Task I.1: Exploratory data analysis

Initializing a Spark session

In [1]:

```
import pandas as pd
from pyspark.sql import SparkSession
spark =
SparkSession.builder.master("local").appName("Movie_recommendation").config("spark.some.config.opti
on","some-value").getOrCreate()
```

The datasets can be easily downloaded from these links:

1. 'rating.csv': https://www.kaggle.com/rounakbanik/the-movies-dataset?select=ratings_small.csv
2. 'movies.csv': https://www.kaggle.com/rounakbanik/the-movies-dataset?select=movies_metadata.csv

Number of rows and columns:

In [2]:

```
rating = spark.read.format("csv").option("header","true").option("inferSchema", "true").load(r"C:\U
sers\74108\Downloads\ratings.csv")
rating = rating.drop('timestamp')

movies = spark.read.format("csv").option("header","true").option("inferSchema", "true").load(r"C:\U
sers\74108\Downloads\movies.csv")
movie_data = rating.join(movies, on='movieId')

Columns = len(movie_data.columns)
Rows = movie_data.count()
print('Number of Columns: {}\nNumber of Rows: {}'.format(Columns, Rows))
movie_data.columns
```

```
Number of Columns: 10
Number of Rows: 100836
```

Out[2]:

```
['movieId',
 'userId',
 'rating',
 'vote_average',
 'vote_count',
 'revenue',
 'budget',
 'adult',
 'title',
 'genres']
```

DATA CLEANING: Replace all the zeros in the abaove mentioned fields (except “Pregnancies”) with NaN.

In [3]:

```
import numpy as np
from pyspark.sql.functions import when

movie_data = movie_data.withColumn("userId",when(movie_data.userId==0,np.nan).otherwise(movie_data.
userId))
movie_data =
movie_data.withColumn("movieId",when(movie_data.movieId==0,np.nan).otherwise(movie_data.movieId))
movie_data = movie_data.withColumn("rating",when(movie_data.rating==0,np.nan).otherwise(movie_data.
rating))
movie_data =
movie_data.withColumn("title",when(movie_data.title==0,np.nan).otherwise(movie_data.title))
from pyspark.sql.types import IntegerType
movie_data = movie_data.withColumn("budget", movie_data["budget"].cast(IntegerType()))
```

```
movie_data.show()
```

```
+-------+------+------+------------+----------+---------+--------+-----+--------------------+------
-----------+
|movieId|userId|rating|vote_average|vote_count|  revenue|  budget|adult|               title|
genres|
+-------+------+------+------------+----------+---------+--------+-----+--------------------+------
-----------+
|    1.0|   1.0|   4.0|         7.7|      5415|373554033|30000000|FALSE|    Toy Story
(1995)|Adventure|Animati...|
|    3.0|   1.0|   4.0|         6.5|        92|        0|       0|FALSE|Grumpier Old Men ...|
omedy|Romance|
|    6.0|   1.0|   4.0|         7.7|      1886|187436818|60000000|FALSE|         Heat (1995)|Action
Crime|Thri...|
|   47.0|   1.0|   5.0|         5.4|       452|122195920|18000000|FALSE|Seven (a.k.a. Se7...|
Mystery|Thriller|
|   50.0|   1.0|   5.0|         8.1|      5915|327311859|33000000|FALSE|Usual Suspects,
T...|Crime|Mystery|Thr...|
|   70.0|   1.0|   3.0|         6.2|       192|        0|       0|FALSE|From Dusk Till
Da...|Action|Comedy|Hor...|
|  101.0|   1.0|   5.0|         6.7|         3|        0|       0|FALSE|Bottle Rocket
(1996)|Adventure|Comedy|...|
|  110.0|   1.0|   4.0|         6.8|         4|        0|       0|FALSE|   Braveheart (1995)|    Ac
ion|Drama|War|
|  151.0|   1.0|   5.0|         4.4|       105|        0|17000000|FALSE|      Rob Roy (1995)|Action
Drama|Roma...|
|  157.0|   1.0|   5.0|         5.1|        45|  9851610|50000000|FALSE|Canadian Bacon (1...|
Comedy|War|
|  163.0|   1.0|   5.0|         7.2|         5|        0|       0|FALSE|    Desperado
(1995)|Action|Romance|We...|
|  216.0|   1.0|   5.0|         5.8|        14|   472370|  150000|FALSE|Billy Madison (1995)|
Comedy|
|  223.0|   1.0|   3.0|         4.9|       261| 20350754|45000000|FALSE|       Clerks (1994)|
Comedy|
|  231.0|   1.0|   5.0|         4.6|         9|  5780000|14000000|FALSE|Dumb & Dumber (Du...|
Adventure|Comedy|
|  235.0|   1.0|   4.0|         5.6|       217|104324083|60000000|FALSE|      Ed Wood (1994)|
Comedy|Drama|
|  260.0|   1.0|   5.0|         5.3|         9|        0|       0|FALSE|Star Wars:
Episod...|Action|Adventure|...|
|  296.0|   1.0|   3.0|         7.0|       222|        0|18000000|FALSE| Pulp Fiction
(1994)|Comedy|Crime|Dram...|
|  316.0|   1.0|   3.0|         7.0|        11|        0| 5500000|FALSE|     Stargate (1994)|Action
Adventure|...|
|  333.0|   1.0|   5.0|         7.0|         2|        0|       0|FALSE|    Tommy Boy (1995)|
Comedy|
|  349.0|   1.0|   4.0|         4.7|         3|        0|       0|FALSE|Clear and
Present...|Action|Crime|Dram...|
+-------+------+------+------------+----------+---------+--------+-----+--------------------+------
-----------+
only showing top 20 rows
```

Correlations between independent variables using data visualization. Selecting different numeric and string columns, and drawing 1 plot (e.g. bar chart, histogram, boxplot, etc.) for each to summarise it

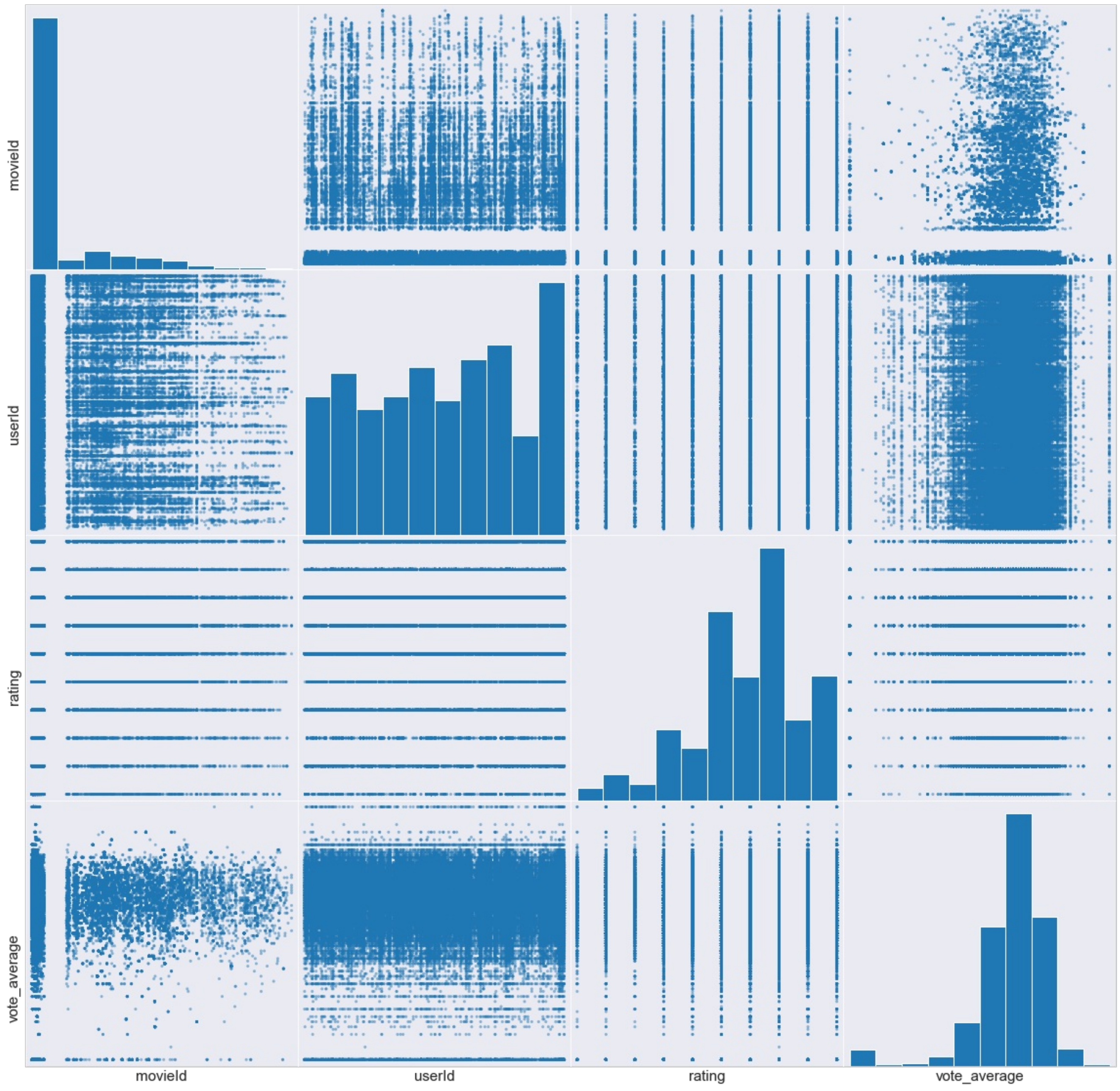
In [4]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.gridspec as gs
sns.set_style('dark')
%matplotlib inline


numeric_features = [t[0] for t in movie_data.dtypes if (t[1] == 'double') or (t[1] == 'str') ]
movie_data.select(numeric_features).describe().toPandas().transpose()

numeric_data = movie_data.select(numeric_features).toPandas()
axs = pd.plotting.scatter_matrix(numeric_data, figsize=(20, 20));
n = len(numeric_data.columns)
for i in range(n):
    v = axs[i, 0]
    v.yaxis.label.set_size(15)
    v.yaxis.label.set_ha('right')
    v.set_yticks(())
    h = axs[n-1, i]
    h.xaxis.label.set_size(15)
    h.set_xticks(())
```

```
C:\Users\74108\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\tools.py:298:
MatplotlibDeprecationWarning:
The rowNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later
. Use ax.get_subplotspec().rowspan.start instead.
  layout[ax.rowNum, ax.colNum] = ax.get_visible()
C:\Users\74108\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\tools.py:298:
MatplotlibDeprecationWarning:
The colNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later
. Use ax.get_subplotspec().colspan.start instead.
  layout[ax.rowNum, ax.colNum] = ax.get_visible()
C:\Users\74108\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\tools.py:304:
MatplotlibDeprecationWarning:
The rowNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later
. Use ax.get_subplotspec().rowspan.start instead.
  if not layout[ax.rowNum + 1, ax.colNum]:
C:\Users\74108\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\tools.py:304:
MatplotlibDeprecationWarning:
The colNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later
. Use ax.get_subplotspec().colspan.start instead.
  if not layout[ax.rowNum + 1, ax.colNum]:
```



Task I.2: Recommendation engine

Build the recommendation model using ALS on the training data

In [5]:

```
(training,test)=movie_data.randomSplit([0.8, 0.2]) # split into training and testing sets
```

This subtask requires you to implement a recommender system on Collaborative filtering with Alternative Least Squares Algorithm.

In [6]:

```
# Build the recommendation model using ALS on the training data
#Fitting the Alternating Least Squares Model

from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.recommendation import ALS

# Note we set cold start strategy to 'drop' to ensure we don't get NaN evaluation metrics
als = ALS(maxIter=5,regParam=0.09,rank=25,userCol="userId",itemCol="movieId",ratingCol="rating",col
dStartStrategy="drop",nonnegative=True)
model = als.fit(training) # fit the ALS model to the training set
```

Generating Predictions & Model Evaluation: Evaluating a model is a core part of building an effective machine learning model. In PySpark we will be using RMSE(Root mean squared Error) as our evaluation metric. The RMSE described our error in terms of the rating column.

In [7]:

```
evaluator=RegressionEvaluator(metricName="rmse",labelCol="rating",predictionCol="prediction")
predictions=model.transform(test)
rmse=evaluator.evaluate(predictions)
print("RMSE="+str(rmse))
predictions.show()
```

```
RMSE=0.8780123904093605
+-------+------+------+------------+----------+--------+--------+-----+--------------------+-------
----------+----------+
|movieId|userId|rating|vote_average|vote_count| revenue|  budget|adult|               title|
genres|prediction|
+-------+------+------+------------+----------+--------+--------+-----+--------------------+-------
----------+----------+
|  471.0| 602.0|   4.0|         5.1|        42|21011500|17080000|FALSE|Hudsucker Proxy, ...|
Comedy| 3.2045844|
|  471.0| 599.0|   2.5|         5.1|        42|21011500|17080000|FALSE|Hudsucker Proxy, ...|
Comedy| 2.8620176|
|  471.0| 474.0|   3.0|         5.1|        42|21011500|17080000|FALSE|Hudsucker Proxy, ...|
Comedy| 3.5551262|
|  471.0| 136.0|   4.0|         5.1|        42|21011500|17080000|FALSE|Hudsucker Proxy, ...|
Comedy| 3.5937843|
|  471.0| 469.0|   5.0|         5.1|        42|21011500|17080000|FALSE|Hudsucker Proxy, ...|
Comedy| 3.3475266|
|  833.0| 599.0|   1.5|         5.4|        46|       0|       0|FALSE|High School High ...|
Comedy| 1.5394657|
| 1088.0| 599.0|   2.5|         6.6|        97| 3011195| 2962051|FALSE|Dirty Dancing
(1987)|Drama|Musical|Rom...|  2.811266|
| 1088.0| 479.0|   4.0|         6.6|        97| 3011195| 2962051|FALSE|Dirty Dancing
(1987)|Drama|Musical|Rom...| 2.6540227|
| 1088.0| 387.0|   1.5|         6.6|        97| 3011195| 2962051|FALSE|Dirty Dancing
(1987)|Drama|Musical|Rom...| 2.6207194|
| 1088.0| 555.0|   4.0|         6.6|        97| 3011195| 2962051|FALSE|Dirty Dancing
(1987)|Drama|Musical|Rom...| 3.3191097|
| 1088.0|  51.0|   4.0|         6.6|        97| 3011195| 2962051|FALSE|Dirty Dancing
(1987)|Drama|Musical|Rom...|    3.9688|
| 1088.0| 226.0|   1.0|         6.6|        97| 3011195| 2962051|FALSE|Dirty Dancing
(1987)|Drama|Musical|Rom...|  3.612993|
| 1088.0| 414.0|   3.0|         6.6|        97| 3011195| 2962051|FALSE|Dirty Dancing
(1987)|Drama|Musical|Rom...| 3.2411554|
| 1088.0|  42.0|   3.0|         6.6|        97| 3011195| 2962051|FALSE|Dirty Dancing
```

```
(1987)|Drama|Musical|Rom...| 3.6864858|
| 1088.0| 104.0|   3.0|         6.6|        97| 3011195| 2962051|FALSE|Dirty Dancing
(1987)|Drama|Musical|Rom...| 3.5908737|
| 1238.0| 268.0|   5.0|         7.4|       217|       0|       0|FALSE|   Local Hero (1983)|
Comedy| 3.8799565|
| 1342.0| 599.0|   2.5|         6.8|       255| 4505922|  200000|FALSE|     Candyman (1992)|
Horror|Thriller| 2.3511457|
| 1342.0| 608.0|   2.0|         6.8|       255| 4505922|  200000|FALSE|     Candyman (1992)|
Horror|Thriller| 3.0458868|
| 1580.0|  34.0|   2.5|         7.3|       118|       0|       0|FALSE|Men in Black
(a.k...|Action|Comedy|Sci-Fi| 2.6829376|
| 1580.0| 183.0|   4.0|         7.3|       118|       0|       0|FALSE|Men in Black (a.k...|Action|
omedy|Sci-Fi| 3.7511015|
+-------+------+------+------------+----------+--------+--------+-----+--------------------+-------
----------+----------+
only showing top 20 rows
```

Recommending Movies with ALS: The approach here will be simple We will be taking a single userid example 29 as features and pass it to trained ALS Model. The same way we did with the test data!

In [ ]:

```python
single_user = test.filter(test['userId']==29).select(['movieId','userId','title','genres'])
# User had 10 ratings in the test data set
# Realistically this should be some sort of hold out set!
single_user.show()
```

In [ ]:

```python
#Now we will use model.transform() function in order to generate recommended movies along with the
ir predicted features.

recomendations = model.transform(single_user)
recomendations.orderBy('prediction',ascending=False).show()
```

Task I.3: Classification

This subtask requires you to implement a classification system with Logistic regression. You need to include

Building a classification model using Logistic Regression (LR)

In [8]:

```python
from pyspark.ml.feature import Imputer
imputer=Imputer(inputCols=['vote_average','vote_count',
'revenue','budget'],outputCols=['vote_average','vote_count', 'revenue','budget'])
model=imputer.fit(movie_data)
movie_data=model.transform(movie_data)
movie_data.show(5)
```

```
+-------+------+------+------------+----------+---------+--------+-----+--------------------+------
-----------+
|movieId|userId|rating|vote_average|vote_count|  revenue|  budget|adult|               title|
genres|
+-------+------+------+------------+----------+---------+--------+-----+--------------------+------
-----------+
|    1.0|   1.0|   4.0|         7.7|      5415|373554033|30000000|FALSE|    Toy Story
(1995)|Adventure|Animati...|
|    3.0|   1.0|   4.0|         6.5|        92|        0|       0|FALSE|Grumpier Old Men ...|
omedy|Romance|
|    6.0|   1.0|   4.0|         7.7|      1886|187436818|60000000|FALSE|         Heat (1995)|Actio
Crime|Thri...|
|   47.0|   1.0|   5.0|         5.4|       452|122195920|18000000|FALSE|Seven (a.k.a. Se7...|
Mystery|Thriller|
|   50.0|   1.0|   5.0|         8.1|      5915|327311859|33000000|FALSE|Usual Suspects,
T...|Crime|Mystery|Thr...|
+-------+------+------+------------+----------+---------+--------+-----+--------------------+------
-----------+
only showing top 5 rows
```

In [9]:

```
cols=movie_data.columns
cols.remove("userId")
cols.remove("movieId")
cols.remove("title")
cols.remove("genres")
cols.remove("adult")

from pyspark.ml.feature import VectorAssembler
vectorAssembler = VectorAssembler(inputCols = cols, outputCol = "features")

# Now let us use the transform method to transform our dataset
movie_data = vectorAssembler.transform(movie_data)
```

In [10]:

```
#Randomly split data into train and test sets, and set seed for reproducibility.

(training,test)=movie_data.randomSplit([0.8, 0.2], seed = 2018)
print("Training Dataset Count: " + str(training.count()))
print("Test Dataset Count: " + str(test.count()))
```

```
Training Dataset Count: 80792
Test Dataset Count: 20044
```

Logistic Regression Model

In [11]:

```
from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression(featuresCol = 'features', labelCol = 'userId', maxIter=10)
lrModel = lr.fit(training)
```

Make predictions on the test set.

In [12]:

```
predict_train=lrModel.transform(training)
predict_test=lrModel.transform(test)
predict_test.select("userId", "prediction").show(10)
```

```
+------+----------+
|userId|prediction|
+------+----------+
|   1.0|     603.0|
|  27.0|     294.0|
|  33.0|     294.0|
|  40.0|     603.0|
|  46.0|     603.0|
|  57.0|     603.0|
|  68.0|     294.0|
|  89.0|     294.0|
|  91.0|     603.0|
|  93.0|     294.0|
+------+----------+
only showing top 10 rows
```

Evaluate our Logistic Regression Model

In [14]:

```
from pyspark.ml.evaluation import BinaryClassificationEvaluator
evaluator=BinaryClassificationEvaluator(rawPredictionCol='rawPrediction',labelCol='userId')
predict_test.select("userId","rawPrediction","prediction","probability").show(5)
```

```
print("The area under ROC for train set is {}".format(evaluator.evaluate(predict_train)))
print("The area under ROC for test set is {}".format(evaluator.evaluate(predict_test)))
```

```
+-------+--------------------+----------+--------------------+
|userId|       rawPrediction|prediction|         probability|
+-------+--------------------+----------+--------------------+
|   1.0|[-4.3539941842314...|     603.0|[1.20837400355320...|
|  27.0|[-4.3451377886585...|     294.0|[1.13916436063618...|
|  33.0|[-4.3451377886585...|     294.0|[1.13916436063618...|
|  40.0|[-4.3628505798043...|     603.0|[1.23724613854051...|
|  46.0|[-4.3628505798043...|     603.0|[1.23724613854051...|
+-------+--------------------+----------+--------------------+
only showing top 5 rows

The area under ROC for train set is 1.0
The area under ROC for test set is 1.0
```

In [ ]: