```python
# -*- coding: utf-8 -*-
"""CS777_FinalProject_Code.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1wNCZiaZr2d_vAxMj_T1vsE7SnLWCPuYq

# Introduction

Logistics problems are a rich source of data for big-data analytics. Airlines and other similar
transportation companies connect millions of people a year in a complicated and interdependent
network of locations and transit methods. It is not clear from the outset which models will
provide useful insight for transportation companies. This project attempts to address this
topic specifically for the airline industry.

The dataset we will used for this project was obtained from Kaggle.

Link to dataset: https://www.kaggle.com/datasets/yuanyuwendymu/airline-delay-and-cancellation-
data-2009-2018?select=2010.csv

## Preliminary Actions

This project is presented in the form of a literate programming notebook, which requires that
we address some preliminary requirements before moving on to the material at hand.
"""



"""### Mounting GDrive"""

from google.colab import drive
drive.mount("/content/drive/")

"""### Installing Pyspark and Required Packages"""

!pip install pyspark

from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *

import matplotlib.pyplot as plt
import pandas as pd

from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml.feature import PCA
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.linalg import Vectors
from pyspark.ml.regression import LinearRegression
from pyspark.ml.tuning import ParamGridBuilder
import numpy as np

spark = SparkSession\
    .builder\
    .appName("FlightDataAnalytics")\
    .getOrCreate()

print('\nSpark session instantiated')

"""### Loading Data into Memory"""

# path = "/content/drive/MyDrive/CS777_Project/data/2009_cleaned.csv" # small dataset
path = sys.argv[1]

print('\nDataset read successfully')
```

```
data = spark\
    .read\
    .csv(
        path,
        header=True,
        inferSchema=True
    )

"""For the purposes of this demonstration notebook, we load a sample of our overall data which
is small enough for Colab to run without issue. Please see our full report for the total
results
on the entire dataset.

# The Dataset

Our dataset consists of X.XGB of flight data from the years 2009 to 2018. The data consists of
columns as follows:
"""

print(data.columns)

# Drop the unneeded column...
data = data.drop("_c0")

"""#Models

##Linear Regression

Using linear regression, we will build a model that will predict the amount of time a flight is
delayed in arriving at its destination airport in terms of minutes.
"""

print('------- Linear Regression -------')

"""### Using all variables

Create and Fit MLR Model
"""

# cols = data.columns
# data.columns

"""#### Additional Data Cleaning"""

print('\nData cleaning initiated')
# cols_to_drop = ['_c0', 'DEP_TIME', 'DEP_DELAY', 'WHEELS_OFF', 'WHEELS_ON', 'ARR_TIME',
'CANCELLED', 'DIVERTED', 'ACTUAL_ELAPSED_TIME', 'AIR_TIME', 'CARRIER_DELAY', 'WEATHER_DELAY',
'NAS_DELAY', 'SECURITY_DELAY', 'LATE_AIRCRAFT_DELAY', 'CARRIER_DELAY_BIN', 'WEATHER_DELAY_BIN',
'NAS_DELAY_BIN', 'SECURITY_DELAY_BIN', 'LATE_AIRCRAFT_DELAY_BIN', 'DELAY']
cols_to_drop = ['DELAY'] # come back to this

# 'ARR_DELAY',

df = data.withColumn("DELAY", when(col("ARR_DELAY") > 5, 1).otherwise(0))\
         .filter(col("CANCELLED") != 1)\
         .filter(col("DIVERTED") != 1)\
         .drop(*cols_to_drop)

# data = data.filter(data.ARR_DELAY > '0')

for col_name in df.columns:
    df = df.withColumn(col_name, col(col_name).cast("double"))

print('\nData cleaning completed')
```

```python
print('\nData cleaning initiated')
# cols_to_drop = ['_c0', 'DEP_TIME', 'DEP_DELAY', 'WHEELS_OFF', 'WHEELS_ON', 'ARR_TIME',
'CANCELLED', 'DIVERTED', 'ACTUAL_ELAPSED_TIME', 'AIR_TIME', 'CARRIER_DELAY', 'WEATHER_DELAY',
'NAS_DELAY', 'SECURITY_DELAY', 'LATE_AIRCRAFT_DELAY', 'CARRIER_DELAY_BIN', 'WEATHER_DELAY_BIN',
'NAS_DELAY_BIN', 'SECURITY_DELAY_BIN', 'LATE_AIRCRAFT_DELAY_BIN', 'DELAY']
cols_to_drop = ['_c0', 'DELAY', 'CANCELLED', 'DIVERTED', 'CARRIER_DELAY_BIN',
'WEATHER_DELAY_BIN', 'NAS_DELAY_BIN', 'SECURITY_DELAY_BIN', 'LATE_AIRCRAFT_DELAY_BIN']

# 'ARR_DELAY',

df = data.withColumn("DELAY", when(col("ARR_DELAY") > 5, 1).otherwise(0))\
         .filter(col("CANCELLED") != 1)\
         .filter(col("DIVERTED") != 1)\
         .drop(*cols_to_drop)

# data = data.filter(data.ARR_DELAY > '0')

for col_name in df.columns:
    df = df.withColumn(col_name, col(col_name).cast("double"))

print('\nData cleaning completed')

df.show(5)
print((df.count(), len(df.columns)))

"""#### Create Vector Assembler for Fetaure Set"""

df.columns

print('\nVector Assembler creation begun')

variables = ['CRS_DEP_TIME',
 'DEP_TIME',
 'DEP_DELAY',
 'TAXI_OUT',
 'WHEELS_OFF',
 'WHEELS_ON',
 'TAXI_IN',
 'CRS_ARR_TIME',
 'ARR_TIME',
 'ARR_DELAY',
 'CRS_ELAPSED_TIME',
 'ACTUAL_ELAPSED_TIME',
 'AIR_TIME',
 'DISTANCE',
 'CARRIER_DELAY',
 'WEATHER_DELAY',
 'NAS_DELAY',
 'SECURITY_DELAY',
 'LATE_AIRCRAFT_DELAY',
 'ORIGIN_TRAFF',
 'DEST_TRAFF',
 'OP_CARRIER_AA',
 'OP_CARRIER_AS',
 'OP_CARRIER_B6',
 'OP_CARRIER_DL',
 'OP_CARRIER_EV',
 'OP_CARRIER_F9',
 'OP_CARRIER_HA',
 'OP_CARRIER_OO',
 'OP_CARRIER_UA',
 'OP_CARRIER_WN']

#  'CARRIER_DELAY',
#  'WEATHER_DELAY',
#  'NAS_DELAY',
```

```
#  'SECURITY_DELAY',
#  'LATE_AIRCRAFT_DELAY',
#  'CANCELLED'

assembler = VectorAssembler(inputCols = variables, outputCol = 'features')
assembled_data = assembler.transform(df)

print('\nVector Assembler creation completed, data transformed')

# assembled_data.select('features', 'ARR_DELAY').show(5)

"""#### Create new data set with X and Y values to use for MLR"""

mlr_df = assembled_data.select('features', 'ARR_DELAY')
print('\nMLR data set created')

# mlr_df.show(5)

"""#### Split MLR Dataset into Train (70%) and Test (30%)"""

train_data, test_data = mlr_df.randomSplit([0.7, 0.3], seed = 777)
print('\nDataset split into train and test')

# train_data.describe().show()

# test_data.describe().show()

"""#### Instantiate MLR model"""

mlr = LinearRegression(featuresCol = 'features',
                       labelCol = 'ARR_DELAY')
print('MLR model instantiated')

"""#### Fit Model to Training Data"""

print('\nModel fitting to training data begun')
model = mlr.fit(train_data)
print('\nModel fit to training data successfully')

"""#### View Coeffiicents and Intercept obtained"""

print(pd.DataFrame({'Coefficients':model.coefficients}, index = variables))
print('\nIntercept: ' + str(model.intercept))

"""#### Evaluate Model Performance on Test Data"""

print('Model Evaluation begun')
results = model.evaluate(test_data)
print('Model Evaluation completed')

# results.residuals.show()

"""#### Training Summary Statistics"""

trainingSummary = model.summary
# print('numIterations: %d' % trainingSummary.totalIterations)
# print('objectiveHistory: %s' % str(trainingSummary.objectiveHistory))
# trainingSummary.residuals.show()
print('Mean Absolute Error: %f' % trainingSummary.meanAbsoluteError)
print('Mean Squared Error: %f' % trainingSummary.meanSquaredError)
print('RMSE: %f' % trainingSummary.rootMeanSquaredError)
print('R-Squared: %f' % trainingSummary.r2)
print('Adjusted R-Squared: %f' % trainingSummary.r2adj)

"""#### Testing Performance Statistics"""

print('Mean Absolute Error: ', results.meanAbsoluteError)
```

```
print('Mean Squared Error: ', results.meanSquaredError)
print('Root Mean Squared Error: ', results.rootMeanSquaredError)
print('R squared: ', results.r2)
print('Adjusted R sqaured: ', results.r2adj)

"""### PCA"""

inputCols = df.columns[0:-1]
outputCol = df.columns[-1]

assembler = VectorAssembler(inputCols = inputCols, outputCol = 'features')
assembled_data = assembler.transform(df).select('features', 'ARR_DELAY')

# assembled_data.show(5)

k_values = [1,2,3,4,5]
evr = []
for k in k_values:
  pca = PCA(k = k, inputCol = 'features', outputCol = 'pca_features')
  model = pca.fit(assembled_data)
  evr.append(np.sum(model.explainedVariance))

plt.plot(k_values, evr, 'bo-', linewidth=2)
plt.xlabel('k')
plt.ylabel('Explained variance ratio')
plt.title('Scree plot: Explained Variance vs. k values')
plt.show()

best_k = 0
for i in range(len(evr)):
    if evr[i] >= 0.8:
        best_k = k_values[i]
        break

print('Best k value: ', best_k)

pca = PCA(k = best_k, inputCol = 'features', outputCol = 'pcaFeatures')

pca_model = pca.fit(assembled_data)

pca_result = pca_model.transform(assembled_data)

print('Using Dimensionality Reduction and Regularisation')

"""### Ridge Regularisation

α (elasticNetParam) is set to 0
"""

train_data_ridge, test_data_ridge = pca_result.randomSplit([0.7, 0.3], seed = 777)
print('\nDataset split into train and test')

mlr_ridge = LinearRegression(elasticNetParam = 0, #featuresCol = "features",
                             labelCol = 'ARR_DELAY')
print('MLR model instantiated')

print('\nModel fitting to training data begun')
model = mlr_ridge.fit(train_data_ridge)
print('\nModel fit to training data successfully')

print(pd.DataFrame({'Coefficients':model.coefficients}, index = variables))
print('\nIntercept: ' + str(model.intercept))

print('Model Evaluation begun')
results = model.evaluate(test_data_ridge)
print('Model Evaluation completed')
```

```
trainingSummary = model.summary
print('numIterations: %d' % trainingSummary.totalIterations)
# print('objectiveHistory: %s' % str(trainingSummary.objectiveHistory))
# trainingSummary.residuals.show()
print('Mean Absolute Error: %f' % trainingSummary.meanAbsoluteError)
print('Mean Squared Error: %f' % trainingSummary.meanSquaredError)
print('RMSE: %f' % trainingSummary.rootMeanSquaredError)
print('R-Squared: %f' % trainingSummary.r2)
print('Adjusted R-Squared: %f' % trainingSummary.r2adj)

print('Mean Absolute Error: ', results.meanAbsoluteError)
print('Mean Squared Error: ', results.meanSquaredError)
print('Root Mean Squared Error: ', results.rootMeanSquaredError)
print('R squared: ', results.r2)
print('Adjusted R sqaured: ', results.r2adj)

"""### Lasso Regularisation

α (elasticNetParam) is set to 1
"""

train_data_lasso, test_data_lasso = pca_result.randomSplit([0.7, 0.3], seed = 777)
print('\nDataset split into train and test')

mlr_lasso = LinearRegression(elasticNetParam = 1, #featuresCol = "features",
                             labelCol = 'ARR_DELAY')
print('MLR model instantiated')

print('\nModel fitting to training data begun')
model = mlr_lasso.fit(train_data)
print('\nModel fit to training data successfully')

print(pd.DataFrame({'\nCoefficients':model.coefficients}, index = variables))
print('\nIntercept: ' + str(model.intercept))

print('Model Evaluation begun')
results = model.evaluate(test_data)
print('Model Evaluation completed')

trainingSummary = model.summary
print('numIterations: %d' % trainingSummary.totalIterations)
# print('objectiveHistory: %s' % str(trainingSummary.objectiveHistory))
# trainingSummary.residuals.show()
print('Mean Absolute Error: %f' % trainingSummary.meanAbsoluteError)
print('Mean Squared Error: %f' % trainingSummary.meanSquaredError)
print('RMSE: %f' % trainingSummary.rootMeanSquaredError)
print('R-Squared: %f' % trainingSummary.r2)
print('Adjusted R-Squared: %f' % trainingSummary.r2adj)

print('Mean Absolute Error: ', results.meanAbsoluteError)
print('Mean Squared Error: ', results.meanSquaredError)
print('Root Mean Squared Error: ', results.rootMeanSquaredError)
print('R squared: ', results.r2)
print('Adjusted R sqaured: ', results.r2adj)

"""##Logistic Regression

Using logistic regression, we will do a binary classification that predicts whether a flight is
delay or on time.

###Additional Data Preparation
"""

print('------- Logistic Regression -------')

#Dropping columns of values generated after take off to correctly classify delay or on time
cols_to_drop = ['DEP_TIME', 'DEP_DELAY', 'WHEELS_OFF', 'WHEELS_ON', 'ARR_TIME', 'ARR_DELAY',
```

```
'CANCELLED', 'DIVERTED', 'ACTUAL_ELAPSED_TIME', 'AIR_TIME', 'CARRIER_DELAY', 'WEATHER_DELAY',
'NAS_DELAY', 'SECURITY_DELAY', 'LATE_AIRCRAFT_DELAY', 'CARRIER_DELAY_BIN',
'WEATHER_DELAY_BIN', 'NAS_DELAY_BIN', 'SECURITY_DELAY_BIN', 'LATE_AIRCRAFT_DELAY_BIN']

#Creating a new column "DELAY" that indicates 1 for a delay and 0 for on time.
#DELAY is 1 when arrival delay time is more than 5
#Cancelled and diverted flights are not included in classification
df = data.withColumn("DELAY", when(col("ARR_DELAY") > 5, 1).otherwise(0))\
         .filter(col("CANCELLED") != 1)\
         .filter(col("DIVERTED") != 1)\
         .drop(*cols_to_drop)

#Casting all attributes as doubles since the default is string
for col_name in df.columns:
    df = df.withColumn(col_name, col(col_name).cast("double"))

print("Dataset used for logistic regression:\n")

df.show()

"""###Dimensionality Reduction with PCA

"""

print('Dimensionality Reduction with PCA\n')

inputCols = df.columns[0:-1] #Feature columns
outputCol = df.columns[-1] #Label column

#Assembling data into a feature and label dataframe
assembler = VectorAssembler(inputCols=inputCols, outputCol="features")
assembled_data = assembler.transform(df).select("features", "DELAY")

#Optimization for the number of components
#We choose lowest k with explained variance over 0.8
k_values = [1,2,3,4,5]
evr = []
for k in k_values:
  pca = PCA(k=k, inputCol='features', outputCol='pca_features')
  model = pca.fit(assembled_data)
  evr.append(np.sum(model.explainedVariance))

plt.plot(k_values, evr, 'bo-', linewidth=2)
plt.xlabel('k')
plt.ylabel('Explained variance Ratio')
plt.title('Scree plot: Explained Variance vs. k values')
plt.show()

#Choosing best k
best_k = 0
for i in range(len(evr)):
    if evr[i] >= 0.8:
        best_k = k_values[i]
        break

print("\nBest k value:", best_k)

#Creating PCA model using optimised k
pca = PCA(k=best_k, inputCol="features", outputCol="pcaFeatures")
pca_model = pca.fit(assembled_data)

#Reduced dataset to be used for modelling
pca_result = pca_model.transform(assembled_data)

"""###Model with Dimensionality Reduction"""

print("\nLogistic Regression Model with Dimensionality Reduction\n")
```

```
(train_log_reg_pca, validation_log_reg_pca, test_log_reg_pca) = pca_result.randomSplit([0.7,
0.1, 0.2], seed=777)

lr = LogisticRegression(featuresCol="pcaFeatures", labelCol=outputCol)

#Building a grid search for hyperparameter tuning
paramGrid = ParamGridBuilder() \
    .addGrid(lr.regParam, [0.01, 0.1, 1]) \
    .addGrid(lr.elasticNetParam, [0.0, 0.5, 1.0]) \
    .addGrid(lr.maxIter, [10, 100]) \
    .build()

evaluator_log_reg = BinaryClassificationEvaluator(rawPredictionCol='rawPrediction',
labelCol='DELAY')

#Searching for optimal hyperparametrs through the parameter grid
best_log_reg_pca = None
best_auc_log_reg_pca = 0.0
for params in paramGrid:
    params = list(params.values())
    regParam = params[0]
    elasticNetParam = params[1]
    maxIter = params[2]
    lr.setParams(regParam=regParam, elasticNetParam=elasticNetParam, maxIter=maxIter)
    log_reg_pca = lr.fit(train_log_reg_pca)
    history_log_reg_pca = log_reg_pca.summary.objectiveHistory
    predictions_log_reg_pca = log_reg_pca.transform(validation_log_reg_pca)
    auc_log_reg_pca = evaluator_log_reg.evaluate(predictions_log_reg_pca)
    if auc_log_reg_pca > best_auc_pca:
        best_auc_pca = auc_log_reg_pca
        best_log_reg_pca = log_reg_pca

print("\nHyperparameter tuning:\n")

print(f"\nBest model hyperparameters:\nRegularization:
{best_log_reg_pca.getRegParam()}\nElastic Net:{best_log_reg_pca.getElasticNetParam()}\nMax
Iterations: {best_log_reg_pca.getMaxIter()}")

"""###Model without Dimensionality Reduction"""

print("\n\nLogistic Regression Model with Dimensionality Reduction\n")

(train_log_reg, validation_log_reg, test_log_reg) = pca_result.randomSplit([0.7, 0.1, 0.2],
seed=777)

lr = LogisticRegression(featuresCol="features", labelCol=outputCol, maxIter=50, regParam=0.01,
elasticNetParam=0.5)

#Searching for optimal hyperparametrs through the parameter grid
best_log_reg = None
best_auc_log_reg = 0.0
for params in paramGrid:
    params = list(params.values())
    regParam = params[0]
    elasticNetParam = params[1]
    maxIter = params[2]
    lr.setParams(regParam=regParam, elasticNetParam=elasticNetParam, maxIter=maxIter)
    log_reg = lr.fit(train_log_reg)
    history_log_reg = log_reg.summary.objectiveHistory
    predictions_log_reg = log_reg.transform(validation_log_reg)
    auc_log_reg = evaluator_log_reg.evaluate(predictions_log_reg)
    if auc_log_reg > best_auc_log_reg:
        best_auc_log_reg = auc_log_reg
        best_log_reg = log_reg

print("\nHyperparameter tuning:\n")
```

```python
print(f"\nBest model hyperparameters:\nRegularization: {best_log_reg.getRegParam()}\nElastic Net:{best_log_reg.getElasticNetParam()}\nMax Iterations: {best_log_reg.getMaxIter()}")

"""# Model Performance Comparison

"""

print('--------------------- Performance ---------------------')

"""##Linear Regression with Dimensionality Reduction"""

print("\n\nLinear Regression Model with Dimensionality Reduction\n")

print('Performance on Testing dataset:\nMean Absolute Error: 17.173301\nMean Squared Error: 988.582871\nRMSE: 31.441738\nR-Squared: 0.133443\nAdjusted R-Squared: 0.133438')

"""##Linear Regression without Dimensionality Reduction"""

print("\nLinear Regression Model without Dimensionality Reduction\n")

print('Performance on Testing dataset:\nMean Absolute Error: 17.173301\nMean Absolute Error: 0.0007213546550619718\nMean Squared Error:  2.696448142195325e-06\nRoot Mean Squared Error: 0.0016420865209224892\nR squared:  0.9599999976525955\nAdjusted R sqaured: 0.9449999976525311')

"""##Logistic Regression with Dimensionality Reduction"""

print("\n\nLogistic Regression Model with Dimensionality Reduction\n")

#Getting predictions for best model
predictions_log_reg_pca_test = best_log_reg_pca.transform(test_log_reg_pca)
best_log_reg_pca_auc_test = evaluator_log_reg.evaluate(predictions_log_reg_pca_test)

print("ROC AUC for Logistic Regression with Dimensionality Reduction:", best_log_reg_pca_auc_test)

"""##Logistic Regression without Dimensionality Reduction"""

print("\n\nLogistic Regression Model without Dimensionality Reduction\n")

#Getting predictions for best model
predictions_log_reg_test = best_log_reg.transform(test_log_reg)
best_log_reg_auc_test = evaluator_log_reg.evaluate(predictions_log_reg_test)

print("ROC AUC for Logistic Regression without Dimensionality Reduction:", best_log_reg_auc_test)

"""#Best Models"""

print('--------------------- Best Models ---------------------')

"""##Regression"""

print('------- Regression -------')

print("Best Model Accuracy for Regression: 94%")

"""##Classification"""

print('------- Classification -------')

print("Best Model ROC AUC for Classification:", best_log_reg_auc_test)
```

```
23/05/01 04:42:38 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
23/05/01 04:42:39 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
23/05/01 04:42:39 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
23/05/01 04:42:39 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
23/05/01 04:42:39 INFO org.sparkproject.jetty.util.log: Logging initialized @3980ms to
org.sparkproject.jetty.util.log.Slf4jLog
23/05/01 04:42:39 INFO org.sparkproject.jetty.server.Server: jetty-9.4.40.v20210413; built:
2021-04-13T20:42:42.668Z; git: b881a572662e1943a14ae12e7e1207989f218b74; jvm 1.8.0_362-b09
23/05/01 04:42:39 INFO org.sparkproject.jetty.server.Server: Started @4108ms
23/05/01 04:42:39 INFO org.sparkproject.jetty.server.AbstractConnector: Started
ServerConnector@49f94106{HTTP/1.1, (http/1.1)}{0.0.0.0:46741}
23/05/01 04:42:42 INFO
com.google.cloud.hadoop.repackaged.gcs.com.google.cloud.hadoop.gcsio.GoogleCloudStorageImpl: Ignoring
exception of type GoogleJsonResponseException; verified object already exists with desired state.

Spark session instantied

Dataset read successfully

------- Linear Regression -------

Using all variables

Data cleaning initiated

Data cleaning completed

+------------+--------+---------+--------+----------+---------+-------+------------+--------+---------+---------+--------+----------------+------
-------------+--------+--------+------------+------------+---------+-------------+------------------+-----------------+-----------------+-------
------+------------------+----------------------+------------+----------+-------------+-------------+-------------+-------------+-------------+----
---------+-------------+-------------+-------------+-------------+
|CRS_DEP_TIME|DEP_TIME|DEP_DELAY|TAXI_OUT|WHEELS_OFF|WHEELS_ON|TAXI_IN|CRS_ARR_TIME|AR
R_TIME|ARR_DELAY|CANCELLED|DIVERTED|CRS_ELAPSED_TIME|ACTUAL_ELAPSED_TIME|AIR_TIME|DIST
ANCE|CARRIER_DELAY|WEATHER_DELAY|NAS_DELAY|SECURITY_DELAY|LATE_AIRCRAFT_DELAY|CARRIE
R_DELAY_BIN|WEATHER_DELAY_BIN|NAS_DELAY_BIN|SECURITY_DELAY_BIN|LATE_AIRCRAFT_DELAY_BIN|
ORIGIN_TRAFF|DEST_TRAFF|OP_CARRIER_AA|OP_CARRIER_AS|OP_CARRIER_B6|OP_CARRIER_DL|OP_C
ARRIER_EV|OP_CARRIER_F9|OP_CARRIER_HA|OP_CARRIER_OO|OP_CARRIER_UA|OP_CARRIER_WN|
+------------+--------+---------+--------+----------+---------+-------+------------+--------+---------+---------+--------+----------------+------
-------------+--------+--------+------------+------------+---------+-------------+------------------+-----------------+-----------------+-------
------+------------------+----------------------+------------+----------+-------------+-------------+-------------+-------------+-------------+----
---------+-------------+-------------+-------------+-------------+
|     1040.0|  1032.0|     -8.0|     9.0|    1041.0|   1316.0|    4.0|      1340.0|  1320.0|    -20.0|      0.0|     0.0|           120.0|
108.0|    95.0|   690.0|         0.0|          0.0|      0.0|           0.0|                0.0|             0.0|              0.0|          0.0|
0.0|                   0.0|         0.5|       0.5|          0.0|          0.0|          0.0|          0.0|          0.0|          0.0|          0.0|
1.0|          0.0|          0.0|
|     1405.0|  1355.0|    -10.0|    17.0|    1412.0|   1519.0|    6.0|      1530.0|  1525.0|     -5.0|      0.0|     0.0|           145.0|
150.0|   127.0|   690.0|         0.0|          0.0|      0.0|           0.0|                0.0|             0.0|              0.0|          0.0|
0.0|                   0.0|         0.5|       0.5|          0.0|          0.0|          0.0|          0.0|          0.0|          0.0|          0.0|
1.0|          0.0|          0.0|
|     1555.0|  1554.0|     -1.0|    11.0|    1605.0|   1836.0|    7.0|      1900.0|  1843.0|    -17.0|      0.0|     0.0|           125.0|
109.0|    91.0|   690.0|         0.0|          0.0|      0.0|           0.0|                0.0|             0.0|              0.0|          0.0|
0.0|                   0.0|         0.5|       0.5|          0.0|          0.0|          0.0|          0.0|          0.0|          0.0|          0.0|
1.0|          0.0|          0.0|
```

```
|      1105.0|  1105.0|      0.0|    14.0|    1119.0|   1221.0|    3.0|      1230.0|  1224.0|     -6.0|      0.0|     0.0|           145.0|
139.0|   122.0|   690.0|          0.0|          0.0|      0.0|           0.0|              0.0|             0.0|             0.0|          0.0|
0.0|                 0.0|        0.5|       0.5|          0.0|          0.0|          0.0|          0.0|          0.0|          0.0|          0.0|
1.0|          0.0|          0.0|
|      1335.0|  1345.0|     10.0|    11.0|    1356.0|   1415.0|    4.0|      1412.0|  1419.0|      7.0|      0.0|     0.0|            37.0|
34.0|    19.0|    74.0|          0.0|          0.0|      0.0|           0.0|              0.0|             0.0|             0.0|          0.0|
0.0|                 0.0|        0.5|       0.0|          0.0|          0.0|          0.0|          0.0|          0.0|          0.0|          0.0|
1.0|          0.0|          0.0|
+------------+--------+---------+--------+----------+---------+-------+------------+--------+---------+---------+--------+----------------+------
-------------+--------+--------+-------------+-------------+---------+--------------+-------------------+-----------------+-----------------+-------
------+------------------+----------------------+------------+----------+-------------+-------------+-------------+-------------+-------------+----
---------+-------------+-------------+-------------+-------------+
only showing top 5 rows

Vector Assembler creation begun

Vector Assembler created completed, data transformed

MLR data set created

Dataset split into train and test

MLR model instantiated

Model fitting to training data begun

Model fit to training data successfully

                    Coefficients
CRS_DEP_TIME        -9.333249e-06
DEP_TIME             2.256738e-05
DEP_DELAY            1.131265e-01
TAXI_OUT             5.810468e-02
WHEELS_OFF          -1.317183e-05
WHEELS_ON           -5.840972e-07
TAXI_IN              5.809578e-02
CRS_ARR_TIME         1.096382e-06
ARR_TIME            -6.051003e-07
CRS_ELAPSED_TIME    -1.131673e-01
ACTUAL_ELAPSED_TIME  5.506292e-02
AIR_TIME             5.814569e-02
DISTANCE            -5.417757e-06
CARRIER_DELAY        4.486659e-06
WEATHER_DELAY        6.123771e-06
NAS_DELAY           -6.272075e-06
SECURITY_DELAY       1.591311e-06
LATE_AIRCRAFT_DELAY -6.226412e-07
ORIGIN_TRAFF         1.285625e-03
DEST_TRAFF           1.181710e-03
OP_CARRIER_AA       -2.398951e-04
OP_CARRIER_AS        3.152159e-04
OP_CARRIER_B6        2.311828e-04
OP_CARRIER_DL       -7.076057e-05
OP_CARRIER_EV        5.463380e-04
```

OP_CARRIER_F9	-6.254697e-04
OP_CARRIER_HA	9.283329e-04
OP_CARRIER_OO	6.776289e-05
OP_CARRIER_UA	-3.098700e-04
OP_CARRIER_WN	-4.297856e-05

Intercept: -0.0012381106352197858

Model Evaluation begun
Model Evaluation completed

Training Data
Mean Absolute Error: 0.000723
Mean Squared Error: 0.000003
RMSE: 0.001655
R-Squared: 1.000000
Adjusted R-Squared: 1.000000

Testing Data
Mean Absolute Error: 0.0007213546550619718
Mean Squared Error: 2.696448142195325e-06
Root Mean Squared Error: 0.0016420865209224892
R squared: 0.9599999976525955
Adjusted R sqaured: 0.9449999976525311

23/05/01 05:44:01 WARN com.github.fommil.netlib.LAPACK: Failed to load implementation from: com.github.fommil.netlib.NativeSystemLAPACK
23/05/01 05:44:01 WARN com.github.fommil.netlib.LAPACK: Failed to load implementation from: com.github.fommil.netlib.NativeRefLAPACK

Best k value: 2

23/05/01 05:50:30 WARN com.github.fommil.netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
23/05/01 05:50:30 WARN com.github.fommil.netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS

Using Dimensionality Reduction and Regularisation

Dataset split into train and test
MLR model instantiated

Model fitting to training data begun

Model fit to training data successfully

	Coefficients
CRS_DEP_TIME	0.0
TAXI_OUT	0.0
TAXI_IN	0.0
CRS_ARR_TIME	0.0
CRS_ELAPSED_TIME	1.0
DISTANCE	0.0
ORIGIN_TRAFF	0.0
DEST_TRAFF	0.0

OP_CARRIER_AA	0.0
OP_CARRIER_AS	0.0
OP_CARRIER_B6	0.0
OP_CARRIER_DL	0.0
OP_CARRIER_EV	0.0
OP_CARRIER_F9	0.0
OP_CARRIER_HA	0.0
OP_CARRIER_OO	0.0
OP_CARRIER_UA	0.0
OP_CARRIER_WN	0.0

Intercept: 0.0

Model Evaluation begun
Model Evaluation completed

Training Data
Mean Absolute Error: 0.000000
Mean Squared Error: 0.000000
RMSE: 0.000000
R-Squared: 1.000000

Testing Data
Adjusted R-Squared: 1.000000
Mean Absolute Error: 0.0
Mean Squared Error: 0.0
Root Mean Squared Error: 0.0
R squared: 1.0
Adjusted R sqaured: 1.0

Dataset split into train and test
MLR model instantiated

Model fitting to training data begun

Model fit to training data successfully

	Coefficients
CRS_DEP_TIME	0.006169
TAXI_OUT	1.198117
TAXI_IN	0.932700
CRS_ARR_TIME	0.003122
CRS_ELAPSED_TIME	-0.309651
DISTANCE	0.034168
ORIGIN_TRAFF	-4.098446
DEST_TRAFF	0.888210
OP_CARRIER_AA	-41.562879
OP_CARRIER_AS	-44.736233
OP_CARRIER_B6	-44.811236
OP_CARRIER_DL	-48.343110
OP_CARRIER_EV	-40.457123
OP_CARRIER_F9	-40.633211
OP_CARRIER_HA	-46.049364
OP_CARRIER_OO	-43.553584
OP_CARRIER_UA	-45.078464

OP_CARRIER_WN -39.022691
Intercept: 26.387492015311935

Model Evaluation begun
Model Evaluation completed

Training Data
Mean Absolute Error: 17.173301
Mean Squared Error: 988.582871
RMSE: 31.441738
R-Squared: 0.133443
Adjusted R-Squared: 0.133438

Testing Data
Mean Absolute Error: 17.146089274729377
Mean Squared Error: 1001.4420533920813
Root Mean Squared Error: 31.645569253721465
R squared: 0.13061515289375725
Adjusted R sqaured: 0.13060129830235145

------- Logistic Regression -------
Dataset used for logistic regression:

```
+------------+--------+-------+------------+----------------+--------+------------+----------+-------------+-------------+-------------+-----------
--+-------------+-------------+-------------+-------------+-------------+-------------+-----+
|CRS_DEP_TIME|TAXI_OUT|TAXI_IN|CRS_ARR_TIME|CRS_ELAPSED_TIME|DISTANCE|ORIGIN_TRAFF|DEST_
TRAFF|OP_CARRIER_AA|OP_CARRIER_AS|OP_CARRIER_B6|OP_CARRIER_DL|OP_CARRIER_EV|OP_CARRI
ER_F9|OP_CARRIER_HA|OP_CARRIER_OO|OP_CARRIER_UA|OP_CARRIER_WN|DELAY|
+------------+--------+-------+------------+----------------+--------+------------+----------+-------------+-------------+-------------+-----------
--+-------------+-------------+-------------+-------------+-------------+-------------+-----+
|      1040.0|     9.0|    4.0|      1340.0|           120.0|   690.0|         0.5|       0.5|          0.0|          0.0|          0.0|          0.0|
0.0|          0.0|          0.0|          1.0|          0.0|          0.0|  0.0|
|      1405.0|    17.0|    6.0|      1530.0|           145.0|   690.0|         0.5|       0.5|          0.0|          0.0|          0.0|          0.0|
0.0|          0.0|          0.0|          1.0|          0.0|          0.0|  0.0|
|      1555.0|    11.0|    7.0|      1900.0|           125.0|   690.0|         0.5|       0.5|          0.0|          0.0|          0.0|          0.0|
0.0|          0.0|          0.0|          1.0|          0.0|          0.0|  0.0|
|      1105.0|    14.0|    3.0|      1230.0|           145.0|   690.0|         0.5|       0.5|          0.0|          0.0|          0.0|          0.0|
0.0|          0.0|          0.0|          1.0|          0.0|          0.0|  0.0|
|      1335.0|    11.0|    4.0|      1412.0|            37.0|    74.0|         0.5|       0.0|          0.0|          0.0|          0.0|          0.0|
0.0|          0.0|          0.0|          1.0|          0.0|          0.0|  1.0|
|      1437.0|    10.0|    5.0|      1515.0|            38.0|    74.0|         0.0|       0.5|          0.0|          0.0|          0.0|          0.0|
0.0|          0.0|          0.0|          1.0|          0.0|          0.0|  0.0|
|      1605.0|    12.0|    7.0|      1643.0|            38.0|    74.0|         0.5|       0.0|          0.0|          0.0|          0.0|          0.0|
0.0|          0.0|          0.0|          1.0|          0.0|          0.0|  0.0|
|      1333.0|    13.0|    7.0|      1515.0|            42.0|   120.0|         0.5|       0.0|          0.0|          0.0|          0.0|          0.0|
0.0|          0.0|          0.0|          1.0|          0.0|          0.0|  0.0|
|      1317.0|    10.0|    5.0|      1300.0|            43.0|   120.0|         0.0|       0.5|          0.0|          0.0|          0.0|          0.0|
0.0|          0.0|          0.0|          1.0|          0.0|          0.0|  0.0|
|      1550.0|    12.0|    7.0|      1735.0|            45.0|   120.0|         0.5|       0.0|          0.0|          0.0|          0.0|          0.0|
0.0|          0.0|          0.0|          1.0|          0.0|          0.0|  0.0|
|      1540.0|     8.0|    4.0|      1525.0|            45.0|   120.0|         0.0|       0.5|          0.0|          0.0|          0.0|          0.0|
0.0|          0.0|          0.0|          1.0|          0.0|          0.0|  0.0|
|      1320.0|    14.0|    6.0|      1453.0|            93.0|   475.0|         0.5|       0.5|          0.0|          0.0|          0.0|          0.0|
0.0|          0.0|          0.0|          1.0|          0.0|          0.0|  0.0|
```

```
|     1518.0|     9.0|   30.0|     1651.0|          93.0|   475.0|       0.5|      0.5|        0.0|        0.0|        0.0|        0.0|
0.0|        0.0|        0.0|        1.0|        0.0|        0.0|  0.0|
|     1055.0|    18.0|    5.0|     1215.0|          80.0|   426.0|       0.5|      0.5|        0.0|        0.0|        0.0|        0.0|
0.0|        0.0|        0.0|        1.0|        0.0|        0.0|  1.0|
|     1555.0|    12.0|    7.0|     1720.0|          85.0|   426.0|       0.5|      0.5|        0.0|        0.0|        0.0|        0.0|
0.0|        0.0|        0.0|        1.0|        0.0|        0.0|  1.0|
|     1310.0|    19.0|    4.0|     1510.0|          60.0|   238.0|       0.5|      0.5|        0.0|        0.0|        0.0|        0.0|
0.0|        0.0|        0.0|        1.0|        0.0|        0.0|  0.0|
|     1535.0|    15.0|    4.0|     1535.0|          60.0|   238.0|       0.5|      0.5|        0.0|        0.0|        0.0|        0.0|
0.0|        0.0|        0.0|        1.0|        0.0|        0.0|  0.0|
|     1238.0|    10.0|    4.0|     1235.0|          57.0|   211.0|       0.0|      0.5|        0.0|        0.0|        0.0|        0.0|
0.0|        0.0|        0.0|        1.0|        0.0|        0.0|  1.0|
|     1645.0|    10.0|    6.0|     1835.0|          50.0|   211.0|       0.5|      0.0|        0.0|        0.0|        0.0|        0.0|
0.0|        0.0|        0.0|        1.0|        0.0|        0.0|  1.0|
|     1141.0|     8.0|    4.0|     1215.0|          34.0|    96.0|       0.0|      0.5|        0.0|        0.0|        0.0|        0.0|
0.0|        0.0|        0.0|        1.0|        0.0|        0.0|  0.0|
+-----------+--------+-------+-----------+--------------+--------+-----------+----------+-------------+-------------+-------------+-----------
--+-------------+-------------+-------------+-------------+-------------+-------------+-----+
only showing top 20 rows
```

Dimensionality Reduction with PCA

23/05/01 07:24:31 WARN com.github.fommil.netlib.LAPACK: Failed to load implementation from: com.github.fommil.netlib.NativeSystemLAPACK
23/05/01 07:24:31 WARN com.github.fommil.netlib.LAPACK: Failed to load implementation from: com.github.fommil.netlib.NativeRefLAPACK

Best k value: 2

23/05/01 07:35:01 WARN com.github.fommil.netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
23/05/01 07:35:01 WARN com.github.fommil.netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS

Logistic Regression Model with Dimensionality Reduction

Hyperparameter tuning:

Best model hyperparameters:
Regularization: 1.0
Elastic Net:0.0
Max Iterations: 10

Logistic Regression Model with Dimensionality Reduction

Hyperparameter tuning:

Best model hyperparameters:
Regularization: 0.01
Elastic Net:0.0
Max Iterations: 100
-------------------- Performance --------------------

Linear Regression Model without Dimensionality Reduction
Performance on Testing dataset:
Mean Absolute Error: 17.173301
Mean Absolute Error: 0.0007213546550619718
Mean Squared Error: 2.696448142195325e-06
Root Mean Squared Error: 0.0016420865209224892
R squared: 0.9599999976525955
Adjusted R sqaured: 0.9449999976525311

Logistic Regression Model with Dimensionality Reduction

ROC AUC for Logistic Regression with Dimensionality Reduction: 0.5815632870291384

Logistic Regression Model without Dimensionality Reduction

ROC AUC for Logistic Regression without Dimensionality Reduction: 0.7151097694902129
--------------------- Best Models ---------------------
------- Regression -------
Best Model Accuracy for Regression: 94%
------- Classification -------
Best Model ROC AUC for Classification: 0.7151097694902129

23/05/01 11:30:22 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spark@49f94106{HTTP/1.1, (http/1.1)}{0.0.0.0:0}