ICS 474: PROJECT

Date of submission: 16-12-2022

Ali Alnujaidi 201839260

ICS 474: Project

Due: Saturday Week 14

Use Spark Machine Learning library to complete all the three parts of this project.

Part A: Clustering - 7%

1. **[2 pts]** Find a dataset in kaggle or any other source. Make sure that each dataset is at least 500 MB.

   Found this dataset:
   https://www.kaggle.com/datasets/robikscube/flight-delay-dataset-20182022?select=Combined_Flights_2022.csv
   And I will use the dataset: **Combined_Flights_2022 (1.42 GB)**

2. **[2 pt]** Write a detailed description of the dataset.

   This dataset contains all flight information including cancellation and delays by airline for dates back to January 2018.
   I will use the airtime, distance and DepTime to cluster the data using k-means

3. **[6 pt]** Preprocess the dataset.

   First, I dropped all rows with null value.

   Then, I selected these three columns to cluster the data and I minimize the data because it takes too long time if I didn't.

```
df = df.select("AirTime","Distance","DepTime")
```

   And this is the new data set like:

```
+-------+--------+-------+
|AirTime|Distance|DepTime|
+-------+--------+-------+
|   40.0|   212.0| 1123.0|
|   55.0|   295.0|  728.0|
|   47.0|   251.0| 1514.0|
|   57.0|   376.0| 1430.0|
|   49.0|   251.0| 1135.0|
+-------+--------+-------+
```

Then I dropped any null value and cast the three columns into integers:

```
df = df.withColumn("AirTime",col("AirTime").cast("int")).dropna("any")
df = df.withColumn("Distance",col("Distance").cast("int")).dropna("any")
df = df.withColumn("DepTime",col("DepTime").cast("int")).dropna("any")
```

Then I took the max number of each column then divided it by its columns to make the columns values between 0-1.

```
df = df.withColumn("AirTime",col("AirTime").cast("int")).dropna("any")
df = df.withColumn("Distance",col("Distance").cast("int")).dropna("any")
df = df.withColumn("DepTime",col("DepTime").cast("int")).dropna("any")
```

```
df.describe().show()
```

```
+-------+------------------+-----------------+-----------------+
|summary|           AirTime|         Distance|          DepTime|
+-------+------------------+-----------------+-----------------+
|  count|           3944916|          3944916|          3944916|
|   mean|111.00754870319165|798.8928636249796|1334.120380763494|
| stddev| 69.96245895307061|593.1635201368039|505.7184102507513|
|    min|                 8|               31|                1|
|    max|               727|             5095|             2400|
+-------+------------------+-----------------+-----------------+
```

```
df = df.withColumn("AirTime", df.AirTime/727)
df = df.withColumn("Distance", df.Distance/5095)
df = df.withColumn("DepTime", df.DepTime/2400)
```

```
df.describe().show()
```

```
+-------+--------------------+--------------------+--------------------+
|summary|             AirTime|            Distance|             DepTime|
+-------+--------------------+--------------------+--------------------+
|  count|             3944916|             3944916|             3944916|
|   mean| 0.15269263920660694| 0.15679938442095578|  0.5558834919849465|
| stddev| 0.09623446898634216|  0.1164207105273417| 0.21071600427114648|
|    min|0.011004126547455296|0.006084396467124632|4.166666666666667E-4|
|    max|                 1.0|                 1.0|                 1.0|
+-------+--------------------+--------------------+--------------------+
```

Then I used the vector to start the ML model.

```
from pyspark.ml.linalg import Vector
from pyspark.ml.feature import VectorAssembler
input_cols = ['AirTime','Distance','DepTime']
vec_assembler = VectorAssembler(inputCols=input_cols, outputCol="features")
df = vec_assembler.transform(df)
```

4. **[8 pts]** Using K-means algorithm to cluster the dataset.

```
from pyspark.ml.clustering import KMeans
from pyspark.ml.evaluation import ClusteringEvaluator
```

```
kmeans = KMeans().setK(4).setSeed(1)
model = kmeans.fit(df)
```

```
predictions = model.transform(df)
```

```
predictions.show()
```

```
+--------------------+--------------------+------------------+--------------------+----------+
|             AirTime|            Distance|           DepTime|            features|prediction|
+--------------------+--------------------+------------------+--------------------+----------+
|0.055020632737276476|0.041609421000981354|0.46791666666666665|[0.05502063273727...|         1|
| 0.07565337001375516| 0.05789990186457311|0.30333333333333334|[0.07565337001375...|         3|
| 0.06464924346629987|0.049263984298331696| 0.6308333333333334|[0.06464924346629...|         1|
| 0.07840440165061899| 0.07379784102060843| 0.5958333333333333|[0.07840440165061...|         1|
| 0.06740027510316368|0.049263984298331696|0.47291666666666665|[0.06740027510316...|         1|
| 0.10591471801925723| 0.10618253189401373|0.39666666666666667|[0.10591471801925...|         3|
| 0.03576341127922971| 0.02492639842983317|              0.89|[0.03576341127922...|         0|
| 0.18707015130674004| 0.15132482826300295|0.46541666666666665|[0.18707015130674...|         1|
| 0.05089408528198074| 0.03729146221786065| 0.5891666666666666|[0.05089408528198...|         1|
| 0.08253094910591471| 0.08380765456329735|0.39458333333333334|[0.08253094910591...|         3|
|0.061898211829436035|0.045142296368989206| 0.6441666666666667|[0.06189821182943...|         1|
|  0.1155433287482806| 0.08380765456329735|0.29333333333333333|[0.11554332874828...|         3|
| 0.09491059147180192|  0.0830225711481845| 0.5658333333333333|[0.09491059147180...|         1|
|  0.0811554332874828|  0.0775269872423945|              0.42|[0.08115543328748...|         1|
| 0.14167812929848694| 0.11030421982335623| 0.5229166666666667|[0.14167812929848...|         1|
| 0.05364511691884457|0.041609421000981354| 0.3904166666666667|[0.05364511691884...|         3|
|0.061898211829436035|0.049263984298331696| 0.5508333333333333|[0.06189821182943...|         1|
|0.061898211829436035|0.059470068694798824| 0.4633333333333333|[0.06189821182943...|         1|
| 0.16368638239339753| 0.15544651619234542|              0.75|[0.16368638239339...|         0|
| 0.07427785419532325| 0.08380765456329735| 0.8141666666666667|[0.07427785419532...|         0|
+--------------------+--------------------+------------------+--------------------+----------+
only showing top 20 rows
```

```
evaluator = ClusteringEvaluator()
```

```
silhouette = evaluator.evaluate(predictions)
print("Silhouette with squared euclidean distance = " + str(silhouette))
```

```
Silhouette with squared euclidean distance = 0.5307547787592565
```

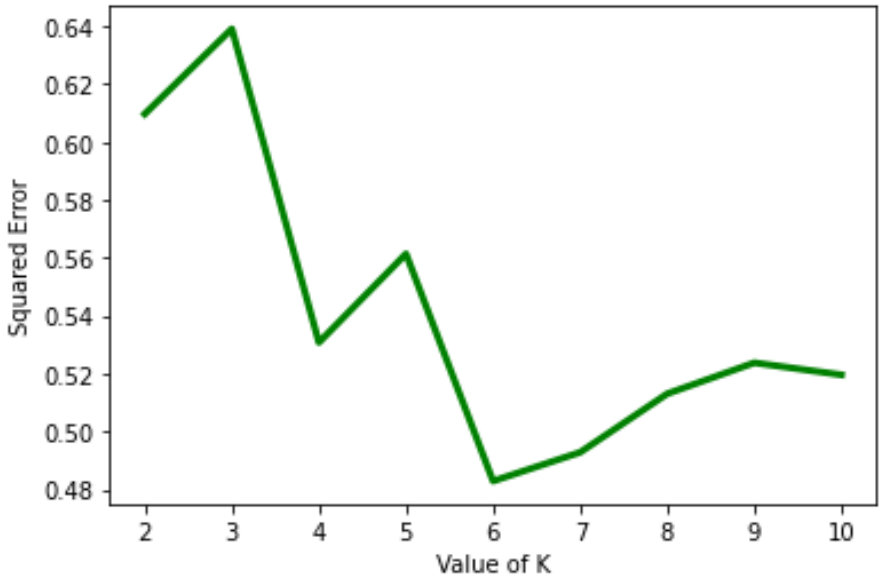
```
centers = model.clusterCenters()
print("Cluster Centers: ")
for center in centers:
    print(center)
```

```
Cluster Centers:
[0.12571497 0.12432134 0.79310947]
[0.11827202 0.11452456 0.53378847]
[0.35967546 0.40935606 0.5822787 ]
[0.14107441 0.14243007 0.30415446]
```

5. **[4 pts]** Use the Elbow method and the Silhouette method to find the optimal K.

```
import matplotlib.pyplot as plt
cost =[]
for i in range(2, 11):
    kmeans = KMeans().setK(i).setSeed(1)
    model = kmeans.fit(df)
    predictions = model.transform(df)
    silhouette = evaluator.evaluate(predictions)
    cost.append(silhouette)


plt.plot(range(2, 11), cost, color ='g', linewidth ='3')
plt.xlabel("Value of K")
plt.ylabel("Squared Error")
plt.show()
```



The optimal value is both 4 or 6.

Part B: Regression - 7%

1. **[2 pts]** Find one or two datasets in Kaggle or any other source. Make sure that each dataset is at least 500 MB.

   Found this dataset:
   https://www.kaggle.com/datasets/phamtheds/predict-flight-delays?select=Predict_Flight_Delays_2022_M1_M4.csv
   And I will use the dataset: **Predict_Flight_Delays_2022_M1_M4.csv (654.96 MB)**

2. **[2 pts]** Write a detailed description of each dataset.

   This dataset contains all flight information including cancellation and delays by airline.
   I will use the Depart time, depart delay, airtime and distance to build regression model.

3. **[6 pts]** Preprocess each dataset.

First, I dropped all rows with null value.

```
df = df.withColumn("DEP_TIME",col("DEP_TIME").cast("double")).dropna("any")
df = df.withColumn("DEP_DELAY",col("DEP_DELAY").cast("double")).dropna("any")
df = df.withColumn("AIR_TIME",col("AIR_TIME").cast("double")).dropna("any")
df = df.withColumn("DISTANCE",col("DISTANCE").cast("double")).dropna("any")
```

Then I used the vector to start the ML model.

```
from pyspark.ml.linalg import Vector
from pyspark.ml.feature import VectorAssembler
input_cols = ['DEP_TIME','DISTANCE','AIR_TIME']
assembler = VectorAssembler(inputCols=input_cols, outputCol='features')
df = assembler.transform(df)
```

```
df = df.select("features","DEP_DELAY")
```

4. **[2 pts]** Divide each dataset into training and testing.

```
(train_data, test_data) = df.randomSplit([0.7, 0.3])
```

5. **[12 pts]** Build two regression models.

First regression model I will use linear regression.

```
(train_data, test_data) = df.randomSplit([0.7, 0.3])
```

```
from pyspark.ml.regression import LinearRegression
```

```
from pyspark.ml.evaluation import RegressionEvaluator
```

```
reg = LinearRegression(labelCol = "DEP_DELAY")
```

```
model = reg.fit(train_data)
```

```
res = model.evaluate(test_data)
```

```
unlabeled_data = test_data.select("features")
```

```
predictions = model.transform(unlabeled_data)
```

```
predictions.show(5)
```

```
+-----------------+-------------------+
|         features|         prediction|
+-----------------+-------------------+
| [1.0,150.0,32.0]|-5.2567932199345355|
| [1.0,152.0,28.0]| -5.278999231410023|
| [1.0,351.0,53.0]| -4.843497116322919|
| [1.0,391.0,58.0]|  -4.75611611217702|
|[1.0,888.0,108.0]| -3.746224223406286|
+-----------------+-------------------+
only showing top 5 rows
```

second regression model I will use Factorization machines regressor

```
from pyspark.ml.regression import FMRegressor
```

```
from pyspark.ml.evaluation import RegressionEvaluator
```

```
reg = FMRegressor(labelCol = "DEP_DELAY")
```

```
model = reg.fit(train_data)
```

```
predictions = model.transform(test_data)
```

```
unlabeled_data = test_data.select("features")
```

```
predictions.show(5)
```

```
+-----------------+---------+-------------------+
|         features|DEP_DELAY|         prediction|
+-----------------+---------+-------------------+
| [1.0,150.0,32.0]|    176.0|-2475.7978944209754|
| [1.0,351.0,53.0]|     66.0| -9265.678590179945|
| [1.0,395.0,56.0]|    116.0|-10985.361222069128|
| [1.0,602.0,84.0]|    116.0|-24912.176027404843|
|[1.0,888.0,108.0]|    196.0| -47015.50104131433|
+-----------------+---------+-------------------+
only showing top 5 rows
```

6. **[4 pts]** Test the models and compute their accuracy.

Accuracy for the first model:

```
print("MAE Reg model: ", res.meanAbsoluteError)
print("MSE Reg model: ", res.meanSquaredError)
print("RMSE Reg model: ", res.rootMeanSquaredError)
print("R2 Reg model:", res.r2)
print("Adj R2 Reg model:", res.r2adj)
```

```
MAE Reg model:  22.243838985084892
MSE Reg model:  2637.0922212583196
RMSE Reg model:  51.35262623526006
R2 Reg model: 0.013693216594809154
Adj R2 Reg model: 0.013688456013371542
```

Accuracy for the second model:

```
evaluator = RegressionEvaluator(
    labelCol="DEP_DELAY", predictionCol="prediction", metricName="rmse")
rmse = evaluator.evaluate(predictions)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)
```

```
Root Mean Squared Error (RMSE) on test data = 749040
```

Part C: Classification - 6%

1. **[2 pts]** Find one or two datasets in kaggle or any other source. Make sure that each dataset is at least one 500MB.

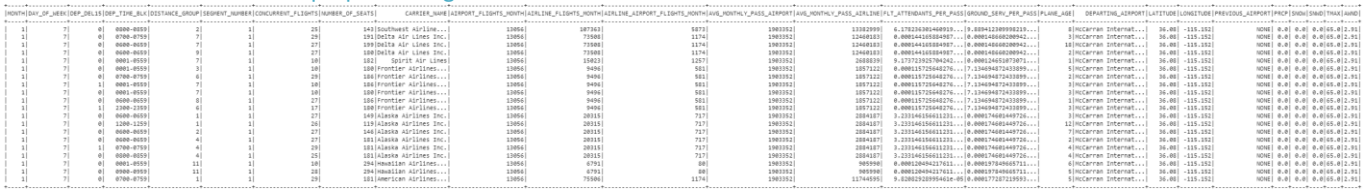
Found this dataset:
https://www.kaggle.com/datasets/threnjen/2019-airline-delays-and-cancellations?select=full_data_flightdelay.csv

And I will use the dataset: **full_data_flightdelay.csv (1.37 GB)**

2. **[2 pts]** Write a detailed description of each dataset.

This is a classification dataset with detailed airline, weather, airport and employment information. All raw data files are also included for customization of the dataset, including adding cancellation, specific delay reasons, and/or arrival delays in order to create a multiclass problem. Note: Raw files for weather include only the top 90% of airports for passenger traffic, as all weather data was downloaded manually. And I will use this data set to predict if the flight will be delayed or not

And this is the dataset before preprocessing



3. **[6 pts]** Preprocess each dataset.

A. For the first classification model:

I Will select these columns to predict if the flight will delay more than 15 min or not.

(DEP_DEL15) 1 = Yes, 0 =No

```
df = df.select('CONCURRENT_FLIGHTS','PLANE_AGE','DEPARTING_AIRPORT','PRCP','SNOW','SNWD','TMAX','AWND','DEP_DEL15')
```

```
df.show()
```

```
+-----------------+---------+-------------------+----+----+----+----+----+---------+
|CONCURRENT_FLIGHTS|PLANE_AGE|  DEPARTING_AIRPORT|PRCP|SNOW|SNWD|TMAX|AWND|DEP_DEL15|
+-----------------+---------+-------------------+----+----+----+----+----+---------+
|               25|        8|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|               29|        3|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|               27|       18|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|               27|        2|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|               10|        1|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|               10|        5|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|               29|        2|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|               10|        3|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        1|
|               10|        3|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|               27|        1|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|               17|        3|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        1|
|               27|        3|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|               26|       12|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|               27|        7|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|               27|        2|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|               29|        4|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        1|
|               25|        4|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|               10|        6|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|               28|        5|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|               29|        5|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|
+-----------------+---------+-------------------+----+----+----+----+----+---------+
```

Then I convert DEPARTING_AIRPORT column to index from 1 to 96

```
from pyspark.ml.feature import StringIndexer
```

```
indexer = StringIndexer(inputCol="DEPARTING_AIRPORT", outputCol="AirportIndex")
indexed = indexer.fit(df).transform(df)
```

```
indexed.show()
```

```
+-----------------+---------+--------------------+----+----+----+----+----+---------+------------+
|CONCURRENT_FLIGHTS|PLANE_AGE|   DEPARTING_AIRPORT|PRCP|SNOW|SNWD|TMAX|AWND|DEP_DEL15|AirportIndex|
+-----------------+---------+--------------------+----+----+----+----+----+---------+------------+
|               25|        8|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|
|               29|        3|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|
|               27|       18|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|
|               27|        2|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|
|               10|        1|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|
|               10|        5|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|
|               29|        2|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|
|               10|        3|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        1|        10.0|
|               10|        3|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|
|               27|        1|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|
|               17|        3|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        1|        10.0|
|               27|        3|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|
|               26|       12|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|
|               27|        7|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|
|               27|        2|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|
|               29|        4|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        1|        10.0|
|               25|        4|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|
|               10|        6|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|
|               28|        5|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|
|               29|        5|McCarran Internat...| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|
+-----------------+---------+--------------------+----+----+----+----+----+---------+------------+
```

Then I dropped the DEPARTING_AIRPORT column and I will use AirportIndex column instead

```
df = indexed.drop('DEPARTING_AIRPORT')
```

```
df = df.withColumn("CONCURRENT_FLIGHTS",col("CONCURRENT_FLIGHTS").cast("double")).dropna("any")
df = df.withColumn("PLANE_AGE",col("PLANE_AGE").cast("double")).dropna("any")
df = df.withColumn("PRCP",col("PRCP").cast("double")).dropna("any")
df = df.withColumn("SNOW",col("SNOW").cast("double")).dropna("any")
df = df.withColumn("SNWD",col("SNWD").cast("double")).dropna("any")
df = df.withColumn("TMAX",col("TMAX").cast("double")).dropna("any")
df = df.withColumn("AWND",col("AWND").cast("double")).dropna("any")
df = df.withColumn("DEP_DEL15",col("DEP_DEL15").cast("int")).dropna("any")
```

Then I used the vector to start the ML model.

```
from pyspark.ml.linalg import Vector
from pyspark.ml.feature import VectorAssembler
input_cols = ['CONCURRENT_FLIGHTS','PLANE_AGE','PRCP','SNOW','SNWD','TMAX','AWND','AirportIndex']
assembler = VectorAssembler(inputCols=input_cols, outputCol='features')
assembler_temp = assembler.transform(df)
```

```
assembler_temp.show()
```

```
+------------------+---------+----+----+----+----+----+---------+------------+--------------------+
|CONCURRENT_FLIGHTS|PLANE_AGE|PRCP|SNOW|SNWD|TMAX|AWND|DEP_DEL15|AirportIndex|            features|
+------------------+---------+----+----+----+----+----+---------+------------+--------------------+
|              25.0|      8.0| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|[25.0,8.0,0.0,0.0...|
|              29.0|      3.0| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|[29.0,3.0,0.0,0.0...|
|              27.0|     18.0| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|[27.0,18.0,0.0,0....|
|              27.0|      2.0| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|[27.0,2.0,0.0,0.0...|
|              10.0|      1.0| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|[10.0,1.0,0.0,0.0...|
|              10.0|      5.0| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|[10.0,5.0,0.0,0.0...|
|              29.0|      2.0| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|[29.0,2.0,0.0,0.0...|
|              10.0|      3.0| 0.0| 0.0| 0.0|65.0|2.91|        1|        10.0|[10.0,3.0,0.0,0.0...|
|              10.0|      3.0| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|[10.0,3.0,0.0,0.0...|
|              27.0|      1.0| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|[27.0,1.0,0.0,0.0...|
|              17.0|      3.0| 0.0| 0.0| 0.0|65.0|2.91|        1|        10.0|[17.0,3.0,0.0,0.0...|
|              27.0|      3.0| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|[27.0,3.0,0.0,0.0...|
|              26.0|     12.0| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|[26.0,12.0,0.0,0....|
|              27.0|      7.0| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|[27.0,7.0,0.0,0.0...|
|              27.0|      2.0| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|[27.0,2.0,0.0,0.0...|
|              29.0|      4.0| 0.0| 0.0| 0.0|65.0|2.91|        1|        10.0|[29.0,4.0,0.0,0.0...|
|              25.0|      4.0| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|[25.0,4.0,0.0,0.0...|
|              10.0|      6.0| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|[10.0,6.0,0.0,0.0...|
|              28.0|      5.0| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|[28.0,5.0,0.0,0.0...|
|              29.0|      5.0| 0.0| 0.0| 0.0|65.0|2.91|        0|        10.0|[29.0,5.0,0.0,0.0...|
+------------------+---------+----+----+----+----+----+---------+------------+--------------------+
only showing top 20 rows
```

Then I dropped the columns I will use to predict and keep the features column And the predicted DEP_DEL15

```
new_df = assembler_temp.drop('CONCURRENT_FLIGHTS','PLANE_AGE','PRCP','SNOW','SNWD','TMAX','AWND','AirportIndex')
```

```
new_df.show()
```

```
+---------+--------------------+
|DEP_DEL15|            features|
+---------+--------------------+
|        0|[25.0,8.0,0.0,0.0...|
|        0|[29.0,3.0,0.0,0.0...|
|        0|[27.0,18.0,0.0,0....|
|        0|[27.0,2.0,0.0,0.0...|
|        0|[10.0,1.0,0.0,0.0...|
|        0|[10.0,5.0,0.0,0.0...|
|        0|[29.0,2.0,0.0,0.0...|
|        1|[10.0,3.0,0.0,0.0...|
|        0|[10.0,3.0,0.0,0.0...|
|        0|[27.0,1.0,0.0,0.0...|
|        1|[17.0,3.0,0.0,0.0...|
|        0|[27.0,3.0,0.0,0.0...|
|        0|[26.0,12.0,0.0,0....|
|        0|[27.0,7.0,0.0,0.0...|
|        0|[27.0,2.0,0.0,0.0...|
|        1|[29.0,4.0,0.0,0.0...|
|        0|[25.0,4.0,0.0,0.0...|
|        0|[10.0,6.0,0.0,0.0...|
|        0|[28.0,5.0,0.0,0.0...|
|        0|[29.0,5.0,0.0,0.0...|
+---------+--------------------+
only showing top 20 rows
```

B. For the second classification model:

Note: they are all the same dataset, but the columns used to predict the delay is different. Also, the preprocessing is almost similar.

I Will select these columns to predict if the flight will delay more than 15 min or not. (DEP_DEL15) 1 = Yes, 0 =No

```
df = df.select('DISTANCE_GROUP','NUMBER_OF_SEATS','AIRPORT_FLIGHTS_MONTH','GROUND_SERV_PER_PASS','PRCP','SNOW','SNWD','TMAX','AWND','DEP_DEL15')
```

```
df.show()
```

```
+--------------+---------------+---------------------+--------------------+----+----+----+----+----+---------+
|DISTANCE_GROUP|NUMBER_OF_SEATS|AIRPORT_FLIGHTS_MONTH|GROUND_SERV_PER_PASS|PRCP|SNOW|SNWD|TMAX|AWND|DEP_DEL15|
+--------------+---------------+---------------------+--------------------+----+----+----+----+----+---------+
|             2|            143|                13056|9.889412309998219...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|             7|            191|                13056|0.000148660200942...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|             7|            199|                13056|0.000148660200942...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|             9|            180|                13056|0.000148660200942...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|             7|            182|                13056|0.000124651073071...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|             3|            180|                13056|7.134694872433899...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|             6|            186|                13056|7.134694872433899...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|             7|            186|                13056|7.134694872433899...| 0.0| 0.0| 0.0|65.0|2.91|        1|
|             7|            180|                13056|7.134694872433899...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|             8|            186|                13056|7.134694872433899...| 0.0| 0.0| 0.0|65.0|2.91|        0|
|             6|            180|                13056|7.134694872433899...| 0.0| 0.0| 0.0|65.0|2.91|        1|
|             1|            149|                13056|0.000174601449726...| 0.0| 0.0| 0.0|65.0|2.91|        0|
```

Then I cast all the columns to numbers.

```
df = df.withColumn("DISTANCE_GROUP",col("DISTANCE_GROUP").cast("double")).dropna("any")
df = df.withColumn("NUMBER_OF_SEATS",col("NUMBER_OF_SEATS").cast("double")).dropna("any")
df = df.withColumn("AIRPORT_FLIGHTS_MONTH",col("AIRPORT_FLIGHTS_MONTH").cast("double")).dropna("any")
df = df.withColumn("GROUND_SERV_PER_PASS",col("GROUND_SERV_PER_PASS").cast("double")).dropna("any")
df = df.withColumn("PRCP",col("PRCP").cast("double")).dropna("any")
df = df.withColumn("SNOW",col("SNOW").cast("double")).dropna("any")
df = df.withColumn("SNWD",col("SNWD").cast("double")).dropna("any")
df = df.withColumn("TMAX",col("TMAX").cast("double")).dropna("any")
df = df.withColumn("AWND",col("AWND").cast("double")).dropna("any")
df = df.withColumn("DEP_DEL15",col("DEP_DEL15").cast("int")).dropna("any")
```

Then I used the vector to start the ML model.

```
from pyspark.ml.linalg import Vector
from pyspark.ml.feature import VectorAssembler
input_cols = ['DISTANCE_GROUP','NUMBER_OF_SEATS','AIRPORT_FLIGHTS_MONTH','GROUND_SERV_PER_PASS','PRCP','SNOW','SNWD','TMAX','AWND']
assembler = VectorAssembler(inputCols=input_cols, outputCol='features')
assembler_temp = assembler.transform(df)
```

```
assembler_temp.show()
```

```
+--------------+---------------+---------------------+--------------------+----+----+----+----+----+---------+--------------------+
|DISTANCE_GROUP|NUMBER_OF_SEATS|AIRPORT_FLIGHTS_MONTH|GROUND_SERV_PER_PASS|PRCP|SNOW|SNWD|TMAX|AWND|DEP_DEL15|            features|
+--------------+---------------+---------------------+--------------------+----+----+----+----+----+---------+--------------------+
|           2.0|          143.0|              13056.0|9.889412309998219E-5| 0.0| 0.0| 0.0|65.0|2.91|        0|[2.0,143.0,13056....|
|           7.0|          191.0|              13056.0|1.486602009422039E-4| 0.0| 0.0| 0.0|65.0|2.91|        0|[7.0,191.0,13056....|
|           7.0|          199.0|              13056.0|1.486602009422039E-4| 0.0| 0.0| 0.0|65.0|2.91|        0|[7.0,199.0,13056....|
|           9.0|          180.0|              13056.0|1.486602009422039E-4| 0.0| 0.0| 0.0|65.0|2.91|        0|[9.0,180.0,13056....|
|           7.0|          182.0|              13056.0|1.246510730715623...| 0.0| 0.0| 0.0|65.0|2.91|        0|[7.0,182.0,13056....|
|           3.0|          180.0|              13056.0|7.134694872433899E-6| 0.0| 0.0| 0.0|65.0|2.91|        0|[3.0,180.0,13056....|
|           6.0|          186.0|              13056.0|7.134694872433899E-6| 0.0| 0.0| 0.0|65.0|2.91|        0|[6.0,186.0,13056....|
|           7.0|          186.0|              13056.0|7.134694872433899E-6| 0.0| 0.0| 0.0|65.0|2.91|        1|[7.0,186.0,13056....|
|           7.0|          180.0|              13056.0|7.134694872433899E-6| 0.0| 0.0| 0.0|65.0|2.91|        0|[7.0,180.0,13056....|
|           8.0|          186.0|              13056.0|7.134694872433899E-6| 0.0| 0.0| 0.0|65.0|2.91|        0|[8.0,186.0,13056....|
|           6.0|          180.0|              13056.0|7.134694872433899E-6| 0.0| 0.0| 0.0|65.0|2.91|        1|[6.0,180.0,13056....|
```

Then I dropped the columns I will use to predict and keep the features column And the predicted DEP_DEL15

```
new_df = assembler_temp.drop('DISTANCE_GROUP','NUMBER_OF_SEATS','AIRPORT_FLIGHTS_MONTH','GROUND_SERV_PER_PASS','PRCP','SNOW','SNWD','TMAX','AWND')
```

```
new_df.show()
```

```
+---------+--------------------+
|DEP_DEL15|            features|
+---------+--------------------+
|        0|[2.0,143.0,13056....|
|        0|[7.0,191.0,13056....|
|        0|[7.0,199.0,13056....|
|        0|[9.0,180.0,13056....|
|        0|[7.0,182.0,13056....|
|        0|[3.0,180.0,13056....|
|        0|[6.0,186.0,13056....|
|        1|[7.0,186.0,13056....|
|        0|[7.0,180.0,13056....|
|        0|[8.0,186.0,13056....|
|        1|[6.0,180.0,13056....|
|        0|[1.0,149.0,13056....|
|        0|[1.0,119.0,13056....|
|        0|[2.0,146.0,13056....|
|        0|[4.0,181.0,13056....|
|        1|[4.0,181.0,13056....|
```

4. **[2 pts]** Divide each dataset into training and testing.

```
(trainData, testData) = new_df.randomSplit([0.7, 0.3])
```

5. **[12 pts]** Build two classification models.

the first classification model (Decision tree):

```
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

```
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
```

```
dt_model = DecisionTreeClassifier(labelCol="DEP_DEL15", featuresCol="features", maxBins=100)
```

```
model = dt_model.fit(trainData)
```

```
prediction = model.transform(testData)
prediction.show()
```

```
+---------+--------------------+--------------------+--------------------+----------+
|DEP_DEL15|            features|       rawPrediction|         probability|prediction|
+---------+--------------------+--------------------+--------------------+----------+
|        0|(8,[0,1,5,6],[2.0...|[2021179.0,422817.0]|[0.82699767102728...|       0.0|
|        0|(8,[0,1,5,6],[4.0...|[2021179.0,422817.0]|[0.82699767102728...|       0.0|
|        0|(8,[0,1,5,6],[4.0...|[2021179.0,422817.0]|[0.82699767102728...|       0.0|
|        0|(8,[0,1,5,6],[4.0...|[2021179.0,422817.0]|[0.82699767102728...|       0.0|
|        0|(8,[0,1,5,6],[4.0...|[2021179.0,422817.0]|[0.82699767102728...|       0.0|
|        0|(8,[0,1,5,6],[4.0...|[2021179.0,422817.0]|[0.82699767102728...|       0.0|
|        0|(8,[0,1,5,6],[4.0...|[2021179.0,422817.0]|[0.82699767102728...|       0.0|
|        0|(8,[0,1,5,6],[4.0...|[2021179.0,422817.0]|[0.82699767102728...|       0.0|
|        0|(8,[0,1,5,6],[4.0...|[2021179.0,422817.0]|[0.82699767102728...|       0.0|
|        0|(8,[0,1,5,6],[4.0...|[2021179.0,422817.0]|[0.82699767102728...|       0.0|
|        0|(8,[0,1,5,6],[4.0...|[2021179.0,422817.0]|[0.82699767102728...|       0.0|
|        0|(8,[0,1,5,6],[4.0...|[2021179.0,422817.0]|[0.82699767102728...|       0.0|
|        0|(8,[0,1,5,6],[5.0...|[2021179.0,422817.0]|[0.82699767102728...|       0.0|
|        0|(8,[0,1,5,6],[5.0...|[2021179.0,422817.0]|[0.82699767102728...|       0.0|
|        0|(8,[0,1,5,6],[5.0...|[2021179.0,422817.0]|[0.82699767102728...|       0.0|
|        0|(8,[0,1,5,6],[5.0...|[2021179.0,422817.0]|[0.82699767102728...|       0.0|
|        0|(8,[0,1,5,6],[5.0...|[2021179.0,422817.0]|[0.82699767102728...|       0.0|
|        0|(8,[0,1,5,6],[5.0...|[2021179.0,422817.0]|[0.82699767102728...|       0.0|
|        0|(8,[0,1,5,6],[5.0...|[2021179.0,422817.0]|[0.82699767102728...|       0.0|
|        0|(8,[0,1,5,6],[9.0...|[2021179.0,422817.0]|[0.82699767102728...|       0.0|
+---------+--------------------+--------------------+--------------------+----------+
only showing top 20 rows
```

the second classification model (random forest):

```
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

```
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
```

```
dt_model = RandomForestClassifier(labelCol="DEP_DEL15", featuresCol="features", maxBins=100)
```

```
model = dt_model.fit(trainData)
```

```
prediction = model.transform(testData)
prediction.show()
```

```
+---------+--------------------+--------------------+--------------------+----------+
|DEP_DEL15|            features|       rawPrediction|         probability|prediction|
+---------+--------------------+--------------------+--------------------+----------+
|        0|(9,[0,1,2,3],[1.0...|[16.3011575706836...|[0.81505787853418...|       0.0|
|        0|(9,[0,1,2,3],[1.0...|[16.3011575706836...|[0.81505787853418...|       0.0|
|        0|(9,[0,1,2,3],[5.0...|[16.3011575706836...|[0.81505787853418...|       0.0|
|        0|(9,[0,1,2,3],[5.0...|[16.3011575706836...|[0.81505787853418...|       0.0|
|        0|(9,[0,1,2,3],[5.0...|[16.3011575706836...|[0.81505787853418...|       0.0|
|        0|(9,[0,1,2,3],[5.0...|[16.3011575706836...|[0.81505787853418...|       0.0|
|        0|(9,[0,1,2,3],[5.0...|[16.3011575706836...|[0.81505787853418...|       0.0|
|        0|(9,[0,1,2,3],[5.0...|[16.3011575706836...|[0.81505787853418...|       0.0|
|        0|(9,[0,1,2,3],[5.0...|[16.3011575706836...|[0.81505787853418...|       0.0|
|        0|(9,[0,1,2,3],[5.0...|[16.3646883971287...|[0.81823441985643...|       0.0|
|        0|(9,[0,1,2,3],[5.0...|[16.3011575706836...|[0.81505787853418...|       0.0|
|        0|(9,[0,1,2,3],[5.0...|[16.3646883971287...|[0.81823441985643...|       0.0|
|        0|(9,[0,1,2,3],[7.0...|[16.3011575706836...|[0.81505787853418...|       0.0|
|        0|(9,[0,1,2,3],[7.0...|[16.3011575706836...|[0.81505787853418...|       0.0|
|        0|(9,[0,1,2,3],[7.0...|[16.3011575706836...|[0.81505787853418...|       0.0|
|        0|(9,[0,1,2,3],[7.0...|[16.3011575706836...|[0.81505787853418...|       0.0|
|        0|(9,[0,1,2,3],[7.0...|[16.3646883971287...|[0.81823441985643...|       0.0|
|        0|(9,[0,1,2,3],[7.0...|[16.3646883971287...|[0.81823441985643...|       0.0|
|        0|(9,[0,1,2,3],[7.0...|[16.3646883971287...|[0.81823441985643...|       0.0|
|        0|(9,[0,1,2,3],[7.0...|[16.3646883971287...|[0.81823441985643...|       0.0|
+---------+--------------------+--------------------+--------------------+----------+
only showing top 20 rows
```

6. **[4 pts]** Test the models and compute their accuracy.

Accuracy for the first classification model:

```
evaluator = MulticlassClassificationEvaluator(labelCol="DEP_DEL15",predictionCol="prediction",metricName="accuracy")
accuracy = evaluator.evaluate(prediction)
print("Accuracy :",accuracy)
print("Test Error = %g" %(1.0 - accuracy ))
```

```
Accuracy : 0.8110163700797225
Test Error = 0.188984
```

Accuracy for the second classification model:

```
evaluator = MulticlassClassificationEvaluator(labelCol="DEP_DEL15",predictionCol="prediction",metricName="accuracy")
accuracy = evaluator.evaluate(prediction)
print("Accuracy For the Second Classification model: ",accuracy)
print("Test Error For the Second Classification model = %g" %(1.0 - accuracy ))
```

```
Accuracy For the Second Classification model:  0.8108965301804528
Test Error For the Second Classification model = 0.189103
```