APPENDIX

(A)

```
---
title: "Distributed Data Analysis"
author: "2012754"
date: "21/03/21"
output: html_notebook
---

This is an [R Markdown](http://rmarkdown.rstudio.com) Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

#HOUSING DATA
```{r}
library(dplyr)
#import dataset and inspect file
housing <- read.csv("C:/Users/tochi/Desktop/Housing_Data_GreaterLondon_2018-20.csv", header = TRUE)
str(housing)
summary(housing)

#drop columns
housing_clean <- select(housing, -'saon', -'paon', -'estate_type', -'street', -'linked_data_uri', -'county', -'transaction_category', -'street', -'locality')
view(housing_clean)
#convert into correct date format
housing_clean$deed_date <- as.Date(housing_clean$deed_date, format = "%Y-%m-%d")

#convert into categorical
housing_clean$property_type <- as.factor(housing_clean$property_type)
housing_clean$new_build <- as.factor(housing_clean$new_build)

#change deed date to date prior to our join
names(housing_clean)[3] <- 'date'

#format to just month and year
housing_clean$date <- format(housing_clean$date, "%m/%Y")

```
````

(B)

```r
```
##DATA JOINING STREET CRIME WITH HOUSING

```{r}
library(zoo)
library(anytime)


#read in the street data and inspect it
street_crime <- read.csv('C:/Users/tochi/Desktop/metropolitan-street-final.csv')
str(street_crime)

#convert into categorical
street_crime$Crime.type <- as.factor(street_crime$Crime.type)
street_crime$postcode <- as.factor(street_crime$postcode)

street_crime$Month <- anytime(street_crime$Month)
street_crime$Month <- as.Date(street_crime$Month, format = "%Y-%m-%d")
street_crime$Month <- format(street_crime$Month, '%m/%Y')

names(street_crime)[2] <- 'date'

#inner join on postcode
Crime_Housing <- merge(housing_clean, street_crime, by = c("postcode", "date"), all.x = TRUE, all.y = TRUE)
#create a csv file with our df
write.csv(Crime_Housing, 'C:/Users/tochi/Desktop/Crime_Housing.csv', row.names = FALSE)
str(Crime_Housing)
plot(Crime_Housing$price_paid, ylab = "price_paid")
plot(Crime_Housing$Crime.type, ylab -= "Crimetype")

```
```
```

(C)

```
#Removing Outliers and Duplicate Rows
```{r}

str(Crime_Housing)
summary(Crime_Housing)
boxplot(Crime_Housing$price_paid)
Crime_Housing_boxplot <-
boxplot(Crime_Housing$price_paid)


Crime_Housing_boxplot$out

min(Crime_Housing_boxplot$out)

Crime_Housing[Crime_Housing$price_paid >=
min(Crime_Housing_boxplot$out), ]

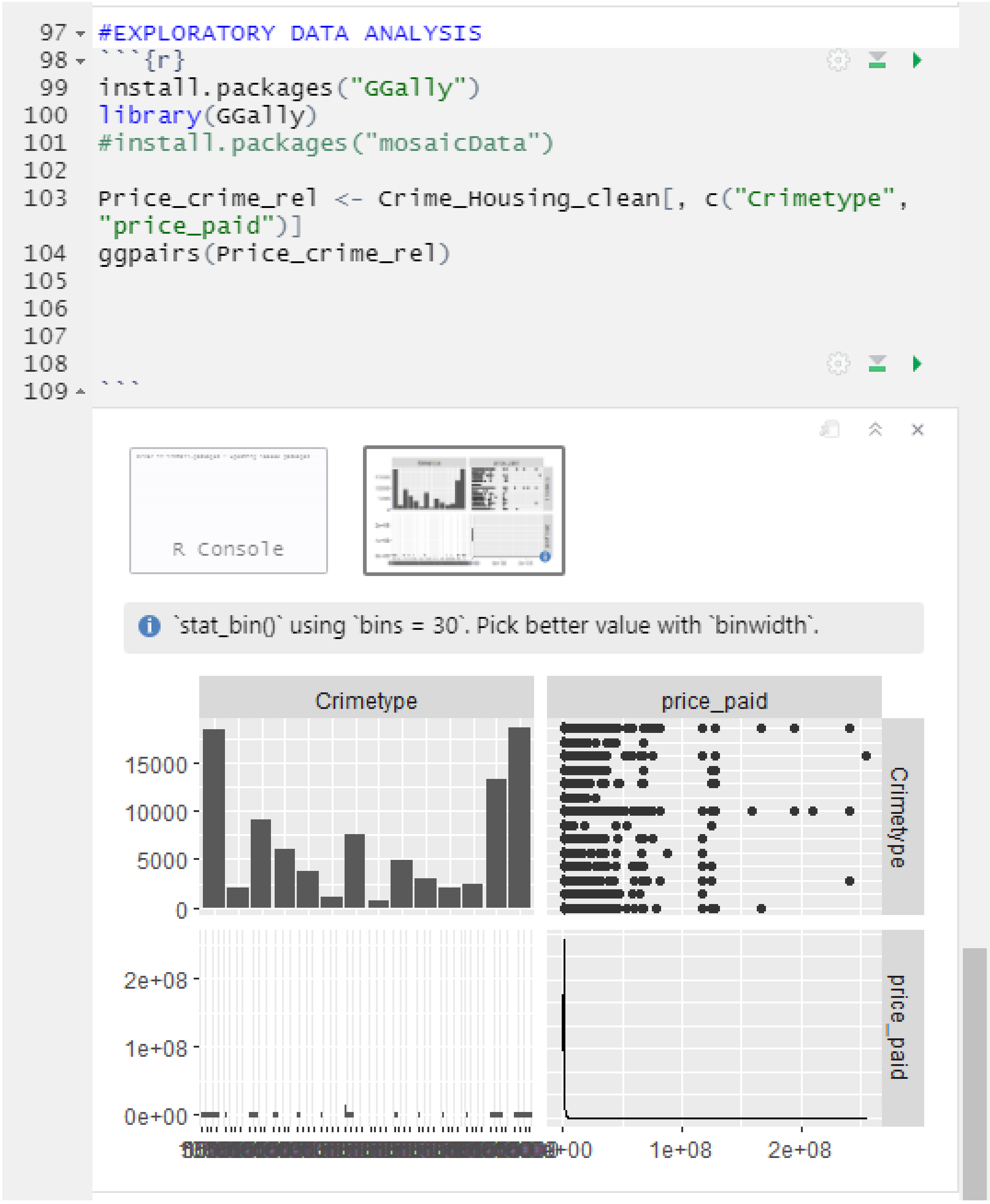
#to remove duplicate rows
Crime_Housing_clean <- unique(Crime_Housing)
str(Crime_Housing_clean)

##to visualise graphically
Crime.type.freq <- xtabs(~ Crimetype, data =
Crime_Housing)
Crime.type.prop <- prop.table(Crime.type.freq)
Crime.type.prop

Price_paid.hist <- ggplot(data = Crime_Housing, aes(x =
price_paid)) + geom_histogram(binwidth =10, color =
"black", fill = "lightgrey") + xlab("Price_Paid") +
ylab("Frequency")


```
```

(D)



#EXPLORATORY DATA ANALYSIS
```{r}
install.packages("GGally")
library(GGally)
#install.packages("mosaicData")
Price_crime_rel <- Crime_Housing_clean[, c("Crimetype", "price_paid")]
ggpairs(Price_crime_rel)
```
R Console
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
Crimetype
price_paid
15000
10000
5000
0
2e+08
1e+08
0e+00
1e+08
2e+08
Crimetype
price_paid

(E)

```
In [1]: import pandas as pd
        import datetime#, timedelta

        import numpy as np
        import pandas as pd
        from matplotlib import pyplot as plt
        from sklearn.datasets import make_blobs
        from sklearn.cluster import KMeans
```

```
In [2]: housing_Crime_Data = pd.read_csv(r"C:\Users\tochi\Desktop\Crime_Housing.csv")
        housing_Crime_Data.describe()
```

Out[2]:

| | price_paid |
|---|---|
| count | 1.344350e+05 |
| mean | 9.119420e+05 |
| std | 5.360636e+06 |
| min | 1.000000e+02 |
| 25% | 3.170000e+05 |
| 50% | 4.350000e+05 |
| 75% | 6.450000e+05 |
| max | 2.550000e+08 |

```
In [3]: housing_Crime_Data.dtypes
```

```
Out[3]: Month            object
        Crime.type       object
        postcode         object
        price_paid        int64
        property_type    object
        new_build        object
        town             object
        district         object
        dtype: object
```

```
In [4]: housing_Crime_Data.columns
```

(F)

```
In [4]: housing_Crime_Data.columns

Out[4]: Index(['Month', 'Crime.type', 'postcode', 'price_paid', 'property_type',
               'new_build', 'town', 'district'],
              dtype='object')

In [5]: housing_Crime_Data['Crime.type'] = pd.Categorical(housing_Crime_Data['Crime.type'] )
        housing_Crime_Data['postcode'] = pd.Categorical(housing_Crime_Data['postcode'] )
        housing_Crime_Data['Month'] = pd.Categorical(housing_Crime_Data['Month'])
        housing_Crime_Data['property_type'] = pd.Categorical(housing_Crime_Data['property_type'] )
        housing_Crime_Data['new_build'] = pd.Categorical(housing_Crime_Data['new_build'] )
        housing_Crime_Data['district'] = pd.Categorical(housing_Crime_Data['district'] )
        housing_Crime_Data['town'] =  pd.Categorical(housing_Crime_Data.loc[:,'town'])

In [6]: housing_Crime_Data.dtypes

Out[6]: Month            category
        Crime.type       category
        postcode         category
        price_paid          int64
        property_type    category
        new_build        category
        town             category
        district         category
        dtype: object

In [8]: Housing_CrimeData_Clean_Cat = housing_Crime_Data.select_dtypes(include=['category'])
        Housing_CrimeData_Clean_num = housing_Crime_Data.select_dtypes(exclude=['category'])
        X_encoded = pd.get_dummies(Housing_CrimeData_Clean_Cat)

In [9]: frames = [X_encoded, Housing_CrimeData_Clean_num]
        combo_enc = pd.concat(frames, axis = 1)

In [10]: Housing_CrimeData_Clean_Cat_freq = Housing_CrimeData_Clean_Cat.copy()
         for c in Housing_CrimeData_Clean_Cat_freq.columns.to_list():
             Housing_CrimeData_Clean_Cat_freq[c] = Housing_CrimeData_Clean_Cat_freq.groupby(c).transform('count')/len(Housing_CrimeData_Cl

In [11]: frames_freq = [Housing_CrimeData_Clean_Cat_freq, Housing_CrimeData_Clean_num]
         combo_enc_freq = pd.concat(frames_freq, axis = 1)
```

(G)

In [8]:
```
Housing_CrimeData_Clean_Cat = housing_Crime_Data.select_dtypes(include=['category'])
Housing_CrimeData_Clean_num = housing_Crime_Data.select_dtypes(exclude=['category'])
X_encoded = pd.get_dummies(Housing_CrimeData_Clean_Cat)
```

In [9]:
```
frames = [X_encoded, Housing_CrimeData_Clean_num]
combo_enc = pd.concat(frames, axis = 1)
```

In [10]:
```
Housing_CrimeData_Clean_Cat_freq = Housing_CrimeData_Clean_Cat.copy()
for c in Housing_CrimeData_Clean_Cat_freq.columns.to_list():
    Housing_CrimeData_Clean_Cat_freq[c] = Housing_CrimeData_Clean_Cat_freq.groupby(c).transform('count')/len(Housing_CrimeData_Cl
```

In [11]:
```
frames_freq = [Housing_CrimeData_Clean_Cat_freq, Housing_CrimeData_Clean_num]
combo_enc_freq = pd.concat(frames_freq, axis = 1)
combo_enc_freq.shape
```

Out[11]:
```
(134435, 8)
```

In [12]:
```
combo_enc_freq.tail()
combo_enc_freq.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 134435 entries, 0 to 134434
Data columns (total 8 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Month          134435 non-null  float64
 1   Crime.type     134435 non-null  float64
 2   postcode       134435 non-null  float64
 3   property_type  134435 non-null  float64
 4   new_build      134435 non-null  float64
 5   town           134435 non-null  float64
 6   district       134435 non-null  float64
 7   price_paid     134435 non-null  int64
dtypes: float64(7), int64(1)
memory usage: 8.2 MB
```

In [13]:
```
from sklearn.model_selection import train_test_split
Xtrain, Xtest, ytrain, ytest = train_test_split(combo_enc_freq.filter(items =[ 'Month', 'Crime.type', 'postcode', 'property_type'
       'new_build', 'town', 'district']), combo_enc_freq['price_paid'], random_state=0)
```

(H)

```
In [14]: from sklearn.neural_network import MLPRegressor

In [15]: from sklearn.datasets import make_regression

In [43]: regr = MLPRegressor(random_state=1, max_iter=15000).fit(Xtrain, ytrain)

C:\Users\tochi\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:582: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (15000) reached and the optimization hasn't converged yet.
  warnings.warn(

In [44]: regr.predict(Xtest)

Out[44]: array([ 672271.06936196, 3195774.85376639,  489987.51534799, ...,
        3268704.75471761,  279349.94606583,  555175.15235468])

In [45]: regr.score(Xtest, ytest)

Out[45]: 0.027650184831768065

In [51]: #R_Square and Adjusted R Square
         import statsmodels.api as sm
         X_addC = sm.add_constant(Xtest)
         result = sm.OLS(ytest, X_addC).fit()
         print(result.rsquared, result.rsquared_adj)

0.019493542853790125 0.019289276754566154

In [69]: from sklearn.metrics import mean_squared_error
         import math
         pred_y = regr.predict(Xtest)
         print(mean_squared_error(ytest, pred_y))
         print(math.sqrt(mean_squared_error(ytest, pred_y)))

30351495581054.434
5509219.14440281

In [67]: from sklearn.metrics import mean_absolute_error
         print(mean_absolute_error(ytest, pred_y))

807765.0093318874
```

(I-1)

In [1]:
```
!pip install spark
```

```
Collecting spark
  Downloading spark-0.2.1.tar.gz (41 kB)
     |████████████████████████████████| 41 kB 204 kB/s
Building wheels for collected packages: spark
  Building wheel for spark (setup.py) ... done
  Created wheel for spark: filename=spark-0.2.1-py3-none-any.whl size=58738 sha256=d2ef5ba9f0c493d0136c93f2823ea36a0b39720c8858
7af5f94bfa2849ac1357
  Stored in directory: /Users/sne2909/Library/Caches/pip/wheels/c5/19/ff/9b16f354528bc9698ec3286be7947ebbf1f8391325553961d4
Successfully built spark
Installing collected packages: spark
Successfully installed spark-0.2.1
```

In [2]:
```
!pip install imblearn
```

```
Collecting imblearn
  Downloading imblearn-0.0-py2.py3-none-any.whl (1.9 kB)
Collecting imbalanced-learn
  Downloading imbalanced_learn-0.8.0-py3-none-any.whl (206 kB)
     |████████████████████████████████| 206 kB 4.8 MB/s
Requirement already satisfied: joblib>=0.11 in /Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages (f
rom imbalanced-learn->imblearn) (0.15.1)
Requirement already satisfied: scipy>=0.19.1 in /Users/sne2909/Library/Python/3.8/lib/python/site-packages (from imbalanced-lea
rn->imblearn) (1.4.1)
Requirement already satisfied: numpy>=1.13.3 in /Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from imbalanced-learn->imblearn) (1.19.5)
Collecting scikit-learn>=0.24
  Downloading scikit_learn-0.24.1-cp38-cp38-macosx_10_13_x86_64.whl (7.2 MB)
     |████████████████████████████████| 7.2 MB 15.4 MB/s
Requirement already satisfied: threadpoolctl>=2.0.0 in /Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-pac
kages (from scikit-learn>=0.24->imbalanced-learn->imblearn) (2.1.0)
Installing collected packages: scikit-learn, imbalanced-learn, imblearn
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 0.23.1
    Uninstalling scikit-learn-0.23.1:
      Successfully uninstalled scikit-learn-0.23.1
Successfully installed imbalanced-learn-0.8.0 imblearn-0.0 scikit-learn-0.24.1
```

In [3]:
```
!pip install pyspark
```

```
Collecting pyspark
```

(I-2)

```
    Uninstalling scikit-learn-0.23.1:
      Successfully uninstalled scikit-learn-0.23.1
Successfully installed imbalanced-learn-0.8.0 imblearn-0.0 scikit-learn-0.24.1
```

In [3]:
```
!pip install pyspark
```

```
Collecting pyspark
  Downloading pyspark-3.1.1.tar.gz (212.3 MB)
     |████████████████████████████████| 212.3 MB 57 kB/s
Collecting py4j==0.10.9
  Downloading py4j-0.10.9-py2.py3-none-any.whl (198 kB)
     |████████████████████████████████| 198 kB 71.7 MB/s
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.1.1-py2.py3-none-any.whl size=212767604 sha256=3e650e1c9e5e6e78d7daaa54fe39db83
f4d79199faeb7529e3ec45b21f5325fa
  Stored in directory: /Users/sne2909/Library/Caches/pip/wheels/b3/0e/81/264aeed961e43b9f6ba9ec81c8c540d2d7dccc52c6b51cbf22
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9 pyspark-3.1.1
Note: you may need to restart the kernel to use updated packages.
```

In [4]:
```
from pyspark.sql import SQLContext
from pyspark.sql import DataFrameNaFunctions
from pyspark.ml import Pipeline
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import Binarizer
from pyspark.ml.feature import OneHotEncoder, VectorAssembler, StringIndexer, VectorIndexer
from pyspark.ml.classification import RandomForestClassifier
from pyspark.sql.functions import avg
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from imblearn.over_sampling import SMOTE
from imblearn.combine import SMOTEENN
from pyspark.mllib.evaluation import MulticlassMetrics
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from sklearn.model_selection import train_test_split
from collections import Counter
```

In [5]:
```
from pyspark.context import SparkContext
```

(I-3)

```
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9 pyspark-3.1.1
Note: you may need to restart the kernel to use updated packages.
```

In [4]:
```
from pyspark.sql import SQLContext
from pyspark.sql import DataFrameNaFunctions
from pyspark.ml import Pipeline
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import Binarizer
from pyspark.ml.feature import OneHotEncoder, VectorAssembler, StringIndexer, VectorIndexer
from pyspark.ml.classification import RandomForestClassifier
from pyspark.sql.functions import avg
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from imblearn.over_sampling import SMOTE
from imblearn.combine import SMOTEENN
from pyspark.mllib.evaluation import MulticlassMetrics
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from sklearn.model_selection import train_test_split
from collections import Counter
```

In [5]:
```
from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
sc = SparkContext()
spark = SparkSession(sc)
```

In [8]:
```
crime_housing = sqlContext.read.csv("./Crime_Housing.csv", header=True, inferSchema= True)
crime_housing.columns
```

Out[8]:
```
['Month',
 'Crimetype',
 'postcode',
 'price_paid',
 'property_type',
 'new_build',
 'town',
 'district']
```

In [10]:
```
crime_housing.printSchema()
```

(J)

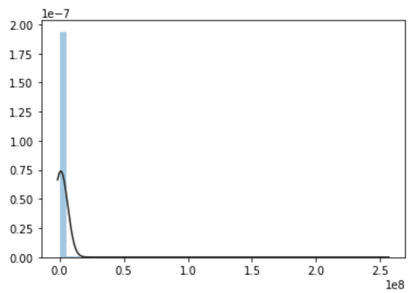
```
'town',
'district']
```

In [10]:
```
crime_housing.printSchema()
```

```
root
 |-- Month: string (nullable = true)
 |-- Crimetype: string (nullable = true)
 |-- postcode: string (nullable = true)
 |-- price_paid: integer (nullable = true)
 |-- property_type: string (nullable = true)
 |-- new_build: string (nullable = true)
 |-- town: string (nullable = true)
 |-- district: string (nullable = true)
```

In [25]:
```
import seaborn as sns
#tO VISUALISE 1D view of my dataset
from scipy.stats import norm
sns.distplot(dataframe_2[['price_paid']], fit=norm, kde=False)
```

Out[25]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe2015eeeb0>
```




In [28]:
```
dataframe_1.groupby('Crimetype').count().sort_values('Month', ascending=False)
```

Out[28]:

| | Month | postcode | property_type | new_build | town | district |
|---|---|---|---|---|---|---|
| Crimetype | | | | | | |

(K)

```
In [28]: dataframe_1.groupby('Crimetype').count().sort_values('Month', ascending=False)
```

Out[28]:

| Crimetype | Month | postcode | property_type | new_build | town | district |
|---|---|---|---|---|---|---|
| Anti-social behaviour | 28256 | 28256 | 28256 | 28256 | 28256 | 28256 |
| Violence and sexual offences | 23130 | 23130 | 23130 | 23130 | 23130 | 23130 |
| Vehicle crime | 12724 | 12724 | 12724 | 12724 | 12724 | 12724 |
| Burglary | 8759 | 8759 | 8759 | 8759 | 8759 | 8759 |
| Other theft | 8561 | 8561 | 8561 | 8561 | 8561 | 8561 |
| Criminal damage and arson | 5754 | 5754 | 5754 | 5754 | 5754 | 5754 |
| Public order | 4719 | 4719 | 4719 | 4719 | 4719 | 4719 |
| Drugs | 3706 | 3706 | 3706 | 3706 | 3706 | 3706 |
| Shoplifting | 2980 | 2980 | 2980 | 2980 | 2980 | 2980 |
| Robbery | 2977 | 2977 | 2977 | 2977 | 2977 | 2977 |
| Theft from the person | 2735 | 2735 | 2735 | 2735 | 2735 | 2735 |
| Bicycle theft | 1826 | 1826 | 1826 | 1826 | 1826 | 1826 |
| Other crime | 860 | 860 | 860 | 860 | 860 | 860 |
| Possession of weapons | 561 | 561 | 561 | 561 | 561 | 561 |

```
In [43]: # replace predictor variable with categorical labels instead... using pandas quantile cut we can do that
crime_housing_clean = crime_housing.toPandas().copy()

pd.qcut(crime_housing_clean['price_paid'], q=[0, .1, 0.25, .5, .75, .9, 1]).unique()

Out[43]: [(220000.0, 317000.0], (99.999, 220000.0], (317000.0, 435000.0], (435000.0, 645000.0], (645000.0, 1100000.0], (1100000.0, 25500
0000.0]]
Categories (6, interval[float64]): [(99.999, 220000.0] < (220000.0, 317000.0] < (317000.0, 435000.0] < (435000.0, 645000.0] <
(645000.0, 1100000.0] < (1100000.0, 255000000.0]]
```

```
In [52]: crime_housing_clean['price_paid_categories'] = pd.qcut(crime_housing_clean['price_paid'], q=[0, .1, 0.25, .5, .75, .9, 1], labels
# extract month from Month (date) column
crime_housing_clean['month_int'] = crime_housing_clean['Month'].apply(lambda x: pd.to_datetime(x).month)
```

(L-1)

```
In [53]: crime_housing_clean.head()
```

Out[53]:

| | Month | Crimetype | postcode | price_paid | property_type | new_build | town | district | price_paid_categories | month_int |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 01/11/2019 | Anti-social behaviour | RM6 5PJ | 250000 | T | N | ROMFORD | BARKING AND DAGENHAM | 220k−317k | 1 |
| **1** | 01/11/2019 | Anti-social behaviour | RM6 5JJ | 215000 | F | N | ROMFORD | BARKING AND DAGENHAM | < $220k | 1 |
| **2** | 01/11/2019 | Burglary | RM6 5JP | 290000 | T | N | ROMFORD | BARKING AND DAGENHAM | 220k−317k | 1 |
| **3** | 01/11/2019 | Criminal damage and arson | RM6 5PJ | 250000 | T | N | ROMFORD | BARKING AND DAGENHAM | 220k−317k | 1 |
| **4** | 01/11/2019 | Criminal damage and arson | RM6 5PJ | 250000 | T | N | ROMFORD | BARKING AND DAGENHAM | 220k−317k | 1 |

```
In [107]: crime_housing_clean.town.nunique()
```

Out[107]: 72

```
In [77]: crime_housing_clean.columns = crime_housing_clean.columns.str.lower()
cols = crime_housing_clean.columns
```

```
In [108]: sdf_crime_housing = spark.createDataFrame(crime_housing_clean)
# Encode categorical variables before test/train split
categoricalCols = ['crimetype','property_type','town','new_build','district']
stages = []

featureIndexers = [StringIndexer(inputCol=catCol, outputCol=catCol+'Index') for catCol in categoricalCols]
stages += featureIndexers

labelIndexer = [StringIndexer(inputCol=labelCol, outputCol='label') for labelCol in ['price_paid_categories']]
stages += labelIndexer

numericCols = ['month_int']

assemblerInputs = [col + 'Index' for col in categoricalCols] + numericCols
assembler = VectorAssembler(inputCols=assemblerInputs, outputCol="features")
stages += [assembler]

pipeline = Pipeline(stages=stages)
sdf = pipeline.fit(sdf_crime_housing).transform(sdf_crime_housing)

selectedCols = ['label','features'] + cols.tolist()
```

(L-2)

```
pipeline = Pipeline(stages=stages)
sdf = pipeline.fit(sdf_crime_housing).transform(sdf_crime_housing)

selectedCols = ['label','features'] + cols.tolist()
sdf = sdf.select(selectedCols)
sdf.printSchema()
```

```
root
 |-- label: double (nullable = false)
 |-- features: vector (nullable = true)
 |-- month: string (nullable = true)
 |-- crimetype: string (nullable = true)
 |-- postcode: string (nullable = true)
 |-- price_paid: long (nullable = true)
 |-- property_type: string (nullable = true)
 |-- new_build: string (nullable = true)
 |-- town: string (nullable = true)
 |-- district: string (nullable = true)
 |-- price_paid_categories: string (nullable = true)
 |-- month_int: long (nullable = true)
```

In [109]:
```
sdf.show(5)
```

```
+-----+--------------------+----------+--------------------+--------+----------+-------------+---------+-------+---------------
-----+--------------------+---------+
|label|            features|     month|           crimetype|postcode|price_paid|property_type|new_build|   town|            dis
trict|price_paid_categories|month_int|
+-----+--------------------+----------+--------------------+--------+----------+-------------+---------+-------+---------------
-----+--------------------+---------+
|  3.0|[0.0,1.0,1.0,0.0,...|01/11/2019|Anti-social behav...| RM6 5PJ|    250000|            T|        N|ROMFORD|BARKING AND DAG
ENHAM|         $220k - $317k|        1|
|  4.0|[0.0,0.0,1.0,0.0,...|01/11/2019|Anti-social behav...| RM6 5JJ|    215000|            F|        N|ROMFORD|BARKING AND DAG
ENHAM|               < $220k|        1|
|  3.0|[3.0,1.0,1.0,0.0,...|01/11/2019|            Burglary| RM6 5JP|    290000|            T|        N|ROMFORD|BARKING AND DAG
ENHAM|         $220k - $317k|        1|
|  3.0|[5.0,1.0,1.0,0.0,...|01/11/2019|Criminal damage a...| RM6 5PJ|    250000|            T|        N|ROMFORD|BARKING AND DAG
ENHAM|         $220k - $317k|        1|
|  3.0|[5.0,1.0,1.0,0.0,...|01/11/2019|Criminal damage a...| RM6 5PJ|    250000|            T|        N|ROMFORD|BARKING AND DAG
ENHAM|         $220k - $317k|        1|
+-----+--------------------+----------+--------------------+--------+----------+-------------+---------+-------+---------------
-----+--------------------+---------+
only showing top 5 rows
```

(M)

```
 |-- new_build: string (nullable = true)
 |-- town: string (nullable = true)
 |-- district: string (nullable = true)
 |-- price_paid_categories: string (nullable = true)
 |-- month_int: long (nullable = true)
```

In [109]:

```
sdf.show(5)
```

```
+-----+--------------------+----------+--------------------+--------+----------+-------------+---------+-------+---------------
-----+--------------------+---------+
|label|            features|     month|           crimetype|postcode|price_paid|property_type|new_build|   town|            dis
trict|price_paid_categories|month_int|
+-----+--------------------+----------+--------------------+--------+----------+-------------+---------+-------+---------------
-----+--------------------+---------+
|  3.0|[0.0,1.0,1.0,0.0,...|01/11/2019|Anti-social behav...| RM6 5PJ|    250000|            T|        N|ROMFORD|BARKING AND DAG
ENHAM|         $220k - $317k|        1|
|  4.0|[0.0,0.0,1.0,0.0,...|01/11/2019|Anti-social behav...| RM6 5JJ|    215000|            F|        N|ROMFORD|BARKING AND DAG
ENHAM|               < $220k|        1|
|  3.0|[3.0,1.0,1.0,0.0,...|01/11/2019|            Burglary| RM6 5JP|    290000|            T|        N|ROMFORD|BARKING AND DAG
ENHAM|         $220k - $317k|        1|
|  3.0|[5.0,1.0,1.0,0.0,...|01/11/2019|Criminal damage a...| RM6 5PJ|    250000|            T|        N|ROMFORD|BARKING AND DAG
ENHAM|         $220k - $317k|        1|
|  3.0|[5.0,1.0,1.0,0.0,...|01/11/2019|Criminal damage a...| RM6 5PJ|    250000|            T|        N|ROMFORD|BARKING AND DAG
ENHAM|         $220k - $317k|        1|
+-----+--------------------+----------+--------------------+--------+----------+-------------+---------+-------+---------------
-----+--------------------+---------+
only showing top 5 rows
```

In [164]:

```
# split data in train & test

train, test = sdf.randomSplit([0.8, 0.2], seed=25)
print("Training Dataset Count: " + str(train.count()))
print("Test Dataset Count: " + str(test.count()))
```

```
Training Dataset Count: 107432
Test Dataset Count: 27003
```

(N)

In [165]:
```
dt = DecisionTreeClassifier(featuresCol = 'features', labelCol = 'label', maxDepth = 3, maxBins=100)
dtModel = dt.fit(train)
predictions = dtModel.transform(test)
predictions.select('crimetype','postcode','property_type','new_build','town','district', 'month_int','label','prediction').show(1
```

```
+--------------------+--------+-------------+---------+------+--------+---------+-----+----------+
|           crimetype|postcode|property_type|new_build|  town|district|month_int|label|prediction|
+--------------------+--------+-------------+---------+------+--------+---------+-----+----------+
|Violence and sexu...|SW16 2SN|            F|        N|LONDON| LAMBETH|        1|  0.0|       0.0|
|Violence and sexu...| SW9 9UQ|            F|        N|LONDON| LAMBETH|        1|  0.0|       0.0|
|       Vehicle crime| SW8 2EU|            F|        N|LONDON| LAMBETH|        1|  0.0|       0.0|
|Criminal damage a...|SW16 6JD|            F|        N|LONDON| LAMBETH|        1|  0.0|       0.0|
|             Robbery|SW16 5LJ|            F|        N|LONDON| LAMBETH|        1|  0.0|       0.0|
|       Bicycle theft|SE24 0NS|            F|        N|LONDON| LAMBETH|        1|  0.0|       0.0|
|Anti-social behav...| SW9 8QH|            T|        N|LONDON| LAMBETH|        1|  0.0|       1.0|
|Anti-social behav...| SW9 8QH|            T|        N|LONDON| LAMBETH|        1|  0.0|       1.0|
|Anti-social behav...| SW9 8QH|            T|        N|LONDON| LAMBETH|        1|  0.0|       1.0|
|Anti-social behav...|SW16 3LJ|            F|        Y|LONDON| LAMBETH|        1|  0.0|       0.0|
+--------------------+--------+-------------+---------+------+--------+---------+-----+----------+
only showing top 10 rows
```

In [166]:
```
# evaluate decision tree
evaluator = MulticlassClassificationEvaluator(predictionCol='prediction', labelCol='label', metricName='accuracy')
accuracy = evaluator.evaluate(predictions)

print("Test Error = %g " % (1.0 - accuracy))
print("Accuracy = %g " % accuracy)
```

```
Test Error = 0.65415
Accuracy = 0.34585
```

(O)

In [167]:
```
from pyspark.ml.classification import RandomForestClassifier
rf = RandomForestClassifier(featuresCol = 'features', labelCol = 'label', maxBins=100)
rfModel = rf.fit(train)
predictions = rfModel.transform(test)
predictions.select('crimetype','postcode','property_type','new_build','town','district', 'month_int','label','prediction').show(1
```

```
+--------------------+--------+-------------+---------+------+--------+---------+-----+----------+
|           crimetype|postcode|property_type|new_build|  town|district|month_int|label|prediction|
+--------------------+--------+-------------+---------+------+--------+---------+-----+----------+
|Violence and sexu...|SW16 2SN|            F|        N|LONDON| LAMBETH|        1|  0.0|       1.0|
|Violence and sexu...| SW9 9UQ|            F|        N|LONDON| LAMBETH|        1|  0.0|       1.0|
|       Vehicle crime| SW8 2EU|            F|        N|LONDON| LAMBETH|        1|  0.0|       1.0|
|Criminal damage a...|SW16 6JD|            F|        N|LONDON| LAMBETH|        1|  0.0|       1.0|
|             Robbery|SW16 5LJ|            F|        N|LONDON| LAMBETH|        1|  0.0|       1.0|
|       Bicycle theft|SE24 0NS|            F|        N|LONDON| LAMBETH|        1|  0.0|       1.0|
|Anti-social behav...| SW9 8QH|            T|        N|LONDON| LAMBETH|        1|  0.0|       1.0|
|Anti-social behav...| SW9 8QH|            T|        N|LONDON| LAMBETH|        1|  0.0|       1.0|
|Anti-social behav...| SW9 8QH|            T|        N|LONDON| LAMBETH|        1|  0.0|       1.0|
|Anti-social behav...|SW16 3LJ|            F|        Y|LONDON| LAMBETH|        1|  0.0|       1.0|
+--------------------+--------+-------------+---------+------+--------+---------+-----+----------+
only showing top 10 rows
```

In [168]:
```
evaluator = MulticlassClassificationEvaluator(predictionCol='prediction', labelCol='label', metricName='accuracy')
accuracy = evaluator.evaluate(predictions)

print("Test Error = %g " % (1.0 - accuracy))
print("Accuracy = %g " % accuracy)
```

```
Test Error = 0.598193
Accuracy = 0.401807
```

(P)

```
In [169]: lr = LogisticRegression(featuresCol = 'features', labelCol = 'label', maxIter=10)
          lrModel = lr.fit(train)

In [170]: predictions = lrModel.transform(test)
          predictions.select('crimetype','postcode','property_type','new_build','town','district', 'month_int','label','prediction').show(1
```

```
+--------------------+--------+-------------+---------+------+--------+---------+-----+----------+
|           crimetype|postcode|property_type|new_build|  town|district|month_int|label|prediction|
+--------------------+--------+-------------+---------+------+--------+---------+-----+----------+
|Violence and sexu...|SW16 2SN|            F|        N|LONDON| LAMBETH|        1|  0.0|       0.0|
|Violence and sexu...| SW9 9UQ|            F|        N|LONDON| LAMBETH|        1|  0.0|       0.0|
|       Vehicle crime| SW8 2EU|            F|        N|LONDON| LAMBETH|        1|  0.0|       0.0|
|Criminal damage a...|SW16 6JD|            F|        N|LONDON| LAMBETH|        1|  0.0|       0.0|
|             Robbery|SW16 5LJ|            F|        N|LONDON| LAMBETH|        1|  0.0|       0.0|
|       Bicycle theft|SE24 0NS|            F|        N|LONDON| LAMBETH|        1|  0.0|       0.0|
|Anti-social behav...| SW9 8QH|            T|        N|LONDON| LAMBETH|        1|  0.0|       1.0|
|Anti-social behav...| SW9 8QH|            T|        N|LONDON| LAMBETH|        1|  0.0|       1.0|
|Anti-social behav...| SW9 8QH|            T|        N|LONDON| LAMBETH|        1|  0.0|       1.0|
|Anti-social behav...|SW16 3LJ|            F|        Y|LONDON| LAMBETH|        1|  0.0|       1.0|
+--------------------+--------+-------------+---------+------+--------+---------+-----+----------+
only showing top 10 rows
```

```
In [171]: evaluator = MulticlassClassificationEvaluator(predictionCol='prediction', labelCol='label', metricName='accuracy')
          accuracy = evaluator.evaluate(predictions)

          print("Test Error = %g " % (1.0 - accuracy))
          print("Accuracy = %g " % accuracy)
```

```
Test Error = 0.714254
Accuracy = 0.285746
```

```
In [ ]:
```