Homework 9

Argiro Chatzis

Part 1&2: Downloading the new test2_new.txt data and the output1.txt file from HW 5 and using the userID in test2_new.txt to fetch the related rating scores in the training data for the same users

In [1]:
```
import pandas as pd
test = pd.read_csv('test2_new.txt', sep = '|', header=None)
test.columns = ['UserID', 'ItemID', 'Predictor']
test.head( )
```

Out[1]:

	UserID	ItemID	Predictor
0	200031	30877	1
1	200031	8244	1
2	200031	130183	0
3	200031	198762	0
4	200031	34503	1

In [2]:
```
train = pd.read_csv('output1.txt', sep='|', header=None)
train.columns = ['UserID', 'ItemID', 'Rating1', 'Rating2']
train.head( )
```

Out[2]:

	UserID	ItemID	Rating1	Rating2
0	199810	208019	0.0	0.0
1	199810	74139	0.0	0.0

	UserID	ItemID	Rating1	Rating2
2	199810	9903	0.0	0.0
3	199810	242681	0.0	0.0
4	199810	18515	0.0	70.0

In [3]:
```
test['Rating1'] = 0.0
test['Rating2'] = 0.0
test = test[['UserID', 'ItemID', 'Rating1', 'Rating2', 'Predictor']]
test.head( )
```

Out[3]:

	UserID	ItemID	Rating1	Rating2	Predictor
0	200031	30877	0.0	0.0	1
1	200031	8244	0.0	0.0	1
2	200031	130183	0.0	0.0	0
3	200031	198762	0.0	0.0	0
4	200031	34503	0.0	0.0	1

In [4]:
```
for i in range(len(test)):
    user = test['UserID'][i]
    item = test['ItemID'][i]
    test['Rating1'][i] = train[train['UserID']==user][train['ItemID']==item]['Rating1']
    test['Rating2'][i] = train[train['UserID']==user][train['ItemID']==item]['Rating2']
```

In [5]:
```
test.to_csv('test.csv', index=False)
```

In [6]:
```
train.to_csv('train.csv', index=False)
```

Getting the Data into a Pyspark Framework in Order to Apply the ML Algorithms to it:

```
In [7]: import findspark
        findspark.init( )
        from pyspark.sql import SparkSession

        from pyspark import SparkContext
        sc = SparkContext()
```

```
In [8]: from pyspark.ml.evaluation import RegressionEvaluator
        from pyspark.ml.recommendation import ALS
        from pyspark.sql import Row
```

```
In [9]: if __name__ == "__main__":
            spark = SparkSession\
                .builder\
                .appName("HOMEWORK9")\
                .getOrCreate()
```

```
In [10]: df = spark.read.csv('test.csv', header=True, inferSchema=True)
         df.printSchema( )
```

```
root
 |-- UserID: integer (nullable = true)
 |-- ItemID: integer (nullable = true)
 |-- Rating1: double (nullable = true)
 |-- Rating2: double (nullable = true)
 |-- Predictor: integer (nullable = true)
```

```
In [11]: df.show(10)
```

```
+------+------+-------+-------+---------+
|UserID|ItemID|Rating1|Rating2|Predictor|
+------+------+-------+-------+---------+
|200031| 30877|   90.0|   50.0|        1|
|200031|  8244|   90.0|    0.0|        1|
|200031|130183|    0.0|    0.0|        0|
|200031|198762|    0.0|    0.0|        0|
|200031| 34503|   90.0|   50.0|        1|
|200031|227283|    0.0|   90.0|        0|
|200032|218377|    0.0|    0.0|        0|
|200032|110262|    0.0|    0.0|        0|
|200032| 18681|   90.0|   90.0|        1|
|200032|138493|   90.0|   90.0|        1|
+------+------+-------+-------+---------+
only showing top 10 rows
```

```
In [12]: cols = df.columns

In [13]: from pyspark.ml.feature import VectorAssembler
         assembler =VectorAssembler(inputCols=['Rating1','Rating2'],
                                    outputCol= 'features')

In [14]: from pyspark.ml import Pipeline

         pipeline=Pipeline(stages=[assembler])
         model=pipeline.fit(df)
         df=model.transform(df)
         selectedCols = ['features'] + cols
         df = df.select(selectedCols)

In [15]: train_1, test_1 = df.randomSplit([0.8, 0.2], seed=2018)
         print('Training Dataset Count: '+str(train_1.count( )))
         print('Test Dataset Count: '+str(test_1.count( )))

Training Dataset Count: 4849
Test Dataset Count: 1151
```

Part 3&4: Applying the 4 different classifiers and submitting results to Kaggle

1. Support Vector Machine Classifier:

```
In [16]: from pyspark.ml.classification import LinearSVC
         lsvc = LinearSVC(labelCol = 'Predictor',maxIter=10, regParam=0.1)
         lsvcModel = lsvc.fit(df)
         predictions = lsvcModel.transform(df)
         predictions.show(5)
```

```
+-----------+------+------+-------+-------+---------+--------------------+---
--------+
|   features|UserID|ItemID|Rating1|Rating2|Predictor|       rawPrediction|pre
diction|
+-----------+------+------+-------+-------+---------+--------------------+---
--------+
|[90.0,50.0]|200031| 30877|   90.0|   50.0|        1|[-1.5238711219942...|
1.0|
```

```
| [90.0,0.0]|200031|  8244|   90.0|    0.0|        1|[-0.4222426652233...|
1.0|
|  (2,[],[])|200031|130183|    0.0|    0.0|        0|[1.00004604469945...|
0.0|
|  (2,[],[])|200031|198762|    0.0|    0.0|        0|[1.00004604469945...|
0.0|
|[90.0,50.0]|200031| 34503|   90.0|   50.0|        1|[-1.5238711219942...|
1.0|
+-----------+------+------+-------+-------+---------+--------------------+---
-------+
only showing top 5 rows
```

In [17]:
```
from pyspark.ml.evaluation import BinaryClassificationEvaluator
```

In [18]:
```
evaluator = BinaryClassificationEvaluator(labelCol='Predictor')
print("Test Area Under ROC: " + str(evaluator.evaluate(predictions)))
```

```
Test Area Under ROC: 0.8728997222222222
```

In [19]:
```
main_df = spark.read.csv('train.csv', header=True, inferSchema=True)
main_df.printSchema( )
```

```
root
 |-- UserID: integer (nullable = true)
 |-- ItemID: integer (nullable = true)
 |-- Rating1: double (nullable = true)
 |-- Rating2: double (nullable = true)
```

In [20]:
```
main_cols = main_df.columns
main_df = model.transform(main_df)
selCols = ['features'] + main_cols
main_df = main_df.select(selCols)
main_df.printSchema( )
```

```
root
 |-- features: vector (nullable = true)
 |-- UserID: integer (nullable = true)
 |-- ItemID: integer (nullable = true)
 |-- Rating1: double (nullable = true)
 |-- Rating2: double (nullable = true)
```

In [21]:
```
SVM_predictions = lsvcModel.transform(main_df)
```

In [22]:
```
SVM_predictions.show(10)
```

```
+-------------+------+------+-------+-------+--------------------+----------+
|     features|UserID|ItemID|Rating1|Rating2|       rawPrediction|prediction|
+-------------+------+------+-------+-------+--------------------+----------+
|    (2,[],[])|199810|208019|    0.0|    0.0|[1.00004604469945...|       0.0|
|    (2,[],[])|199810| 74139|    0.0|    0.0|[1.00004604469945...|       0.0|
|    (2,[],[])|199810|  9903|    0.0|    0.0|[1.00004604469945...|       0.0|
|    (2,[],[])|199810|242681|    0.0|    0.0|[1.00004604469945...|       0.0|
|   [0.0,70.0]|199810| 18515|    0.0|   70.0|[-0.5422337947798...|       1.0|
|   [0.0,90.0]|199810|105760|    0.0|   90.0|[-0.9828851774881...|       1.0|
|    (2,[],[])|199812|276940|    0.0|    0.0|[1.00004604469945...|       0.0|
|[100.0,100.0]|199812|142408|  100.0|  100.0|[-2.7835316576454...|       1.0|
|[100.0,100.0]|199812|130023|  100.0|  100.0|[-2.7835316576454...|       1.0|
|    (2,[],[])|199812| 29189|    0.0|    0.0|[1.00004604469945...|       0.0|
+-------------+------+------+-------+-------+--------------------+----------+
only showing top 10 rows
```

In [23]:
```
SVM_answer = SVM_predictions.select('UserID', 'ItemID', 'prediction')
SVM_answer = SVM_answer.toPandas( )
```

In [24]:
```
SVM_answer['TrackID'] = ''
SVM_answer.head( )
```

Out[24]:

	UserID	ItemID	prediction	TrackID
0	199810	208019	0.0	
1	199810	74139	0.0	
2	199810	9903	0.0	
3	199810	242681	0.0	
4	199810	18515	1.0	

In [25]:
```
for i in range(len(SVM_answer)):
    SVM_answer['TrackID'][i] = str(SVM_answer['UserID'][i])+'_'+str(SVM_answer['ItemID'][i])
```

In [26]:
```
SVM_answer = SVM_answer[['TrackID', 'UserID', 'ItemID', 'prediction']]
SVM_answer = SVM_answer.rename(columns={'prediction': 'Predictor'})
SVM_answer.head( )
```

Out[26]:

	TrackID	UserID	ItemID	Predictor
0	199810_208019	199810	208019	0.0
1	199810_74139	199810	74139	0.0
2	199810_9903	199810	9903	0.0
3	199810_242681	199810	242681	0.0
4	199810_18515	199810	18515	1.0

In [27]:
```
SVM_answer.drop(columns={'UserID', 'ItemID'}, inplace=True)
SVM_answer.head( )
```

Out[27]:

	TrackID	Predictor
0	199810_208019	0.0
1	199810_74139	0.0
2	199810_9903	0.0
3	199810_242681	0.0
4	199810_18515	1.0

In [28]:
```
SVM_answer.to_csv('SVM.csv', index=False)
```

The SVM Classifier got us an accuracy of 85.578% when submitted to kaggle.

2. Factorization Machine:

In [29]:
```
from pyspark.ml.classification import FMClassifier
from pyspark.ml.feature import MinMaxScaler, StringIndexer
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

fm = FMClassifier(featuresCol = 'features', labelCol = 'Predictor', stepSize=0.001)
fmmodel = fm.fit(df)
predictions_fm = fmmodel.transform(main_df)
predictions_fm.show(5)
```

```
+----------+------+------+-------+-------+--------------------+--------------
------+----------+
|  features|UserID|ItemID|Rating1|Rating2|       rawPrediction|         proba
bility|prediction|
+----------+------+------+-------+-------+--------------------+--------------
------+----------+
| (2,[],[])|199810|208019|    0.0|    0.0|[0.10276937000588...|[0.52566975379
244...|       0.0|
| (2,[],[])|199810| 74139|    0.0|    0.0|[0.10276937000588...|[0.52566975379
244...|       0.0|
| (2,[],[])|199810|  9903|    0.0|    0.0|[0.10276937000588...|[0.52566975379
244...|       0.0|
| (2,[],[])|199810|242681|    0.0|    0.0|[0.10276937000588...|[0.52566975379
244...|       0.0|
|[0.0,70.0]|199810| 18515|    0.0|   70.0|[-1.0901217713695...|[0.25159534874
361...|       1.0|
+----------+------+------+-------+-------+--------------------+--------------
------+----------+
only showing top 5 rows
```

In [30]:
```
fm_answer = predictions_fm.select('UserID', 'ItemID', 'prediction')
fm_answer = fm_answer.toPandas()
```

In [31]:
```
fm_answer['TrackID'] = ''
fm_answer.head( )
```

Out[31]:

	UserID	ItemID	prediction	TrackID
0	199810	208019	0.0	
1	199810	74139	0.0	
2	199810	9903	0.0	
3	199810	242681	0.0	
4	199810	18515	1.0	

In [32]:
```
for i in range(len(fm_answer)):
    fm_answer['TrackID'][i] = str(fm_answer['UserID'][i])+'_'+str(fm_answer['ItemID'][i])
```

In [33]:
```
fm_answer = fm_answer[['TrackID', 'UserID', 'ItemID', 'prediction']]
fm_answer = fm_answer.rename(columns={'prediction': 'Predictor'})
fm_answer.head( )
```

Out[33]:

	TrackID	UserID	ItemID	Predictor
0	199810_208019	199810	208019	0.0
1	199810_74139	199810	74139	0.0
2	199810_9903	199810	9903	0.0
3	199810_242681	199810	242681	0.0
4	199810_18515	199810	18515	1.0

In [34]:
```
fm_answer.drop(columns={'UserID', 'ItemID'}, inplace=True)
fm_answer.head( )
```

Out[34]:

	TrackID	Predictor
0	199810_208019	0.0
1	199810_74139	0.0
2	199810_9903	0.0
3	199810_242681	0.0
4	199810_18515	1.0

In [35]:
```
fm_answer.to_csv('factorization_machine.csv', index=False)
```

The Factorization Machine Classifier got us an accuracy of 85.745% when submitted to kaggle.

3. Logistic Regression:

In [41]:
```
from pyspark.ml.classification import LogisticRegression

lr = LogisticRegression(featuresCol ='features', labelCol = 'Predictor', maxIter=10)
lrmodel = lr.fit(df)
predictions_lr = lrmodel.transform(main_df)
predictions_lr.show(5)
```

```
+-----------+------+------+-------+-------+---------+--------------------+---
-----------------+----------+
|   features|UserID|ItemID|Rating1|Rating2|Predictor|       rawPrediction|
probability|prediction|
+-----------+------+------+-------+-------+---------+--------------------+---
-----------------+----------+
|[90.0,50.0]|200031| 30877|   90.0|   50.0|        1|[-4.6449289927726...|[0.
00951873496864...|       1.0|
| [90.0,0.0]|200031|  8244|   90.0|    0.0|        1|[-3.0251701626248...|[0.
04630163526226...|       1.0|
```

```
|  (2,[],[])|200031|130183|    0.0|    0.0|         0|[1.42369930626282...|[0.
80591769594253...|       0.0|
|  (2,[],[])|200031|198762|    0.0|    0.0|         0|[1.42369930626282...|[0.
80591769594253...|       0.0|
|[90.0,50.0]|200031| 34503|   90.0|   50.0|         1|[-4.6449289927726...|[0.
00951873496864...|       1.0|
+-----------+------+------+-------+-------+----------+--------------------+---
-----------------+----------+
only showing top 5 rows
```

In [43]:
```
evaluator = BinaryClassificationEvaluator(labelCol = 'Predictor')
print("Test Area Under ROC: " + str(evaluator.evaluate(predictions_lr)))
```

```
Test Area Under ROC: 0.8749316111111111
```

In [48]:
```
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
paramGrid = (ParamGridBuilder()
             .addGrid(lr.regParam, [0.01, 0.1, 0.5])
             .addGrid(lr.elasticNetParam, [0.0, 0.5, 1.0])
             .addGrid(lr.maxIter, [1, 5, 10])
             .build( ))

 cv = CrossValidator(estimator=lr, estimatorParamMaps=paramGrid, evaluator=evaluator, nu
                                                                          mFolds=5)

 cvModel = cv.fit(df)
 predictions_lr_1 = cvModel.transform(df)
 print('Test Area Under ROC', evaluator.evaluate(predictions_lr_1))
```

```
Test Area Under ROC 0.8747172777777777
```

In [49]:
```
lr_answer = predictions_lr.select('UserID', 'ItemID', 'prediction')
lr_answer = lr_answer.toPandas()
```

In [50]:
```
lr_answer['TrackID'] = ''
lr_answer.head( )
```

Out[50]:

	UserID	ItemID	prediction	TrackID
0	200031	30877	1.0	
1	200031	8244	1.0	
2	200031	130183	0.0	
3	200031	198762	0.0	
4	200031	34503	1.0	

In [51]:
```
for i in range(len(lr_answer)):
    lr_answer['TrackID'][i] = str(lr_answer['UserID'][i])+'_'+str(lr_answer['ItemID'][i])
```

In [52]:
```
lr_answer = lr_answer[['TrackID', 'UserID', 'ItemID', 'prediction']]
lr_answer = lr_answer.rename(columns={'prediction': 'Predictor'})
lr_answer.head( )
```

Out[52]:

	TrackID	UserID	ItemID	Predictor
0	200031_30877	200031	30877	1.0
1	200031_8244	200031	8244	1.0
2	200031_130183	200031	130183	0.0
3	200031_198762	200031	198762	0.0
4	200031_34503	200031	34503	1.0

In [53]:
```
lr_answer.drop(columns={'UserID', 'ItemID'}, inplace=True)
lr_answer.head( )
```

Out[53]:

	TrackID	Predictor
0	200031_30877	1.0
1	200031_8244	1.0
2	200031_130183	0.0
3	200031_198762	0.0
4	200031_34503	1.0

In [54]:
```
lr_answer.to_csv('logistic_regress.csv', index=False)
```

The Logistic Regression Classifier got us an accuracy of 85.771% when submitted to kaggle.

4. Gradient_Boosted Tree Classifier:

In [63]:
```
from pyspark.ml.classification import GBTClassifier

gbt = GBTClassifier(featuresCol = 'features', labelCol = 'Predictor', maxIter=10)
gbtmodel = gbt.fit(df)
predictions_gbt = gbtmodel.transform(main_df)
predictions_gbt.show(5)
```

```
+----------+------+------+-------+-------+--------------------+--------------
------+----------+
|  features|UserID|ItemID|Rating1|Rating2|       rawPrediction|         proba
bility|prediction|
+----------+------+------+-------+-------+--------------------+--------------
------+----------+
| (2,[],[])|199810|208019|    0.0|    0.0|[0.69621496476798...|[0.80097988422
856...|       0.0|
| (2,[],[])|199810| 74139|    0.0|    0.0|[0.69621496476798...|[0.80097988422
856...|       0.0|
| (2,[],[])|199810|  9903|    0.0|    0.0|[0.69621496476798...|[0.80097988422
856...|       0.0|
```

```
| (2,[],[])|199810|242681|    0.0|    0.0|[0.69621496476798...|[0.80097988422
856...|       0.0|
|[0.0,70.0]|199810| 18515|    0.0|   70.0|[-0.3823633131076...|[0.31762094530
726...|       1.0|
+----------+------+------+-------+-------+--------------------+-------------
------+----------+
only showing top 5 rows
```

```
In [71]: evaluator = BinaryClassificationEvaluator(labelCol = 'prediction')
         print("Test Area Under ROC: " + str(evaluator.evaluate(predictions_gbt)))
```

```
Test Area Under ROC: 1.0
```

```
In [72]: gbt_answer = predictions_gbt.select('UserID', 'ItemID', 'prediction')
         gbt_answer = gbt_answer.toPandas()
```

```
In [73]: gbt_answer['TrackID'] = ''
         gbt_answer.head( )
```

Out[73]:

	UserID	ItemID	prediction	TrackID
0	199810	208019	0.0	
1	199810	74139	0.0	
2	199810	9903	0.0	
3	199810	242681	0.0	
4	199810	18515	1.0	

```
In [74]: for i in range(len(gbt_answer)):
             gbt_answer['TrackID'][i] = str(gbt_answer['UserID'][i])+'_'+str(gbt_answer['ItemID'][i])
```

```
In [75]: gbt_answer = gbt_answer[['TrackID', 'UserID', 'ItemID', 'prediction']]
         gbt_answer = gbt_answer.rename(columns={'prediction': 'Predictor'})
         gbt_answer.head( )
```

Out[75]:

	TrackID	UserID	ItemID	Predictor
0	199810_208019	199810	208019	0.0
1	199810_74139	199810	74139	0.0
2	199810_9903	199810	9903	0.0
3	199810_242681	199810	242681	0.0
4	199810_18515	199810	18515	1.0

In [76]:
```
gbt_answer.drop(columns={'UserID', 'ItemID'}, inplace=True)
gbt_answer.head( )
```

Out[76]:

	TrackID	Predictor
0	199810_208019	0.0
1	199810_74139	0.0
2	199810_9903	0.0
3	199810_242681	0.0
4	199810_18515	1.0

In [77]:
```
gbt_answer.to_csv('gradient_boosted_tree.csv', index=False)
```

The Gradient-Boosted Tree Classifier got us an accuracy of 85.753% when submitted to kaggle.