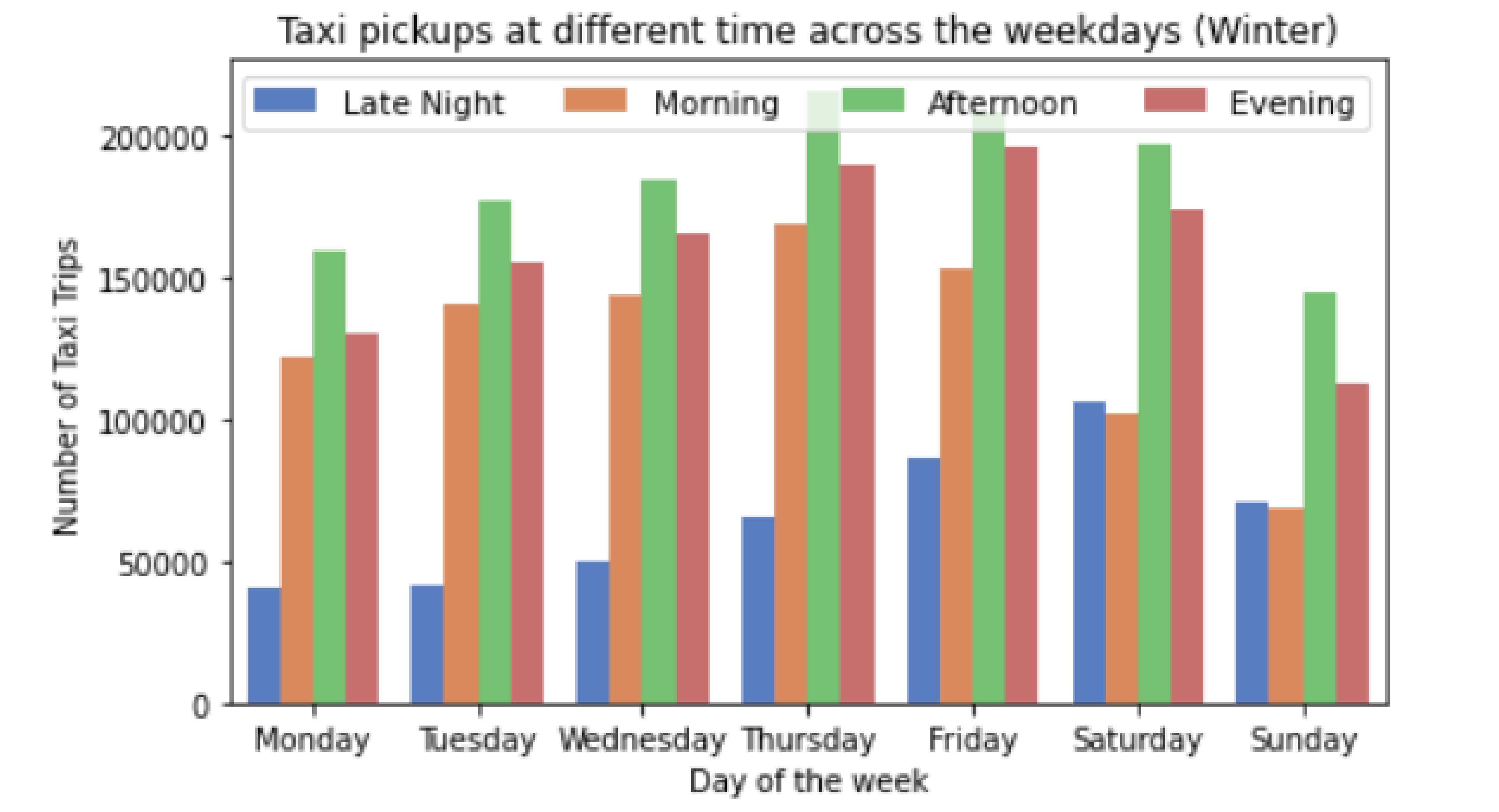
In [40]:
```
gbr_train_pred = reg.predict(X_train)
gbr_test_pred = reg.predict(X_test)
train_rmse = mean_squared_error(y_train, gbr_train_pred, squared=False)
test_rmse = mean_squared_error(y_test, gbr_test_pred, squared=False)
train_r2 = r2_score(y_train, gbr_train_pred)
test_r2 = r2_score(y_test, gbr_test_pred)

print("Gradient Boost Regression")
print("Train RMSE:", train_rmse)
print("Test RMSE:", test_rmse)
print("Train R2", train_r2)
print("Test R2", test_r2)
```

```
Gradient Boost Regression
Train RMSE: 1.4955406565471216
Test RMSE: 1.5333279112993963
Train R2 0.6150890325112002
Test R2 0.5990238410658886
```

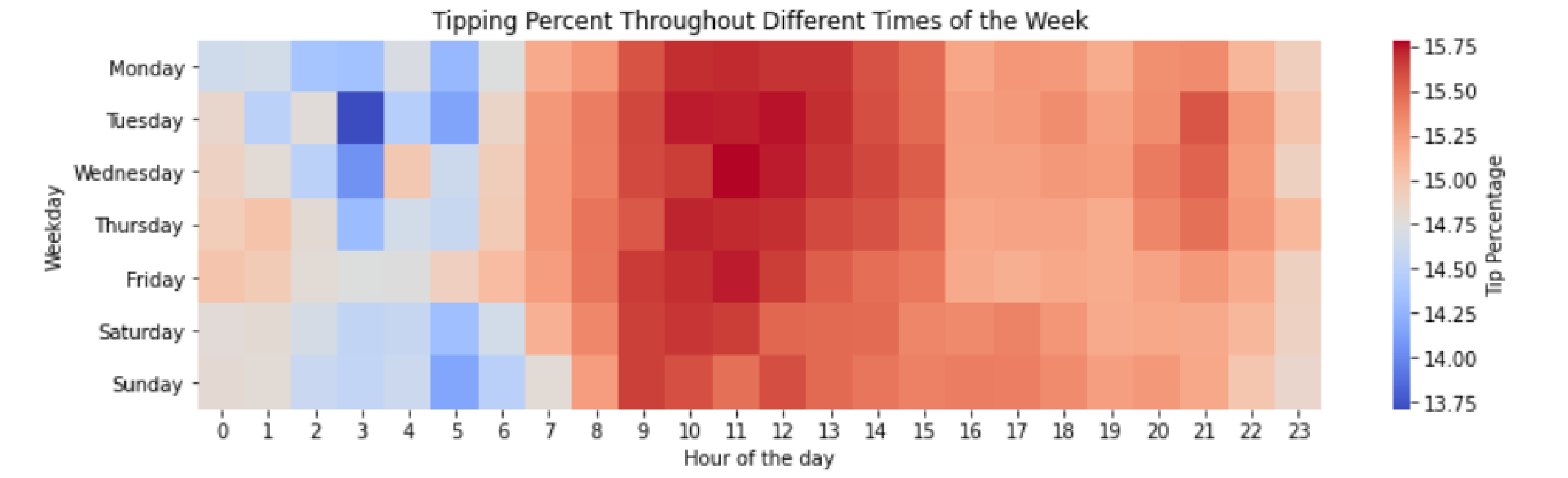

Residual plot
tip_amount
Fitted values
20
10
0
−10
−20
−30
5
10
15
20
25
30


Standarized residuals
Fitted value
0
1
2
3
4
5
5
10
15
20
25
30

Analysis



Taxi pickups at different time across the weekdays (Winter)
Late Night
Morning
Afternoon
Evening
Number of Taxi Trips
0
50000
100000
150000
200000
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Day of the week



Tipping Percent Throughout Different Times of the Week
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Weekday
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour of the day
15.75
15.50
15.25
15.00
14.75
14.50
14.25
14.00
13.75
Tip Percentage

THANK YOU