HW2_Regression

June 1, 2021

Gamze Atmaca , Bulut Fıçıcı

[1]:
```
import sys
from pyspark import SparkConf, SparkContext, SQLContext
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
```

[2]:
```
from pyspark.sql.types import StructType,StructField, StringType, IntegerType
from pyspark.sql.types import ArrayType, DoubleType, BooleanType
from pyspark.sql.functions import corr
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.regression import LinearRegression
from pyspark.ml.linalg import Vector
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.sql.functions import *
```

[3]:
```
sc = SparkContext.getOrCreate()
sqlcont = SQLContext(sc)
```

[4]:
```
# Using schema to create df

schema = StructType() \
      .add("Open_Time",DoubleType(),True) \
      .add("Open",DoubleType(),True) \
      .add("High",DoubleType(),True) \
      .add("Low",DoubleType(),True) \
      .add("Close_btc",DoubleType(),True) \
      .add("Volume",DoubleType(),True) \

df_btc = sqlcont.read.format("csv") \
      .option("header", True) \
      .schema(schema) \
      .load('csv/btc_2021_hourly.csv')

schema = StructType() \
      .add("Open_Time",DoubleType(),True) \
```

```
      .add("Open",DoubleType(),True) \
      .add("Low",DoubleType(),True) \
      .add("Close_eth",DoubleType(),True) \
      .add("Volume",DoubleType(),True)

df_eth = sqlcont.read.format("csv") \
      .option("header", True) \
      .schema(schema) \
      .load('csv/eth_2021_hourly.csv')

schema = StructType() \
      .add("Open_Time",DoubleType(),True) \
      .add("Open",DoubleType(),True) \
      .add("High",DoubleType(),True) \
      .add("Low",DoubleType(),True) \
      .add("Close_xrp",DoubleType(),True) \
      .add("Volume",DoubleType(),True)

df_xrp = sqlcont.read.format("csv") \
      .option("header", True) \
      .schema(schema) \
      .load('csv/xrp_2021_hourly.csv')

schema = StructType() \
      .add("Open_Time",DoubleType(),True) \
      .add("Open",DoubleType(),True) \
      .add("High",DoubleType(),True) \
      .add("Low",DoubleType(),True) \
      .add("Close_ada",DoubleType(),True) \
      .add("Volume",DoubleType(),True)

df_ada = sqlcont.read.format("csv") \
      .option("header", True) \
      .schema(schema) \
      .load('csv/ada_2021_hourly.csv')

schema = StructType() \
      .add("Open_Time",DoubleType(),True) \
      .add("Open",DoubleType(),True) \
      .add("High",DoubleType(),True) \
      .add("Low",DoubleType(),True) \
      .add("Close_bnb",DoubleType(),True) \
      .add("Volume",DoubleType(),True)

df_bnb = sqlcont.read.format("csv") \
      .option("header", True) \
      .schema(schema) \
```

```
      .load('csv/bnb_2021_hourly.csv')
```

[5]:
```
df_btc=df_btc.drop("Open").drop("High").drop("Low").drop("Volume")
df_eth=df_eth.drop("Open").drop("High").drop("Low").drop("Volume")
df_xrp=df_xrp.drop("Open").drop("High").drop("Low").drop("Volume")
df_ada=df_ada.drop("Open").drop("High").drop("Low").drop("Volume")
df_bnb=df_bnb.drop("Open").drop("High").drop("Low").drop("Volume")
```

[6]:
```
# Joined dataframes with respect to their Open_Time

df=df_btc.join(df_bnb,["Open_Time"])
df=df.join(df_eth,["Open_Time"])
df=df.join(df_ada,["Open_Time"])
df=df.join(df_xrp,["Open_Time"])
```

[7]:
```
# Timestamp to datetime

df=df.withColumn("Open_Time", from_unixtime(col("Open_Time")))
```

[48]:
```
df.show(3,vertical=True)
```

```
-RECORD 0------------------------
 Open_Time | 2021-01-01 01:00:00
 Close_btc | 29409.99
 Close_bnb | 37.6134
 Close_eth | 733.37
 Close_ada | 0.18358
 Close_xrp | 0.22382
-RECORD 1------------------------
 Open_Time | 2021-01-01 02:00:00
 Close_btc | 29194.65
 Close_bnb | 37.96
 Close_eth | 742.27
 Close_ada | 0.18368
 Close_xrp | 0.22361
-RECORD 2------------------------
 Open_Time | 2021-01-01 03:00:00
 Close_btc | 29278.4
 Close_bnb | 37.925
 Close_eth | 743.1
 Close_ada | 0.18292
 Close_xrp | 0.2273
only showing top 3 rows
```

[9]:
```
# There is no NULL value in out dataset.
```

```
df.filter(df["Open_Time"].isNull()).count()+df.filter(df_btc["Close_btc"].
 ↪isNull()).count()\
+df.filter(df["Close_bnb"].isNull()).count()+df.filter(df["Close_eth"].
 ↪isNull()).count()\
+df.filter(df["Close_ada"].isNull()).count()+df.filter(df["Close_xrp"].
 ↪isNull()).count()
```

[9]:
```
0
```

[10]:
```
# Shape of our dataset

(df.count(),len(df.columns))
```

[10]:
```
(3632, 6)
```

[11]:
```
# All values are distinct

df.distinct().count()
```

[11]:
```
3632
```

[73]:
```
# Description

df.describe().show()
```

```
+-------+-------------------+-----------------+------------------+-------------
----+-------------------+-------------------+
|summary|          Open_Time|        Close_btc|         Close_bnb|
Close_eth|          Close_ada|          Close_xrp|
+-------+-------------------+-----------------+------------------+-------------
----+-------------------+-------------------+
|  count|               3632|             3632|              3632|
3632|               3632|               3632|
|   mean|               null|47765.06958700446|
289.3430961178408|1988.5438436123306|  1.052209691629956| 0.7552170870044059|
| stddev|               null|9755.696878505894|199.37845900084326|
735.3322081769288|0.48754714146259637|0.46263710089047844|
|    min|2021-01-01 01:00:00|         29000.01|           35.8756|
714.29|            0.17064|            0.21743|
|    max|2021-06-01 15:00:00|         64577.26|            684.22|
4297.42|             2.4197|            1.93777|
+-------+-------------------+-----------------+------------------+-------------
----+-------------------+-------------------+
```

[44]:
```
# Max values of BTCUSDT
```

```
df.orderBy("Close_btc",ascending=False).show(10)
```

```
+------------------+---------+---------+---------+---------+---------+
|          Open_Time|Close_btc|Close_bnb|Close_eth|Close_ada|Close_xrp|
+------------------+---------+---------+---------+---------+---------+
|2021-04-14 06:00:00| 64577.26| 583.1688|  2353.56|   1.5195|  1.89768|
|2021-04-14 11:00:00| 64511.21| 574.7138|  2357.99|  1.47108|  1.83783|
|2021-04-14 07:00:00|  64288.8| 575.7693|  2369.75|  1.50496|  1.90294|
|2021-04-14 04:00:00| 64268.97| 574.2624|   2338.0|  1.53841|  1.93777|
|2021-04-14 10:00:00| 64099.99| 571.4354|  2332.74|  1.48703|  1.82924|
|2021-04-14 05:00:00| 64038.49| 579.6999|   2353.0|  1.50858|  1.89836|
|2021-04-14 08:00:00| 63928.57| 580.8758|  2327.75|  1.49377|  1.81327|
|2021-04-14 12:00:00| 63871.42| 565.0001|   2350.0|  1.42307|  1.77777|
|2021-04-14 13:00:00| 63772.67|  555.727|  2350.32|  1.41792|  1.73382|
|2021-04-14 03:00:00| 63749.19| 568.3266|  2290.25|   1.4589|  1.81586|
+------------------+---------+---------+---------+---------+---------+
only showing top 10 rows
```

[14]:
```
# BNB has the greatest correlation with BTC

df.select(corr("Close_btc","Close_bnb")).show()
df.select(corr("Close_btc","Close_eth")).show()
df.select(corr("Close_btc","Close_xrp")).show()
df.select(corr("Close_btc","Close_ada")).show()
```

```
+--------------------------+
|corr(Close_btc, Close_bnb)|
+--------------------------+
|        0.6851279938636956|
+--------------------------+

+--------------------------+
|corr(Close_btc, Close_eth)|
+--------------------------+
|       0.46032378356795545|
+--------------------------+

+--------------------------+
|corr(Close_btc, Close_xrp)|
+--------------------------+
|         0.516116173912515|
+--------------------------+

+--------------------------+
|corr(Close_btc, Close_ada)|
+--------------------------+
|        0.5841987920185967|
```

```
+-------------------------+
```

[15]:
```
# Constructing feature matrix

vec_assembler=VectorAssembler(inputCols=["Close_bnb","Close_eth","Close_xrp","Close_ada"],outp
features=vec_assembler.transform(df)
features.select("features").show(10,False)
```

```
+--------------------------------+
|features                        |
+--------------------------------+
|[37.6134,733.37,0.22382,0.18358]|
|[37.96,742.27,0.22361,0.18368]  |
|[37.925,743.1,0.2273,0.18292]   |
|[37.8702,739.3,0.23898,0.1818]  |
|[37.7129,739.5,0.23819,0.18297] |
|[37.6463,737.04,0.23428,0.181]  |
|[37.6812,734.4,0.22976,0.17955] |
|[37.4172,725.1,0.22874,0.17762] |
|[37.9477,728.77,0.23259,0.17904]|
|[38.827,733.27,0.23692,0.18021] |
+--------------------------------+
only showing top 10 rows
```

[51]:
```
# Main dataset to train and test

model=features.select("features","Close_btc")
model.show(5, False)
```

```
+--------------------------------+---------+
|features                        |Close_btc|
+--------------------------------+---------+
|[37.6134,733.37,0.22382,0.18358]|29409.99 |
|[37.96,742.27,0.22361,0.18368]  |29194.65 |
|[37.925,743.1,0.2273,0.18292]   |29278.4  |
|[37.8702,739.3,0.23898,0.1818]  |29220.31 |
|[37.7129,739.5,0.23819,0.18297] |29187.01 |
+--------------------------------+---------+
only showing top 5 rows
```

0.1 Linear Regression

[17]:
```
# Linear regression model and fit

train_df,test_df=model.randomSplit([0.7,0.3])
reg=LinearRegression(labelCol="Close_btc")
linear=reg.fit(train_df)
```

[58]:
```
print("Intercept is:",linear.intercept)
print("\nCoefficients are:", linear.coefficients)
```

```
Intercept is: 46939.720952994074

Coefficients are:
[86.15099731110614,-7.954388397981677,-14999.29639551094,2894.692519484616]
```

[61]:
```
test_predictions=linear.evaluate(test_df)
print("R-square value:",test_predictions.r2)
```

```
R-square value: 0.659413921058608
```

[62]:
```
test_predictions.residuals.show(5)
```

```
+-------------------+
|          residuals|
+-------------------+
| -7501.564139406608|
| -6537.100556508296|
|  -11915.0241065974|
|-5684.2292298508255|
| -12149.07651986189|
+-------------------+
only showing top 5 rows
```

0.2 Logistic Regression

[24]:
```
# Creating a new dataframe to use logistic regression.

schema = StructType() \
      .add("Open Time",DoubleType(),True) \
      .add("Open",DoubleType(),True) \
      .add("High",DoubleType(),True) \
      .add("Low",DoubleType(),True) \
      .add("Close",DoubleType(),True) \
      .add("Volume",DoubleType(),True) \
      .add("Close Time",DoubleType(),True) \
      .add("Quote asset volume",DoubleType(),True) \
```

```
      .add("Number of trades",IntegerType(),True) \
      .add("Taker buy base asset volume",DoubleType(),True) \
      .add("Taker buy quote asset volume",DoubleType(),True) \
      .add("Ignore",StringType(),True) \


btc_df = sqlcont.read.format("csv") \
      .option("header", True) \
      .schema(schema) \
      .load('csv/btc_2021_hourly.csv')
```

[64]:
```
# Using different columns for logistic regression.

btc_df=btc_df.drop("Open Time").drop("High").drop("Low").drop("Close Time").
 ↪drop("Ignore")
btc_df.show(3, vertical=True)
```

```
-RECORD 0-----------------------------------------
 Open                          | 28995.13
 Close                         | 29409.99
 Volume                        | 5403.068471
 Quote asset volume            | 1.583578168180572E8
 Number of trades              | 103896
 Taker buy base asset volume   | 3160.041701
 Taker buy quote asset volume | 9.261399193555292E7
 Up/Down                       | 1
-RECORD 1-----------------------------------------
 Open                          | 29410.0
 Close                         | 29194.65
 Volume                        | 2384.23156
 Quote asset volume            | 6.98426536734203E7
 Number of trades              | 57646
 Taker buy base asset volume   | 1203.433506
 Taker buy quote asset volume | 3.525274990832606E7
 Up/Down                       | 0
-RECORD 2-----------------------------------------
 Open                          | 29195.25
 Close                         | 29278.4
 Volume                        | 1461.345077
 Quote asset volume            | 4.276077672551646E7
 Number of trades              | 42510
 Taker buy base asset volume   | 775.915666
 Taker buy quote asset volume | 2.270554798307977E7
 Up/Down                       | 1
only showing top 3 rows
```

[65]:
```
# Target variable is Up/Down which takes the value of 1 if BTC rises, 0 if it␣
↪falls between the opening and closing time.

btc_df=btc_df.withColumn("Up/Down", \
   when(((btc_df.Close-btc_df.Open) >= 0), lit(1)) \
     .when(((btc_df.Close-btc_df.Open) < 0), lit(0)) \
  )
```

[68]:
```
btc_df.show(2, vertical=True)
```

```
-RECORD 0-----------------------------------------
 Open                         | 28995.13
 Close                        | 29409.99
 Volume                       | 5403.068471
 Quote asset volume           | 1.583578168180572E8
 Number of trades             | 103896
 Taker buy base asset volume  | 3160.041701
 Taker buy quote asset volume | 9.261399193555292E7
 Up/Down                      | 1
-RECORD 1-----------------------------------------
 Open                         | 29410.0
 Close                        | 29194.65
 Volume                       | 2384.23156
 Quote asset volume           | 6.98426536734203E7
 Number of trades             | 57646
 Taker buy base asset volume  | 1203.433506
 Taker buy quote asset volume | 3.525274990832606E7
 Up/Down                      | 0
only showing top 2 rows
```

[28]:
```
# Constructing features vector

vec_assembler_log=VectorAssembler(inputCols=["Volume","Quote asset␣
↪volume","Number of trades","Taker buy base asset volume","Taker buy quote␣
↪asset volume"],outputCol="features")
features_btc=vec_assembler_log.transform(btc_df)
features_btc.select("features").show(5,False)
```

```
+-------------------------------------------------------------------------------+
|features                                                                       |
+-------------------------------------------------------------------------------+
|[5403.068471,1.583578168180572E8,103896.0,3160.041701,9.261399193555292E7]|
|[2384.23156,6.98426536734203E7,57646.0,1203.433506,3.525274990832606E7]    |
|[1461.345077,4.276077672551646E7,42510.0,775.915666,2.270554798307977E7]   |
|[2038.046803,5.961463730352874E7,55414.0,1003.342834,2.934638188020654E7]  |
|[1469.956262,4.286453870435811E7,41800.0,679.846742,1.982719029247262E7]   |
```

```
|[1420.726291,4.144601301819005E7,46400.0,699.142676,2.039832275992173E7]  |
|[2380.180918,6.90346190948994E7,53158.0,1054.720991,3.059845740248306E7]  |
|[2008.165739,5.827419069862189E7,55012.0,1022.06617,2.966244772598589E7]  |
|[2022.056022,5.9006512260986E7,43674.0,1208.477578,3.527271735224266E7]   |
|[1944.255841,5.688192387605724E7,46783.0,1014.538319,2.968302781899852E7] |
+--------------------------------------------------------------------------+
only showing top 10 rows
```

[70]:
```
model_btc=features_btc.select("features","Up/Down")
model_btc.show(5)
```

```
+--------------------+-------+
|            features|Up/Down|
+--------------------+-------+
|[5403.068471,1.58…|      1|
|[2384.23156,6.984…|      0|
|[1461.345077,4.27…|      1|
|[2038.046803,5.96…|      0|
|[1469.956262,4.28…|      0|
+--------------------+-------+
only showing top 5 rows
```

[30]:
```
trainbtc_df,testbtc_df=model_btc.randomSplit([0.7,0.3])
logreg=LogisticRegression(labelCol="Up/Down")
logistic=logreg.fit(trainbtc_df)
```

[71]:
```
print("Intercept is:",logistic.intercept)
print("\nCoefficients are:",logistic.coefficients)
```

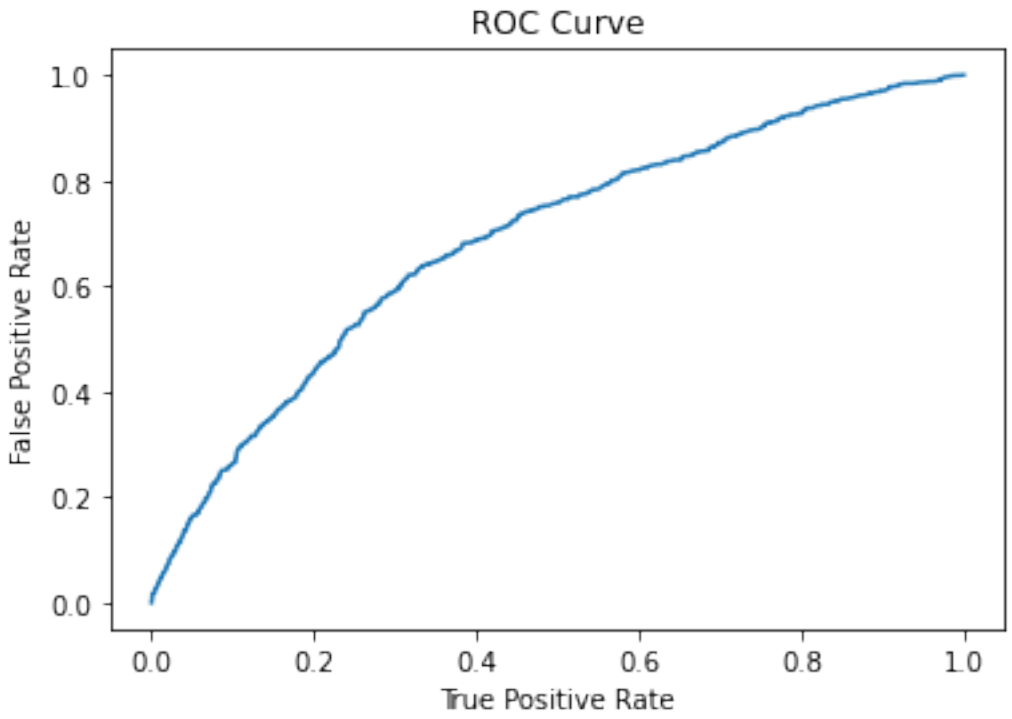
```
Intercept is: 0.04683552109420736

Coefficients are: [-0.0004609331702100972,-3.727613924771374e-08,4.8172719527566
43e-07,0.0010854380242368207,7.277716245768393e-08]
```

[34]:
```
test_predictions=logistic.transform(testbtc_df)
```

0.3 Metrics

[35]:
```
trainingSummary = logistic.summary
roc = trainingSummary.roc.toPandas()
plt.plot(roc['FPR'],roc['TPR'])
plt.ylabel('False Positive Rate')
plt.xlabel('True Positive Rate')
plt.title('ROC Curve')
plt.show()
```

```
print('Training set areaUnderROC: ' + str(trainingSummary.areaUnderROC))
```




```
Training set areaUnderROC: 0.6873306723599124
```

[36]:
```
evaluator = BinaryClassificationEvaluator(labelCol='Up/Down')
print('Test Area Under ROC:', evaluator.evaluate(test_predictions))
```

```
Test Area Under ROC: 0.704043095615854
```

[37]:
```
accuracy = test_predictions.filter(test_predictions['Up/Down'] ==␣
 ↪test_predictions.prediction).count() / float(test_predictions.count())
print("Accuracy : ",accuracy)
```

```
Accuracy :  0.6421343146274149
```

[38]:
```
class_names=[1.0,0.0]
y_true = test_predictions.select("Up/Down")
y_true = y_true.toPandas()

y_pred = test_predictions.select("prediction")
y_pred = y_pred.toPandas()

print(confusion_matrix(y_true, y_pred,labels=class_names))
```

```
[[375 148]
 [241 323]]
```