Coventry University
FACULTY OF ENGINEERING, ENVIRONMENT & COMPUTING

(7082CEM)

Coursework

Demonstration of a Big Data Program

MODULE LEADER: Dr. Marwan Fuad

Student Name: Antonio Mpembe Franco

SID:

Descriptive and predictive analytics using Hadoop & Anaconda

I can confirm that all work submitted is my own: Yes

Contents

Introduction

The purpose of this paper is to transcribe the coursework of the module, demonstrating and explaining how the project was carried out.

The project applies big data analysis concepts. Store the data in distributed storage, Hadoop, and then process it using the MapReduce programming model. The data is processed for two different main reasons. First, get a complete picture of the data and for a descriptive analysis, where Jupiter Notebook helps with the visualization of the data. And the second is in the static domain, where Machine Learning is applied for predictive analysis.

To fulfil the tasks established for the project. The paper demonstrates how to set the Hadoop in pseudo-distributed mode and how to write and execute MapReduce programs, also evaluating performance. The article explains the thinking process and some parts of the code that are considered crucial. The paper explains the data cleaning process, data exploration and analysis which is the descriptive analysis section then finally the machine learning section which deals with predictive analysis.

Hadoop

Hadoop is an open-source framework for writing and running distributed applications that process large amounts of data. It provides storage and computational resources. It has many characteristics that make it appealing for big data, such as being a fault-tolerant system and capable of scaling out. Being a practical platform to store and process large data with HDFS and MapReduce. And with the latest versions of the framework, it has one more core component, YARN, for job scheduling and managing resources (DR. Marwan Fuad 2020).

The paper explains HDFS and MapReduce more in-depth because they are essential for the project.

HDFS

The Hadoop distributed file system is a storage component of the framework, which is built and optimized to support very large files. HDFS is compatible with MapReduce but can be used independently as it can also be used with Spark (DR. Marwan Fuad 2020).

HDFS works on the master-slave architecture, where one of the two main nodes is the master node, node name, and the other is slave, data node. As the name implies, files can be saved in a distributed manner. Where the pattern stores 3 replicas. And divide them into 64MB or 128MB block assets (DR. Marwan Fuad 2020).

In this project, the dataset is stored into the data node. And the metadata is stored in the name node.

MapReduce

This is a parallel programming model, an algorithm distributed in a cluster. The computational component of Hadoop. With the objective of every computational task being implemented as a mapper and reducer. And in this project will use the Mapper and Reducer to implement the programs (DR. Marwan Fuad 2020).

```
public static class TokenizerMapper
        extends Mapper<Object, Text, Text, IntWritable>{
public static class IntSumReducer
        extends Mapper<Text, IntWritable, Text, IntWritable>{
```

Code snippet 1: Apache Hadoop Word Count example (Apache Hadoop 2020)

As we can see from, *code snippet 1*, the word count example program, the class, "TokenizerMapper", that extends the Mapper has 4 variables, the first 2 define the type of the input, where the first represents the key and the second the value. And the other 2 define the output and work in the same matter as the input. This format is also visible in the class, "IntSumReducer", that extends the Reducer.

Environment Setup

In the project, the Hadoop framework is set in the Linux operative system. And before that, there is a need to set the Java runtime environment specifically the version 1.8, because the Hadoop framework to be set is 2.7.3.

Java – Environment Setup

To get java 1.8 there is a need to install the Java development kit version 8. From the "Home" directory open the terminal.

```
$ sudo apt-get install openjdk-8-jre
```

Code snippet 2: Install Java development kit (Oracle n.d)

And to set the java path, open the ubuntu ".bashrc" file and add in Java path and save the file.

```
$ export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
$ PATH=$JAVA_HOME/bin:$PATH
```

Code snippet 3: Set Java path (Dr.Marwan Faud 2020)

Hadoop - Environment Setup

it is necessary to download the Hadoop framework, and then unzipping it. Add the Hadoop path in the ".bashrc" file and save the file.

```
$ export HADOOP_HOME=~/hadoop-2.7.3
$ PATH=$HADOOP_HOME/bin:$PATH
```

Code snippet 4: set Hadoop path (Dr. Marwan Faud 2020)

The Hadoop environment is set, In the project, the setup used is Hadoop – Pseudo distributed mode. This is not needed to replicate the project. To set the Pseudo distributed mode. It is necessary to add the property "fs.default" to the core-site.xml and "dfs.replication" to the hdfs-site.xml, in both cases in the configuration section.

```
<property>
<name>fs.defaultFS</name>
<value>hdfs://localhost:9000</value>
</property>
```

Code snippet 5: core-site.xml (Apache Hadoop 2020)

```
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
```

Code snippet 6: hdfs-site.xml (Apache Hadoop 2020)

And to run a MapReduce job on YARN in a pseudo-distributed, there is a need to add a few parameters. Add the property “mapreduce.framework.name” to the mapred-site.xml and “yarn.nodemanager.aux-services” to the yarn-site.xml, in both cases in the configuration section. (Apache Hadoop 2016).

```
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
```

Code snippet 7: mapred-site.xml (Apache Hadoop 2020)

```
<property>
<name>yarn.nodemanager.aux-
services</name>
<value>mapreduce_shuffle</value></prop
erty>
```

Code snippet 8: yarn-site.xml (Apache Hadoop 2020)

Running Hadoop

To run HDFS, it is necessary to run the 2 main nodes, name node and data node.

```
$ hdfs namenode -format
$ hdfs namenode > namenode.log 2>&1 &
$ hdfs datanode > datanode.log 2>&1 &
```

Code snippet 9: run HDFS (Dr. Marwan Faud 2020)

To run YARN, it is also necessary to run the 2 main component of YARN, resource manager and node manager.

```
$ yarn resourcemanager > resourcemanager.txt 2>&1 &
$ yarn nodemanager > nodemanager.txt 2>&1 &
```

Code snippet 10: run YARN (Dr. Marwan Faud 2020)

Anaconda – Environment Setup

Anaconda is recommended to install because it contains libraries important for this project, such as Matplotlib, pandas and Jupyter Notebook. This is a package manager for scientific libraries in python.

To install Anaconda, search for the name. The download the Windows versions and follow the default installation preferences.

Now that is ready to run the Jupyter Notebook.

Data acquisition

The dataset used in the project is called “Predict the house prices in India” from Kaggle (Anmol Kumar 2020). The project only uses the train.csv file. The description of the dataset in the time of making the project.

This dataset contains 29451 examples and 12 features.

Feature	Description	Data type
POSTED_BY	Category marking who has listed the property	String
UNDER_CONSTRUCTION	Under Construction or Not	Boolean
RERA	Rera approved or Not	Boolean
BHK_NO	Number of Rooms	Numeric
BHK_OR_RK	Type of property	String
SQUARE_FT	The total area of the house in square feet	Numeric
READY_TO_MOVE	Category marking Ready to move or Not	Boolean
RESALE	Category marking Resale or not	Boolean
ADDRESS	Address of the property	String
LONGITUDE	Longitude of the property	Numeric
LATITUDE	Latitude of the property	Numeric
TARGET(PRICE_IN_LACS)	Price in (1/100000) rupees	Numeric

Table 1: Dataset description

BHK means bedroom, hall, and kitchen. RK means room and kitchen. And RERA stands for real estate regulatory authority.

Some of the information is irrelevant such as the address because it is described by longitude and latitude. Still, it is important to visualize the data, because it is easier for humans to remember names than numbers. And the BHK_OR_RK feature is useless because all the examples in the dataset are BHK.

Data cleaning

It is important to check for missing features in each example. Find and remove any replicated examples that could lead to incorrect data set analysis. In the description of the data set, it is noticed that some characteristics are irrelevant, and others are a little difficult to understand the magnitude, such as feet and rupee.

For all this, it is necessary to create some Hadoop MapReduce program in Java. For this, an integrated development environment (IDE) is used. The use of IntelliJ or Eclipse is recommended.

Setting an integrated development environment

IntelliJ is the IDE of choice, but choosing Eclipse is also good. Installing IntelliJ is simple.

```
$ sudo snap install intellij-idea-community --classic
```

Code snippet 11: install IntelliJ (JetBrains n.d)

After installing the IDE, create a maven project and add the dependencies. Or to create a Java project and add the necessary Hadoop jar file.

```
<dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-mapreduce-examples</artifactId>
        <version>2.7.3</version>
        <scope>provided</scope>
</dependency>
...
<dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-common</artifactId>
        <version>2.7.3</version>
        <scope>provided</scope>
</dependency>
```

Code snippet 12: Hadoop dependecies (Microsoft 2020)

Also add a plugin within the build, specifying the version of Java required in this project, 1.8.

```
<plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.6.1</version>
        <configuration>
        <source>1.8</source>
        <target>1.8</target>
        </configuration>
</plugin>
```

Code snippet 13: Hadoop plugin (Microsoft 2020)

Missing features

In this process, it is necessary to check each example and see if any features are missing. There is no need for the Reducer class for this process, as it is enough to select which line is missing and which feature is missing. Code of the author of the article.

```
int lines = 0;
    ...
    int i=0; lines++;
    while (i<words.length){  //word is each feature of one example
        if(words[i].trim().isEmpty()){
            context.write(new Text(String.valueOf(lines)), new
                  Text(String.valueOf(i)) );
        ...
```

Code snippet 14: EmptyFeatures.java (Antonio Franco 2020)

As seen from the code above, there is a value called “lines” that is the line of examples and a value called “i” which is the missing feature. On this dataset, there is no missing feature.

Repeated Example

To do this, you need to find and remove all replicated examples. To replicate the project, you can use the program shown in the article that removes replicated examples. But on paper, there is a program that finds repeated examples, one that counts examples considered extra and another that removes replicas.

Program to find repeated examples.

This code finds the repeated examples and how many times it has been replicated. For this program, there is the use of a Mapper class and a Reducer. Code of the author of the article.

```
public void map(Object key, Text value, Context context)
...
        context.write(new Text(wordOut), new IntWritable(1));
        // word is the example
        ...

public void reduce(Text key, Iterable<IntWritable> values,Context context)
...
        for (IntWritable val : values) {
                sum += val.get();
        if(sum >= 2){
                context.write(key, result);//result is sum as IntWritable
```

Code snippet 15: DublicateChecker.java (Antonio Franco 2020)

As it can be seen in the code above in the reduce function. The code prints the example that appears more than once. The example is the "key" value, and its frequency is the "result" value. As seen in output example has appeared 2 times.

```
Builder,0,0,1,BHK,422.15336950000005,1,0,"Hoshangabad Road,Bhopal",22.75,77.72,10.9 2
                                ...
```

Output 1: Repeated examples

Count extra examples

This program counts extra example from the output of the "*Program to find repeated examples*". From the output of this program and the knowledge of the dataset. The output of this program and the knowledge of the dataset helps verify that the replicas have been removed correctly. Code of the author of the article.

```
public void map(Object key, Text value, Context context)
...
        int replication = Integer.parseInt(words[1]) - 1;
        context.write(new Text("Number of replicas is:"), new
                IntWritable(replication));
        ...

public void reduce(Text key, Iterable<IntWritable> values,Context context)
...
        for (IntWritable val : values) {
                sum += val.get();
        context.write(key, result); //result is sum but as IntWritable
```

Code snippet 16: DublicateCounter.java (Antonio Franco 2020)

As can be seen in the code above, the value "replication" reads the frequency of the example minus the original example. Adding all the extra frequency, you get the number of all the extra examples. As seen in the output example, 401 replicas need to be removed.

```
Number of replicas is:    401
```

Output 2: Count extra examples

Remove replicated examples

In this program, replicas are removed. Meaning the output file will contain examples without any replica. Code of the author of the article.

```
        public void map(LongWritable key, Text value, Context context)
        ...
                context.write(value, new Text("1"));
                ...

        public void reduce(Text key, Iterable<Text> values,Context context)
        ...
                context.write(key, new Text(""));
```

Code snippet 17: DublicateRemover.java (Antonio Franco 2020)

As can be seen in the code above. There is just a need to print out the key. Now that the Reducer as is applied on the dataset, the new dataset is sorted.

```
Builder,0,0,1,BHK,422.15336950000005,1,0,"Hoshangabad Road,Bhopal",22.75,77.72,10.9
                                        ...
```

Output 3: Remove replicated examples

Constant feature

With the knowledge that is shown about the dataset. Is known that there is one of the features is constant. BHK_OR_RK where all the examples are BHK. There is a need to change the feet to meters and Indian Rupees to British Pound because the project is from the United Kingdom. Easier for the values to be understood.

Feature remove

For this there only need for the Mapper class. Code of the author of the article.

```
        public void map(LongWritable key, Text value, Context context)
                ...
                if(lines == 29050){// variable lines starts from 0
                }else{
                        Double GBP = (Double.parseDouble(words[11])/94.94)*100000;
                        Double m = Double.parseDouble(words[5])/10.7639104;
                        ...
                        String wordOut = ... words[3] + "," + words[5] + ...;
                        context.write(new Text(wordOut), new Text(""));
```

Code snippet 18: FeatureRemover.java (Antonio Franco 2020)

As can be seen in the code above, the code does not select the 5th feature, index 4 because it is where the where the name of the feature are. And it does not print them because it ideal for them to be at the top.

```
Builder,0,0,1,39.21933143367674,1,0,"Hoshangabad Road,Bhopal",22.75,77.72,11480...
                                        ...
```

Output 4: Feature remove

Features names on top

In this program will put the features names no the top of the dataset, meaning on the first line of the dataset. Code of the author of the article, file FeatureRemover.java (Antonio Franco 2020).

```
public void map(LongWritable key, Text value, Context context)
        if(lines == 0){
                context.write(new Text("...,SQUARE_M,...,TARGET(PRICE_IN_BGP)"),...
           }
           context.write(value, new Text(""));                                  ...
```

As can be seen from the code above, the title or features names are printed in the first then the rest of dataset is printed.

```
          ...,BHK_NO.,SQUARE_M,...,TARGET(PRICE_IN_GBP)
                                    ...
```

Data exploration and analysis

This section is focused on analysing the dataset by analysing each feature and relationship between them. By visualizing the dataset and its statistical characteristics.

Now is time to change to Windows but it is not necessary because Jupyter Notebook can be used in both operative systems. For this, it is only necessary to install Anaconda. That comes with the Jupyter Notebook and other important things for data science.

First is necessary to import the dataset file, to the Jupyter Notebook. And change the data type of the feature because the CSV files save everything as Strings.

```
import unicodecsv
prices = []
f = open('train.csv', 'rb')
with open('train.csv', 'rb') as f:
    reader = unicodecsv.DictReader(f)
    prices = list(reader)
for price in prices:
    price['UNDER_CONSTRUCTION'] = bool(int(price['UNDER_CONSTRUCTION']))
         ...
    price['TARGET(PRICE_IN_BGP)'] = price['TARGET(PRICE_IN_BGP)\t']
    del price['TARGET(PRICE_IN_BGP)\t']
         ...                                  ...
```

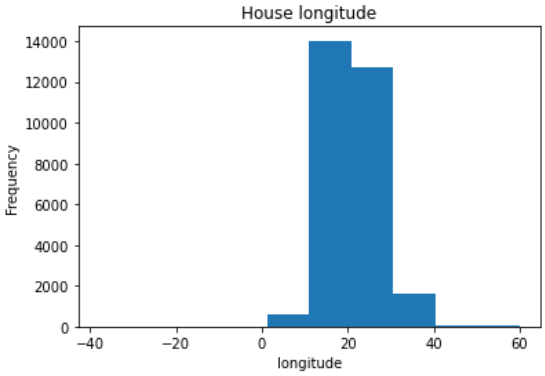
Individual features

Continuous data

Continuous data are numerical data with meaningful information that represent values in a set that are uncountable. In the dataset of this project, there are some features with continuous variables such as “SQUARE_M”, “LONGITUDE”, “LATITUDE”, “TARGET(PRICE_IN_BGP)”.

Using a Histogram is a good way to see if the feature has a normal distribution. Kolmogorov Smirnov Statistic test. “If the P-Value of the KS Test is smaller than 0.05, we do not assume a normal distribution” (Joos Korstanje 2019).

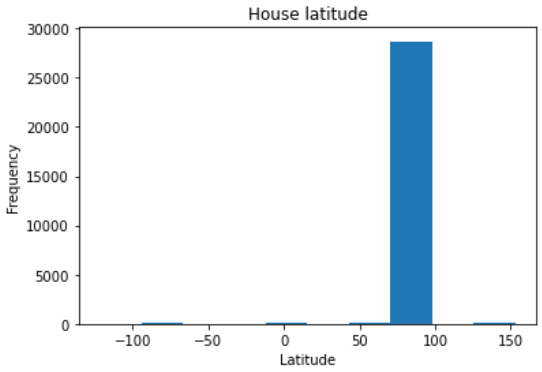
```
import matplotlib.pyplot as plt
import pandas as pd
from scipy.stats import kstest, norm
pricesDataset = pd.DataFrame(prices)
b = np.arange(pricesDataset['SQUARE_M'].min(), pricesDataset['SQUARE_M'].max())
print(pricesDataset['SQUARE_M'].describe())
print(np.median(data))
plt.hist(pricesDataset['SQUARE_M'])
plt.title("House area in square metre")
plt.xlabel("Square meter")
plt.ylabel("Frequency")
plt.show()
my_data = norm.rvs(data["BHK_NO."])
ks_statistic, p_value = kstest(my_data, 'norm')
print(ks_statistic, p_value)
```



The data description of “SQUARE_M”:

```
count     2.905000e+04
mean      1.861007e+03
std       1.778546e+05
min       2.787091e-01
25%       8.361274e+01
50%       1.089513e+02
75%       1.440154e+02
max       2.364805e+07
Median    108.9512622661742
P_value   0.0
```

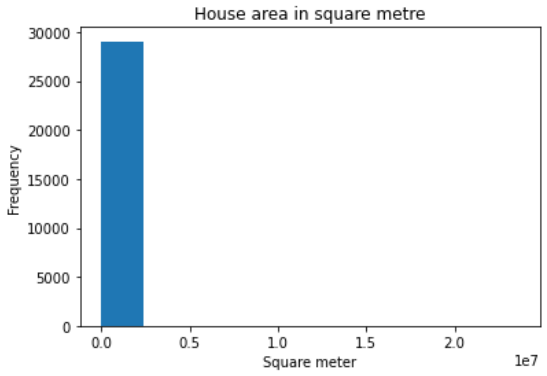
Starting with the area of the house feature, “SQUARE_M”. As can be seen from this histogram most of the examples of the houses have a big range which can be observer by the massive difference between the mean and median. It has no normal distribution.



The data description of “LONGITUDE”:

```
count    29050.000000
mean        21.270272
std          6.195973
min        -37.713008
25%         18.452663
50%         20.631532
75%         26.886881
max         59.912884
Median      20.631532
P_value     0.0
```

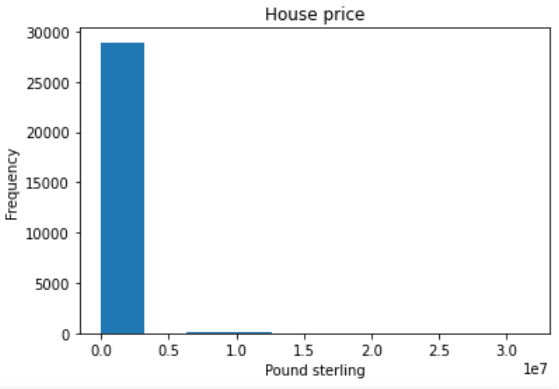
There is no need to show the code for the other features because is the same thought process. The same amount of example as the “SQUARE_M”, all the feature has the same number of examples. the feature, “LONGITUDE”. As can be seen from the histogram the examples are more distributed compare to “SQUARE_M” feature, and as a smaller ranger. This can also be observed on the mean. It has no normal distribution.



The data description of “LATITUDE”:

```
count     29050.000000
mean         76.829585
std          10.567572
min        -121.761248
25%          73.794800
50%          77.322873
75%          77.912934
max         152.962676
Median       77.322873
p_value       0.0
```

The feature “LONGITUDE” and “LATITUDE” describe the feature “ADDRESS”. As can be seen from the histogram the examples are less distributed compare to the feature “LONGITUDE”, but the values are further away from each other. This can be seen from looking at the standard deviation. It has no normal distribution.



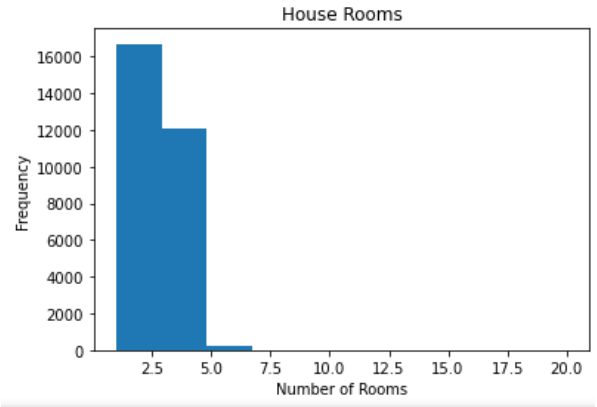
The data description of “TARGET(PRICE_IN_BGP)”:

```
count     2.905000e+04
mean      1.492655e+05
std       6.847017e+05
min       2.633242e+02
25%       4.002528e+04
50%       6.504108e+04
75%       1.053297e+05
max       3.159890e+07
Median    65041.078575942694
P_value   0.0
```

The feature “TARGET(PRICE_IN_BGP)” has a lot of similarities with “SQUARE_M”. Both standard deviations are big and in both, there is a big difference between the mean and median. This may suggest that there is more correlation between these two features compare to the other continuous values. It has no normal distribution.

Discrete data

This is numerical data with meaningful information that represent items that can be counted. In the dataset of this project there a discrete feature “BHK_NO.”.



The data description of “BHK_NO.”:

```
count     29050.000000
mean          2.390809
std           0.880677
min           1.000000
25%           2.000000
50%           2.000000
75%           3.000000
max          20.000000
Median        2.0
p_value       0.0
```

Can be seen by the median that most houses have 2 rooms and more than 75% of the example have 3 or fewer rooms.

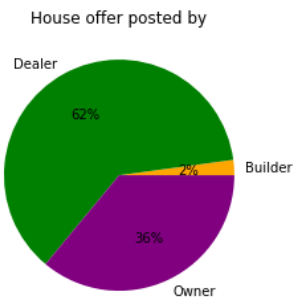
The normal distribution can not be assumed for the numerical feature in the dataset.

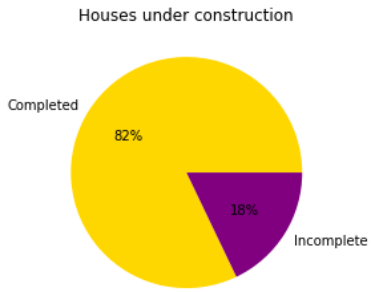
Categorical data

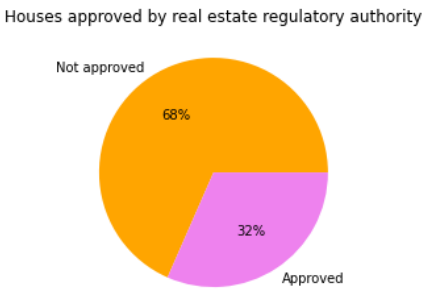
"Categorical variables represent types of data which may be divided into groups (Anon 1997-1998). To represent this visually the use of a bar chart or pie chart is recommended. Because trying to represent this type of data with a histogram does not work. At least it did not work during the execution of the project.

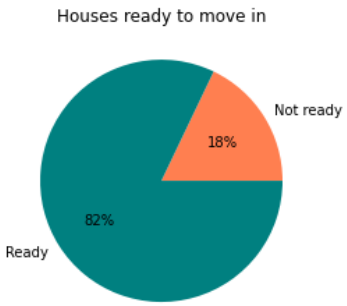
In this data set there are categorical features, such as, "POSTED_BY", "UNDER_CONSTRUCTION", "RERA", "READY_TO_MOVE", "RESALE", "ADDRESS".

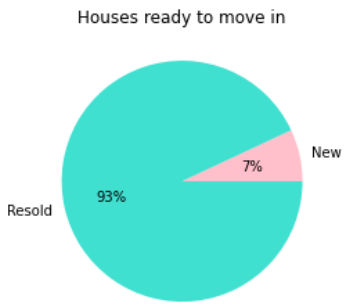
```
data = pricesDataset['POSTED_BY']
(unique, counts) = np.unique(data, return_counts=True)
frequencies = dict(np.asarray((unique, counts)).T)
x = list(frequencies)
y = list(frequencies.values())
plt.pie(y, labels= x, colors=['orange', 'green', 'purple'], autopct='%1.0f%%')
plt.title("House offer posted by")
plt.show()
```









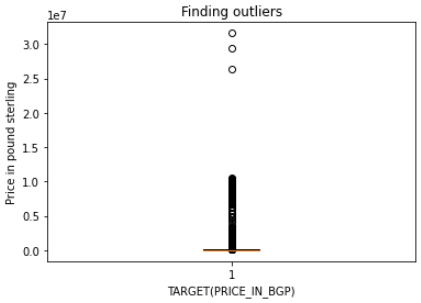


Notice for "ADDRESS" feature it is visually difficult to interpret 6899 unique location. This feature is not represented for that reason. This is not a big problem because the "ADDRESS" feature is described with more detail by the feature "LONGITUDE" and "LATITUDE".

Outlier

"An outlier is a data point that differs significantly from other observations. An outlier may be due to variability in the measurement or it may indicate experimental error; the latter are sometimes excluded from the data set" (Anon 2020). Most of features will and probably have outlier. It is possible to show all outlier because there is a lot of them. And because of having different magnitude, the outlier of some features will not be visible. Finding outliers (Anon 2020).

```
from numpy import percentile
data = pricesDataset['TARGET(PRICE_IN_BGP)']
plt.boxplot(data)
plt.show()
y = list(frequencies.values())
plt.pie(y, labels= x, colors=['orange', 'green', 'purple'], autopct='%1.0f%%')
plt.title("House offer posted by")
plt.show()
q1, q3 = percentile(data, 25), percentile(data, 75)
out = q3 - q1
margin = out * 1.5
lower, upper = q1 - margin, q3 + margin
outliers = [x for x in data if x < lower or x > upper]
print(len(outliers))
```



From the chart, it is difficult to see all the outlier because there are many of them. There 3 noticeable outliers. Having many to say is difficult if all are errors.

```
TARGET(PRICE_IN_BGP) outliers: 3045
BHK_NO. outliers: 283
LATITUDE outliers: 2934
LONGITUDE outliers: 144
SQUARE_M outliers: 1612
```

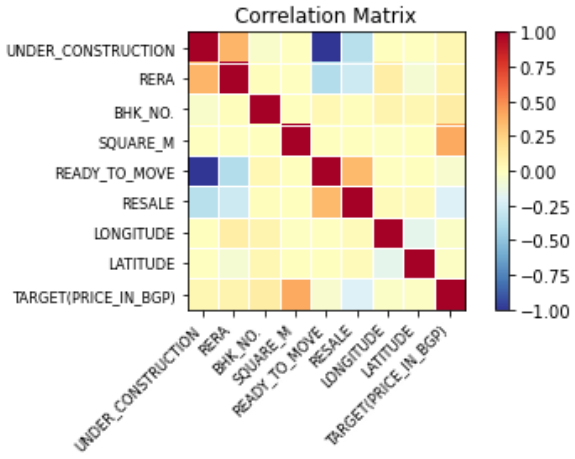
Feature relations

The relationship between each feature and the relationship between price and the rest of the features. To visualize the information is recommended a scatterplot and a correlation map with "matshow matplotlib".

Knowing the relationship between price and other feature will help to choose the feature that most affect the price to represent their relationship in a scatterplot and a correlation matrix.

Correlation matrix

```
from statsmodels import api
corr = data.corr()
print(corr)
api.graphics.plot_corr(corr, xnames=list(corr.columns))
plt.show()
```




```
                        TARGET(PRICE_IN_BGP)
UNDER_CONSTRUCTION                  0.054618
RERA                                0.067314
BHK_NO.                             0.113654
SQUARE_M                            0.409571
READY_TO_MOVE                      -0.054618
RESALE                             -0.203275
LONGITUDE                          -0.030505
LATITUDE                           -0.017813
TARGET(PRICE_IN_BGP)                1.000000
```

As to be expected the area, “SQUAERE_M”, of the house as more correlation with price, “TARGET (PRICE_IN_BGP)”, of the house compared to the other features in this dataset. The area, “SQUARE_M”, the feature is followed by resale, “RESALE”, a feature which has a negative correlation, then there is the number, “BHK_NO.”, and so on.

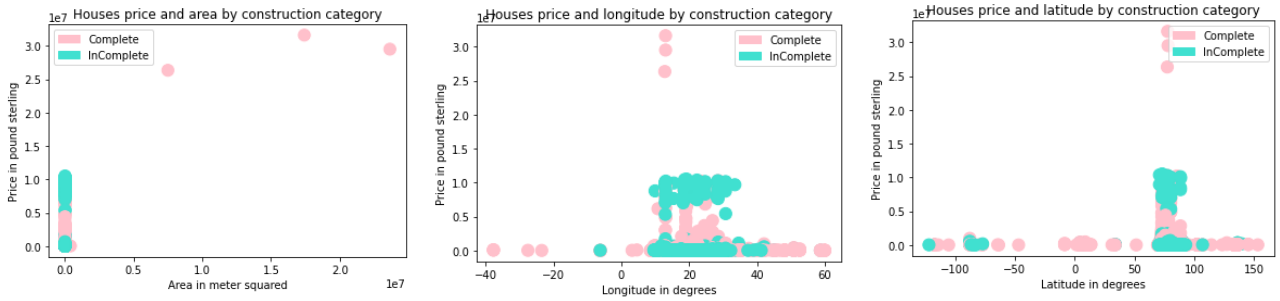
From the correlation value and it is easier to see which features have the most impact on the price of the house.

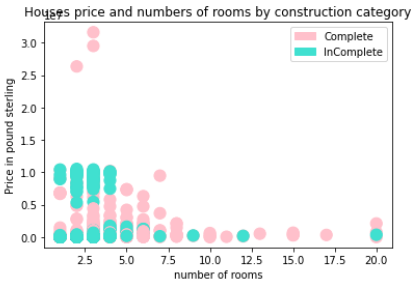
With this knowledge, choosing the most important feature becomes easier.

Using scatterplot is away to visualise the correlation between features. There is already the knowledge of the correlation between features. The plot is used in this situation to represent the correlation between the target feature, “TARGET (PRICE_IN_GBP)”, and numerical features which in this case are continuous variables and discrete, as well as with categorical feature.

Scatter plot to visualize correlation (Ex: Under construction category)

```
import matplotlib.patches as mpatches
colormap = np.array(['pink', 'turquoise'])
rsls = []
for rsl in pricesDataset['UNDER_CONSTRUCTION']:
    if rsl:
        rsls.append(1)
    else:
        rsls.append(0)
category = np.array(rsls)
plt.scatter(pricesDataset['SQUARE_M'], pricesDataset['TARGET(PRICE_IN_BGP)'],
color=colormap[category],s=124 )
...
plt.show()
```





The feature about construction, "UNDER_CONSTRUCTION", as a slight positive correlation with the price. Houses under construction or incomplete seem to be more expensive. Probably because newer houses are more expensive overall. And it seems like the newer houses are approximately in the same area, from what can interpret from the longitude and latitude charts. In all these charts with 3 outliers. This scatter plot could be done with other categories the result will be similar.

Data prediction

The objective of this section is to predict a feature of continuous values. And "Regression algorithms are machine learning techniques for predicting continuous numerical values. They are supervised learning tasks which means they require labelled training examples" (Stacey Ronaghan 2018).

Most common regression algorithms are linear regression, decision trees, neural networks, and k-nearest neighbours.

The chosen algorithm for this project is linear regression because it is quick to compute and easier to understand.

Multiple linear regression Anaconda

The dataset is divided into 3 different datasets that represent the 3 different categories in the "POSTED_BY" feature. Which means the is only 8 features to predict the price of the houses and the outliers are removed.

```
OutdfB, OutdfD, OutdfO = [], [], []
for price in prices:
    if price['TARGET(PRICE_IN_BGP)'] < upper and price['TARGET(PRICE_IN_BGP)'] >
lower:
        und, rer, rea, res = 0, 0, 0, 0
        if price['UNDER_CONSTRUCTION'] == True:
            und = 1
        ...
        case = {'UNDER_CONSTRUCTION': und, 'RERA': rer, 'BHK_NO.': ...
        elif price['POSTED_BY'] == 'Owner':
            OutdfO.append(case)
```

This can be done using the "statsmodels.api" library or using the "sklearn.linear_model" library (Adi Bronshtein 2017; Gurucharan M K 2020).

```
X = dfD[['UNDER_CONSTRUCTION', 'RERA', 'BHK_NO.', ... 'LATITUDE']]
y = dfD['TARGET(PRICE_IN_BGP)']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
from sklearn.metrics import mean_squared_error, r2_score
y_pred = regressor.predict(X_test)
df = pd.DataFrame({'Real Values':y_test, 'Predicted Values':y_pred})
print('Coefficients: \n', regressor.coef_)
print('Coefficient of determination: %.2f'% r2_score(y_test, y_pred))
print('intercept: %.2f'% regressor.intercept_)
```

```
Dataset Dealer
Coefficients:
        [-1.34980027e+03  7.31831105e+03  2.19362102e+04  8.26752731e+00
        1.34980027e+03 -1.35724794e+03 -1.78489322e+03 -6.87626178e+02]
        Coefficient of determination: 0.19
        intercept: 119770.31
Dataset Owner
        Coefficients:
        [-3.31300373e+01  7.96835413e+03  2.37148293e+04 -6.65789331e-02
        3.31300373e+01  1.15095449e+04 -9.11468057e+02 -1.22642010e+02]
        Coefficient of determination: 0.20
        intercept: 15481.42
Dataset Builder
        Coefficients:
        [ 1847.8431142   7773.46673221 13419.03854945   169.13910854
        -1847.8431142   4191.14400871  -949.61508333   -59.41409203]
        Coefficient of determination: 0.36
        intercept: 21805.75
```

This section the feature “POSTED_BY” is being used as part of the equation to predict the price, by attributing each category a value between 0 and 2. Which means there is 9 feature to predict the price of the houses and the outliers will still be removed.

```
Coefficients:
 [-1.40422258e+04 -1.27342166e+03  8.33762585e+03  2.22691667e+04
 -6.14989452e-02  1.27342166e+03  6.57412830e+03 -1.38440043e+03
 -3.75001237e+02]
Coefficient of determination: 0.28
intercept: 79182.19
```

Coefficient of determination is the probability that the prediction will get the right result, the percentage a very low.

```
TARGET(PRICE_IN_BGP) = intercept + coefficient_0 * feature_0 + ...
```

Multiple linear regression Hadoop

This section is the same as the previous one, “Multiple linear regression Anaconda”, but in Hadoop. First the dataset is divided into 3 different datasets that represent the 3 different categories in the “POSTED_BY” feature. Same conditions with the previous section.

```
            if(i > 0) {
                String line = value.toString();
                String[] words = line.split(",(?=(?:[^\"]*\"[^\"]*\")*[^\"]*$)", -1);
                x[lines][0] = Double.parseDouble(words[0]);
                  ...
                x[lines][7] = Double.parseDouble(words[7]);
                y[lines] = Double.parseDouble(words[8]);
                if (i == 596){
                    OLSMultipleLinearRegression regression = new
OLSMultipleLinearRegression();
                    regression.newSampleData(y, x);
                    double[] beta = regression.estimateRegressionParameters();
                    double Rto2 = regression.calculateRSquared();
                    regression.calculateResidualSumOfSquares();
```

The knowloge of how many lines does the documents has it is necessary. But this could be done with a help of reducer. Knowing how many lines the dataset has it necessary but not critical.

```
Dataset Builder
       estimateRegressionParameters
       2.3997131030810665 6E17, -2.39971310308074208E17, 7348.595211490563,
       13339.484268994125, 162.99690441648318, -2.39971310308080352E17,
       25832.168087209775, -1068.4355716195616,
       Intercept -123.82820729308233
         RSquared 0.3529548585255391
Dataset Dealer
       estimateRegressionParameters
       -2.9199077380932356E16, 2.91990773810511E16, 7923.147033127188,
       22028.57255366322, 9.23605557822724, 2.91990773810536E16, -908.3253573806275,
       -1814.0499500582832,
       Intercept -695.0997129812757
         RSquared 0.17990994158298868
Dataset Owner
         estimateRegressionParameters
         5.092639880559639E-9, 17935.073034642734, 7725.002218759843,
         22012.261648864463,-0.06751209352030543,
         17906.769278836255,11602.137110530333,892.1948025735722,
         Intercept -112.97273519692344
         RSquared 0.32021767209381014
```

References

DR. Marwan Fuad, 2020, Introduction to Big Data Analytics, lecture notes, 7082CEM - Big Data Management and Data Visualisation - 2021SEPJAN, The University of Coventry, delivered 23 September 2020.

Anmol Kumar, 2020, House price prediction challenge, Kaggle, viewed 15 October 2020, https://www.kaggle.com/anmolkumar/house-price-prediction-challenge/

Apache Hadoop, 2020, MapReduce Tutorial, Apache Hadoop, viewed 14 October 2020, https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html/

Apache Hadoop, 2016, Hadoop: Setting up a Single Node Cluster, Apache Hadoop, viewed 14 October 2020, https://hadoop.apache.org/docs/r2.7.3/hadoop-project-dist/hadoop-common/SingleCluster.html/

Oracle, n.d, How to download and install prebuilt OpenJDK packages, Oracle, viewed 14 October 2020, https://openjdk.java.net/install/

DR. Marwan Fuad, 2020, Hadoop - Local Mode, lab notes, 7082CEM - Big Data Management and Data Visualisation - 2021SEPJAN, University of Coventry, delivered 23 September 2020.

DR. Marwan Fuad, 2020, Hadoop - Pseudo-distributed Mode, lab notes, 7082CEM - Big Data Management and Data Visualisation - 2021SEPJAN, University of Coventry, delivered 30 September 2020.

JetBrains, n.d, Install as a snap package on Linux, JetBrains, viewed 14 October 2020, https://www.jetbrains.com/help/idea/installation-guide.html#snap/

Microsoft, 2020, Develop Java MapReduce programs for Apache Hadoop on HDInsight, viewed 14 October 2020, https://docs.microsoft.com/en-us/azure/hdinsight/hadoop/apache-hadoop-develop-deploy-java-mapreduce-linux/

eNord9, 2014, How to make IPython notebook matplotlib plot inline, Stack Overflow, viewed 17 October 2020, https://stackoverflow.com/questions/19410042/how-to-make-ipython-notebook-matplotlib-plot-inline/24884342#24884342/

Anon, 1997-1998, Categorical Data, Yale, viewed 19 October 2020, http://www.stat.yale.edu/Courses/1997-98/101/catdat.htm/

Anon, 2020, Outlier, Wikipedia, viewed 20 October 2020, https://en.wikipedia.org/wiki/Outlier/

Anon, 2020, How to Calculate Outliers, WikiHow, Viewed 20 October 2020, https://www.wikihow.com/Calculate-Outliers/

Joos Korstanje, 2019, 6 ways to test for a Normal Distribution — which one to use?, towards data science, Viewed 20 October 2020, https://towardsdatascience.com/6-ways-to-test-for-a-normal-distribution-which-one-to-use-9dcf47d8fa93/

Stacey Ronaghan, 2018, Machine Learning: Trying to predict a numerical value, Medium, viewed 21 October 2020, https://medium.com/@srnghn/machine-learning-trying-to-predict-a-numerical-value-8aafb9ad4d36#:~:text=Regression%20algorithms%20are%20machine%20learning,they%20require%20labelled%20training%20examples./

Gurucharan M K, 2020, Machine Learning Basics: Multiple Linear Regression, towards data science, Viewed 20 October 2020, https://towardsdatascience.com/machine-learning-basics-multiple-linear-regression-9c70f796e5e3/

Adi Bronshtein, 2017, Simple and Multiple Linear Regression in Python, towards data science, Viewed 20 October 2020, https://towardsdatascience.com/simple-and-multiple-linear-regression-in-python-c928425168f9/

Antonio Franco, 2020, DataFilter, https://github.com/tonyamf/DataFilter/