# PySpark Environment Setting

```
# Please run this cell to get Java and spark installed
!apt-get update
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
!wget -q https://downloads.apache.org/spark/spark-2.4.7/spark-2.4.7-bin-hadoop2.7.tgz
!tar xf spark-2.4.7-bin-hadoop2.7.tgz
!pip install pyspark==2.4.7

import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-2.4.7-bin-hadoop2.7"
```

```
Get:1 https://cloud.r-project.org/bin/linux/ubuntu bionic-cran40/ InRelease [3,626 B]
Ign:2 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64  InRel
```

# ICT707 Task 3

## Please implement your assignment in this notebook.

```
Get:11 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
```

```
# Please enter your NAME and student ID
NAME = "GAGANDEEP KAUR"
ID = "1121869"
```

```
Get:13 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
```

# Connect GDrive for data set files

```
Get:20 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [1,733 kB]
```

```
# Mount the cloud folder for data file storage
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
Collecting py4j==0.10.7
```

```
# Make sure you have relevant data files uploaded
# And then use the correct data file names below
datafile = "/content/gdrive/My Drive/Colab Notebooks/ratings.csv"
```

```
  Created wheel for pyspark: filename=pyspark-2.4.7-py2.py3-none-any.whl size=218279466
```

# PART I: EDA

```
Successfully installed py4j-0.10.7 pyspark-2.4.7
```

```
# Code implementation
from pyspark import SparkContext
from pyspark.sql import SQLContext
sc = SparkContext()
sqlContext = SQLContext(sc)


df = sqlContext.read.csv(datafile,header=True,inferSchema=True)
df.show(10)
print("Total number of Rows",df.count())
```

```
+-------+-------+------+
|book_id|user_id|rating|
+-------+-------+------+
|      1|    314|     5|
|      1|    439|     3|
|      1|    588|     5|
|      1|   1169|     4|
|      1|   1185|     4|
|      1|   2077|     4|
|      1|   2487|     4|
|      1|   2900|     5|
```

```
df = df.sample(withReplacement = False,
                        fraction = 0.01, # 1% of observation
                        seed = 2019)
print("Total number of rows in current sample we are using", df.count())
```

```
Total number of rows in current sample we are using 9899
```

```
df.schema
```

```
StructType(List(StructField(book_id,IntegerType,true),StructField(user_id,IntegerType,t
```

```
df.columns
```

```
['book_id', 'user_id', 'rating']
```

# PART II: Collaborative filtering with Alternative Least Squares Algorithm

```
# Code implementation
training, test=df.randomSplit([0.6, 0.4])
print("number of rows in training dataset:",training.count())
print("number of rows in testing dataset:",test.count())
```

```
number of rows in training dataset: 5943
number of rows in testing dataset: 3956
```

```
from pyspark.ml.recommendation import ALS
als=ALS(userCol="user_id",itemCol="book_id",ratingCol="rating",
        nonnegative=True,
        coldStartStrategy="drop",
        implicitPrefs=False)
als_model=als.fit(training)
```

```
pred= als_model.transform(test)


from pyspark.ml.evaluation import RegressionEvaluator
evaluator = RegressionEvaluator(metricName="rmse", labelCol="rating",
                                predictionCol="prediction")
rmse = evaluator.evaluate(pred)
print ("RMSE: ", rmse)
```

```
RMSE:  2.841999713901338
```

## PART III: Classification system with Logistic regression

```
vector_df=df.withColumn("label",df.rating>3)
vector_df.show()
```

```
+-------+-------+------+-----+
|book_id|user_id|rating|label|
+-------+-------+------+-----+
|      1|  30681|     5| true|
|      2|  14546|     5| true|
|      3|   5885|     4| true|
|      8|  17228|     4| true|
|      8|  49288|     5| true|
|     10|   9246|     2|false|
|     10|  23612|     4| true|
|     10|  39423|     3|false|
|     10|  51166|     3|false|
|     14|   8484|     3|false|
|     14|  48559|     3|false|
|     14|  51460|     3|false|
|     17|   3922|     5| true|
|     20|  24845|     3|false|
|     21|   9771|     4| true|
|     24|  16569|     4| true|
|     24|  47800|     5| true|
|     27|    439|     5| true|
|     27|  19942|     4| true|
|     27|  45554|     3|false|
+-------+-------+------+-----+
only showing top 20 rows
```

```
from pyspark.ml.feature import VectorAssembler
assembler=VectorAssembler().setInputCols(["book_id","user_id","rating"]).setOutputCol("featur
ml_df=assembler.transform(vector_df)
ml_df=ml_df.select(ml_df.features,ml_df.label.astype('int'))
ml_df.show()
```

```
+------------------+-----+
|          features|label|
+------------------+-----+
| [1.0,30681.0,5.0]|    1|
| [2.0,14546.0,5.0]|    1|
|  [3.0,5885.0,4.0]|    1|
| [8.0,17228.0,4.0]|    1|
| [8.0,49288.0,5.0]|    1|
| [10.0,9246.0,2.0]|    0|
|[10.0,23612.0,4.0]|    1|
|[10.0,39423.0,3.0]|    0|
|[10.0,51166.0,3.0]|    0|
| [14.0,8484.0,3.0]|    0|
|[14.0,48559.0,3.0]|    0|
|[14.0,51460.0,3.0]|    0|
| [17.0,3922.0,5.0]|    1|
|[20.0,24845.0,3.0]|    0|
| [21.0,9771.0,4.0]|    1|
|[24.0,16569.0,4.0]|    1|
|[24.0,47800.0,5.0]|    1|
|  [27.0,439.0,5.0]|    1|
|[27.0,19942.0,4.0]|    1|
|[27.0,45554.0,3.0]|    0|
+------------------+-----+
only showing top 20 rows
```

```
train, test= ml_df.randomSplit([0.8, 0.2])
from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression(maxIter=10, regParam=0.3, elasticNetParam=0.8)
lr_model=lr.fit(train)
prediction=lr_model.transform(test)
```

```
from pyspark.ml.evaluation import BinaryClassificationEvaluator
scored_test=lr_model.transform(test)
scored_train=lr_model.transform(train)
evaluator=BinaryClassificationEvaluator()
```

```
evaluator.evaluate(scored_train,{evaluator.metricName:'areaUnderROC'})
```

```
1.0
```

```
evaluator.evaluate(scored_test,{evaluator.metricName:'areaUnderROC'})
```

```
1.0
```

## Shut down SparkContext when exiting

If you have error messages related to sparkContext, try to run the following cell, and then rerun all cells.

```
sc.stop()
```