Problem Statement

- Predict time delay in Toronto streets given the route, time, day, date, and location. This machine-learning model will help drivers plan their trips ahead to avoid all sort of delays.

Datasets

The dataset contain six different files corresponding to six different years starting from 2014, and ending at 2019. All 12 months of the year with each day of the month are available in each file. Each day is shown with different timing, locations, and routes.

Report Date	Route	Time	Day	Location	Incident	Min Delay	Min Gap	Direction	Vehicle
2014/01/02	505	6:31:00 AM	Thursday	Dundas and Roncesvalles	Late Leaving Garage	4	8	E/B	4018
2014/01/02	504	12:43:00 PM	Thursday	King and Shaw	Utilized Off Route	20	22	E/B	4128
2014/01/02	501	2:01:00 PM	Thursday	Kingston road and Bingham	Held By	13	19	W/B	4016
2014/01/02	504	2:22:00 PM	Thursday	King St. and Roncesvalles Ave.	Investigation	7	11	W/B	4175
2014/01/02	504	4:42:00 PM	Thursday	King and Bathurst	Utilized Off Route	3	6	E/B	4080
2014/01/02	501	5:39:00 PM	Thursday	Queen and Beaconsfeild	Held By	7	13	W/B	4202
2014/01/02	504	6:38:00 PM	Thursday	Roncesvalles and King Street West	Utilized Off Route	4	7	E/B	4100
2014/01/02	510	7:27:00 PM	Thursday	Spadina and St. Andrews	Investigation	20	22	S/B	4123
2014/01/03	504	1:00:00 AM	Friday	Broadview and Queen	Utilized Off Route	7	14	W/B	4079
2014/01/03	512	5:09:00 AM	Friday	Bathurst and St. Clair	Mechanical	3	6	E/B	4160

The following are the features in the datasets with their descriptions

- Report Date: The date (YYYY/MM/DD) when the delay-causing incident occurred
- Route: The number of the streetcar route
- Time: The time when the delay-causing incident occurred
- Day: The day when the delay-causing incident occurred
- Location: The location where the delay-causing incident occurred
- Incident: The description of the delay-causing incident
- Min Delay: The delay, in minutes, to the schedule for the following streetcar
- Min Gap: The total scheduled time, in minutes, from the streetcar ahead of the following streetcar
- Vehicle: Vehicle Number
- Direction: The direction of the bus route where B,b or BW indicates both ways.

Expected patterns in datasets and data transformation

The publicly available API to fetch the Toronto Street delays datasets.

url = "https://ckan0.cf.opendata.inter.prod-toronto.ca/api/3/action/package_show"

There will be six packages corresponding the datasets for the years from 2014 to 2019.

	Report Date	Route	Time	Day	Location	Incident	Min Delay	Min Gap	Direction	Vehicle	Incident ID	Delay	Gap
0	2014-01-02	505	06:31:00	Thursday	Dundas and Roncesvalles	Late Leaving Garage	4.0	8.0	E/B	4018.0	NaN	NaN	NaN
1	2014-01-02	504	12:43:00	Thursday	King and Shaw	Utilized Off Route	20.0	22.0	E/B	4128.0	NaN	NaN	NaN
2	2014-01-02	501	14:01:00	Thursday	Kingston road and Bingham	Held By	13.0	19.0	W/B	4016.0	NaN	NaN	NaN
3	2014-01-02	504	14:22:00	Thursday	King St. and Roncesvalles Ave.	Investigation	7.0	11.0	W/B	4175.0	NaN	NaN	NaN
4	2014-01-02	504	16:42:00	Thursday	King and Bathurst	Utilized Off Route	3.0	6.0	E/B	4080.0	NaN	NaN	NaN

1. The columns Location, Incident, Min Gap, Vehicle, Incident ID, and Gap are dropped.

	Report Date	Route	Time	Day	Min Delay	Direction	Delay
count	77997	77997.000000	77997	77997	74815.000000	77691	3105.000000

2. Regarding the Direction, it has some noisy data, therefore; the top most five frequent values are been chosen.

```
data['Direction'].value_counts()
```

```
W/B    30757
E/B    29949
N/B     5629
S/B     5359
B/W     5204
Name: Direction, dtype: int64
```

3. Regarding the routes, less frequent routes, which might cause noise to the model optimization, are dropped. The routes after filtration are

```
data['Route'].value_counts()
```

```
501      18935
504      14582
506      10474
505       8636
512       5843
510       5025
511       4087
509       2651
502       1561
514       1489
503       1016
301        912
Name: Route, dtype: int64
```

4. As you can see in the figure below, some datasets have the delay named **Min Delay**, and other named **Delay.** All values have been copied to Min Delay column. Delay column is deleted afterward.

	Report Date	Route	Time	Day	Min Delay	Direction	Delay
count	75211	75211.000000	75211	75211	72209.000000	75211	2934.000000
unique	2142	NaN	1444	7	NaN	5	NaN
top	2017-12-28 00:00:00	NaN	21:00:00	Thursday	NaN	W/B	NaN
freq	150	NaN	285	12173	NaN	30074	NaN
first	2014-01-02 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN
last	2019-12-17 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	502.898472	NaN	NaN	12.714523	NaN	14.647239
std	NaN	22.680922	NaN	NaN	29.859279	NaN	39.817765
min	NaN	301.000000	NaN	NaN	0.000000	NaN	0.000000
25%	NaN	501.000000	NaN	NaN	5.000000	NaN	5.000000
50%	NaN	505.000000	NaN	NaN	6.000000	NaN	7.000000

5. The Routes are re indexed and categorized from zero to 11.

```
data["Route"] = data["Route"].astype('category')
data["Route"] = data["Route"].cat.codes
data["Route"].value_counts()
```

```
1     17462
4     13409
6      9387
5      7494
10     5293
8      4545
9      3543
7      2440
11     1488
2      1476
3       900
0       781
Name: Route, dtype: int64
```

6. Days are also re indexed and categorized from zero to 6.

```
data["Day"] = data["Day"].astype('category')
data["Day"] = data["Day"].cat.codes
data["Day"].value_counts()
```

```
4    11123
0    10839
6    10723
5    10693
1    10011
2     7835
3     6994
Name: Day, dtype: int64
```

7. Directions are also re indexed and categorized.

```
data["Direction"] = data["Direction"].astype('category')
data["Direction"] = data["Direction"].cat.codes
data["Direction"].value_counts()
```

```
4    27275
1    26795
2     4918
3     4678
0     4552
Name: Direction, dtype: int64
```

8. Report Date column is dropped, but the day, month, and year have been extracted to new columns.

```
data['Report Year'] = pd.DatetimeIndex(data['Report Date']).year
data['Report Month'] = pd.DatetimeIndex(data['Report Date']).month
data['Report Day'] = pd.DatetimeIndex(data['Report Date']).day
data.drop(['Report Date'],axis=1, inplace=True)
data.tail()
```

	Route	Time	Day	Min Delay	Direction	Report Year	Report Month	Report Day
70575	10	18:18:00	3	4.0	1	2019	3	31
70576	5	18:38:00	3	8.0	4	2019	3	31
70577	5	20:45:00	3	8.0	4	2019	3	31
70578	5	21:30:00	3	8.0	1	2019	3	31
70579	10	04:26:00	1	NaN	1	2019	4	1

9. Time is dropped, and only hours are being extracted, hours from zero to 23 are grouped into four groups each group correspond to 6 hours' time span.

```
new_data['Time'].value_counts()
```

```
1     24689
2     21227
3     12654
0      9635
Name: Time, dtype: int64
```

10. All delays between 4 and 25 has been dropped to avoid the noise that these outliers can have on the model optimization, it is creating like a buffer zone where no delay and delay instances have a gap in between. This turned out to be useful since it added 8% to the testing accuracy finally; all columns have been encoded using one hot encoding.

	Min Delay	Time_0	Time_1	Time_2	Time_3	Report_Month_1	Report_Month_2	Report_Month_3
count	12794.000000	12794.000000	12794.000000	12794.000000	12794.000000	12794.000000	12794.000000	12794.000000
mean	31.384555	0.137799	0.361107	0.310693	0.190402	0.095748	0.083555	0.099656
std	64.452905	0.344702	0.480340	0.462795	0.392633	0.294257	0.276730	0.299553
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	3.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	37.000000	0.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000
max	1400.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

11. New column is been created that correspond to the output. All delays less than 4 is considered no delay.

12. The data set after all previous processing had 12794 example. The training set is 80%, and the testing set is 20%.

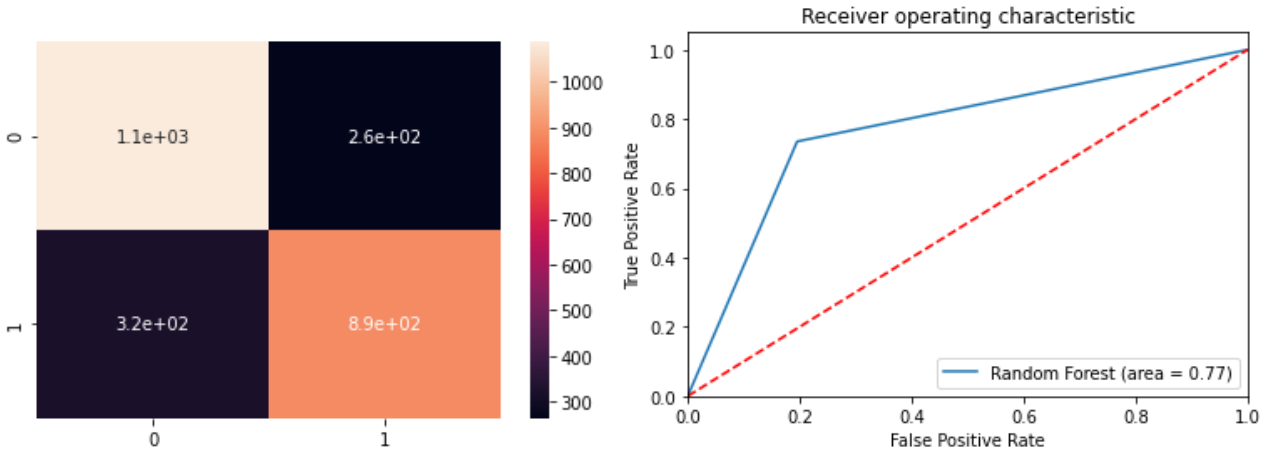
Machine Learning Models

1. Random Forest

The random forest classifier in the sklearn library is used to classify the data sets. The following are the parameter for the random forest classifier.

- bootstrap: True,
- min_samples_leaf: 10,
- n_estimators: 50,
- min_samples_split: 30,
- max_features: sqrt,
- max_depth: 40,
- max_leaf_nodes: None

After fitting the training data in the random forest, the results of the testing set is shown in the figure below. The testing accuracy is 77%.



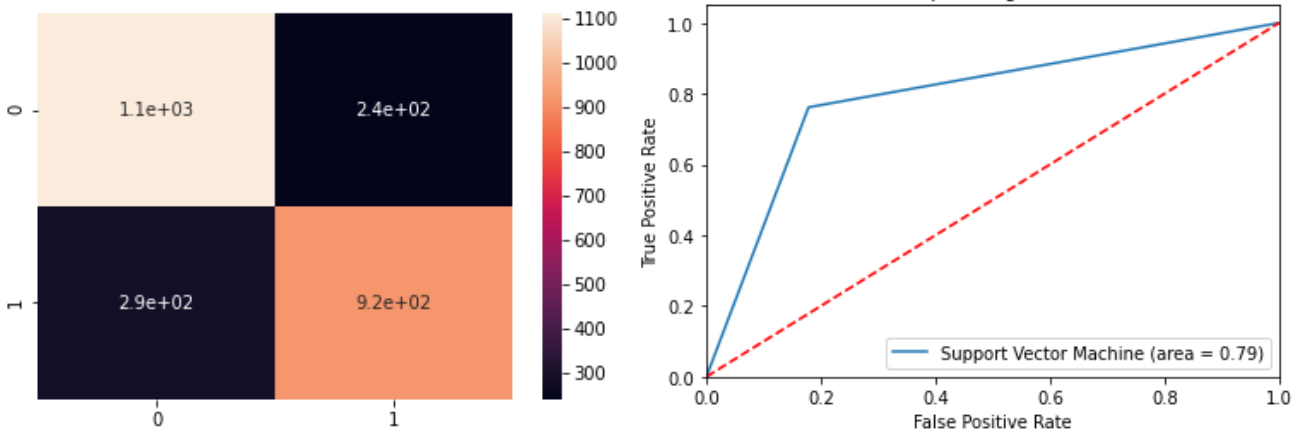
	precision	recall	f1-score	support
0	0.77	0.80	0.79	1351
1	0.77	0.74	0.75	1208
accuracy			0.77	2559
macro avg	0.77	0.77	0.77	2559
weighted avg	0.77	0.77	0.77	2559

2. Support Vector Machine

The support vector machine classifier in the sklearn library is used to classify the data sets. The following are the parameter for the support vector machine.

- C=1.0,
- break_ties=False,
- cache_size=200,
- class_weight=None,
- decision_function_shape=ovr,
- degree=3,
- gamma='scale',
- kernel=rbf,

After fitting the training data in the support vector machine, the results of the testing set is shown in the figure below. The testing accuracy is 79%.



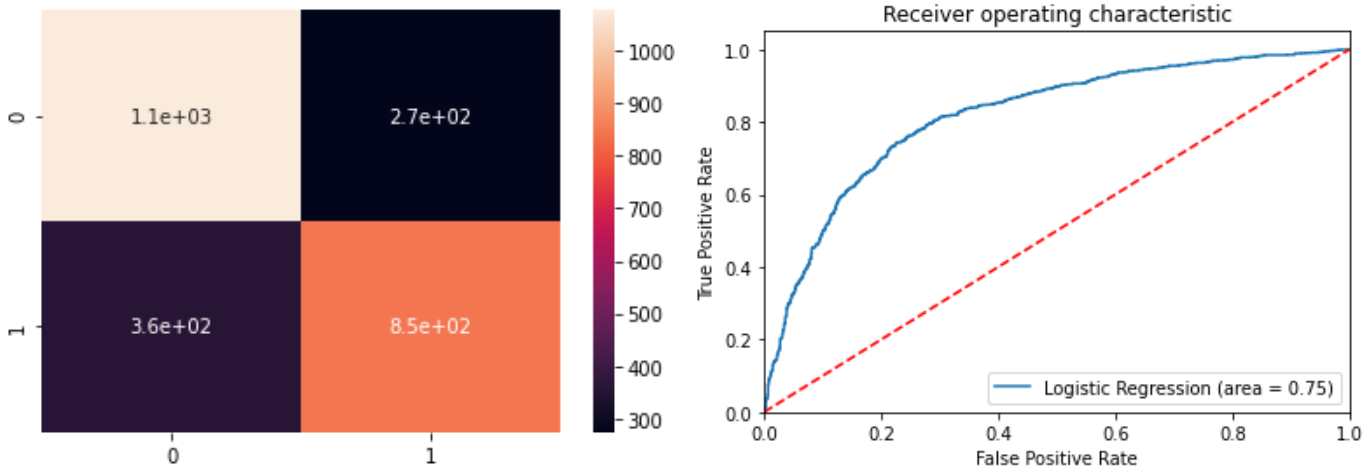
	precision	recall	f1-score	support
0	0.79	0.82	0.81	1351
1	0.79	0.76	0.78	1208
accuracy			0.79	2559
macro avg	0.79	0.79	0.79	2559
weighted avg	0.79	0.79	0.79	2559

3. Logistic Regression

The Logistic Regression in the sklearn library is used to classify the data sets. The following are the parameter for the logistic Regression.

- C=1.0,
- fit_intercept=True
- intercept_scaling=1,
- l1_ratio=None,
- max_iter=100,
- penalty=l2,
- random_state=None,
- solver=lbfgs,
- tol=0.0001

After fitting the training data in the Logistic Regression, the results of the testing set is shown in the figure below. The testing accuracy is 75%.



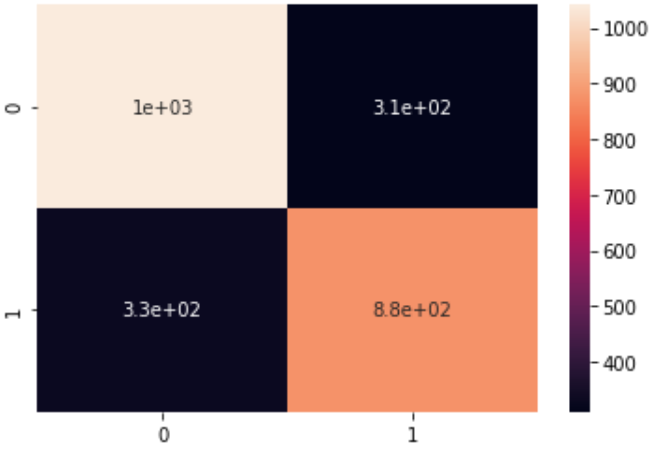
	precision	recall	f1-score	support
0	0.75	0.80	0.77	1351
1	0.76	0.70	0.73	1208
accuracy			0.75	2559
macro avg	0.75	0.75	0.75	2559
weighted avg	0.75	0.75	0.75	2559

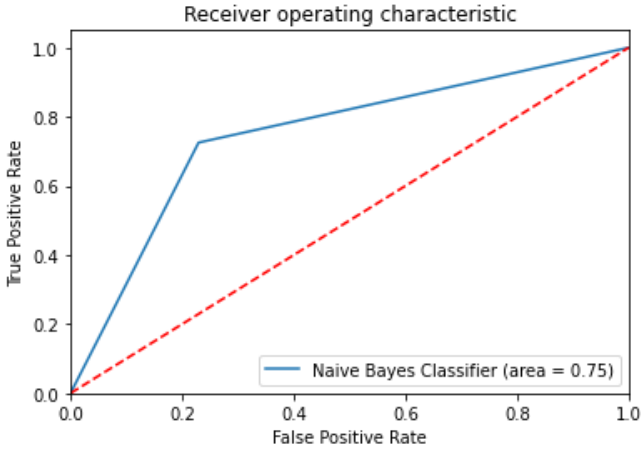
4. Naïve Bayes

The NaïveBayes in the sklearn library is used to classify the data sets. The following are the parameter for the NaïveBayes.

- alpha=1.0,
- class_prior=[0.5, 0.5],
- fit_prior=True
-

After fitting the training data in the NaïveBayes, the results of the testing set is shown in the figure below. The testing accuracy is 75%.





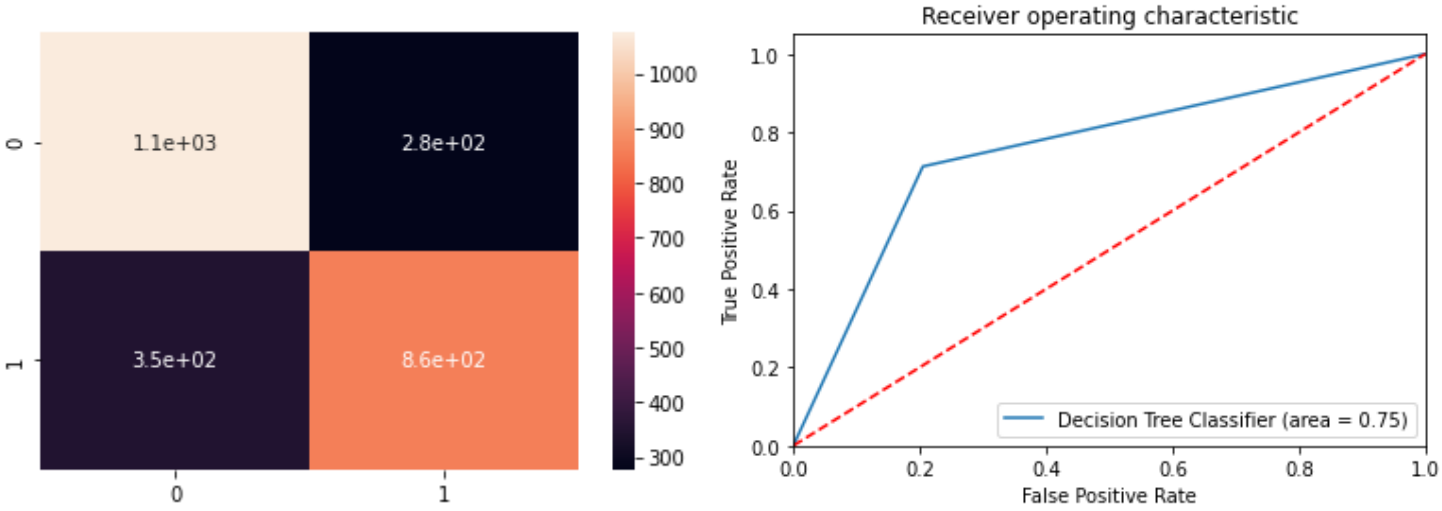
	precision	recall	f1-score	support
0	0.76	0.77	0.76	1351
1	0.74	0.73	0.73	1208
accuracy			0.75	2559
macro avg	0.75	0.75	0.75	2559
weighted avg	0.75	0.75	0.75	2559

5. Decision Tree Classifier

The Decision Tree Classifier in the sklearn library is used to classify the data sets. The following are the parameter for the decision Tree Classifier.

- ccp_alpha=0.0,
- criterion='entropy',
- max_depth=20,
- min_samples_leaf=1,
- min_samples_split=2,
- min_weight_fraction_leaf=0.0,
- random_state=None,
- splitter=best

After fitting the training data in the Decision Tree Classifier, the results of the testing set is shown in the figure below. The testing accuracy is 76%.



	precision	recall	f1-score	support
0	0.76	0.79	0.77	1351
1	0.76	0.71	0.73	1208
accuracy			0.76	2559
macro avg	0.76	0.75	0.75	2559
weighted avg	0.76	0.76	0.76	2559

6. K Nearest Neighbors

The K Nearest Neighbors in the sklearn library is used to classify the data sets. The following are the parameter for the K Nearest Neighbors.

- algorithm='auto',
- leaf_size=30,
- metric='minkowski',
- metric_params=None,
- n_jobs=None,
- n_neighbors=100,
- p=2,
- weights='uniform'
-

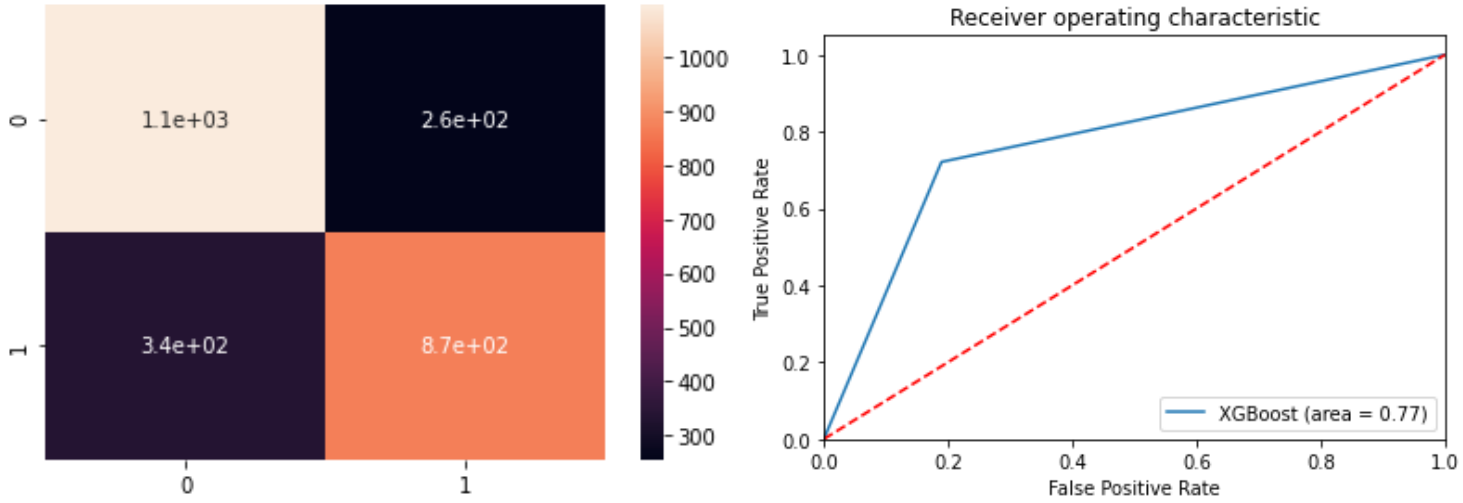
After fitting the training data in the K Nearest Neighbors, the results of the testing set is shown in the figure below. The testing accuracy is **74%**.

7. XGBoost

The XGBoost in the xgboost library is used to classify the data sets. The following are the parameter for the XGBoost.

- base_score=0.5,
- booster=gbtree,
- learning_rate=0.1,
- max_depth=3,
- n_estimators=100,
- n_jobs=1,
- objective=binary:logistic,

After fitting the training data in the XGBoost, the results of the testing set is shown in the figure below. The testing accuracy is 77%.



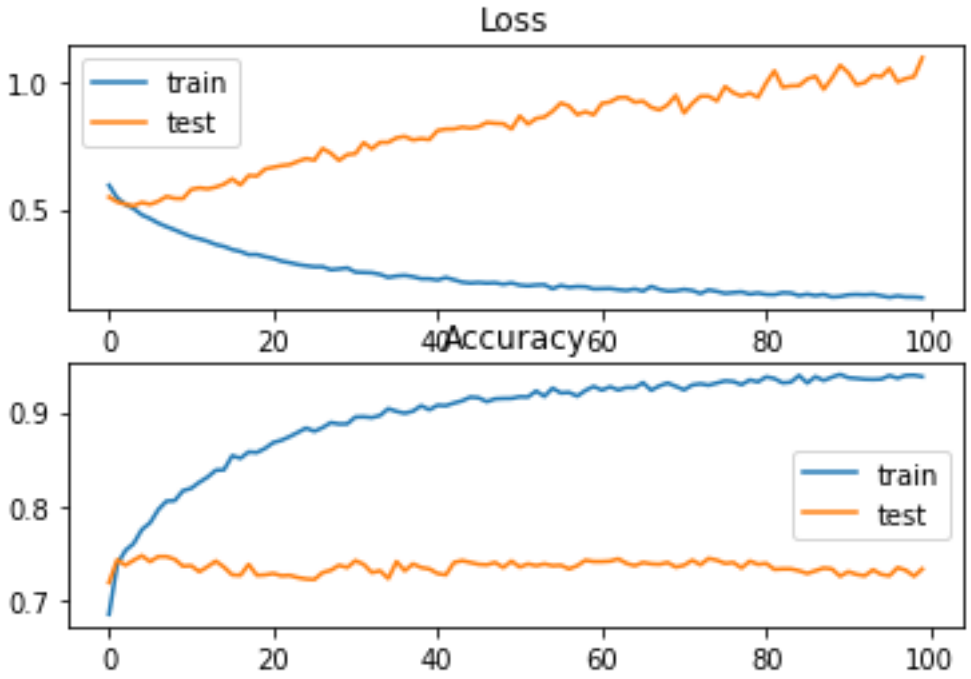
	precision	recall	f1-score	support
0	0.76	0.81	0.79	1351
1	0.77	0.72	0.75	1208
accuracy			0.77	2559
macro avg	0.77	0.77	0.77	2559
weighted avg	0.77	0.77	0.77	2559

8. Neural Network

The neural network from the keras implementation of the tensor flow framework is used to classify the data sets. The following are the parameter for the neural network.

- Number of layer = 4,
- Input shape=(1,77),
- learning rate=0.0001,
- Optimizer=Adam,
- objective= sparse_categorical_crossentropy,

After fitting the training data in the neural network, the results of the testing set is shown in the figure below. The testing accuracy is 74%. The model is overfitting, not learning. Over many attempt by changing the learning rate, the optimizer, the network architecture, and the number of neurons in each layer. The model is not learning, nor generalizing. Finally; the keras tuner package have been used to randomly search for a model that fit the data and learn how to classify based on delay. Again, no improvement at all. Below is the figure of the training and testing loss and accuracy. It is clear that the model is not learning. I would like to say at the end that in the next page I would show how the features are affecting the output; maybe, we can get some insights why the model is not learning.



Results analysis

The figure blow shows the feature scores before and after filtering the data or the addition of the gap between the no delay and delay instances by removing delays between 4 and 25. The left hand side is before, and the right hand side is after filtration. The first finding is that the feature (**Time_0**) has disappeared from the top 20 features affecting the output. Feature (**Time_0**) represent the time span from (00:00 to 06:00 am) which is obvious that the delay will be minimum at that time since no body is driving at night. The second thing to notice is that the (**Direction**) feature vector starts to play a role in the output as we can see three out of four direction vector are present with high score compare to one with high score before filtration. Moreover, the report year should have the least effect, as we want the model to work in every year. The year 2018 had some effect in the output before processing, but after that, we can see that report year is down in the list. Finally, the day features are going up in the list too, which make sense since working days will have more delays compared to holidays. This clearly shows that if more processing time were spent on the data, more techniques to clear and transform the data, the results of the models would be better.

72	Direction_0
62	Route_2
71	Route_11
3	Time_3
63	Route_3
56	Day_3
60	Route_0
1	Time_1
70	Route_10
0	Time_0
69	Route_9
55	Day_2
68	Route_8
67	Route_7
51	Report_Year_2018
74	Direction_2
58	Day_5
2	Time_2
64	Route_4
4	Report_Month_1

72	Direction_0
61	Route_1
70	Route_10
68	Route_8
1	Time_1
3	Time_3
65	Route_5
56	Day_3
74	Direction_2
60	Route_0
55	Day_2
71	Route_11
75	Direction_3
2	Time_2
67	Route_7
64	Route_4
66	Route_6
62	Route_2
49	Report_Year_2016
15	Report_Month_12