Name: Charishma Ravoori

Computing Id: cr2st

Motivation:

The motivation behind this regression analysis is to predict the price (of unit area) of homes in Taipei, Taiwan and to understand the features affecting it. This can be useful for people trying to selling homes as it will help them set a reasonable sale price, and also for people trying to buy homes as it can help them understand the housing market and if the price of the house they are trying to purchase is reasonable.

Code:

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

```
#read into spark dataframe from csv in S3
role = get_execution_role()
bucket='odl-spark19spds6003-001'
data_key = 'cr2st/house.csv'
data_location = 's3://{}/{}'.format(bucket, data_key)
pd.read_csv(data_location)
```

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

```
#write parquet to S3
parquetPath = '/home/ec2-user/SageMaker/tmp-pqt'
df.write.parquet(parquetPath)
```

```
# prep list of files to transfer
files = [f for f in listdir(parquetPath) if isfile(join(parquetPath, f))]

s3 = boto3.resource('s3')
for f in files:
    #print('copying {} to {}'.format(parquetPath+'/'+f,"sample_data/"+f))
    s3.Bucket(bucket).upload_file(parquetPath+'/'+f, "cr2st/pqt/"+f)
```

```
#write to spark dataframe from parquet
df = sqlc.read.parquet(parquetPath)
df
```

This part of the code is used to:

- Initialize the spark environment
- Connect to AWS S3 and read in data
- Convert the csv to parquet (this is to increase efficiency of data storage/extraction as it uses column format
- Convert parquet to spark dataframe

```
from pyspark.sql.types import DoubleType

df = df.withColumn("age", df["age"].cast("long"))
df = df.withColumn("distance", df["distance to the nearest MRT station"].cast("long"))
df = df.withColumn("convenience stores", df["convenience stores"].cast("long"))
#df = df.withColumn("latitude", df["latitude"].cast("long"))
#df = df.withColumn("longitude", df["longitude"].cast("long"))
df = df.withColumn("price", df["price"].cast("long"))
```

```
#view schema
df.printSchema()
```

```
#correlation analysis to see which features affect each other
print("Pearson's r(house age,house price of unit area) = {}".format(df.corr('age', 'price')))
print("Pearson's r(dist from MRT,house price of unit area) = {}".format(df.corr('distance to the nearest MRT station', 'price')))
print("Pearson's r(number of convenience stores, house price of unit area) = {}".format(df.corr('convenience stores', 'price')))
```

Here, basic correlation analysis is performed. The variables are converted to 'long', first as the pearson's correlation function doesn't work for other data types well at the moment.

The correlation analysis can give us an idea of what kind of relationships are present in the data.

```
#create a training and testing datasets
seed = 42
(testDF, trainingDF) = df.randomSplit((0.20, 0.80), seed=seed)
print ('training set N = {}, test set N = {}'.format(trainingDF.count(),testDF.count()))
```

This piece of code splits the data into the train and test sets (with 80% of the data in the training data set)

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

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

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

```
# rename to make ML engine happy
trainingDF = trainingDF.withColumnRenamed("price", "label").withColumnRenamed("distance", "features")
testDF = testDF.withColumnRenamed("price", "label").withColumnRenamed("distance", "features")
```

To use the linear regression function the inputs have to be in vector form

```
#starting linear regression
from pyspark.ml.regression import LinearRegression, LinearRegressionModel

lr = LinearRegression()
lrModel = lr.fit(trainingDF)
```

```
type(lrModel)
```

Here, the training data is being fit to a linear model

```
#transforming test set to get predictions
#appending column 'prediction' tp dataframe
predictionsAndLabelsDF = lrModel.transform(testDF)

print(predictionsAndLabelsDF.orderBy(predictionsAndLabelsDF.label.desc()).take(5))
```

Here, prediction on the test set is being done

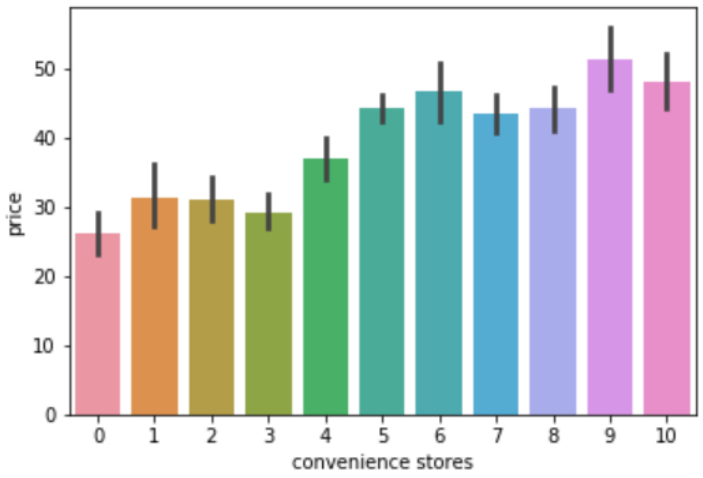
```
#model evaluation
from pyspark.ml.evaluation import RegressionEvaluator
eval = RegressionEvaluator()
print(eval.explainParams())
```

```
eval.setMetricName("rmse").evaluate(predictionsAndLabelsDF)
```

```
eval.setMetricName("r2").evaluate(predictionsAndLabelsDF)
```

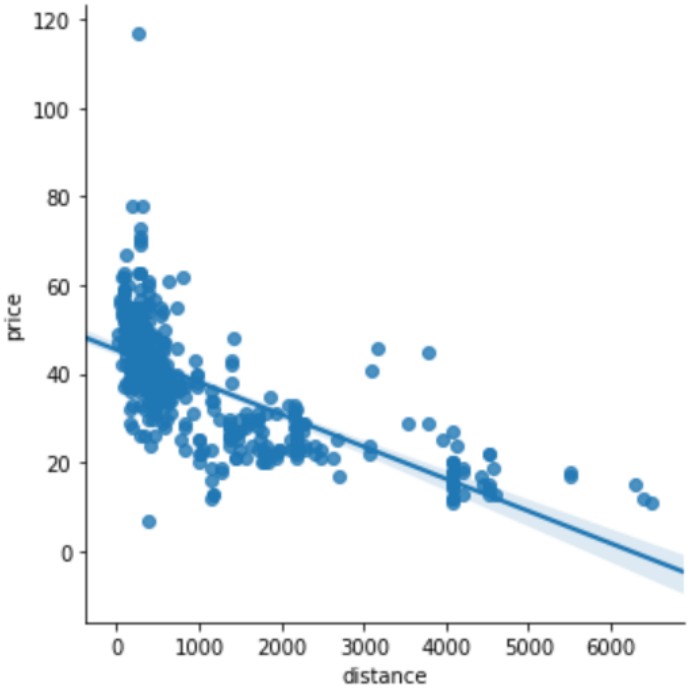
Here, the final part of the code shows the evaluation metric being used : which is the RMSE – mean square error.

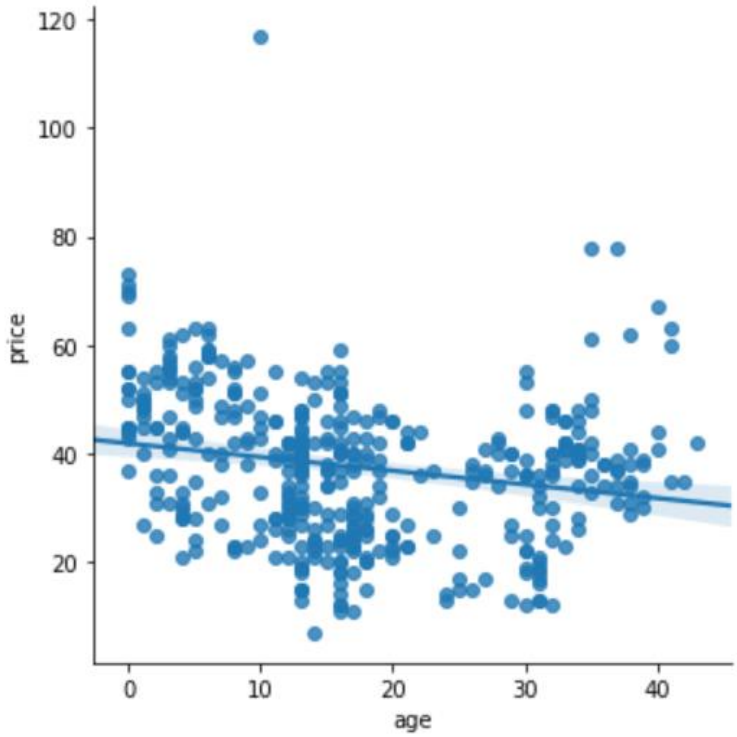
Visualizations:



A bar chart showing the distribution of the variable ‘number of convenience stores’ and how it affects price

Graph showing the distribution of the variable ‘distance from MRT station’ and how it affects price





A graph showing the distribution of the variable ‘age of house’ and how it affects price