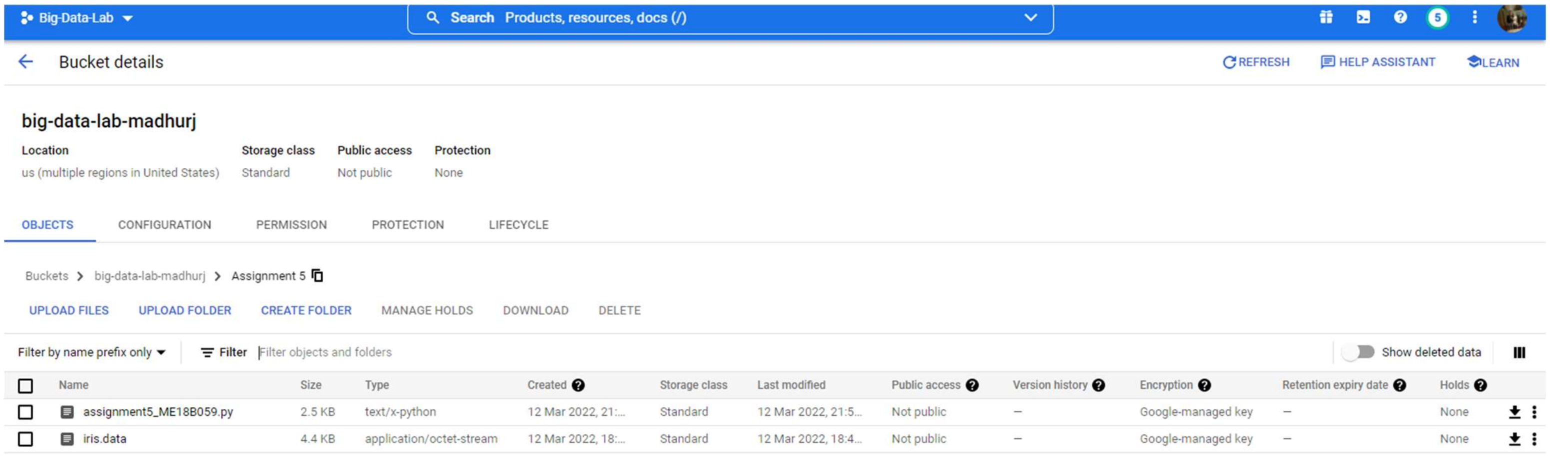
Big Data Lab | Assignment 5 | Madhur Jindal | ME18B059

1. PFA the screenshot to location of the data file and the code for task 3 in the bucket.



big-data-lab-madhurj

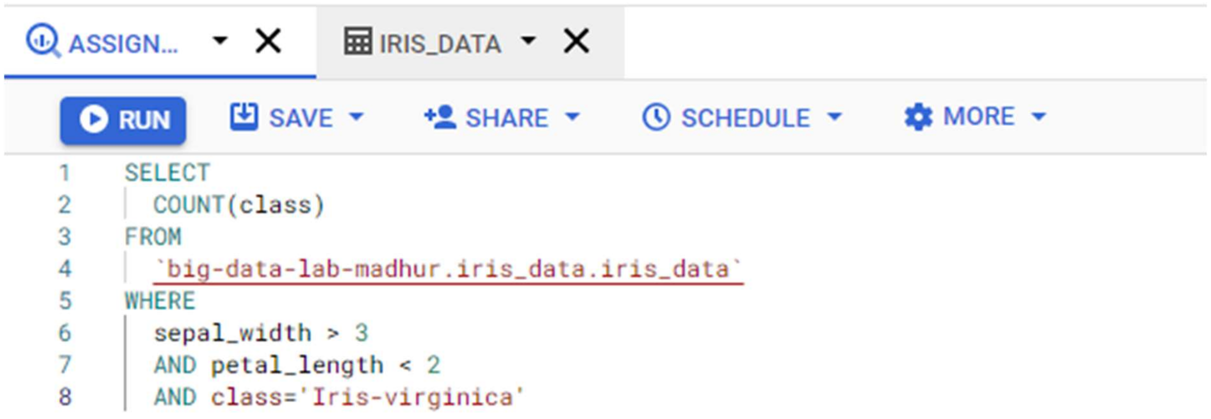
Location	Storage class	Public access	Protection
us (multiple regions in United States)	Standard	Not public	None

Buckets > big-data-lab-madhurj > Assignment 5

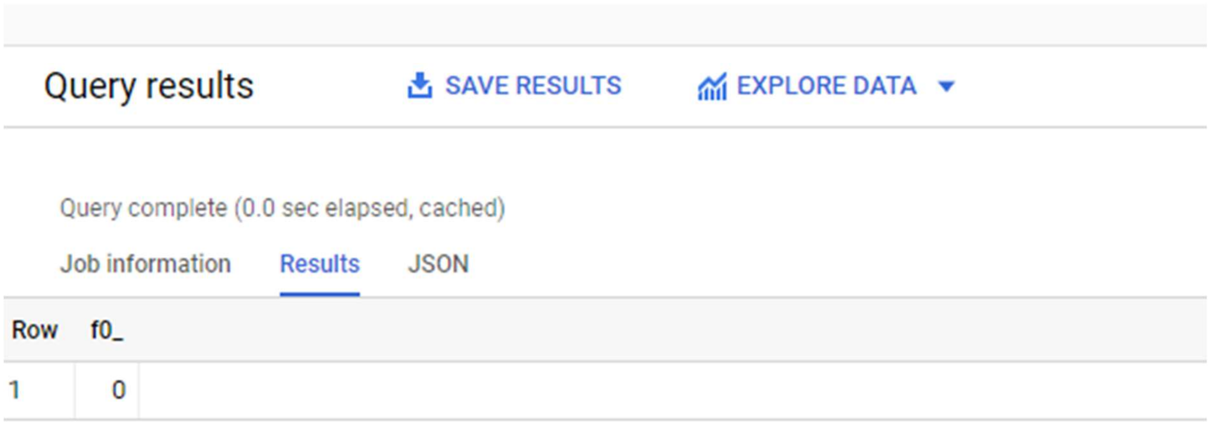
Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption	Retention expiry date	Holds
assignment5_ME18B059.py	2.5 KB	text/x-python	12 Mar 2022, 21:...	Standard	12 Mar 2022, 21:5...	Not public	–	Google-managed key	–	None
iris.data	4.4 KB	application/octet-stream	12 Mar 2022, 18:...	Standard	12 Mar 2022, 18:4...	Not public	–	Google-managed key	–	None

2. PFA the screenshot to the query for task 2 along with the answer as 0. Hence there do not exist any flowers of the given configuration - Iris Virginica flowers which have sepal width greater than 3 cm and petal length smaller than 2 cm.

Big Data Lab | Assignment 5 | Madhur Jindal | ME18B059



```sql
SELECT
  COUNT(class)
FROM
  `big-data-lab-madhur.iris_data.iris_data`
WHERE
  sepal_width > 3
  AND petal_length < 2
  AND class='Iris-virginica'
```



Query results

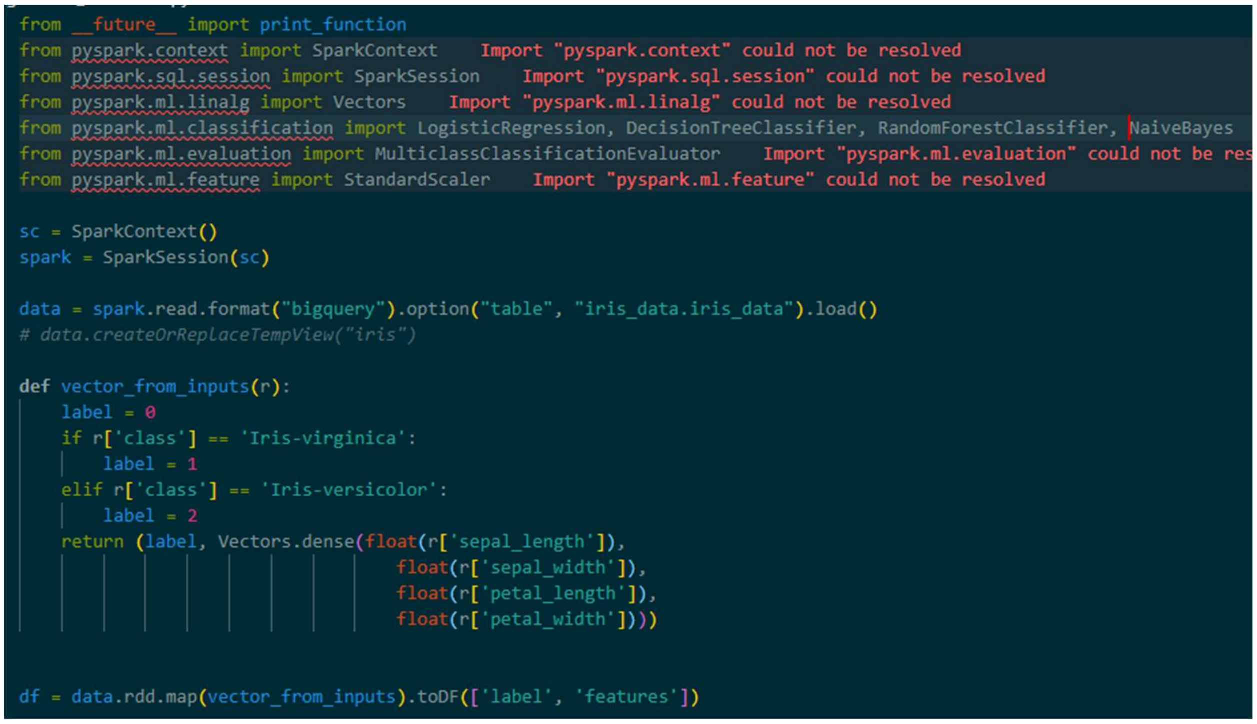
Query complete (0.0 sec elapsed, cached)

Row	f0_
1	0

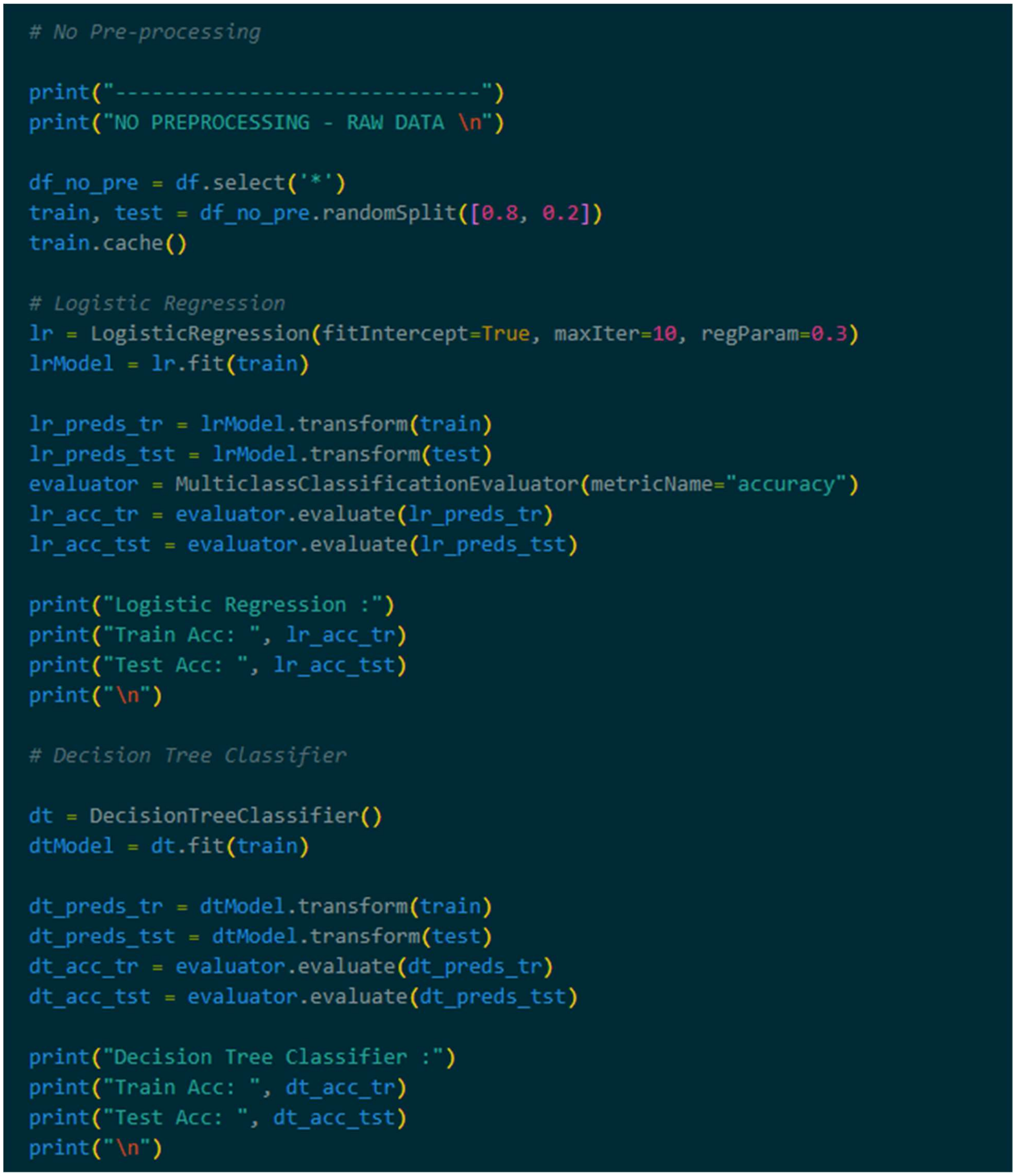
3. The code for the Task 3 is given in the zip named as assignment5_ME18B059.py file. We try to use different classification models along with different pre-processing techniques on the given IRIS dataset and try to predict the class of the iris plant.

   a. We start with importing the required libraries and setting up the Spark Context for connection with spark and also setting up the Spark Session using sc for setting up the connection with sql engine.

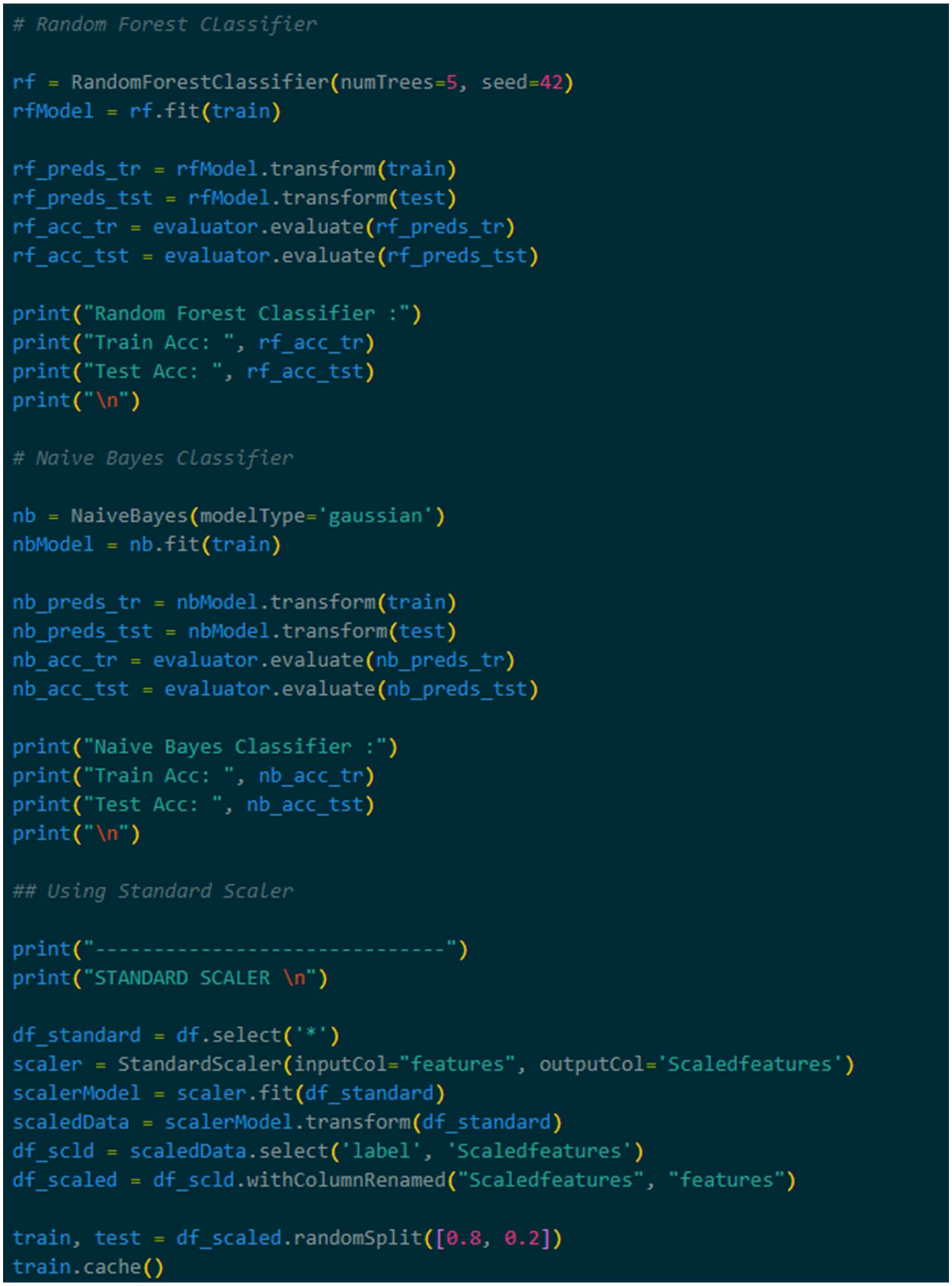
b. Then we read the table using spark sql from our saved tables as an rdd.
c. Then we transform the data in the class column to encoded format and also use Vector.dense to transform the other columns to a features column. Finally, we create a Spark DataFrame from the rdd after applying the transform.
d. We use randomSplit method to create a 4:1 train test split and use MultiClassClassificationEvaluator for calculating accuracy.
e. Following that we try training different models with no pre-processing on the raw data and also using Standard Scaler for transformation.
f. The different models that were trained are:
    a. Logistic Regression
    b. Decision Tree Classifier
    c. Random Forest Classifier
    d. Naïve Bayes Classifier
g. The training and test accuracies for the combinations have been provided.



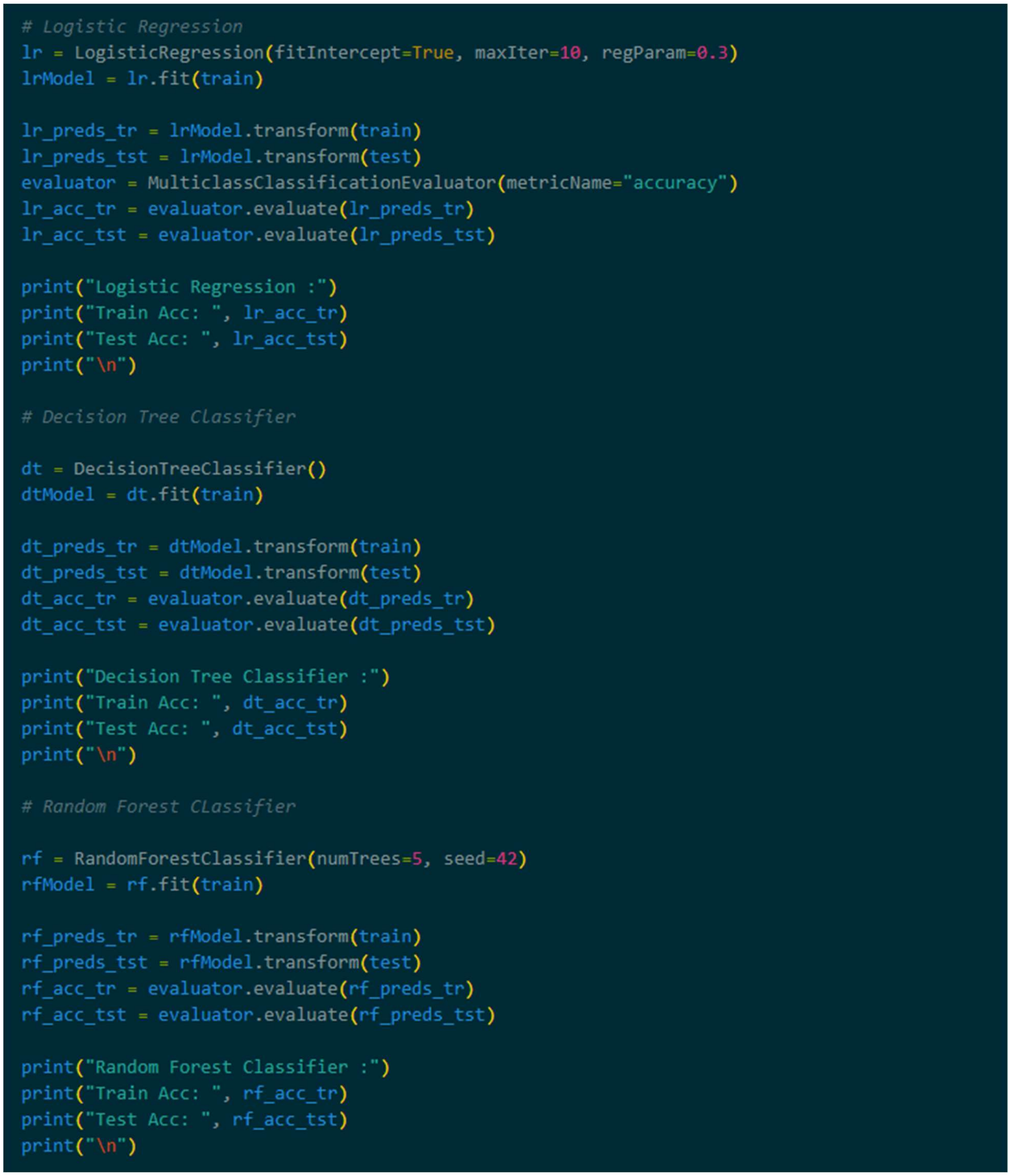
```python
from __future__ import print_function
from pyspark.context import SparkContext    Import "pyspark.context" could not be resolved
from pyspark.sql.session import SparkSession    Import "pyspark.sql.session" could not be resolved
from pyspark.ml.linalg import Vectors    Import "pyspark.ml.linalg" could not be resolved
from pyspark.ml.classification import LogisticRegression, DecisionTreeClassifier, RandomForestClassifier, NaiveBayes
from pyspark.ml.evaluation import MulticlassClassificationEvaluator    Import "pyspark.ml.evaluation" could not be res
from pyspark.ml.feature import StandardScaler    Import "pyspark.ml.feature" could not be resolved

sc = SparkContext()
spark = SparkSession(sc)

data = spark.read.format("bigquery").option("table", "iris_data.iris_data").load()
# data.createOrReplaceTempView("iris")

def vector_from_inputs(r):
    label = 0
    if r['class'] == 'Iris-virginica':
        label = 1
    elif r['class'] == 'Iris-versicolor':
        label = 2
    return (label, Vectors.dense(float(r['sepal_length']),
                                 float(r['sepal_width']),
                                 float(r['petal_length']),
                                 float(r['petal_width'])))


df = data.rdd.map(vector_from_inputs).toDF(['label', 'features'])
```



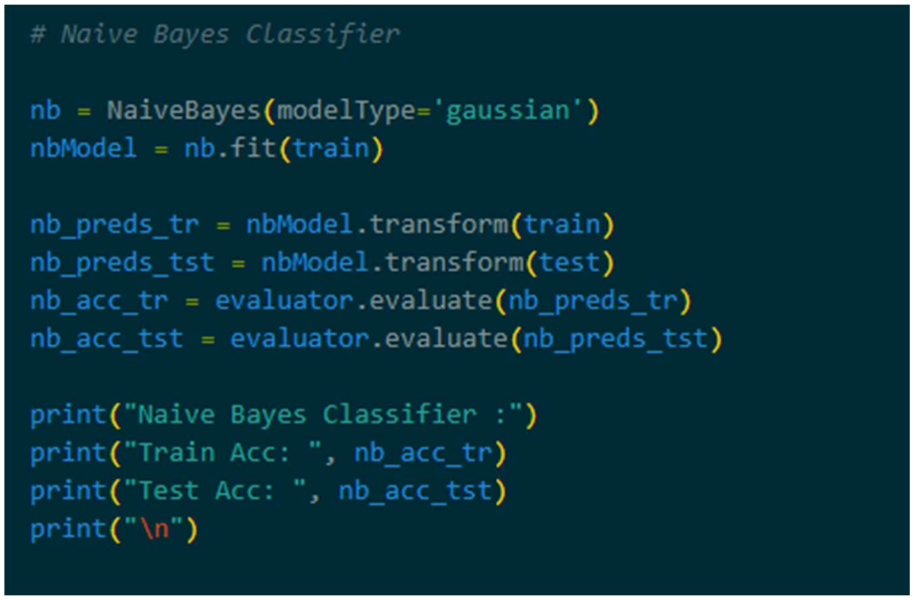
```
# No Pre-processing

print("------------------------------")
print("NO PREPROCESSING - RAW DATA \n")

df_no_pre = df.select('*')
train, test = df_no_pre.randomSplit([0.8, 0.2])
train.cache()

# Logistic Regression
lr = LogisticRegression(fitIntercept=True, maxIter=10, regParam=0.3)
lrModel = lr.fit(train)

lr_preds_tr = lrModel.transform(train)
lr_preds_tst = lrModel.transform(test)
evaluator = MulticlassClassificationEvaluator(metricName="accuracy")
lr_acc_tr = evaluator.evaluate(lr_preds_tr)
lr_acc_tst = evaluator.evaluate(lr_preds_tst)

print("Logistic Regression :")
print("Train Acc: ", lr_acc_tr)
print("Test Acc: ", lr_acc_tst)
print("\n")

# Decision Tree Classifier

dt = DecisionTreeClassifier()
dtModel = dt.fit(train)

dt_preds_tr = dtModel.transform(train)
dt_preds_tst = dtModel.transform(test)
dt_acc_tr = evaluator.evaluate(dt_preds_tr)
dt_acc_tst = evaluator.evaluate(dt_preds_tst)

print("Decision Tree Classifier :")
print("Train Acc: ", dt_acc_tr)
print("Test Acc: ", dt_acc_tst)
print("\n")
```



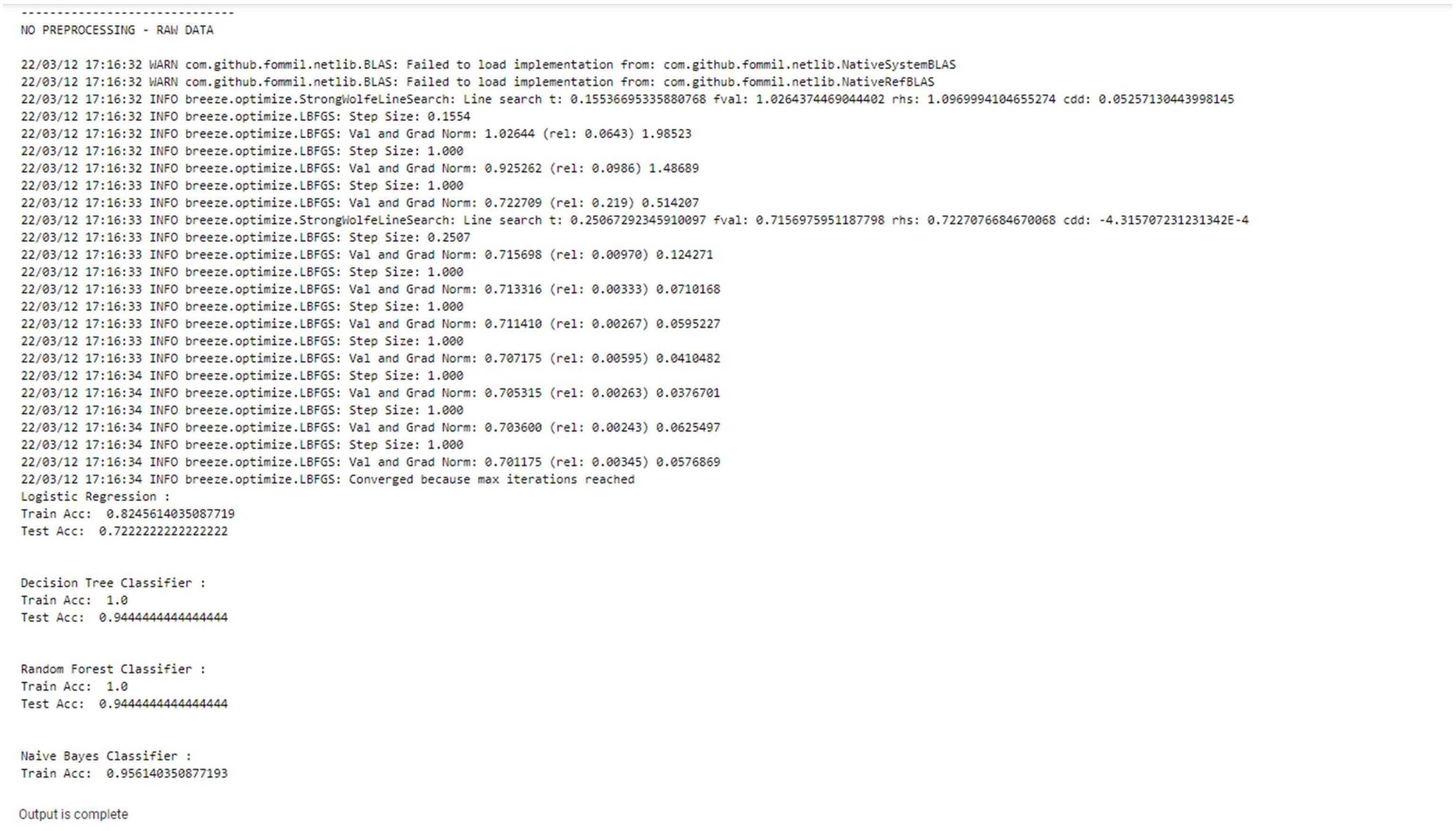
```
# Random Forest Classifier

rf = RandomForestClassifier(numTrees=5, seed=42)
rfModel = rf.fit(train)

rf_preds_tr = rfModel.transform(train)
rf_preds_tst = rfModel.transform(test)
rf_acc_tr = evaluator.evaluate(rf_preds_tr)
rf_acc_tst = evaluator.evaluate(rf_preds_tst)

print("Random Forest Classifier :")
print("Train Acc: ", rf_acc_tr)
print("Test Acc: ", rf_acc_tst)
print("\n")

# Naive Bayes Classifier

nb = NaiveBayes(modelType='gaussian')
nbModel = nb.fit(train)

nb_preds_tr = nbModel.transform(train)
nb_preds_tst = nbModel.transform(test)
nb_acc_tr = evaluator.evaluate(nb_preds_tr)
nb_acc_tst = evaluator.evaluate(nb_preds_tst)

print("Naive Bayes Classifier :")
print("Train Acc: ", nb_acc_tr)
print("Test Acc: ", nb_acc_tst)
print("\n")

## Using Standard Scaler

print("-----------------------------")
print("STANDARD SCALER \n")

df_standard = df.select('*')
scaler = StandardScaler(inputCol="features", outputCol='Scaledfeatures')
scalerModel = scaler.fit(df_standard)
scaledData = scalerModel.transform(df_standard)
df_scld = scaledData.select('label', 'Scaledfeatures')
df_scaled = df_scld.withColumnRenamed("Scaledfeatures", "features")

train, test = df_scaled.randomSplit([0.8, 0.2])
train.cache()
```



```python
# Logistic Regression
lr = LogisticRegression(fitIntercept=True, maxIter=10, regParam=0.3)
lrModel = lr.fit(train)

lr_preds_tr = lrModel.transform(train)
lr_preds_tst = lrModel.transform(test)
evaluator = MulticlassClassificationEvaluator(metricName="accuracy")
lr_acc_tr = evaluator.evaluate(lr_preds_tr)
lr_acc_tst = evaluator.evaluate(lr_preds_tst)

print("Logistic Regression :")
print("Train Acc: ", lr_acc_tr)
print("Test Acc: ", lr_acc_tst)
print("\n")

# Decision Tree Classifier

dt = DecisionTreeClassifier()
dtModel = dt.fit(train)

dt_preds_tr = dtModel.transform(train)
dt_preds_tst = dtModel.transform(test)
dt_acc_tr = evaluator.evaluate(dt_preds_tr)
dt_acc_tst = evaluator.evaluate(dt_preds_tst)

print("Decision Tree Classifier :")
print("Train Acc: ", dt_acc_tr)
print("Test Acc: ", dt_acc_tst)
print("\n")

# Random Forest CLassifier

rf = RandomForestClassifier(numTrees=5, seed=42)
rfModel = rf.fit(train)

rf_preds_tr = rfModel.transform(train)
rf_preds_tst = rfModel.transform(test)
rf_acc_tr = evaluator.evaluate(rf_preds_tr)
rf_acc_tst = evaluator.evaluate(rf_preds_tst)

print("Random Forest Classifier :")
print("Train Acc: ", rf_acc_tr)
print("Test Acc: ", rf_acc_tst)
print("\n")
```



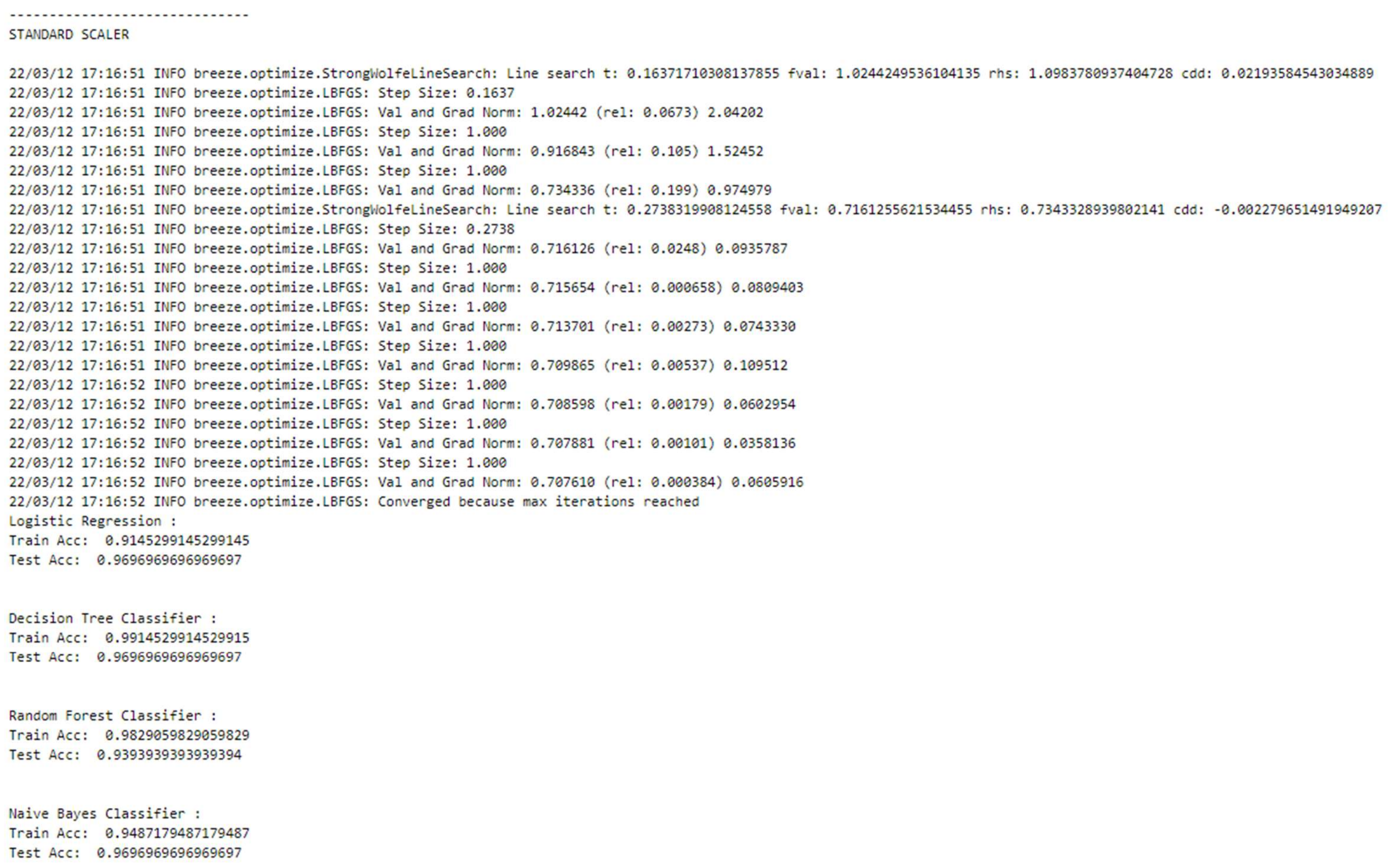
```
# Naive Bayes Classifier

nb = NaiveBayes(modelType='gaussian')
nbModel = nb.fit(train)

nb_preds_tr = nbModel.transform(train)
nb_preds_tst = nbModel.transform(test)
nb_acc_tr = evaluator.evaluate(nb_preds_tr)
nb_acc_tst = evaluator.evaluate(nb_preds_tst)

print("Naive Bayes Classifier :")
print("Train Acc: ", nb_acc_tr)
print("Test Acc: ", nb_acc_tst)
print("\n")
```



```
------------------------------
NO PREPROCESSING - RAW DATA

22/03/12 17:16:32 WARN com.github.fommil.netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
22/03/12 17:16:32 WARN com.github.fommil.netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
22/03/12 17:16:32 INFO breeze.optimize.StrongWolfeLineSearch: Line search t: 0.15536695335880768 fval: 1.0264374469044402 rhs: 1.0969994104655274 cdd: 0.05257130443998145
22/03/12 17:16:32 INFO breeze.optimize.LBFGS: Step Size: 0.1554
22/03/12 17:16:32 INFO breeze.optimize.LBFGS: Val and Grad Norm: 1.02644 (rel: 0.0643) 1.98523
22/03/12 17:16:32 INFO breeze.optimize.LBFGS: Step Size: 1.000
22/03/12 17:16:32 INFO breeze.optimize.LBFGS: Val and Grad Norm: 0.925262 (rel: 0.0986) 1.48689
22/03/12 17:16:33 INFO breeze.optimize.LBFGS: Step Size: 1.000
22/03/12 17:16:33 INFO breeze.optimize.LBFGS: Val and Grad Norm: 0.722709 (rel: 0.219) 0.514207
22/03/12 17:16:33 INFO breeze.optimize.StrongWolfeLineSearch: Line search t: 0.25067292345910097 fval: 0.7156975951187798 rhs: 0.7227076684670068 cdd: -4.315707231231342E-4
22/03/12 17:16:33 INFO breeze.optimize.LBFGS: Step Size: 0.2507
22/03/12 17:16:33 INFO breeze.optimize.LBFGS: Val and Grad Norm: 0.715698 (rel: 0.00970) 0.124271
22/03/12 17:16:33 INFO breeze.optimize.LBFGS: Step Size: 1.000
22/03/12 17:16:33 INFO breeze.optimize.LBFGS: Val and Grad Norm: 0.713316 (rel: 0.00333) 0.0710168
22/03/12 17:16:33 INFO breeze.optimize.LBFGS: Step Size: 1.000
22/03/12 17:16:33 INFO breeze.optimize.LBFGS: Val and Grad Norm: 0.711410 (rel: 0.00267) 0.0595227
22/03/12 17:16:33 INFO breeze.optimize.LBFGS: Step Size: 1.000
22/03/12 17:16:33 INFO breeze.optimize.LBFGS: Val and Grad Norm: 0.707175 (rel: 0.00595) 0.0410482
22/03/12 17:16:34 INFO breeze.optimize.LBFGS: Step Size: 1.000
22/03/12 17:16:34 INFO breeze.optimize.LBFGS: Val and Grad Norm: 0.705315 (rel: 0.00263) 0.0376701
22/03/12 17:16:34 INFO breeze.optimize.LBFGS: Step Size: 1.000
22/03/12 17:16:34 INFO breeze.optimize.LBFGS: Val and Grad Norm: 0.703600 (rel: 0.00243) 0.0625497
22/03/12 17:16:34 INFO breeze.optimize.LBFGS: Step Size: 1.000
22/03/12 17:16:34 INFO breeze.optimize.LBFGS: Val and Grad Norm: 0.701175 (rel: 0.00345) 0.0576869
22/03/12 17:16:34 INFO breeze.optimize.LBFGS: Converged because max iterations reached
Logistic Regression :
Train Acc:  0.8245614035087719
Test Acc:  0.7222222222222222


Decision Tree Classifier :
Train Acc:  1.0
Test Acc:  0.9444444444444444


Random Forest Classifier :
Train Acc:  1.0
Test Acc:  0.9444444444444444


Naive Bayes Classifier :
Train Acc:  0.956140350877193
```

Output is complete

Big Data Lab | Assignment 5 | Madhur Jindal | ME18B059



```
------------------------------
STANDARD SCALER

22/03/12 17:16:51 INFO breeze.optimize.StrongWolfeLineSearch: Line search t: 0.16371710308137855 fval: 1.0244249536104135 rhs: 1.0983780937404728 cdd: 0.02193584543034889
22/03/12 17:16:51 INFO breeze.optimize.LBFGS: Step Size: 0.1637
22/03/12 17:16:51 INFO breeze.optimize.LBFGS: Val and Grad Norm: 1.02442 (rel: 0.0673) 2.04202
22/03/12 17:16:51 INFO breeze.optimize.LBFGS: Step Size: 1.000
22/03/12 17:16:51 INFO breeze.optimize.LBFGS: Val and Grad Norm: 0.916843 (rel: 0.105) 1.52452
22/03/12 17:16:51 INFO breeze.optimize.LBFGS: Step Size: 1.000
22/03/12 17:16:51 INFO breeze.optimize.LBFGS: Val and Grad Norm: 0.734336 (rel: 0.199) 0.974979
22/03/12 17:16:51 INFO breeze.optimize.StrongWolfeLineSearch: Line search t: 0.2738319908124558 fval: 0.7161255621534455 rhs: 0.7343328939802141 cdd: -0.002279651491949207
22/03/12 17:16:51 INFO breeze.optimize.LBFGS: Step Size: 0.2738
22/03/12 17:16:51 INFO breeze.optimize.LBFGS: Val and Grad Norm: 0.716126 (rel: 0.0248) 0.0935787
22/03/12 17:16:51 INFO breeze.optimize.LBFGS: Step Size: 1.000
22/03/12 17:16:51 INFO breeze.optimize.LBFGS: Val and Grad Norm: 0.715654 (rel: 0.000658) 0.0809403
22/03/12 17:16:51 INFO breeze.optimize.LBFGS: Step Size: 1.000
22/03/12 17:16:51 INFO breeze.optimize.LBFGS: Val and Grad Norm: 0.713701 (rel: 0.00273) 0.0743330
22/03/12 17:16:51 INFO breeze.optimize.LBFGS: Step Size: 1.000
22/03/12 17:16:51 INFO breeze.optimize.LBFGS: Val and Grad Norm: 0.709865 (rel: 0.00537) 0.109512
22/03/12 17:16:52 INFO breeze.optimize.LBFGS: Step Size: 1.000
22/03/12 17:16:52 INFO breeze.optimize.LBFGS: Val and Grad Norm: 0.708598 (rel: 0.00179) 0.0602954
22/03/12 17:16:52 INFO breeze.optimize.LBFGS: Step Size: 1.000
22/03/12 17:16:52 INFO breeze.optimize.LBFGS: Val and Grad Norm: 0.707881 (rel: 0.00101) 0.0358136
22/03/12 17:16:52 INFO breeze.optimize.LBFGS: Step Size: 1.000
22/03/12 17:16:52 INFO breeze.optimize.LBFGS: Val and Grad Norm: 0.707610 (rel: 0.000384) 0.0605916
22/03/12 17:16:52 INFO breeze.optimize.LBFGS: Converged because max iterations reached
Logistic Regression :
Train Acc:  0.9145299145299145
Test Acc:  0.9696969696969697


Decision Tree Classifier :
Train Acc:  0.9914529914529915
Test Acc:  0.9696969696969697


Random Forest Classifier :
Train Acc:  0.9829059829059829
Test Acc:  0.9393939393939394


Naive Bayes Classifier :
Train Acc:  0.9487179487179487
Test Acc:  0.9696969696969697
```