DOKUZ EYLUL UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

Delay Estimation and Shortest Flight Path Analysis

By

2014510028 Çağrı Anıl ERBEY

2017510072 Hüseyin Emrecan TAN

2017510076 Oğuzhan TÜRKMEN

April, 2022

İZMİR

• Implementation

In the dataset we use in our application, there is flight information in the cities in America. Within the information in the dataset, there is also the information that the flights have been delayed as a result of some situations. The “ArrTime15 – DepTime 15” features in our dataset tell us about the delay times experienced in these flights. These features are transformed into information whether a flight is delayed more than 15 minutes by setting the 15-minute condition. Thus, it is a "Data Mining" application that determines whether there will be a 15-minute delay in possible flights."

While developing the application, the information in the dataset was first read in the preprocess part. There are 50 features and 72% of them were deleted because they did not contain enough information for the application to be developed. The deleted features were combined and made more meaningful and a new feature was added. For the data that is "null" in the features, "null" values are filled in a certain logic with the KNN approach. 1 instance that may adversely affect the application data has been deleted in the feature with city information. Outlier data in the "FlightLength", which is planned to be used as a target feature, has been cleaned. Thus, we will conclude that “Accuracy, Precision, This cleaning process is performed in order to get the “Recall and F1-Score” values more efficiently. Finally, "KNN - Logistic Regression - Support Vector Machine" models were applied to the dataset in order to obtain the "Accuracy, Precision, Recall and F1-Score" values, and the output of this modeling was taken.

Each of the cities in the dataset is considered as a "node". The “Edge” information between two “Nodes” was determined and calculated as the flight time of the aircraft in order to find the shortest path. Thus, the dataset was turned into a "graph". It has been made into a shortest route application based on the airtime between the two specified cities, and this is a "Graph Theory" application.

While developing the application, our goal is to find the shortest path between two cities. For this, there must be a "Source" and a "Destination" data in the application. A city is selected from the "OriginCityName" feature. This city becomes the “Source” data. A city is selected from the "DestCityName" feature. This city becomes "Destination". In order to find the shortest path, there should also be a "Weight" data. “FlightLength” data is used for this data. With these data, the shortest path was tried to be found by using “Bellman-Ford Algorithm, Dijkstra's Algorithm, A* search Algorithm”. In the event that any city is closed to flights, our application finds an alternative shortest route.

As a result, this project is both a "Data Mining" and a "Graph Theory" application.

• Results

The dataset contains the properties (feature) “DayofMonth - DayOfWeek - DepTimeBlk -ArrTimeBlk - Cancelled - CRSElapsedTime - FlightLength - Distance - NewOriginCityName - NewDestCityName - NewDestStateName - NewOriginStateName - NewUniqueCarrier - Del15”. There are 46130 instances. The data type of the features is "int32".

In the "Data Mining" section of the application, it can predict 89% of the time whether a new flight will have a 15-minute delay over the Del15 feature. We have determined that the highest accuracy values are “Logistic Regression” and “90% train, 10% test” values. We determined that this is the most appropriate model for this dataset.

In the "Graph Theory" section of the application, a city is determined from the "OriginCityName" and "DestCityName" features. The approximate flight time of a flight between these cities is determined by the "FlightLength" feature. By determining the shortest route between the two specified cities, the most suitable route is determined for the customers. As a result of the deletion of any city from the flight route, it finds the shortest route again and determines a suitable route for the customers. The shortest path algorithms “Bellman-Ford Algorithm, Dijkstra's Algorithm, A* search Algorithm” are used. The application works correctly for all three algorithms.

• Performance

One of the points we want to reach as a goal in the "Data Mining" part of the project is to reach the "Accuracy, Precision, Recall and F1-Score" values. For this, 3 different models were applied to the dataset. These are “KNN – Logistic Regression – Support Vector Machine” models. In addition, the model should be tested and applied as “90% train, 10% test” and “n-fold cross validation (n=10)” on the dataset.

K is the number of neighborhood points we would take to decide in which class our test data belongs.K should not be taken in even number it should be odd i.e:- 1,3,5,7,9. But the question is how we take best value of K for prediction of test data belongings.

Output will be the list of average value of error rate for each iteration of k.

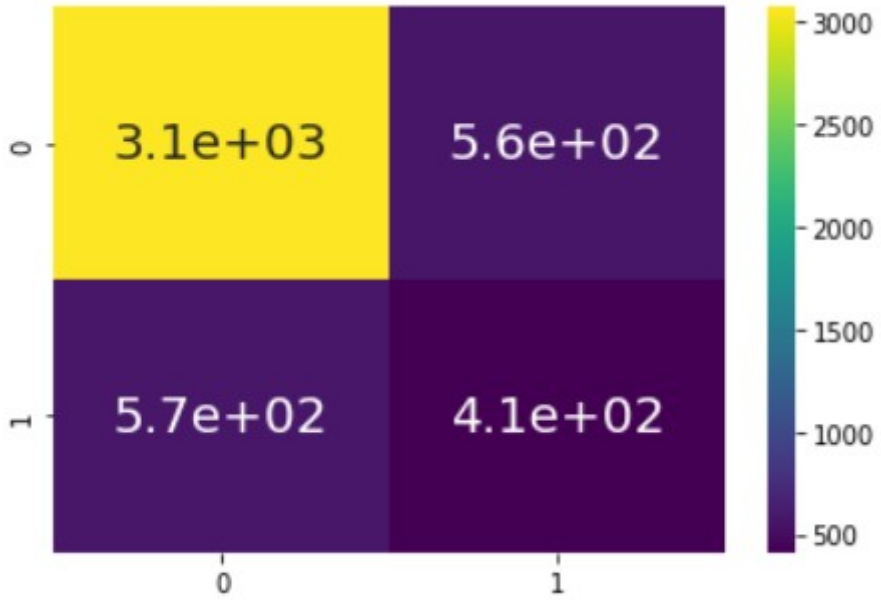


Figure 1 - Before Applying Best Fit K Value

As we see in Figure 2 after k=10 the fluctuation in error rate is not much . Hence whenever we will see the threshold value after which k-value is not fluctuating more we will select that specific threshold value as k value.

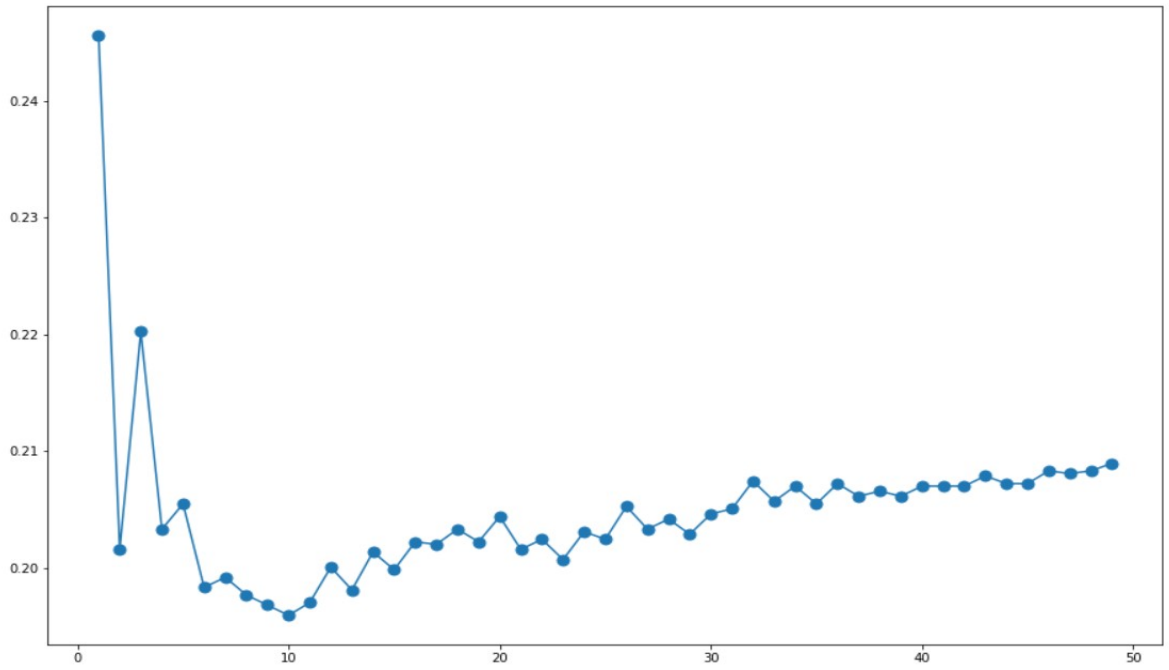


Figure 2 - Finding Best Fit K Value

By comparing both the scenario we can observe that at optimal k value we are able improve the model accuracy.

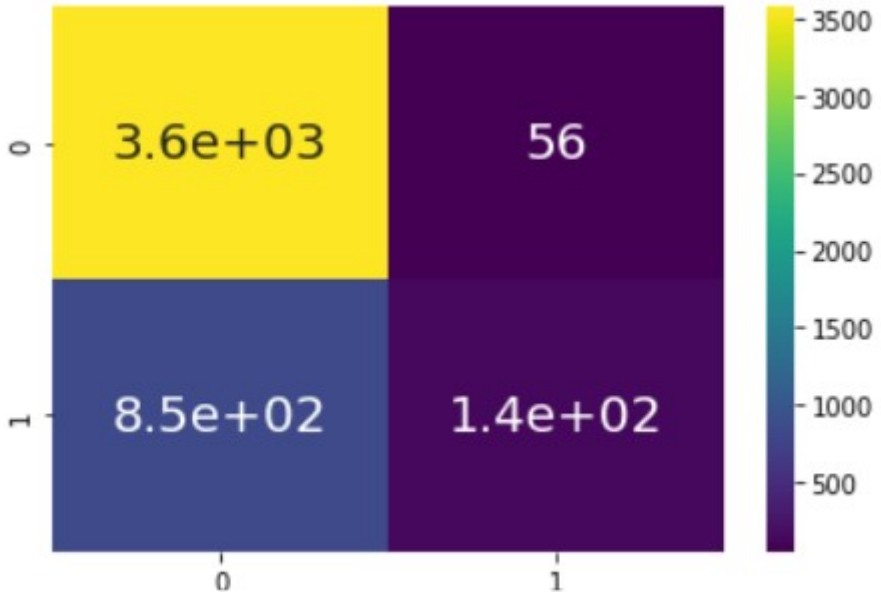


Figure 3 - After Applying Best Fit K Value

K-Nearest Neighbors Algorithm and "%90 train, %10 test"

	precision	recall	f1-score	support
0	0.82	0.95	0.88	3627
1	0.57	0.26	0.35	986
accuracy			0.80	4613
macro avg	0.70	0.60	0.62	4613
weighted avg	0.77	0.80	0.77	4613

Figure 4

K-Nearest Neighbors Algorithm and “n-fold cross validation (n=10)”

```
cv = KFold(n_splits=10, random_state=1, shuffle=True)
# create model
model = KNeighborsClassifier()
# evaluate model
scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)
# report performance
print('Accuracy: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

```
Accuracy: 0.787 (0.003)
```

Figure 5

Logistic Regression Algorithm and “%90 train, %10 test”

	precision	recall	f1-score	support
0	0.91	0.96	0.93	3627
1	0.81	0.64	0.71	986
accuracy			0.89	4613
macro avg	0.86	0.80	0.82	4613
weighted avg	0.89	0.89	0.89	4613

Figure 6

Logistic Regression Algorithm and “n-fold cross validation (n=10)”

```
cv = KFold(n_splits=10, random_state=1, shuffle=True)
# create model
model = LogisticRegression()
# evaluate model
scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)
# report performance
print('Accuracy: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

```
Accuracy: 0.865 (0.014)
```

Figure 7

Support Vector Machine Algorithm and “%90 train, %10 test”

	precision	recall	f1-score	support
0	0.79	1.00	0.88	3627
1	0.00	0.00	0.00	986
accuracy			0.79	4613
macro avg	0.39	0.50	0.44	4613
weighted avg	0.62	0.79	0.69	4613

Figure 8

Support Vector Machine Algorithm and “n-fold cross validation (n=10)”

```
cv = KFold(n_splits=10, random_state=1, shuffle=True)
# create model
model = svm.SVC()
# evaluate model
scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)
# report performance
print('Accuracy: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

```
Accuracy: 0.781 (0.007)
```

Figure 9

As a result, "KNN - Logistic Regression - Support Vector Machine" models were applied. Among these models, “Logistic Regression” was the most efficient model, especially “Accuracy” in general.

“90% train, 10% test” and “n-fold cross validation (n=10) ” should be tested and the model should be applied. As a result of these processes, it was found that the "90% train, 10% test" option, especially "Accuracy", was more efficient.

• UI with sample screenshots

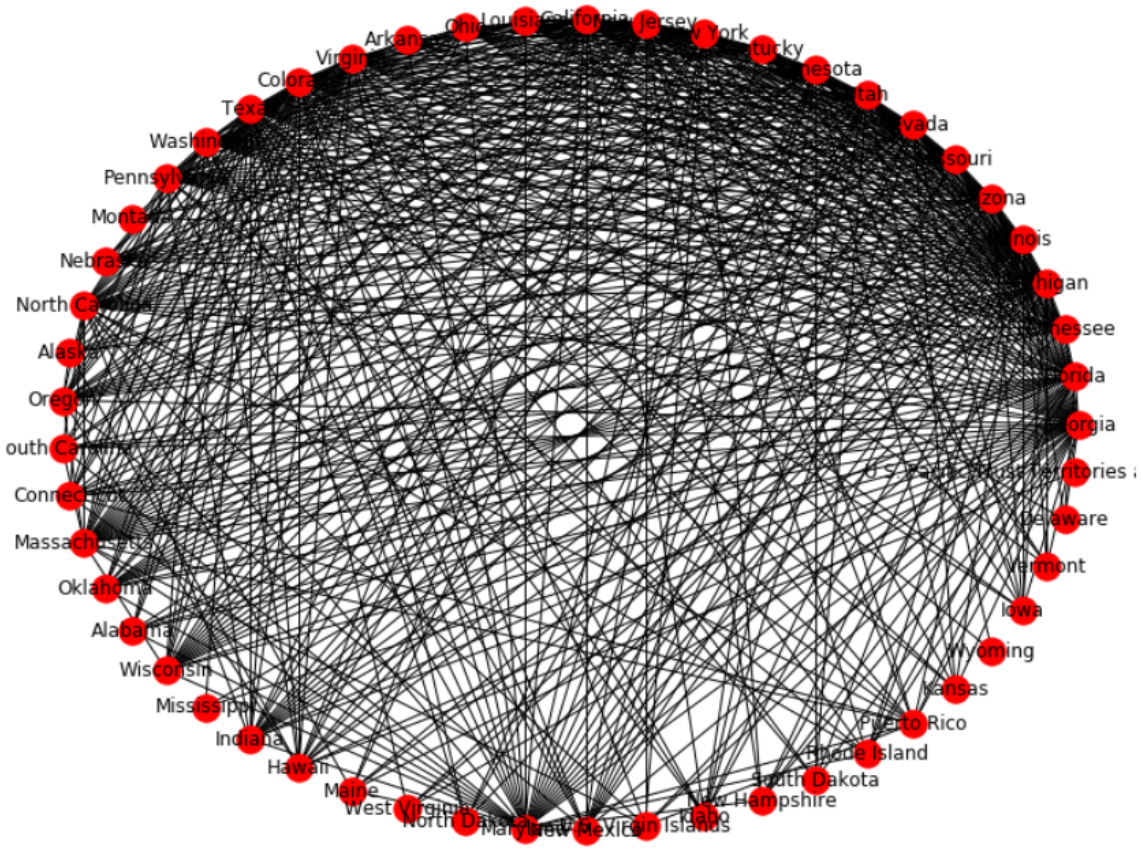


Figure 10 Graph without Distance

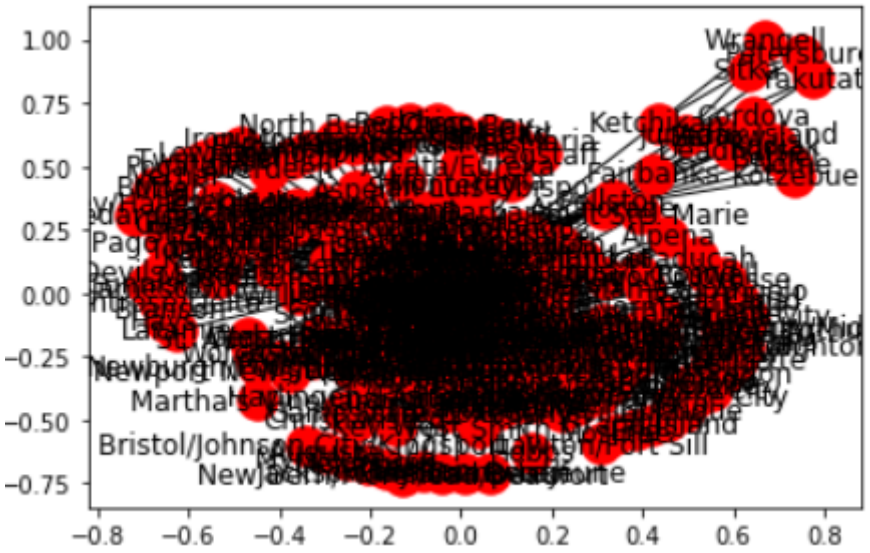


Figure 11 Graph Density

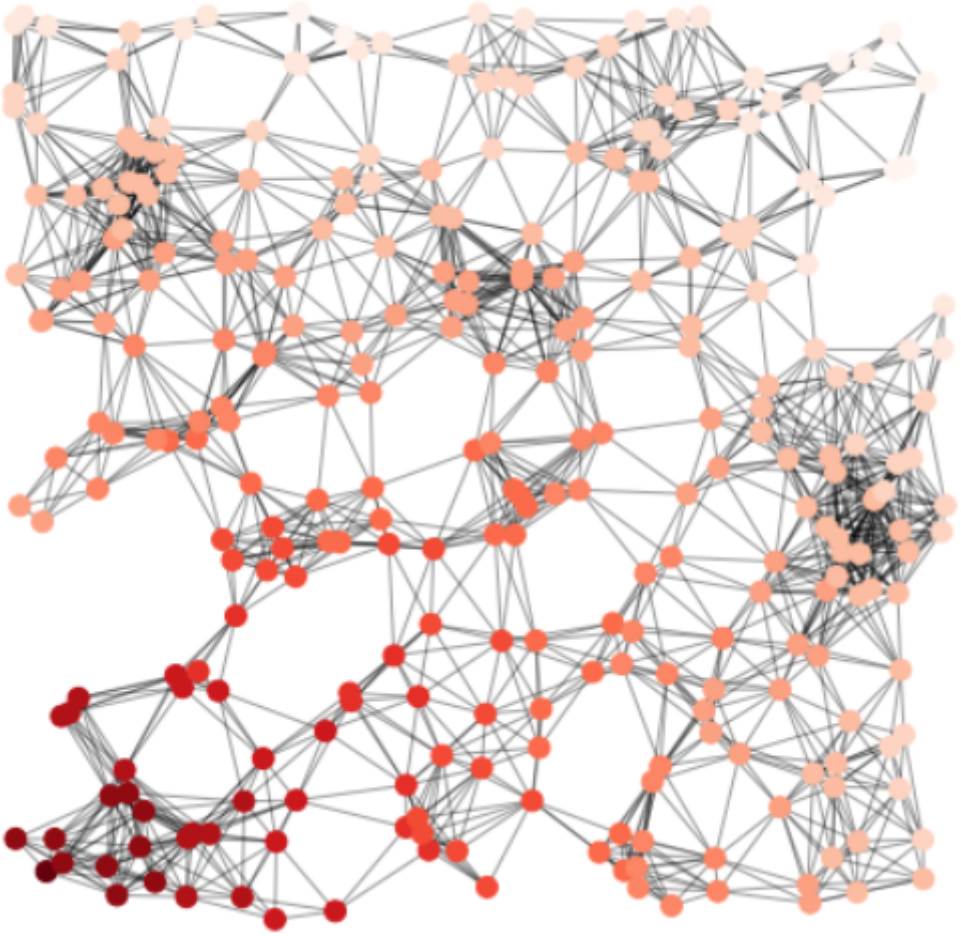
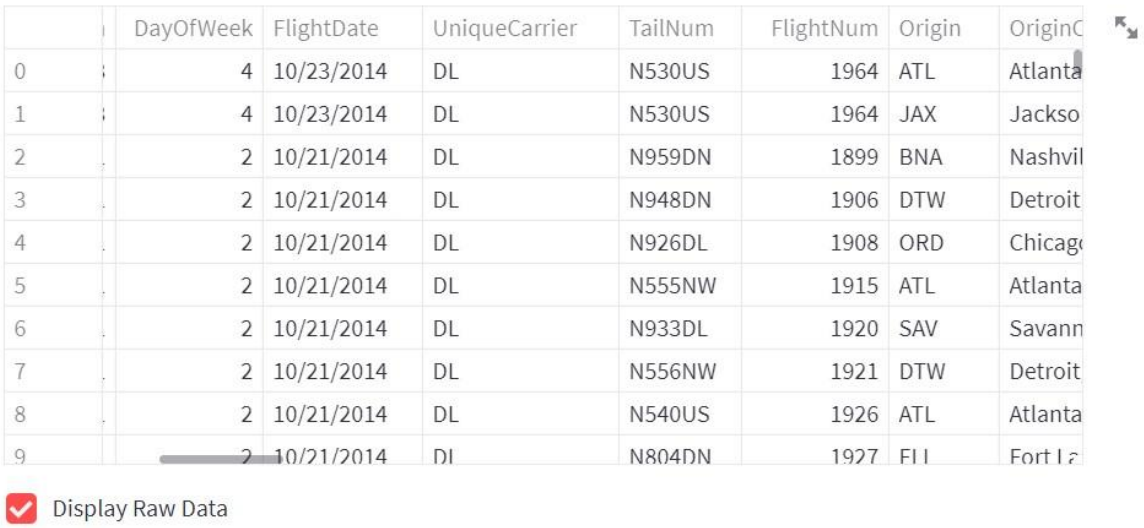


Figure 12 Geometric Graph with Shortest Path

Here is the dataset used in this analysis:



	DayOfWeek	FlightDate	UniqueCarrier	TailNum	FlightNum	Origin	OriginC
0	4	10/23/2014	DL	N530US	1964	ATL	Atlanta
1	4	10/23/2014	DL	N530US	1964	JAX	Jackso
2	2	10/21/2014	DL	N959DN	1899	BNA	Nashvil
3	2	10/21/2014	DL	N948DN	1906	DTW	Detroit
4	2	10/21/2014	DL	N926DL	1908	ORD	Chicag
5	2	10/21/2014	DL	N555NW	1915	ATL	Atlanta
6	2	10/21/2014	DL	N933DL	1920	SAV	Savann
7	2	10/21/2014	DL	N556NW	1921	DTW	Detroit
8	2	10/21/2014	DL	N540US	1926	ATL	Atlanta
9	2	10/21/2014	DL	N804DN	1927	FLL	Fort La

Display Raw Data

Figure 13 Dataset Display in User Interface