CS 5433: Bigdata Management
Programming Assignment 3
Task2 – Program File for Random Forest Algorithm

Group 4

Task 2: Implement prediction algorithm using (a) Linear regression (b) Random Forest and Clearly identify the variable you are predicting. Predict on the same variable for both algorithms.

Part B : Random Forest Algorithm

```
#importing the required packages
## importing all libraries SparkSession,SQLContext
from pyspark.sql import SparkSession,SQLContext
# importing SparkConf,SparkContext
from pyspark import SparkConf,SparkContext
## importing -- StringIndexer,VectorAssembler
from pyspark.ml.feature import StringIndexer,VectorAssembler
## Importing --- RegressionEvaluator
from pyspark.ml.evaluation import RegressionEvaluator
## Importing -- MinMaxScaler
from pyspark.ml.feature import MinMaxScaler
## Importing -- udf
from pyspark.sql.functions import udf
## Importing -- Diuble type
from pyspark.sql.types import DoubleType
## Importing ---- PipeLines
from pyspark.ml import Pipeline
## Importing ---- All miscellancious library from sql types
from pyspark.sql.types import *
## Importing vecor_to_array
from pyspark.ml.functions import vector_to_array
## Importing concat_ws and col ---
from pyspark.sql.functions import concat_ws,col
## Importing Random ForrestRegressor
from pyspark.ml.regression import RandomForestRegressor
## Importing PAndas
import pandas as pd



#Creating the sparkconfiguration
spark = SparkSession.builder.appName("Assignment3_Group4_Task2_PartB").getOrCreate()
#Reading the csv file along with the header data into a dataframe df
df = spark.read.csv("hdfs://hadoop-
nn001.cs.okstate.edu:9000/user/sdarapu/Assign3_Group4_Task1_Output_inpfor_Task2-
4/part-00000-571e77d2-85ae-4579-92f8-dd4dc788ab7f-c000.csv", header = True,
inferSchema = True)
```

```
#printing the schema of the dataframe
df.printSchema()
#converting the String columns into double data type
#InsID_indexer=StringIndexer(inputCol="Institute ID", outputCol="INSTITUTE ID")
#df=InsID_indexer.fit(df).transform(df)
#Name_indexer=StringIndexer(inputCol="Name", outputCol="NAME")
#df=Name_indexer.fit(df).transform(df)
#State_indexer=StringIndexer(inputCol="State", outputCol="STATE")
#df=State_indexer.fit(df).transform(df)
#City_indexer=StringIndexer(inputCol="City", outputCol="CITY")
#df=City_indexer.fit(df).transform(df)
#priting the modified schema
df.printSchema()

## creatin unwarapped list using the udf
UN_WRAPPER = udf(lambda x: round(float(list(x)[0]),3), DoubleType())

## creating col_list witht the required columns --
col_list = ["NAME","STATE","Score","PR Rank","INSTITUTE ID"]

indexx = 0
## Iterating over a while loop
while indexx < len(col_list):
    ## creating the assembler from vector
    assembler =
VectorAssembler(inputCols=[col_list[indexx]],outputCol=col_list[indexx]+"_Vect")

    # MinMaxScaler Transformation

    scaler = MinMaxScaler(inputCol=col_list[indexx]+"_Vect",
outputCol=col_list[indexx]+"_Scaled")

    # Pipeline of VectorAssembler and MinMaxScaler

    pipeline = Pipeline(stages=[assembler, scaler])

    # Fitting pipeline on dataframe

    df = pipeline.fit(df).transform(df).withColumn(col_list[indexx]+"_Scaled",
UN_WRAPPER(col_list[indexx]+"_Scaled")).drop(col_list[indexx]+"_Vect")

    ## incrementer by 1
    indexx = indexx +1

#using vector assembler of coverting the columns into vectors that are used for
prediction
## Vector Assembler ---
vectorAssembler = VectorAssembler(inputCols = df.columns[9:], outputCol = 'features')
```

```
## tranforming the data frame
vectorAssembler.setParams(handleInvalid="skip")#transforming the data frame
## tranforming the data frame
transform_output=vectorAssembler.transform(df)
#selecting the features and Year column and storing into final_df dataframe
final_df=transform_output.select('features','Year')
#splitting the data into training and test data using randomSplit function
(trainingData, testData) = final_df.randomSplit([0.7, 0.3],80)

#printing the number of rows for training and testing data
print("Number of train records are",trainingData.count())
## Printing the testData count
print("Number of test records are",testData.count())
## displaying the traning Data
trainingData.describe().show()
## Displaying the testData
testData.describe().show()
## Using the Random Regressor
## Forming Random Regressor
rf=RandomForestRegressor(featuresCol = 'features', labelCol='Year')
## setting the maxBins
rf.setMaxBins(190)
## random model with tranining Data
rf_model=rf.fit(trainingData)
## lr transforming
rf_predictions = rf_model.transform(testData)
#### random forest predictions
rf=rf_predictions.select("prediction","Year","features")
#####  random forest result
rf.show(200)
## vector to array
rf = rf.withColumn('features', vector_to_array('features'))
# array to string
rf = rf.withColumn("features",concat_ws(",",col("features")))
## writing output
rf.coalesce(1).write.mode("overwrite").option("header","true").csv("hdfs:////user/sdar
apu/Assign3_Group4_Task2_PartB_Output")
```