ML Project Assignment

Guide: Dr. Patrick Healy

CS6502 - Applied Big Data and Visualization

10th May 2020

Submitted by

Jamiu Olashile Salimon

19063679

Table of Contents

Task 1: Select the First 1000 Rows 3
Link to Query on Big Query 3
Actual Query 3
Task 2: Principal Component Analysis (PCA) 4
Summary 4
Exploratory Data Analysis 4
Check for Missing/Null Values 4
Stats 5
Numerical Features 6
Categorical Features 8
Data Preparation 9
Handle Missing Values 9
Encode Categorical Features 9
Scale Numerical Features 9
PCA 11
Task 3: Model Creation 14
Link to Query on Big Query 14
Actual Query 14
Task 4: Model Evaluation 16
Link to Query on Big Query 16
Actual Query 16
Task 5: Model Prediction 18
Link to Query on Big Query 18
Actual Query 18
Task 6: Screenshot of Model Evaluation Report 20
Result from Model Creation Query 20
Result from Model Evaluation Query 20

Task 1: Select the First 1000 Rows

Link to Query on Big Query

https://console.cloud.google.com/bigquery?sq=844523097149:ea0c2dd8f55241f68da15b13ac544ef7

Actual Query

```
select
  extract(dayofweek from trip_start_timestamp) as
trip_start_time_day_of_week,
  extract(hour from trip_start_timestamp) as trip_start_time_hour,
  extract(dayofweek from trip_end_timestamp) as
trip_end_time_day_of_week,
  extract(hour from trip_end_timestamp) as trip_end_time_hour,
  trip_seconds,
  trip_miles,
  pickup_census_tract,
  dropoff_census_tract,
  pickup_community_area,
  dropoff_community_area,
  fare,
  tips,
  tolls,
  extras,
  trip_total,
  payment_type,
  company,
  pickup_latitude,
  pickup_longitude,
  pickup_location,
  dropoff_latitude,
  dropoff_longitude,
  dropoff_location
from
    `bigquery-public-data.chicago_taxi_trips.taxi_trips`
order by
  trip_start_timestamp
asc limit
  1000;
```

Task 2: Principal Component Analysis (PCA)

Summary

Exploratory Data Analysis (EDA) was first carried out on the dataset for the sole purpose of understanding and summarizing it. Secondly, the dataset was **prepared** for modelling. Finally, **Principal Component Analysis (PCA)** was carried out on the prepared dataset.

The work done is described in detail in the next three headings.

You can also find the Jupyter notebook of work done named: “ml-project-pca.ipynb”. It located in the same folder as this document.

You can also find and execute a copy of the notebook on Google Colab via the link below:
https://colab.research.google.com/drive/1VL68e7kdgjMI9VSomWICAkw0Fy7wnKu0?usp=sharing

Exploratory Data Analysis

Check for Missing/Null Values

Rows with missing values include; trip_seconds: 6, pickup_census_tract: 352, dropoff_census_tract: 408, pickup_community_area: 73, dropoff_community_area: 145, company: 635, pickup_latitude: 73, pickup_longitude: 73, pickup_location: 73, dropoff_latitude: 146, dropoff_longitude: 146 and dropoff_location: 146.

```
[59] # displaying data details & checking for nan/null/inf values
     chicago_taxi_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 23 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   trip_start_time_day_of_week  1000 non-null   int64  
 1   trip_start_time_hour       1000 non-null   int64  
 2   trip_end_time_day_of_week  1000 non-null   int64  
 3   trip_end_time_hour         1000 non-null   int64  
 4   trip_seconds               994 non-null    float64
 5   trip_miles                 1000 non-null   float64
 6   pickup_census_tract        648 non-null    float64
 7   dropoff_census_tract       592 non-null    float64
 8   pickup_community_area      927 non-null    float64
 9   dropoff_community_area     854 non-null    float64
 10  fare                       1000 non-null   float64
 11  tips                       1000 non-null   float64
 12  tolls                      1000 non-null   float64
 13  extras                     1000 non-null   float64
 14  trip_total                 1000 non-null   float64
 15  payment_type               1000 non-null   object 
 16  company                    365 non-null    object 
 17  pickup_latitude            927 non-null    float64
 18  pickup_longitude           927 non-null    float64
 19  pickup_location            927 non-null    object 
 20  dropoff_latitude           854 non-null    float64
 21  dropoff_longitude          854 non-null    float64
 22  dropoff_location           854 non-null    object 
dtypes: float64(15), int64(4), object(4)
memory usage: 179.8+ KB
```

Stats

- payment_type, company, pickup_location, and dropoff_location are categorical features.
- Noted distribution are as follows;
  - Normal distribution: trip_start_time_day_of_week, trip_end_time_day_of_week, pickup_census_tract
  - Left skewed: trip_end_time_day_of_week, trip_end_time_hour, extras, pickup_latitude, dropoff_latitude
  - Right skewed: trip_seconds, trip_miles, dropoff_census_tract, pickup_community_area, fare, tips, tolls, trip_total, pickup_longitude, and dropoff_longitude

```
[62] # displaying data stats for numerical features
chicago_taxi_dataset.describe()
```

	trip_start_time_day_of_week	trip_start_time_hour	trip_end_time_day_of_week	trip_end_time_hour	trip_seconds	trip_miles	pickup_census_tract	dropoff_census_tract	pickup_community_area
count	1000.0	1000.0	1000.0	1000.000000	994.000000	1000.000000	6.480000e+02	5.920000e+02	927.000000
mean	3.0	0.0	3.0	0.007000	569.577465	2.706130	1.703122e+10	1.703123e+10	14.450917
std	0.0	0.0	0.0	0.094657	499.971422	6.511342	2.662770e+05	2.660685e+05	13.379230
min	3.0	0.0	3.0	0.000000	0.000000	0.000000	1.703103e+10	1.703103e+10	1.000000
25%	3.0	0.0	3.0	0.000000	300.000000	0.000000	1.703107e+10	1.703107e+10	7.000000
50%	3.0	0.0	3.0	0.000000	480.000000	1.140000	1.703108e+10	1.703108e+10	8.000000
75%	3.0	0.0	3.0	0.000000	720.000000	3.000000	1.703128e+10	1.703132e+10	24.000000
max	3.0	0.0	3.0	2.000000	6600.000000	90.000000	1.703198e+10	1.703184e+10	77.000000

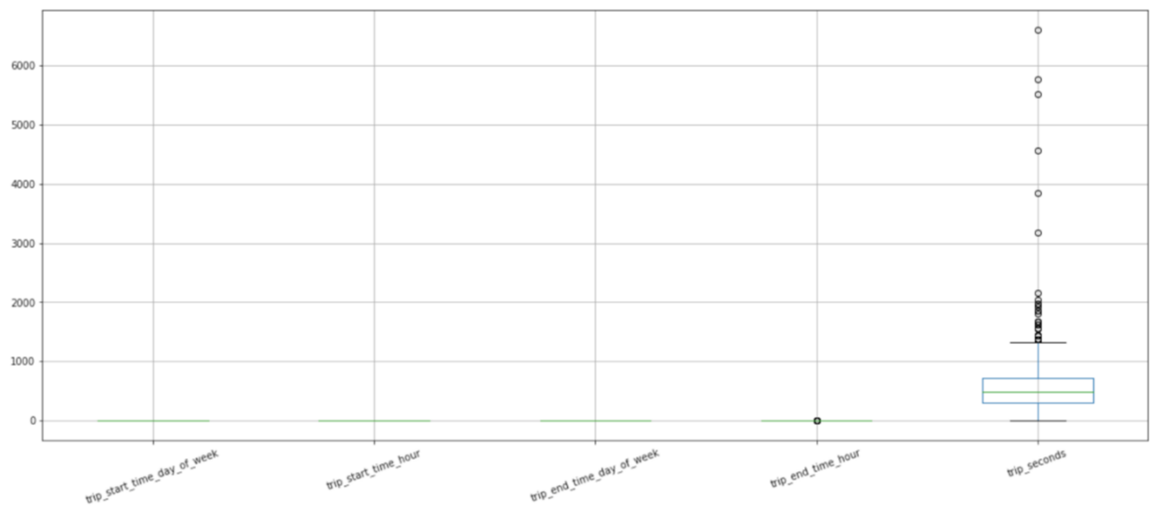
```
[63] # calc variance for each numerical variable
chicago_taxi_dataset.var()
```

```
trip_start_time_day_of_week     0.000000e+00
trip_start_time_hour            0.000000e+00
trip_end_time_day_of_week       0.000000e+00
trip_end_time_hour              8.959960e-03
trip_seconds                    2.499714e+05
trip_miles                      4.239758e+01
pickup_census_tract             7.090345e+10
dropoff_census_tract            7.079247e+10
pickup_community_area           1.790038e+02
dropoff_community_area          2.455722e+02
fare                            8.090699e+04
tips                            5.035703e+00
tolls                           2.500000e+00
extras                          1.510010e+01
trip_total                      8.185993e+04
pickup_latitude                 9.588334e-04
pickup_longitude                1.013319e-03
dropoff_latitude                1.247908e-03
dropoff_longitude               8.895545e-04
dtype: float64
```

Numerical Features

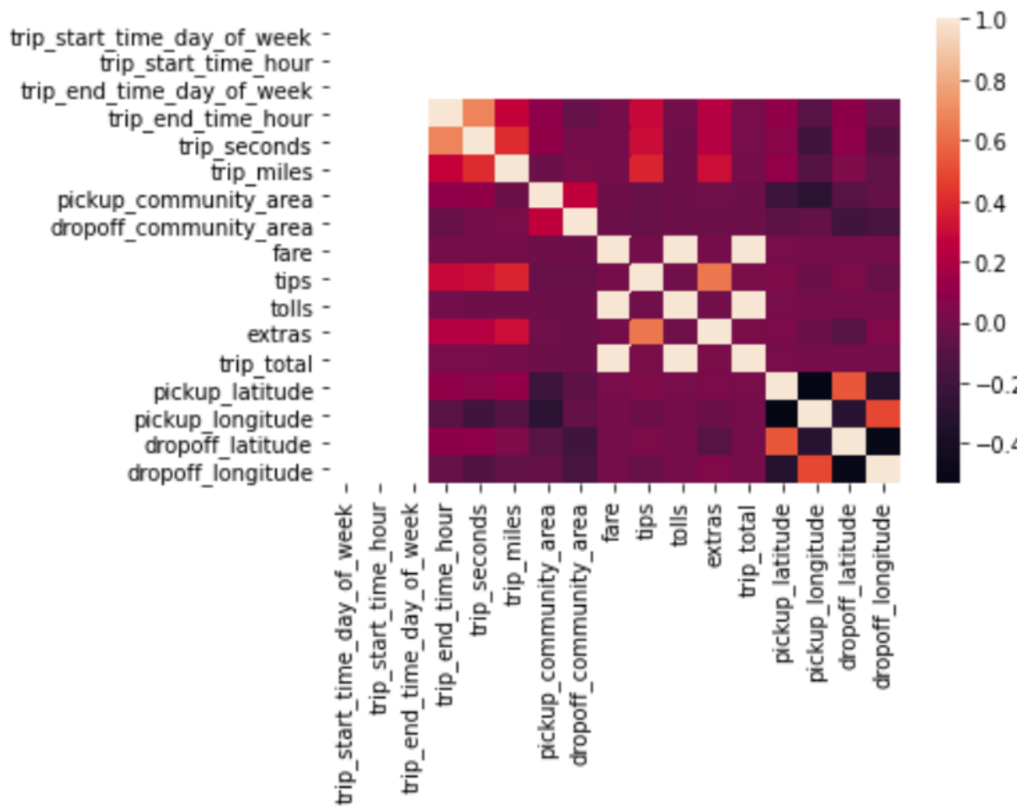
- Check for distribution: I used boxplot and bar chart to visualize the distribution of the numerical features. The visualization shows that trip_seconds contains a lot of outliers was validated and that the extreme values are to the right.

```
[65] # Plot boxplot for numerical features
     column = ['trip_start_time_day_of_week', 'trip_start_time_hour',
               'trip_end_time_day_of_week', 'trip_end_time_hour', 'trip_seconds']
     chicago_taxi_dataset.boxplot(column=column, figsize=(20,8), rot=20)
     pyplot.show()
```



- Check for correlation: I used heatmap to visualize the correlation matrix of the numerical features. I noticed that; tolls, trip_total, and fare strongly correlate with one another.

```
[70] #heatmap of correlation matrix
     sns.heatmap(chicago_taxi_dataset.corr());
     # chicago_taxi_dataset.corr()
```



Categorical Features

Check for distribution: I used a simple frequency table for visualize the distribution of values for payment_type.

```
[71] # function to compose frequency table for categorical variables
     def frequency_table(arr=[]):
       for value in arr:
         print(chicago_taxi_dataset[value].value_counts(normalize=True))
         print('\n')
```

```
[73] cat_lables = ['payment_type']
     frequency_table(cat_lables)
```

```
Cash           0.823
Credit Card    0.166
No Charge      0.008
Unknown        0.002
Dispute        0.001
Name: payment_type, dtype: float64
```

Data Preparation

Handle Missing Values

The columns with missing/ zero values of more than 30% of the columns were removed otherwise the missing values were replaced with the mean for that column. Columns with complex datatype were also removed.

```
[ ] # dropping rows with more than 30% non-null values and rows with non string or number.
    labels_wf_missing_values = ['pickup_census_tract', 'dropoff_census_tract', 'company', 'pickup_location', 'dropoff_location']
    chicago_taxi_dataset = chicago_taxi_dataset.drop(labels_wf_missing_values, axis=1)
```

```
[ ] def imput_missing_val(data, columns=[]):
      for column in columns:
        data[column].fillna(
          data[column].mean(), inplace=True
          )

      return data
```

```
[ ] # fill up missing values with the mean
    columns = ['trip_seconds', 'pickup_community_area', 'dropoff_community_area',
               'pickup_latitude', 'pickup_longitude', 'dropoff_latitude',
               'dropoff_longitude']
    chicago_taxi_dataset = imput_missing_val(chicago_taxi_dataset, columns=columns)
```

Encode Categorical Features

Categorical features were encoded using the one-hot encoding because the PCA algorithm require only numerical data.

```
[ ] # encode categorical features
    chicago_taxi_dataset = pandas.get_dummies(chicago_taxi_dataset, columns=['payment_type'])
    chicago_taxi_dataset
```

›lls	extras	trip_total	pickup_latitude	pickup_longitude	dropoff_latitude	dropoff_longitude	payment_type_Cash	payment_type_Credit Card	payment_type_Dispute	payment_type_No Charge	payment_type_Unknown
0.0	0.0	8.65	41.912566	-87.649121	41.913567	-87.650406	1	0	0	0	0
0.0	2.5	10.55	41.891972	-87.612945	41.879255	-87.642649	1	0	0	0	0
0.0	0.0	16.25	41.912566	-87.649121	41.913567	-87.650406	1	0	0	0	0
0.0	0.0	22.85	41.880994	-87.632746	41.986953	-87.672081	1	0	0	0	0
0.0	1.5	7.95	41.885281	-87.657233	41.879255	-87.642649	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...
0.0	3.0	7.84	41.929047	-87.651311	41.921778	-87.651062	1	0	0	0	0
0.0	0.0	16.05	41.880994	-87.632746	41.938391	-87.638575	1	0	0	0	0
0.0	1.0	6.65	41.870607	-87.622173	41.913567	-87.650406	1	0	0	0	0
0.0	2.0	8.45	41.884987	-87.620993	41.895033	-87.619711	1	0	0	0	0
0.0	1.0	8.85	41.929263	-87.635891	41.890922	-87.618868	1	0	0	0	0

Scale Numerical Features

- Features without outliers were scaled using the Standard Scaler API.
- Log transformation was applied to features with outliers and then Robust Scaler API was used for scaling. Log transformation was applied to normalize the distribution because ML algorithms work better with dataset that are normally distributed and Robust Scaler works best on features with outliers.

```
[ ] # Get min values in trip_seconds
    trip_seconds_min = chicago_taxi_dataset[['trip_seconds']].min() + 1
    trip_seconds_min
```

```
trip_seconds    1.0
dtype: float64
```

```
[ ] # Log and scale trip_seconds
    robust_scaler = RobustScaler()
    logged_ts = numpy.log(chicago_taxi_dataset[['trip_seconds']] + trip_seconds_min)
    chicago_taxi_dataset[['trip_seconds']] = robust_scaler.fit_transform(logged_ts)
```

Code Text

```
[ ] # Scale other features
    standard_scaler = StandardScaler()
    features_to_scaled = ['trip_start_time_day_of_week',  'trip_start_time_hour', 'trip_end_time_day_of_week',
                         'trip_end_time_hour', 'trip_miles','pickup_community_area', 'tips',  'extras',
                          'trip_total', 'pickup_latitude','pickup_longitude', 'dropoff_latitude', 'dropoff_longitude']
    chicago_taxi_dataset[features_to_scaled] = standard_scaler.fit_transform(chicago_taxi_dataset[features_to_scaled])
```

PCA

Goal: The goal of PCA is to find the directions (components) that maximize the variance in the dataset.

The target and independent variables were separated into “y” and “X” respectively. PCA was then carried out on the independent variables using the Scikit-Learn PCA API.

Below are two screenshots showing the implementation of PCA and the visualization of the distribution of the components with respect to their effect on the variance of the dataset;

```
[ ] # Represent dependent variable with y and independent variables with X
    y = chicago_taxi_dataset['fare']
    X = chicago_taxi_dataset.drop('fare', axis=1)
```

```
[ ] X
```

```
y
```

```
[ ] # a copy of list of column names
    column_names = list(X.columns.values)
```

PCA

```
[ ] # Initialize PCA without n_component given and fit with X
    pca = PCA(iterated_power=7)
    pca.fit(X)

    PCA(copy=True, iterated_power=7, n_components=None, random_state=None,
        svd_solver='auto', tol=0.0, whiten=False)
```

```
[ ] # number of features
    pca.n_components_

    21
```

```
[ ] # output the amount of variance explained by each of the selected components
    pca.explained_variance_

    array([2.09826118e+02, 3.49921178e+00, 2.58779966e+00, 2.19364166e+00,
           1.79383840e+00, 1.16414048e+00, 8.23178990e-01, 7.57263084e-01,
           6.64654713e-01, 4.79522003e-01, 4.38568375e-01, 2.11717701e-01,
           1.54993478e-01, 1.22248448e-02, 2.67677792e-03, 1.17221202e-03,
           2.02361914e-04, 6.79706538e-32, 1.65071029e-32, 1.61272429e-33,
           0.00000000e+00])
```

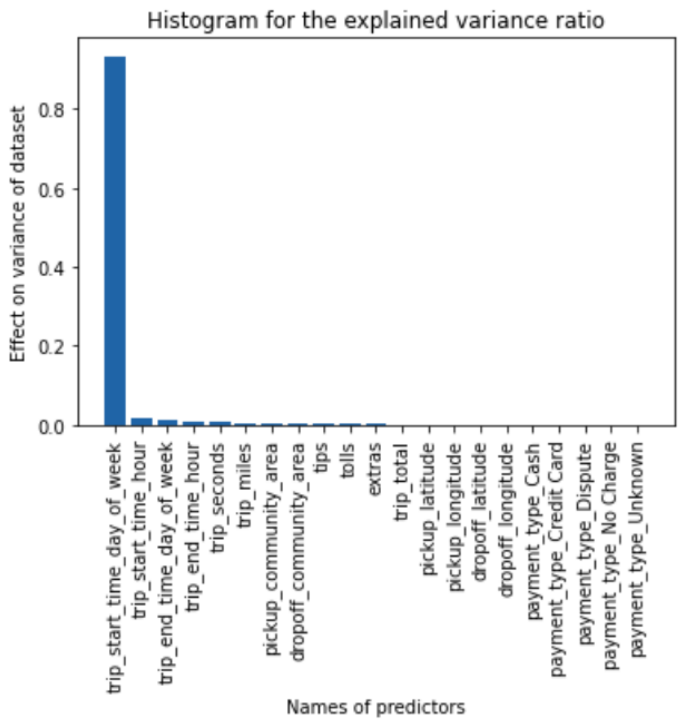
```
[ ] # output the percentage of variance explained by each of the selected components. It should sum up approximately 1.
    pca.explained_variance_ratio_

    array([9.34175924e-01, 1.55789919e-02, 1.15212546e-02, 9.76640681e-03,
           7.98642542e-03, 5.18292009e-03, 3.66491074e-03, 3.37144369e-03,
           2.95913796e-03, 2.13490062e-03, 1.95256921e-03, 9.42597520e-04,
           6.90053160e-04, 5.44267595e-05, 1.19173986e-05, 5.21885579e-06,
           9.00944218e-07, 3.02615084e-34, 7.34919861e-35, 7.18007946e-36,
           0.00000000e+00])
```

```
[ ] explained_variance_ratio = pca.explained_variance_ratio_
    len(explained_variance_ratio)
```

```
21
```

```
# Plot explained variane ratio with a bar chart
pyplot.xticks(range(len(explained_variance_ratio)), column_names)
pyplot.xlabel('Names of predictors')
pyplot.ylabel('Effect on variance of dataset')
pyplot.title('Histogram for the explained variance ratio')
pyplot.xticks(rotation=90)
pyplot.
pyplot.bar(range(len(explained_variance_ratio)), explained_variance_ratio)
pyplot.show()
```



Features to be used for predictive analysis on Big Query based on their effect on the variance of the dataset include:

- trip_start_time_day_of_week: 93.4%
- trip_start_time_hour: 1.55%
- trip_end_time_day_of_week: 1.15%
- trip_end_time_hour: 0.97%
- trip_seconds: 0.79%
- trip_miles: 0.51%
- pickup_community_area: 0.36%
- dropoff_community_area: 0.34%
- tips: 0.29%
- tolls: 0.21%
- extras: 0.19%

Additional feature(s);

- trip_total: It strongly correlates with the target variable even though it only has 0.09% effect on the variance of the dataset.

All in all, the features selected have a total of 99.85% effect on the variance of the dataset.

Task 3: Model Creation

Link to Query on Big Query

https://console.cloud.google.com/bigquery?sq=844523097149:34197bfe890141e79f79b56b41ca472f

Actual Query

```
create or replace model
  `fluted-alloy-266504.chicago_taxi_trips.reg_model_v6`
options
  (
  model_type='LINEAR_REG',
  labels=['fare']
  )
as
  -- standard sql
  select
    trip_start_time_day_of_week,
    trip_start_time_hour,
    trip_end_time_day_of_week,
    trip_end_time_hour,
    trip_seconds,
    trip_miles,
    fare,
    pickup_community_area,
    dropoff_community_area,
    tips,
    tolls,
    extras,
    trip_total
  from
    (
    select
      row_number() over (order by trip_start_timestamp),
      trip_start_timestamp,
      extract(dayofweek from trip_start_timestamp) as
trip_start_time_day_of_week,
      extract(hour from trip_start_timestamp) as
trip_start_time_hour,
      extract(dayofweek from trip_end_timestamp) as
trip_end_time_day_of_week,
      extract(hour from trip_end_timestamp) as trip_end_time_hour,
      trip_seconds,
      trip_miles,
      fare,
      pickup_community_area,
      dropoff_community_area,
      tips,
      tolls,
      extras,
      trip_total
    from
```

```
      `bigquery-public-data.chicago_taxi_trips.taxi_trips`
    )
  where
    fare > 0 and
    trip_miles > 0 and
    trip_seconds > 0 and
    trip_start_time_day_of_week is not null and
    trip_start_time_hour is not null and
    pickup_community_area is not null and
    dropoff_community_area is not null and
    trip_start_timestamp
      between timestamp
      '2013-05-11 04:45:00'
      and
      '2017-05-11 04:45:00' -- 2013-2017: 4 years
;
```

Task 4: Model Evaluation

Link to Query on Big Query

https://console.cloud.google.com/bigquery?sq=844523097149:5c2fd086996348239e5cd0463abc430e

Actual Query

```
select
  mean_squared_error, r2_score, explained_variance,
  case
    when r2_score > .9 and explained_variance > .9 then 'good'
    when r2_score > .8 and explained_variance > .8 then 'fair'
    when r2_score > .7 and explained_variance > .7 then 'decent'
    when r2_score > .6 and explained_variance > .6 then 'not great'
  else 'poor' end as model_quality_based_on_r2_score_and_variance
from
  ml.evaluate(model `fluted-alloy-
266504.chicago_taxi_trips.reg_model_v6`, (
  -- standard sql
  select
    trip_start_time_day_of_week,
    trip_start_time_hour,
    trip_end_time_day_of_week,
    trip_end_time_hour,
    trip_seconds,
    trip_miles,
    fare,
    pickup_community_area,
    dropoff_community_area,
    tips,
    tolls,
    extras,
    trip_total
  from
    (
    select
      row_number() over (order by trip_start_timestamp),
      trip_start_timestamp,
      extract(dayofweek from trip_start_timestamp) as
trip_start_time_day_of_week,
      extract(hour from trip_start_timestamp) as
trip_start_time_hour,
      extract(dayofweek from trip_end_timestamp) as
trip_end_time_day_of_week,
      extract(hour from trip_end_timestamp) as trip_end_time_hour,
      trip_seconds,
      trip_miles,
      fare,
      pickup_community_area,
      dropoff_community_area,
      tips,
      tolls,
```

```
      extras,
      trip_total
    from
      `bigquery-public-data.chicago_taxi_trips.taxi_trips`
    )
  where
    fare > 0 and
    trip_miles > 0 and
    trip_seconds > 0 and
    trip_start_time_day_of_week is not null and
    trip_start_time_hour is not null and
    pickup_community_area is not null and
    dropoff_community_area is not null and
    trip_start_timestamp
      between timestamp
      '2017-05-11 04:46:00'
      and
      '2019-05-11 04:45:00' -- 2017-2019: 2 years
));
```

Task 5: Model Prediction

Link to Query on Big Query

https://console.cloud.google.com/bigquery?sq=844523097149:33325368fe4e43ebb5482e512b3b4502

Actual Query

```
select *
from
  ml.predict(model `fluted-alloy-
266504.chicago_taxi_trips.reg_model_v6`, (
  -- standard sql
  select
    trip_start_time_day_of_week,
    trip_start_time_hour,
    trip_end_time_day_of_week,
    trip_end_time_hour,
    trip_seconds,
    trip_miles,
    fare,
    pickup_community_area,
    dropoff_community_area,
    tips,
    tolls,
    extras,
    trip_total
  from
    (
    select
      row_number() over (order by trip_start_timestamp),
      trip_start_timestamp,
      extract(dayofweek from trip_start_timestamp) as
trip_start_time_day_of_week,
      extract(hour from trip_start_timestamp) as
trip_start_time_hour,
      extract(dayofweek from trip_end_timestamp) as
trip_end_time_day_of_week,
      extract(hour from trip_end_timestamp) as trip_end_time_hour,
      trip_seconds,
      trip_miles,
      fare,
      pickup_community_area,
      dropoff_community_area,
      tips,
      tolls,
      extras,
      trip_total
    from
      `bigquery-public-data.chicago_taxi_trips.taxi_trips`
    )
  where
    fare > 0 and
```

```
    trip_miles > 0 and
    trip_seconds > 0 and
    trip_start_time_day_of_week is not null and
    trip_start_time_hour is not null and
    pickup_community_area is not null and
    dropoff_community_area is not null and
    trip_start_timestamp
      between timestamp
      '2019-05-11 04:46:00'
      and
      '2020-05-11 04:45:00' -- 2019-2020: A year
));
```

Task 6: Screenshot of Model Evaluation Report

Result from Model Creation Query

reg_model_v6

Details | Training | **Evaluation** | Schema

Mean absolute error	0.0397
Mean squared error	0.0100
Mean squared log error	0.0001
Median absolute error	0.0125
R squared	0.9999

Result from Model Evaluation Query

Job information | **Results** | JSON | Execution details

Row	mean_squared_error	r2_score	explained_variance	model_quality_based_on_r2_score_and_variance
1	1.0815584610554083	0.9986835884292433	0.9986972258002985	good