BOSTON HOUSING

Drivers of Value

MOTIVATION

REAL STATE IS AN ECONOMIC DRIVER

Due to its importance in the economy, real state price is an important trend to evaluate the macroeconomics

FINDING NEW COMPONENTS TO CAPTURE REAL STATE VALUE IS KEY TO IMPROVE ACCURACY

As the market gets more complex, it is important to find new components to help accessing house pricing. This dataset brings some non-traditional drivers of value such as neighborhood pollution and amount of industries nearby

A TAGIBLE AND EASY TO WORK DATA

To try out my new Spark capabilities I decided to use a data easy to judge and with a subject I have worked before

CORRELATIONS AND PREDICTION

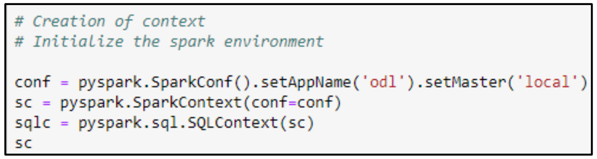
This data set presents opportunity to get some insights from the correlation among variables and to practice some forecasting

DATA SOURCE

https://www.kaggle.com/c/boston-housingsting

HIGHLIGHTS OF CODE USED

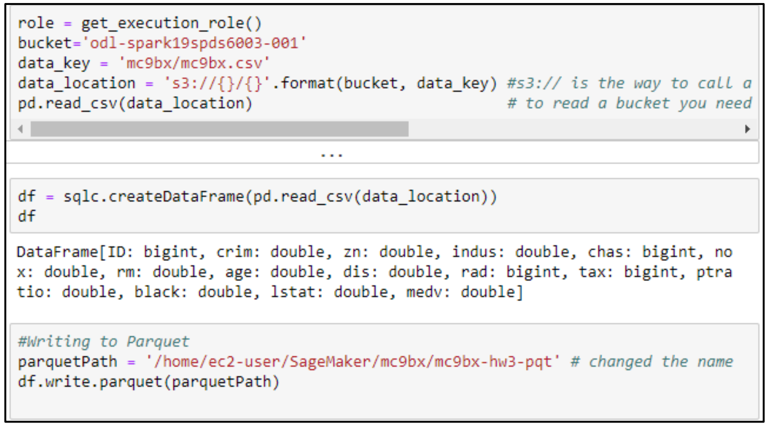
CONTEXT CREATION



```
# Creation of context
# Initialize the spark environment

conf = pyspark.SparkConf().setAppName('odl').setMaster('local')
sc = pyspark.SparkContext(conf=conf)
sqlc = pyspark.sql.SQLContext(sc)
sc
```

PREPARATION OF DATA



```
role = get_execution_role()
bucket='odl-spark19spds6003-001'
data_key = 'mc9bx/mc9bx.csv'
data_location = 's3://{}/{}'.format(bucket, data_key) #s3:// is the way to call a
pd.read_csv(data_location)                            # to read a bucket you need
```

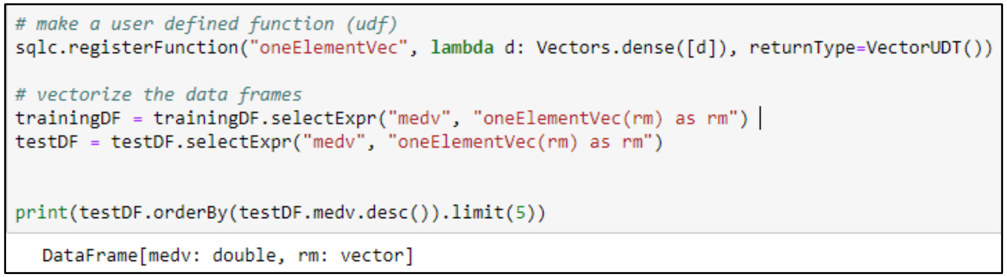
```
...
```

```
df = sqlc.createDataFrame(pd.read_csv(data_location))
df
```

```
DataFrame[ID: bigint, crim: double, zn: double, indus: double, chas: bigint, no
x: double, rm: double, age: double, dis: double, rad: bigint, tax: bigint, ptra
tio: double, black: double, lstat: double, medv: double]
```

```
#Writing to Parquet
parquetPath = '/home/ec2-user/SageMaker/mc9bx/mc9bx-hw3-pqt' # changed the name
df.write.parquet(parquetPath)
```

VECTORIZATION



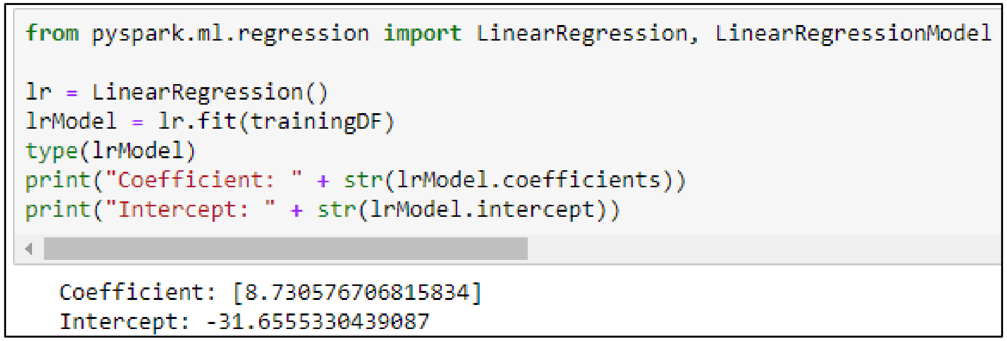
```
# make a user defined function (udf)
sqlc.registerFunction("oneElementVec", lambda d: Vectors.dense([d]), returnType=VectorUDT())

# vectorize the data frames
trainingDF = trainingDF.selectExpr("medv", "oneElementVec(rm) as rm")
testDF = testDF.selectExpr("medv", "oneElementVec(rm) as rm")


print(testDF.orderBy(testDF.medv.desc()).limit(5))
```

```
  DataFrame[medv: double, rm: vector]
```

REGRESSION COEFICCIENTS



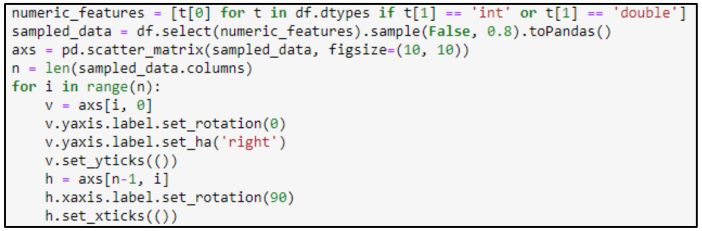
```
from pyspark.ml.regression import LinearRegression, LinearRegressionModel

lr = LinearRegression()
lrModel = lr.fit(trainingDF)
type(lrModel)
print("Coefficient: " + str(lrModel.coefficients))
print("Intercept: " + str(lrModel.intercept))
```

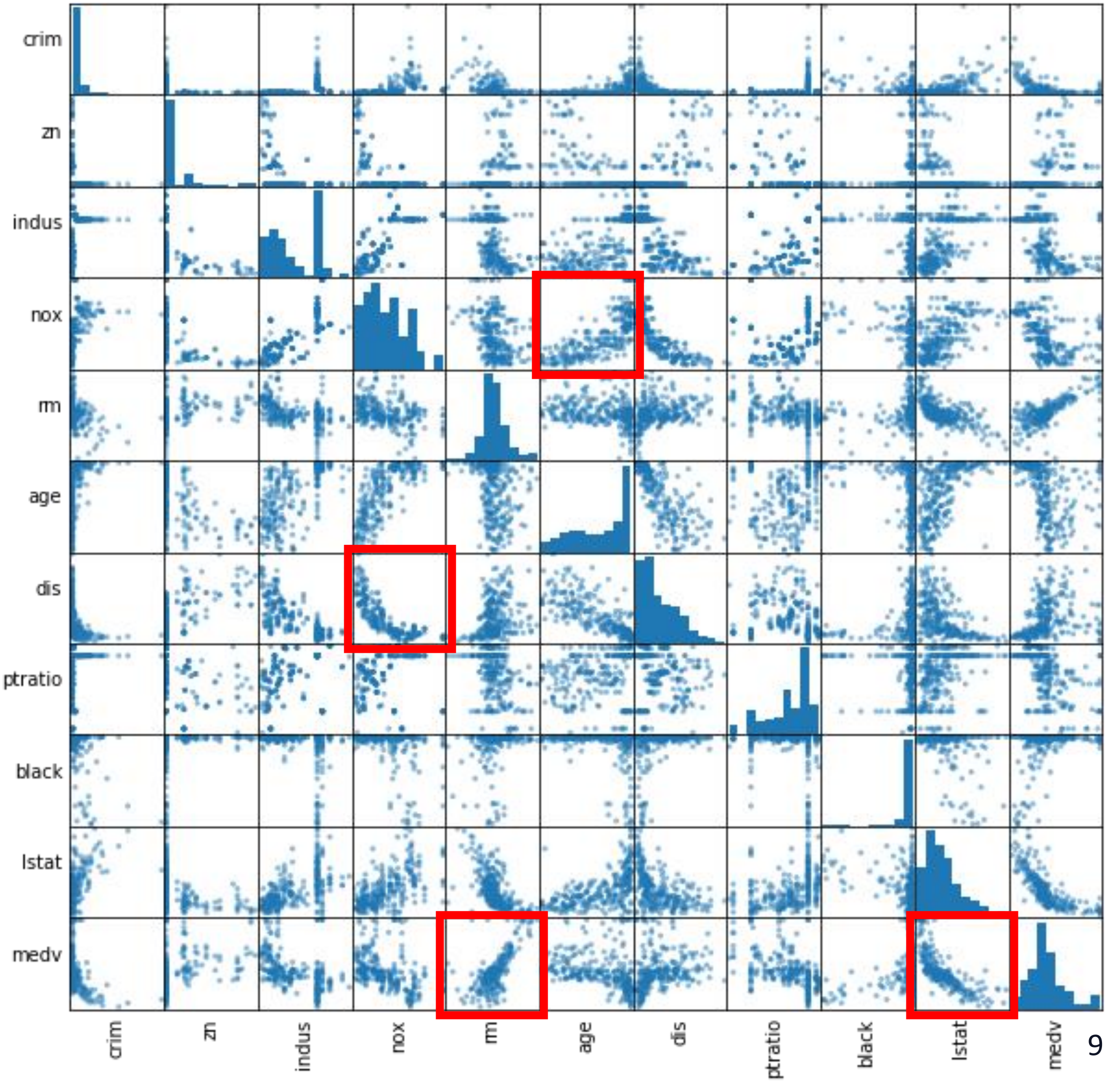
```
  Coefficient: [8.730576706815834]
  Intercept: -31.6555330439087
```

UNIVERSITY OF VIRGINIA
DATA SCIENCE
INSTITUTE

VISUALIZATION



```
numeric_features = [t[0] for t in df.dtypes if t[1] == 'int' or t[1] == 'double']
sampled_data = df.select(numeric_features).sample(False, 0.8).toPandas()
axs = pd.scatter_matrix(sampled_data, figsize=(10, 10))
n = len(sampled_data.columns)
for i in range(n):
    v = axs[i, 0]
    v.yaxis.label.set_rotation(0)
    v.yaxis.label.set_ha('right')
    v.set_yticks(())
    h = axs[n-1, i]
    h.xaxis.label.set_rotation(90)
    h.set_xticks(())
```


crim
zn
indus
nox
rm
age
dis
ptratio
black
lstat
medv
crim
zn
indus
nox
rm
age
dis
ptratio
black
lstat
medv