databricksPredicting Boston House Prices by Using Azure

(http://databricks.com)Databricks

Predicting Boston House Prices by Using Azure Databricks

In this study Boston House Prices were predicted by using Azure Databricks platform. The data includes the home values of Boston in 1970's. Five of the 14 diverse variables were used for this small study. These are:

- CRIM - per capita crime rate by town
- RM - average number of rooms per dwelling
- TAX - full-value property-tax rate per $10,000
- LSTAT - % lower status of the population
- MEDV - Median value of owner-occupied homes in $1000's

The value of homes (median house values) variable was used as the label and the number of rooms, crime per capita, property tax rate, and percent of the population considered lower class variables were used as the features. Predictions were made by using linear regression. Apache Spark Notebook in Azure Databricks were used for running the codes during the analyses.

This small project shows basic machine learning features with Azure Databricks.

The study does not includes some analyses like EDA or Model Tuning etc.!

This is a similar study to Azure Databricks training notebooks. To this study one more feature, new values, and some brief explanations were added.

Sources: https://docs.microsoft.com (https://docs.microsoft.com), https://databricks.com (https://databricks.com), and https://github.com/MicrosoftDocs/mslearn_databricks (https://github.com/MicrosoftDocs/mslearn_databricks)

```
%run "./Predicting Boston House Prices by Using Azure Databricks"
```

```
  Cycle Detected

  Stacktrace:
    /Users/camtaner@hotmail.com/Predicting Boston House Prices by Using Azur
```

```
  e Databricks: python
    /Users/camtaner@hotmail.com/Predicting Boston House Prices by Using Azur
  e Databricks: python
```

Importing Data

```
BostonHouseDF = (spark.read
  .option("HEADER", True)
  .option("inferSchema", True)
  .csv("/mnt/training/bostonhousing/bostonhousing/bostonhousing.csv")
)

display(BostonHouseDF)
```

	_c0	crim	zn	indus	chas	nox
1	1	0.00632	18	2.31	0	0.538
2	2	0.02731	0	7.07	0	0.469
3	3	0.02729	0	7.07	0	0.469
4	4	0.03237	0	2.18	0	0.458
5	5	0.06905	0	2.18	0	0.458
6	6	0.02985	0	2.18	0	0.458
7	7	0.08829	12.5	7.87	0	0.524
8	8	0.14455	12.5	7.87	0	0.524
9	9	0.21124	12.5	7.87	0	0.524
10	10	0.17004	12.5	7.87	0	0.524
11	11	0.22489	12.5	7.87	0	0.524
12	12	0.11747	12.5	7.87	0	0.524
13	13	0.09378	12.5	7.87	0	0.524
14	14	0.62976	0	8.14	0	0.538
15	15	0.63796	0	8.14	0	0.538
16	16	0.62739	0	8.14	0	0.538

Creating features column by using VectorAssembler

```
from pyspark.ml.feature import VectorAssembler

featureCols = ["rm", "crim", "tax", "lstat"]
assembler = VectorAssembler(inputCols=featureCols, outputCol="features")

BostonHouseFeaturizedDF = assembler.transform(BostonHouseDF)

display(BostonHouseFeaturizedDF)
```

	_c0	crim	zn	indus	chas	nox
1	1	0.00632	18	2.31	0	0.538
2	2	0.02731	0	7.07	0	0.469
3	3	0.02729	0	7.07	0	0.469
4	4	0.03237	0	2.18	0	0.458
5	5	0.06905	0	2.18	0	0.458
6	6	0.02985	0	2.18	0	0.458
7	7	0.08829	12.5	7.87	0	0.524
8	8	0.14455	12.5	7.87	0	0.524
9	9	0.21124	12.5	7.87	0	0.524
10	10	0.17004	12.5	7.87	0	0.524
11	11	0.22489	12.5	7.87	0	0.524
12	12	0.11747	12.5	7.87	0	0.524
13	13	0.09378	12.5	7.87	0	0.524
14	14	0.62976	0	8.14	0	0.538
15	15	0.63796	0	8.14	0	0.538
16	16	0.62739	0	8.14	0	0.538
17	17	1.05393	0	8.14	0	0.538
18	18	0.7842	0	8.14	0	0.538
19	19	0.80271	0	8.14	0	0.538
20	20	0.7258	0	8.14	0	0.538
21	21	1.25179	0	8.14	0	0.538
22	22	0.85204	0	8.14	0	0.538
23	23	1.23247	0	8.14	0	0.538
24	24	0.98843	0	8.14	0	0.538
25	25	0.75026	0	8.14	0	0.538
26	26	0.84054	0	8.14	0	0.538
27	27	0.67191	0	8.14	0	0.538

Model Training (Linear Regression)

```
from pyspark.ml.regression import LinearRegression

lr = LinearRegression(labelCol="medv", featuresCol="features")
```

Fitting Linear Regression Model to the Data

```
lrModel = lr.fit(BostonHouseFeaturizedDF)
```

The Coefficients and Intercept of the Model

```
print("Coefficients: {0:.1f}, {1:.1f}, {2:.1f},
{3:.1f}".format(*lrModel.coefficients))
print("Intercept: {0:.1f}".format(lrModel.intercept))
```

```
Coefficients: 5.2, -0.1, -0.0, -0.5
Intercept: -1.4
```

Showing some predictions by creating a sample subDF

```
SubsetBostonHouseFeaturizedDF = (BostonHouseFeaturizedDF
  .limit(15)
  .select("features", "medv")
)

display(SubsetBostonHouseFeaturizedDF)
```

	features	medv
1	▸ [1, 4, [], [6.575, 0.00632, 296, 4.98]]	24
2	▸ [1, 4, [], [6.421, 0.02731, 242, 9.14]]	21.6
3	▸ [1, 4, [], [7.185, 0.02729, 242, 4.03]]	34.7
4	▸ [1, 4, [], [6.998, 0.03237, 222, 2.94]]	33.4
5	▸ [1, 4, [], [7.147, 0.06905, 222, 5.33]]	36.2
6	▸ [1, 4, [], [6.43, 0.02985, 222, 5.21]]	28.7
7	▸ [1, 4, [], [6.012, 0.08829, 311, 12.43]]	22.9

8	▸ [1, 4, [], [6.172, 0.14455, 311, 19.15]]	27.1
9	▸ [1, 4, [], [5.631, 0.21124, 311, 29.93]]	16.5
10	▸ [1, 4, [], [6.004, 0.17004, 311, 17.1]]	18.9
11	▸ [1, 4, [], [6.377, 0.22489, 311, 20.45]]	15
12	▸ [1, 4, [], [6.009, 0.11747, 311, 13.27]]	18.9
13	▸ [1, 4, [], [5.889, 0.09378, 311, 15.71]]	21.7
14	▸ [1, 4, [], [5.949, 0.62976, 307, 8.26]]	20.4
15	▸ [1, 4, [], [6.096, 0.63796, 307, 10.26]]	18.2

Displaying Predictions

```
PredictionSubsetBostonHouseFeaturizedDF =
lrModel.transform(SubsetBostonHouseFeaturizedDF)

display(PredictionSubsetBostonHouseFeaturizedDF)
```

	features	medv	prediction
1	▸ [1, 4, [], [6.575, 0.00632, 296, 4.98]]	24	28.94606828287893
2	▸ [1, 4, [], [6.421, 0.02731, 242, 9.14]]	21.6	26.182557120005452
3	▸ [1, 4, [], [7.185, 0.02729, 242, 4.03]]	34.7	32.92558845397893
4	▸ [1, 4, [], [6.998, 0.03237, 222, 2.94]]	33.4	32.6271050128185
5	▸ [1, 4, [], [7.147, 0.06905, 222, 5.33]]	36.2	32.1286496691088
6	▸ [1, 4, [], [6.43, 0.02985, 222, 5.21]]	28.7	28.431911125436784
7	▸ [1, 4, [], [6.012, 0.08829, 311, 12.43]]	22.9	21.92619176753286
8	▸ [1, 4, [], [6.172, 0.14455, 311, 19.15]]	27.1	19.168430471036622

9	▸ [1, 4, [], [5.631, 0.21124, 311, 29.93]]	16.5	10.559243095325714
10	▸ [1, 4, [], [6.004, 0.17004, 311, 17.1]]	18.9	19.381487793092283
11	▸ [1, 4, [], [6.377, 0.22489, 311, 20.45]]	15	19.54418523310961
12	▸ [1, 4, [], [6.009, 0.11747, 311, 13.27]]	18.9	21.459387205083573
13	▸ [1, 4, [], [5.889, 0.09378, 311, 15.71]]	21.7	19.526001803925226
14	▸ [1, 4, [], [5.949, 0.62976, 307, 8.26]]	20.4	23.81251554874697
15	▸ [1, 4, [], [6.096, 0.63796, 307, 10.26]]	18.2	23.513902245895693

Let's predict off a hypothetical data point of a 4 bedroom home with 2.9 crime rate, 222 property tax rate and 13% average lower class.

	features	prediction
1	▸ [1, 4, [], [4, 2.9, 222, 13]]	11.334412619024187

Showing all 1 rows.