DS 6003 – Spark Assignment

Charu Rawat
Computing ID: cr4zy

Motivation

Each year, over 3 million college applications are filed in the US by about 750,000 students, an average of 4 applications per student. Each of them comes with a certain element of randomness or chance. The intended meritocracy inherent in college admissions gives way to uncertainty, doubt, and anxiety, even for students with exceptional credentials. There are many factors that influence admission decisions. Using regression analysis on the given graduate admissions data can give us an idea as to the likelihood of a student getting admitted into a university given the strength of their profile. The predicted output can be helpful to students as it can give them a fair idea about their chances for admission to a particular university. My motivation for choosing this dataset was to primarily fulfil this objective. As a student who has gone through the rigors of the admissions process, I believe such an analysis can be very helpful in gauging expectations and also aid students in smartly planning the admission application process.
The data was sourced from kaggle.com. Originally, this data was extracted from the applicant's database of UCLA. This dataset consists of multiple quantitative measures of a student's performance such as GRE and TOEFL scores which are crucial in determining admissions into institutions.

Process and Code Walkthrough

The pipeline for this analysis comprised of the following steps-

- Initializing the spark environment
- Reading the data from the csv into a spark dataframe
- Creating s3 bucket and dumping the data there
- Writing parquet to s3 bucket and moving the data
- Creating a dataframe from parquet
- Exploring the data
- Model building – feature selection, vectorization
- Fitting a linear regression model
- Visualization of model fit
- Evaluation of predicted results

Initializing the spark environment

Import necessary modules/packages

In [1]:
```
import pyspark
from os import listdir
from os.path import isfile, join
import boto3
import pandas as pd
from sagemaker import get_execution_role
from pyspark.sql.types import LongType, StringType, StructField, StructType, BooleanType, ArrayType, IntegerType
import pyspark.sql.functions as sf
```

Initialize the spark environment

In [2]:
```
conf = pyspark.SparkConf().setAppName('odl').setMaster('local')
sc = pyspark.SparkContext(conf=conf)
sqlc = pyspark.sql.SQLContext(sc)
sc
```

Out[2]:

SparkContext

Spark UI

Version
`v2.2.1`
Master
`local`
AppName
`odl`

Creating a folder in s3 bucket with the raw data dumped there and then reading the data into a dataframe

Read into spark dataframe from csv

In [4]:
```
role = get_execution_role()
bucket='odl-spark19spds6003-001'
data_key = 'cr4zy/admissions_data.csv'
data_location = 's3://{}/{}'.format(bucket, data_key)
#pd.read_csv(data_location)
```

In [5]:
```
df = sqlc.createDataFrame(pd.read_csv(data_location))
df
```

Out[5]:
```
DataFrame[ID: bigint, GRE: bigint, TOEFL: bigint, UniRating: bigint, SOP: double, LOR: double, CGPA: double, Research: bigint,
Admission: double]
```

Writing data to parquet and reading it into a dataframe which will be used for the analysis

Write parque to s3

In [6]:
```
parquetPath = '/home/ec2-user/SageMaker/cr4zy/parquet_tmp'
df.write.parquet(parquetPath)
```

In [7]:
```
# prep list of files to transfer
files = [f for f in listdir(parquetPath) if isfile(join(parquetPath, f))]

s3 = boto3.resource('s3')
for f in files:
    #print('copying {} to {}'.format(parquetPath+'/'+f,"sample_data/"+f))
    s3.Bucket(bucket).upload_file(parquetPath+'/'+f, "cr4zy/"+f)
```

Make dataframe from parquet

In [8]:
```
# write to spark df
df = sqlc.read.parquet(parquetPath)
df
```

Out[8]:
```
DataFrame[ID: bigint, GRE: bigint, TOEFL: bigint, UniRating: bigint, SOP: double, LOR: double, CGPA: double, Research: bigint,
Admission: double]
```

Exploring relationship between the variables

Correlation between variables

In [11]:
```
# Dependent variable is Admission (Chance of admit)
print("Pearson's r(GRE,TOEFL) = {}".format(df.corr("GRE", "TOEFL")))
print("Pearson's r(SOP,LOR) = {}".format(df.corr("SOP", "LOR")))
print("Pearson's r(CGPA,Admission) = {}".format(df.corr("CGPA", "Admission")))
print("Pearson's r(GRE,Admission) = {}".format(df.corr("GRE", "Admission")))
print("Pearson's r(SOP,Admission) = {}".format(df.corr("SOP", "Admission")))
```

```
Pearson's r(GRE,TOEFL) = 0.8359768030143965
Pearson's r(SOP,LOR) = 0.7295925366175839
Pearson's r(CGPA,Admission) = 0.8732890993553003
Pearson's r(GRE,Admission) = 0.8026104595903504
Pearson's r(SOP,Admission) = 0.6757318583886724
```

Building a model - features are selected and then the data is split into the training and testing set. Post that, vectorization is performed on the dataframe columns and a vectorassembler is used to rename the columns and divide into label and features

Model Building

In [13]:
```
# create train/test sets
seed = 42
(testDF, trainingDF) = df.randomSplit((0.40, 0.60), seed=seed)
print ('training set N = {}, test set N = {}'.format(trainingDF.count(),testDF.count()))
```

```
training set N = 252, test set N = 148
```

Vectorization

In [14]:
```
from pyspark.ml.linalg import Vectors, VectorUDT # nb: bad form, done for pedagogy
```

In [15]:
```
#vectorization
# make a user defined function (udf)
sqlc.registerFunction("oneElementVec", lambda d: Vectors.dense([d]), returnType=VectorUDT())

# vectorize the data frames
trainingDF = trainingDF.selectExpr("Admission", "oneElementVec(GRE) as GRE","oneElementVec(TOEFL) as TOEFL" ,
                                   "oneElementVec(SOP) as SOP","oneElementVec(LOR) as LOR","oneElementVec(CGPA) as CGPA"
testDF = testDF.selectExpr("Admission", "oneElementVec(GRE) as GRE","oneElementVec(TOEFL) as TOEFL" ,
                                   "oneElementVec(SOP) as SOP","oneElementVec(LOR) as LOR","oneElementVec(CGPA) as CGPA")
print(testDF.orderBy(testDF.Admission.desc()).limit(5))
```

In [16]:
```
# rename to make ML engine happy
trainingDF = trainingDF.withColumnRenamed("Admission", "label")
testDF = testDF.withColumnRenamed("Admission", "label")
```

In [17]:
```
from pyspark.ml.feature import VectorAssembler
vectorAssembler = VectorAssembler(inputCols = ['GRE', 'TOEFL' ,'SOP', 'LOR', 'CGPA'], outputCol = 'features')
trainingDF = vectorAssembler.transform(trainingDF)
trainingDF = trainingDF.select(['features', 'label'])
#trainingDF.show(3)

testDF = vectorAssembler.transform(testDF)
testDF = testDF.select(['features', 'label'])
testDF.show(3)
```

A linear regression model is fit on the dataframe. After obtaining a fit, we transform our test set to get predictions. A prediction column gets appended to the dataframe

Fitting a linear regression model

1. Train
2. Predict
3. Evaluate

In [18]:
```
from pyspark.ml.regression import LinearRegression, LinearRegressionModel

lr = LinearRegression()
lrModel = lr.fit(trainingDF)
```

In [19]:
```
trainingDF.take(5)
```

Out[19]:
```
[Row(features=DenseVector([290.0, 100.0, 1.5, 2.0, 7.56]), label=0.47),
 Row(features=DenseVector([290.0, 104.0, 2.0, 2.5, 7.46]), label=0.45),
 Row(features=DenseVector([293.0, 97.0, 2.0, 4.0, 7.8]), label=0.64),
 Row(features=DenseVector([294.0, 93.0, 1.5, 2.0, 7.36]), label=0.46),
 Row(features=DenseVector([294.0, 95.0, 1.5, 1.5, 7.64]), label=0.49)]
```

In [20]:
```
# We are now going to transform our test set to get predictions.
# It will append a prediction column to testDF in the new dataframe predictionsAndLabelsDF.
predictionsAndLabelsDF = lrModel.transform(testDF)
print(predictionsAndLabelsDF.orderBy(predictionsAndLabelsDF.label.desc()))
```

```
DataFrame[features: vector, label: double, prediction: double]
```

Model is evaluated on certain measures like R2, RMSE etc using the RegressionEvaluator function

Model Evaluation

```
In [24]: from pyspark.ml.evaluation import RegressionEvaluator
         eval = RegressionEvaluator()
         print(eval.explainParams())
```

```
labelCol: label column name. (default: label)
metricName: metric name in evaluation - one of:
                    rmse - root mean squared error (default)
                    mse - mean squared error
                    r2 - r^2 metric
                    mae - mean absolute error. (default: rmse)
predictionCol: prediction column name. (default: prediction)
```

```
In [25]: type(eval)
```

```
Out[25]: pyspark.ml.evaluation.RegressionEvaluator
```

```
In [26]: print("RMSE: %g" % eval.setMetricName("rmse").evaluate(predictionsAndLabelsDF))
         print("R-Square: %g" %  eval.setMetricName("r2").evaluate(predictionsAndLabelsDF))
         print("Mean Square Error: %g" % eval.setMetricName("mse").evaluate(predictionsAndLabelsDF))
```

```
RMSE: 0.0643891
R-Square: 0.762574
Mean Square Error: 0.00414595
```

Visualization

This is a plot of the residual's vs the fitted values which is ideally supposed to give us information regarding the validity of the linear assumption, variance of the error terms and point out outliers as well. In this plot, we don't see a uniform horizontal band of the residuals around 0 which could point towards the non-constant variance of the error terms. Significant outlier points are also visible from the pot. Overall, it suggests that improvements could further be made to the model.