Data Science Project-

Citibike

Presented by-

Varun Ramesh (vr2121@nyu.edu)

Jonathan Kung (hk3234@nyu.edu)

Rajan Shantanu Chaturvedi (rsc9044@nyu.edu)

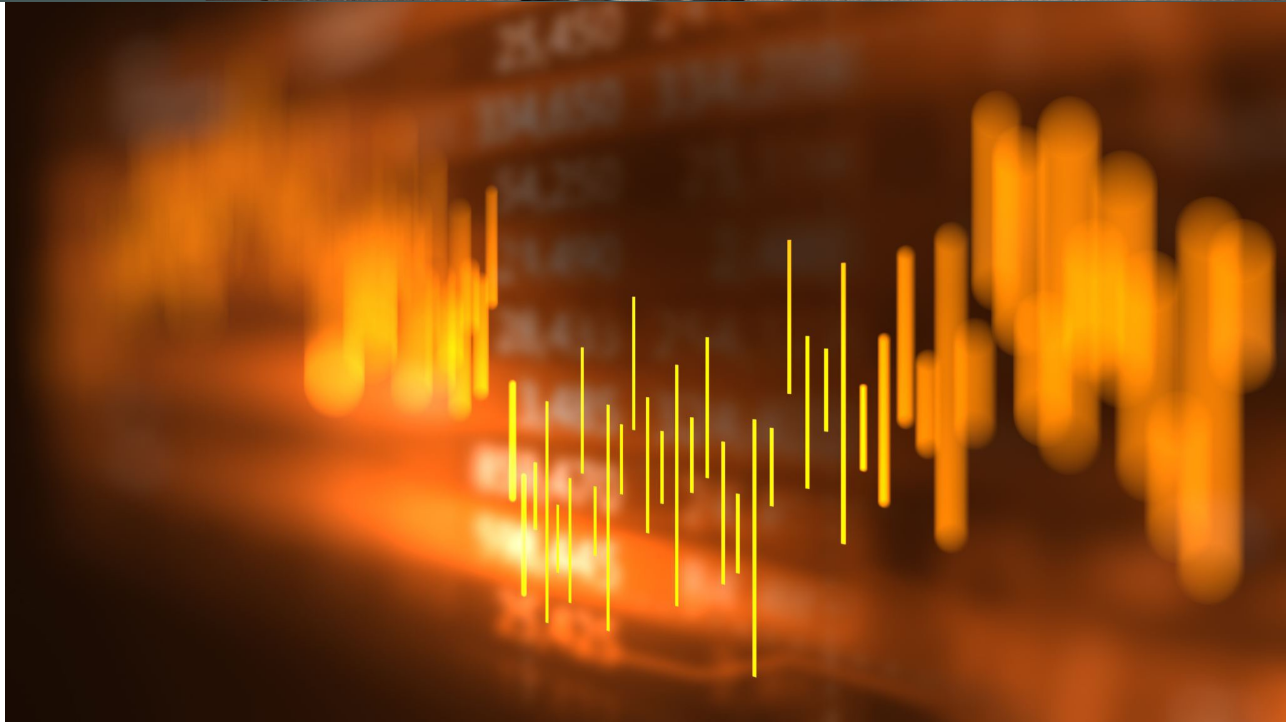


TABLE OF CONTENTS

1. Business problems
2. Technical setup
3. Results
4. Challenges and Changes
5. Deployment/ Next Steps
6. Limitations/ Other Work

Business problems

Problem Statement-

We want to predict the number of trips that will originate at Citibike stations located near either Grand Central Station or near Penn Station, with the time period before midnight on the day of interest.

Current Solution-

Citibike uses machine learning algorithms to help optimize the time of their operations team so that they can be more efficient in moving bikes across stations. They have also implemented Valet Service for expanded bike and dock availability. The company also rewards their customers with points which are redeemable.

Source- https://ride.citibikenyc.com/blog/ridershiprecords

Our Solution-

We will be using Decision Tree Classifier Algorithm for our prediction for the 2019 dataset. We use this because the algorithm needs low level of data preparation which is very much needed in this case, and it will be able to distinguish a large number of classes with less error. The maximum tree depth is set at 9.

We will be using K nearest neighbor (KNN) for our prediction for the 2021 dataset since it is versatile to different calculations of proximity. It is also a memory-based approach.

Technical setup

<u>Data Understanding- </u> The data provided is that of 2019 and 2021 which contains all citibike trips made in September. A weather dataset is also provided which contains the weather information from the year 2018 to 2021 of NYC.

2019 Dataset

In [100]:
```
#Loading the Data in the Dataframe
df_2019= pd.read_csv("/Users/rajanpc/Desktop/DSBA_Project/201909-citibike-tripdata.csv")
df_2019
```

Out[100]:

	tripduration	starttime	stoptime	start station id	start station name	start station latitude	start station longitude	end station id	end station name	end station latitude	end station longitude	bikeid	usertype	birth year
0	327	2019-09-01 00:00:01.9580	2019-09-01 00:05:29.3410	3733	Avenue C & E 18 St	40.730563	-73.973984	504	1 Ave & E 16 St	40.732219	-73.981656	39213	Subscriber	1968
1	1145	2019-09-01 00:00:04.1430	2019-09-01 00:19:09.8360	3329	Degraw St & Smith St	40.682915	-73.993182	270	Adelphi St & Myrtle Ave	40.693083	-73.971789	21257	Customer	1969
2	1293	2019-09-01 00:00:07.3090	2019-09-01 00:21:40.7580	3168	Central Park West & W 85 St	40.784727	-73.969617	423	W 54 St & 9 Ave	40.765849	-73.986905	15242	Customer	1969
3	1753	2019-09-01 00:00:08.0640	2019-09-01 00:29:21.5040	3299	E 98 St & Park Ave	40.788130	-73.952060	3160	Central Park West & W 76 St	40.778968	-73.973747	38760	Subscriber	1990

2021 Dataset:

```
df_2021= pd.read_csv("/Users/rajanpc/Desktop/DSBA_Project/202109-citibike-tripdata.csv")

df_2021
```

	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id	end_station_name	end_station_id	start_lat	start_lng
0	22C33F42C6A0E28E	classic_bike	2021-09-01 10:26:45	2021-09-01 10:43:23	Central Park West & W 72 St	7141.07	E 51 St & 1 Ave	6532.06	40.775794	-73.976206
1	035F743147FCFCDE	classic_bike	2021-09-04 09:52:40	2021-09-04 10:09:19	William St & Pine St	5065.12	NaN	NaN	40.707179	-74.008873
2	9C43CF6A07DACBC6	classic_bike	2021-09-06 17:07:40	2021-09-06 17:34:44	Fulton St & Broadway	5175.08	Jay St & Tech Pl	4710.06	40.711066	-74.009447
3	253A1A5B20CC78EE	classic_bike	2021-09-28 16:53:43	2021-09-28 17:03:00	West Drive & Prospect Park West	3651.04	Ocean Pkwy & Church Ave	3125.09	40.661063	-73.979453
4	5E8F164D6798CEFA	classic_bike	2021-09-19 09:37:47	2021-09-19 09:53:42	Lorimer St & Broadway	4965.01	Jay St & Tech Pl	4710.06	40.704118	-73.948186
...	...	...	...	...	...	...	...	...	...	...
3280216	8A1C8DB4249BF100	classic_bike	2021-09-26 16:00:45	2021-09-26 16:20:51	8 Ave & W 31 St	6450.05	W 67 St & Broadway	7116.04	40.750585	-73.994685
3280217	C290EE73DF58AD79	classic_bike	2021-09-07 08:22:06	2021-09-07 08:38:40	S Portland Ave & Hanson Pl	4354.05	S 3 St & Bedford Ave	5235.05	40.685396	-73.974315

Weather Dataset-

	date	COLD	PRCP	Total_Trips	Day_Type	Day_num	trip_category
0	2019-09-01	70.5	0.00	483	0.0	1	301-500
1	2019-09-02	71.5	0.30	220	0.0	2	0-300
2	2019-09-03	74.5	0.00	703	1.0	3	501-750
3	2019-09-04	77.5	0.00	539	1.0	4	501-750
4	2019-09-05	69.5	0.00	727	1.0	5	501-750
5	2019-09-06	62.5	0.32	272	1.0	6	0-300
6	2019-09-07	65.5	0.02	573	0.0	7	501-750
7	2019-09-08	70.0	0.00	519	0.0	8	501-750
8	2019-09-09	71.0	0.00	626	1.0	9	501-750
9	2019-09-10	71.0	0.01	694	1.0	10	501-750
10	2019-09-11	78.5	0.00	766	1.0	11	More than 751
11	2019-09-12	71.0	0.17	453	1.0	12	301-500
12	2019-09-13	64.5	0.00	673	1.0	13	501-750
13	2019-09-14	67.5	0.00	525	0.0	14	501-750
14	2019-09-15	75.0	0.00	612	0.0	15	501-750
15	2019-09-16	72.5	0.00	566	1.0	16	501-750

Data Preparation- First, we filter out the trips whose duration was less than 120 seconds or 2 minutes. Then we drop the rows which have the same start and end station. In the next step we calculate whether a given day is a weekday or a weekend and whether it is a holiday or not. New columns are created containing both these values. Cold and precipitation columns are also added in the final data preparation.

2019 Cleaned Dataset -

```
#Cleaning and Sorting the Dataframe. Dropped the trips under the 120 seconds of trip duration.
df_2019.sort_values("tripduration", axis = 0, ascending = True, inplace = True, na_position ='last')
trip_dur = df_2019.apply(lambda x: True if x['tripduration'] < 121 else False , axis=1)
rows_num = len(trip_dur[trip_dur == True].index)
print('Total number of trips having duration less than 120 seconds:', rows_num)
df_2019_temp = df_2019.drop(df_2019[(df_2019['tripduration'] < 121)].index)
df_2019_cleaned = df_2019_temp[df_2019_temp['start station id'] != df_2019_temp['end station id']]
df_2019_cleaned
```

```
Total number of trips having duration less than 120 seconds: 36570
```

	tripduration	starttime	stoptime	start station id	start station name	start station latitude	start station longitude	end station id	end station name	end station latitude	end station longitude	bikeid	usertype	bir ye
369443	121	2019-09-06 08:14:05.6490	2019-09-06 08:16:07.5250	3656	E 2 St & Avenue A	40.723077	-73.985836	403	E 2 St & 2 Ave	40.725029	-73.990697	33883	Subscriber	19
1513271	121	2019-09-20 09:10:50.3250	2019-09-20 09:12:51.8200	3714	Division Av & Hooper St	40.706842	-73.954435	3077	Stagg St & Union Ave	40.708771	-73.950953	42013	Subscriber	19
2302025	121	2019-09-29 09:55:47.5610	2019-09-29 09:57:48.7430	238	Bank St & Washington St	40.736197	-74.008592	405	Washington St & Gansevoort St	40.739323	-74.008119	40811	Subscriber	19

```python
#Creating the Date, Hour, Day columns and changing the date format.
df_2019_cleaned['date'] = df_2019_cleaned['starttime'].apply(lambda x:x.split(' ')[0])
df_2019_cleaned["hour"] = df_2019_cleaned['starttime'].apply(lambda x: x.split()[1].split(":")[0])
df_2019_cleaned['day'] = df_2019_cleaned['date'].apply(lambda x: pd.to_datetime(x,format='%Y-%m-%d')).dt.weekday
df_2019_cleaned['date'] = pd.to_datetime(df_2019_cleaned['date'],format = '%Y-%m-%d')
df_2019_cleaned['starttime'] = pd.to_datetime(df_2019_cleaned['starttime'])
df_2019_cleaned['stoptime']= pd.to_datetime(df_2019_cleaned['stoptime'])
df_2019_cleaned
```

;tarttime	stoptime	start station id	start station name	start station latitude	start station longitude	end station id	end station name	end station latitude	end station longitude	bikeid	usertype	birth year	gender	date	hour	day
19-09-06 4:05.649	2019-09-06 08:16:07.525	3656	E 2 St & Avenue A	40.723077	-73.985836	403	E 2 St & 2 Ave	40.725029	-73.990697	33883	Subscriber	1993	1	2019-09-06	08	4
19-09-20 0:50.325	2019-09-20 09:12:51.820	3714	Division Av & Hooper St	40.706842	-73.954435	3077	Stagg St & Union Ave	40.708771	-73.950953	42013	Subscriber	1981	2	2019-09-20	09	4
19-09-29 5:47.561	2019-09-29 09:57:48.743	238	Bank St & Washington St	40.736197	-74.008592	405	Washington St & Gansevoort St	40.739323	-74.008119	40811	Subscriber	1978	1	2019-09-29	09	6

```
# Creating the column of weekend: 0-weekday, 1-weekend
week_end = []
for i in df_2019_cleaned['date']:
  num=i.weekday()
  if num < 5:
    k = 0
  else:
    k = 1
  week_end.append(k)
df_2019_cleaned['Weekend'] = week_end

# Creating the column of holiday: 0 - holiday, 1-Not holiday
holidays = holidays.UnitedStates(years = 2019)
Holi_day = []
for i in df_2019_cleaned['date']:
  if i in holidays:
    Holi_day.append(1)
  else:
    Holi_day.append(0)
df_2019_cleaned['Holiday'] = Holi_day

# Creating the column of Working Day: 0 - holiday, 1-Not holiday
Working_Day = []
for i in range(len(df_2019_cleaned['Weekend'])):
    if (df_2019_cleaned['Weekend'].iloc[i] == 1) or (df_2019_cleaned['Holiday'].iloc[i] == 1):
      Working_Day.append('FALSE')
    else:
      Working_Day.append('TRUE')
df_2019_cleaned['Working_Day'] = Working_Day
df_2019_cleaned
```

start ation id	start station name	start station latitude	start station longitude	end station id	end station name	end station latitude	...	bikeid	usertype	birth year	gender	date	hour	day	Weekend	Holiday	Working_Day
3656	E 2 St & Avenue A	40.723077	-73.985836	403	E 2 St & 2 Ave	40.725029	...	33883	Subscriber	1993	1	2019-09-06	08	4	0	0	TRUE
3714	Division Av & Hooper St	40.706842	-73.954435	3077	Stagg St & Union Ave	40.708771	...	42013	Subscriber	1981	2	2019-09-20	09	4	0	0	TRUE
238	Bank St & Washington St	40.736197	-74.008592	405	Washington St & Gansevoort St	40.739323	...	40811	Subscriber	1978	1	2019-09-29	09	6	1	0	FALSE
497	E 17 St & Broadway	40.737050	-73.990093	285	Broadway & E 14 St	40.734546	...	30855	Subscriber	1987	1	2019-09-10	16	1	0	0	TRUE
	North																

```python
#Loading the weather data of 2019 into the dataframe
dfw= pd.read_csv("/Users/rajanpc/Desktop/DSBA_Project/weather_nyc_20180101_to_20211031.c
dfw = dfw[(dfw['DATE']> "2018-12-31") & (dfw['DATE']< "2020-01-01")]
dfw['DATE']= pd.to_datetime(dfw['DATE'])

#Adding average tempreture columns. Avg = (TMAX + TMIN)/2
dfw['COLD'] = (dfw['TMAX'] + dfw['TMIN'])/2

# Filtering the relevant columns
dfw = dfw.loc[:, dfw.columns.intersection(['NAME','DATE', 'PRCP', 'COLD'])]
dfw
```

	NAME	DATE	PRCP	COLD
365	NY CITY CENTRAL PARK, NY US	2019-01-01	0.06	48.5
366	NY CITY CENTRAL PARK, NY US	2019-01-02	0.00	37.5
367	NY CITY CENTRAL PARK, NY US	2019-01-03	0.00	40.5
368	NY CITY CENTRAL PARK, NY US	2019-01-04	0.00	41.0
369	NY CITY CENTRAL PARK, NY US	2019-01-05	0.50	44.0

```python
#Merging the Cleaned and Weather dataframes
df_2019_final = pd.merge(df_2019_cleaned[['date','start station name','end station name','Weekend','Holiday','Working_D
df_2019_final = df_2019_final.drop(columns='DATE')

#Filtering the Relevant features of stations near to Grand Central Station
df_trips= pd.DataFrame()
for i in ['West St & Chambers St','E 103 St & Lexington Ave','Van Brunt St & Wolcott St*40.677343','Jay St & Tech Pl']:
    dft = df_2019_final[df_2019_final['start station name'].str.lower() == i.lower()]
    dft = dft[['date','Working_Day','COLD','PRCP']].sort_values(by = ['date']).groupby(['date','Working_Day','COLD','PR
    df_trips = pd.concat([df_trips, dft])\
           .groupby(['date','Working_Day','COLD','PRCP'])['Total_Trips']\
           .sum().reset_index()

#Creating the Day type column. No Workday-0, Workday-1
df_trips.loc[df_trips['Working_Day'] == 'FALSE', 'Day_Type'] = 0
df_trips.loc[df_trips['Working_Day'] == 'TRUE', 'Day_Type'] = 1
df_trips.drop('Working_Day', axis=1, inplace=True)
df_trips['Day_num'] = df_trips['date'].apply(lambda x:pd.to_datetime(x,format='%Y-%m-%d')).dt.day

#Categorising the Trips
cus_bins = [0,300,500,750,1000]
trips_label = ['0-300','301-500','501-750','More than 1000']
df_trips['trip_category'] = pd.cut(df_trips['Total_Trips'],bins=cus_bins,labels=trips_label,include_lowest=True)
df_trips['trip_category'].groupby(df_trips['trip_category']).count()
df_trips
```

	date	COLD	PRCP	Total_Trips	Day_Type	Day_num	trip_category
0	2019-09-01	70.5	0.00	483	0.0	1	301-500

2021 Cleaned Dataset-

```
# Sorting the dataframe

df_2021_cleaned = df_2021[df_2021['start_station_id'] != df_2021['end_station_id']]
df_2021_cleaned.dropna()
```

	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id	end_station_name	end_station_id	start_lat	start_lng
0	22C33F42C6A0E28E	classic_bike	2021-09-01 10:26:45	2021-09-01 10:43:23	Central Park West & W 72 St	7141.07	E 51 St & 1 Ave	6532.06	40.775794	-73.976206
2	9C43CF6A07DACBC6	classic_bike	2021-09-06 17:07:40	2021-09-06 17:34:44	Fulton St & Broadway	5175.08	Jay St & Tech Pl	4710.06	40.711066	-74.009447
3	253A1A5B20CC78EE	classic_bike	2021-09-28 16:53:43	2021-09-28 17:03:00	West Drive & Prospect Park West	3651.04	Ocean Pkwy & Church Ave	3125.09	40.661063	-73.979453
4	5E8F164D6798CEFA	classic_bike	2021-09-19 09:37:47	2021-09-19 09:53:42	Lorimer St & Broadway	4965.01	Jay St & Tech Pl	4710.06	40.704118	-73.948186
5	0702265BE26C21F3	classic_bike	2021-09-23 09:35:32	2021-09-23 09:38:00	William St & Pine St	5065.12	Fulton St & Pearl St	5024.09	40.707179	-74.008873
...	...	...	...	...	...	...	...	...	...	...
3280216	8A1C8DB4249BF100	classic_bike	2021-09-26 16:00:45	2021-09-26 16:20:51	8 Ave & W 31 St	6450.05	W 67 St & Broadway	7116.04	40.750585	-73.994685

Using the same method mentioned in the above slides, the final 2021 dataset is obtained.

	date	COLD	PRCP	Total_Trips	Day_Type	Day_num	trip_category
0	2021-09-01	70.5	7.13	230	1.0	1	0-300
1	2021-09-02	69.0	0.10	783	1.0	2	More than 750
2	2021-09-03	66.5	0.00	646	1.0	3	501-750
3	2021-09-04	70.0	0.00	607	0.0	4	501-750
4	2021-09-05	70.5	0.02	423	0.0	5	301-500
5	2021-09-06	75.0	0.00	629	0.0	6	501-750
6	2021-09-07	72.0	0.00	319	1.0	7	301-500
7	2021-09-08	76.0	0.00	175	1.0	8	0-300
8	2021-09-09	72.0	0.26	339	1.0	9	301-500
9	2021-09-10	68.5	0.00	769	1.0	10	More than 750
10	2021-09-11	68.0	0.00	253	0.0	11	0-300
11	2021-09-12	73.5	0.00	123	0.0	12	0-300
12	2021-09-13	75.0	0.12	510	1.0	13	501-750
13	2021-09-14	73.5	0.00	790	1.0	14	More than 750
14	2021-09-15	78.0	0.00	694	1.0	15	501-750
15	2021-09-16	73.5	0.00	278	1.0	16	0-300
16	2021-09-17	72.0	0.00	534	1.0	17	501-750
17	2021-09-18	76.5	0.00	714	0.0	18	501-750

Modeling Techniques- We are going to use Decision Tree Classifier and K Nearest Neighbor for our problem. In the DT classifier we split the data into 70% training and 30% testing data with max depth 9. For KNN we split the data into 65% training and 35% testing data with n neighbors=3.

Performance Measures- We will be using accuracy score and confusion matrix as performance measures.

```
#Model Preparation
np.random.seed(42)
features = ['COLD','PRCP','Day_Type']
X = df_trips[features]
Y = df_trips['trip_category']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.30)
```

```
# Decision Classifier
from sklearn import tree
from sklearn.metrics import accuracy_score
decision_clf = tree.DecisionTreeClassifier(criterion="entropy",max_depth=9)
decision_clf = decision_clf.fit(X_train, Y_train)
print ("Accuracy on training = %.4f" % accuracy_score(decision_clf.predict(X_train), Y_train))
```

```
Accuracy on training = 1.0000
```

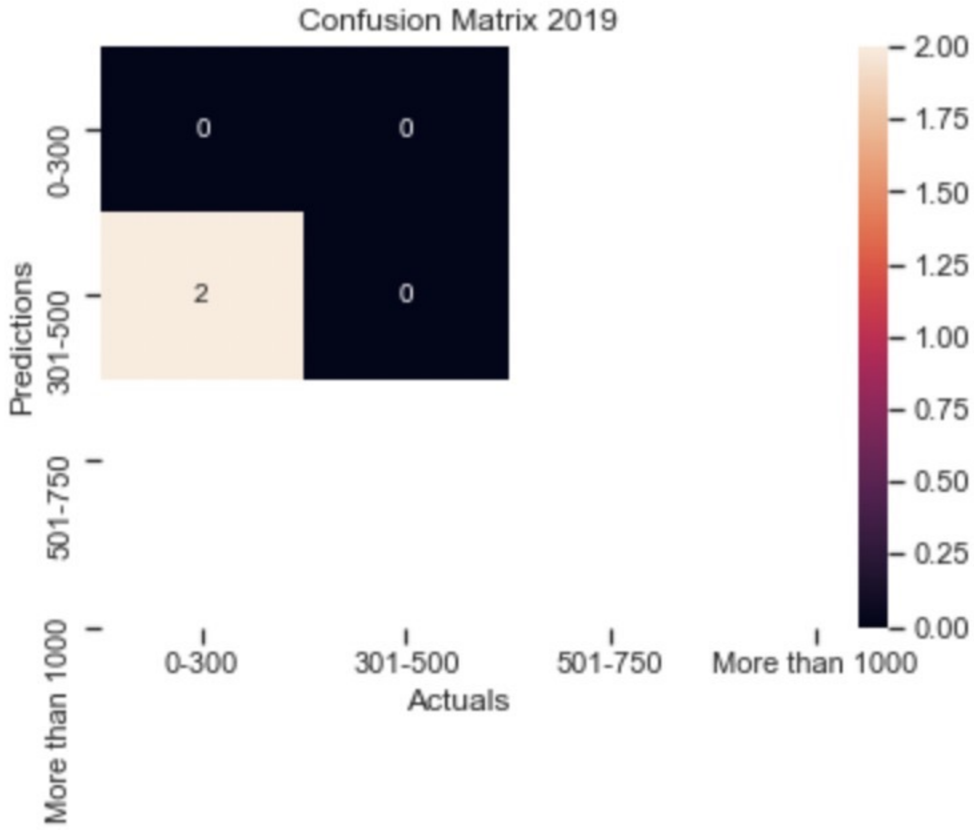
```
decision_clf = tree.DecisionTreeClassifier(criterion="entropy", max_depth=9)
decision_clf = decision_clf.fit(X_train, Y_train)
print ("Accuracy on testing = %.4f" % accuracy_score(decision_clf.predict(X_test), Y_test))
```

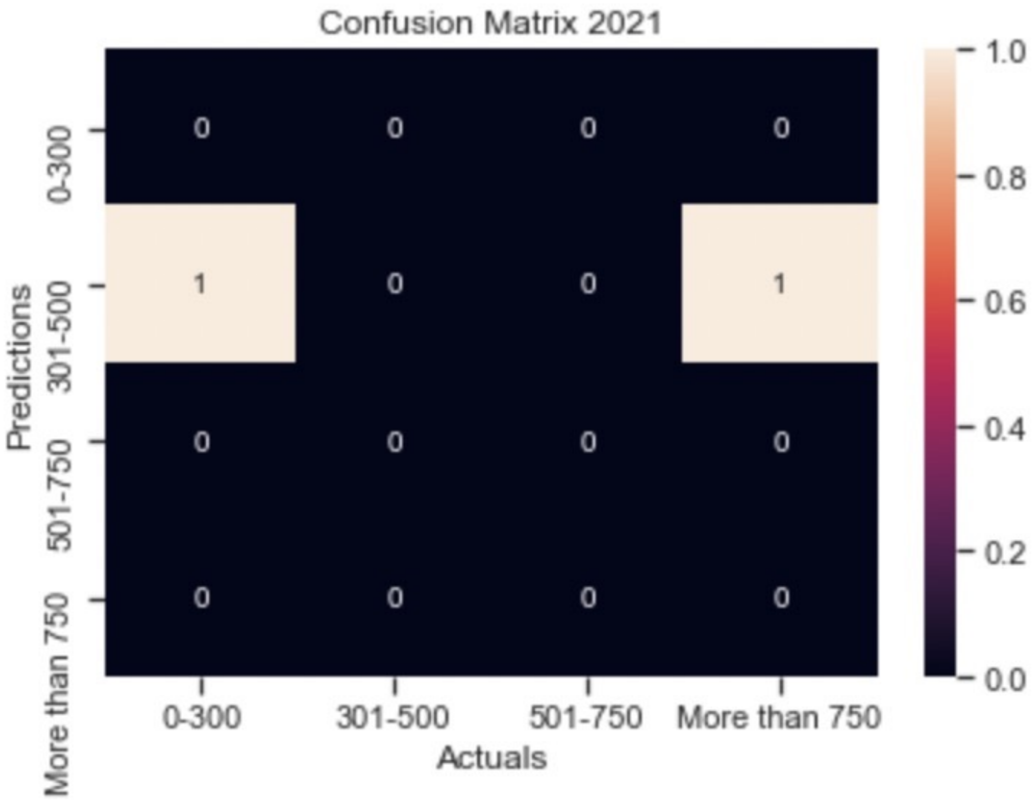
```
Accuracy on testing = 0.7778
```

In [218]:
```
#model Preparation- 2021
np.random.seed(42)
features = ['COLD','PRCP','Day_Type']
X = df_trips[features]
Y = df_trips['trip_category']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.35)
```

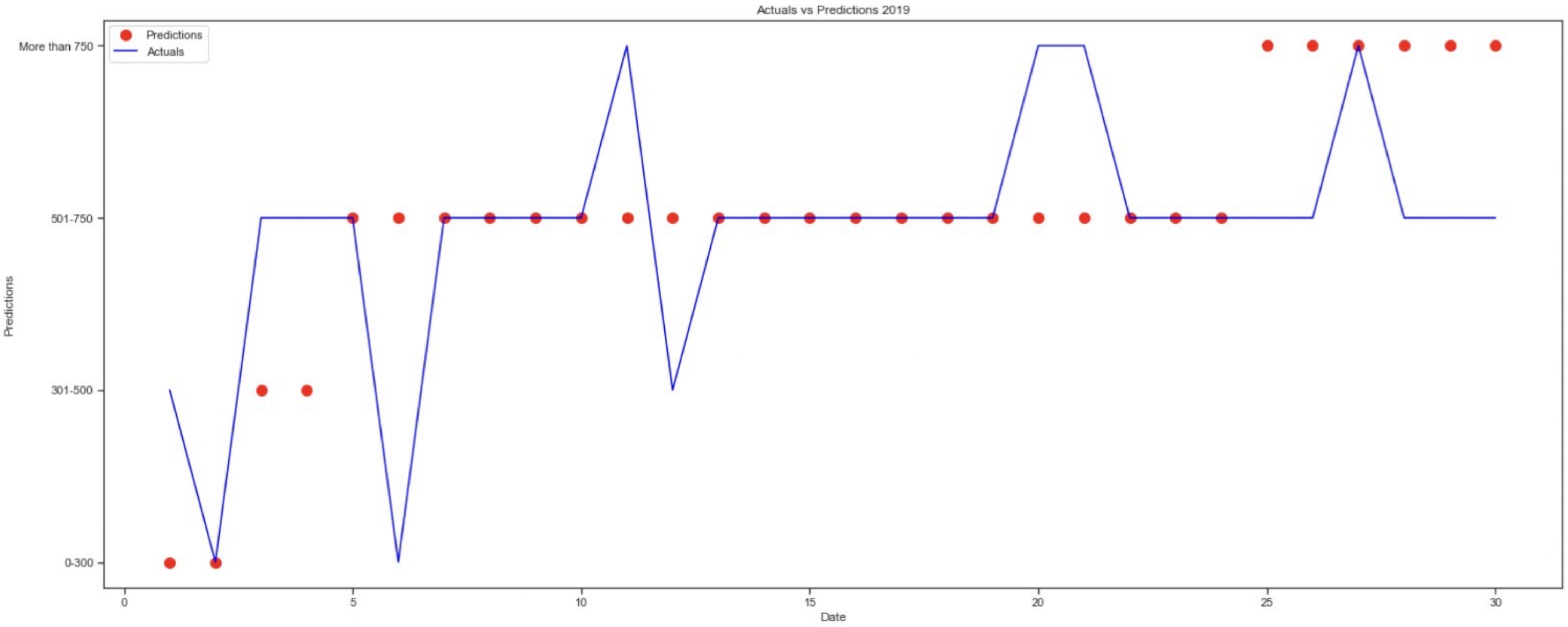
In [262]:
```
#k-nearest neighbors algorithm(KNN)
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train,Y_train)
knn.fit(X_test,Y_test)
print ("Accuracy on testing = %.4f" % accuracy_score(Knnmodel.predict(X_test), Y_test) )
```

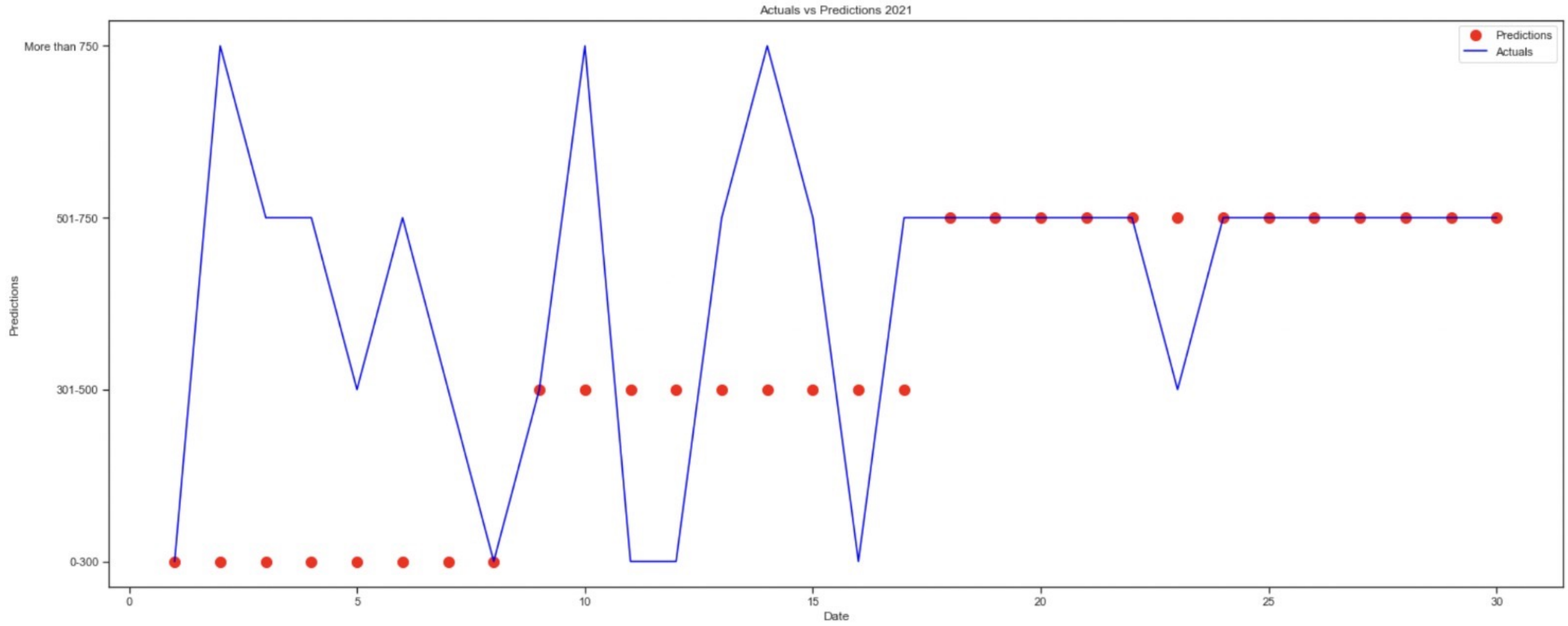
```
Accuracy on testing = 0.8182
```

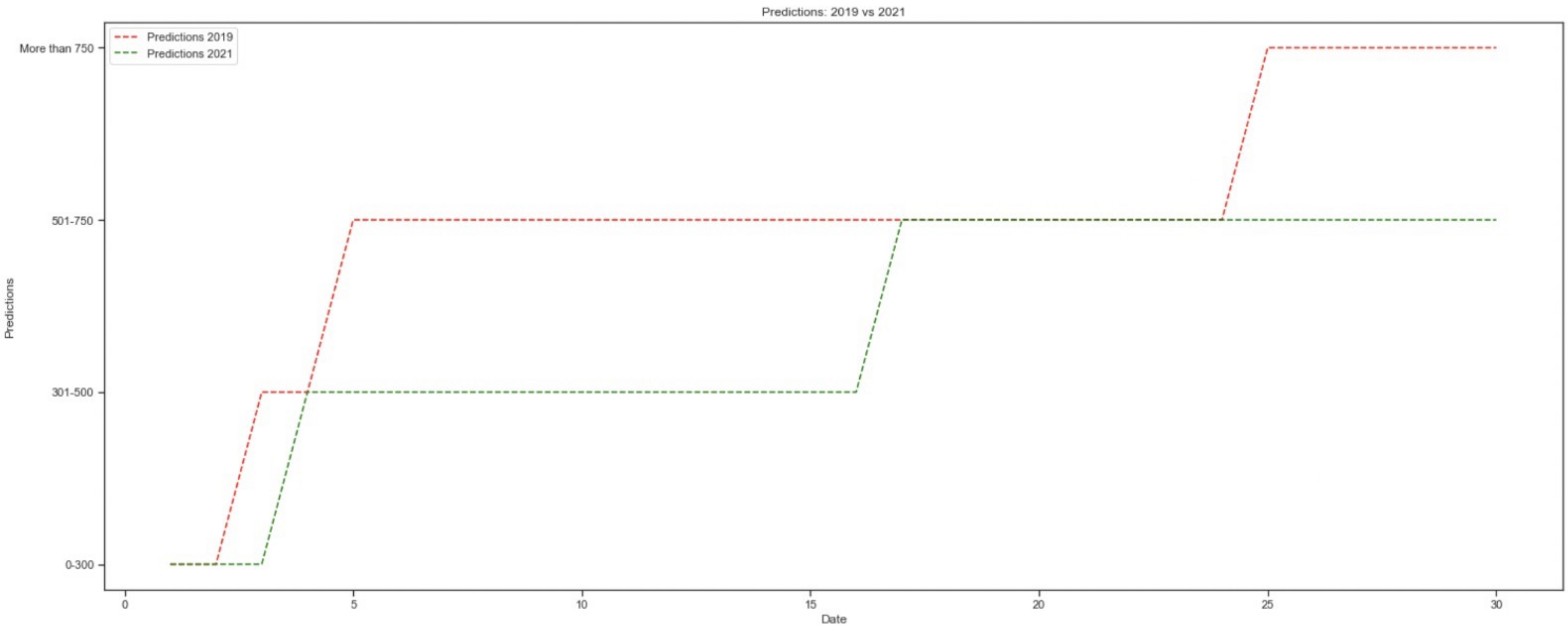

Confusion Matrix 2019
Predictions
0-300
301-500
501-750
More than 1000
Actuals
0
0
2
0
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00


Confusion Matrix 2021
Predictions
0-300
301-500
501-750
More than 750
Actuals
0 0 0 0
1 0 0 1
0 0 0 0
0 0 0 0
1.0
0.8
0.6
0.4
0.2
0.0

Results


Actuals vs Predictions 2019
Predictions
Actuals
More than 750
501-750
301-500
0-300
Predictions
0
5
10
15
20
25
30
Date


Actuals vs Predictions 2021
Predictions
Actuals
More than 750
501-750
301-500
0-300
Predictions
0
5
10
15
20
25
30
Date


Predictions: 2019 vs 2021
Predictions 2019
Predictions 2021
More than 750
501-750
301-500
0-300
Predictions
0
5
10
15
20
25
30
Date

Technical Challenges and Changes

1. We used two different algorithms on both the datasets because using Decision Tree Algorithm on the 2021 dataset only resulted in a 40% accuracy rate. We tried to debug the code, but we didn’t have enough time. This is the reason we used KNN for the 2021 dataset.
2. We would probably use all the algorithms for prediction and choose the most fitting algorithm. We would use the same algorithm for both the datasets.
3. We would also have a better prediction if both the datasets had similar columns and structure.

Deployment/ Next Steps

1. If the results were successful, we would like to predict the number of rides for the whole year. This will provide a better insight into customer behaviors, patterns, etc.
2. With a successful code, Citi can predict customer patterns which will be in the next month. Keeping this in consideration they can release related advertisement campaigns, promos, etc.

Limitation/ Other Work

1. If there were bike types in both the datasets, Citi could have potentially identified which of their bike types are used the most and those bike types could have been deployed in its stations.

2. There were lot of ‘NA’ values in the dataset. These values severely affect the prediction.

3. In the dataset, a member is only listed as a member whereas there are three types of membership in Citibike namely monthly, yearly and daily passes. A more detailed database is needed for detailed predictions.

4. We have filtered stations which have the same names. This will also filter out the customers who actually ride for a long time and end up at the same station.