Analysis and predictions on land sales in Ireland

Abstract.

Determining land prices is very challenging task now a days for investors and customers both as land prices are dynamically changes.

The focus of this project is to apply data science and machine learning techniques that can help investors by predicting what prices they can maintain.

Introduction

Project scenario

In Ireland an investor wants to know about land prices. But he get confused when he comes to know different prices rates.

To overcome above mentioned problems this this investor decides to make a robust prediction system for its company.

Data Description:

The dataset which I am going to use here is dataset of land sales in Ireland. It is openly available on Kaggle.

Dataset source: https://data.gov.ie/dataset/ara02-agricultural-land-sales?package_type=dataset

The important features are:

- **Year**
- **Type of Land Use**
- **Region**
- **UNIT**
- **VALUE**

Reason for Analysis:

The reasons of doing this Analysis are following:

- To extract information, facts and figures from raw data
- To understand pattern of data.
- To know about trends.
- To make visualizations to represents data in the form of graphs and charts. As a pictu re is worth than 1000 words.
- To show what relationships different feature are having among them.
- To make an accurate Machine Learning Model which can be used to land sales on th e basis of input data?

Technical requirements:

There are two main tools which are used to do analysis which are following:

- **Python:**

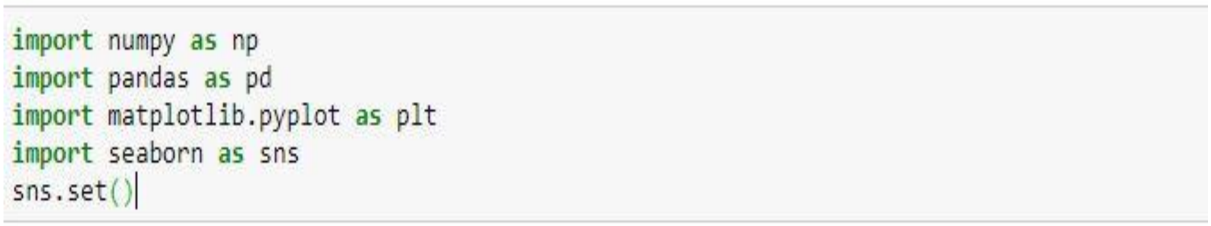
  Python is programming language which is used here .I have to perform this data analysis usi ng python

- **Jupiter notebook:**

  Jupiter notebook is tool on have to I have to write python script

Importing libraries and dataset

The first step is to import all required libraries.so here I am importing four Major Libraries which are here:



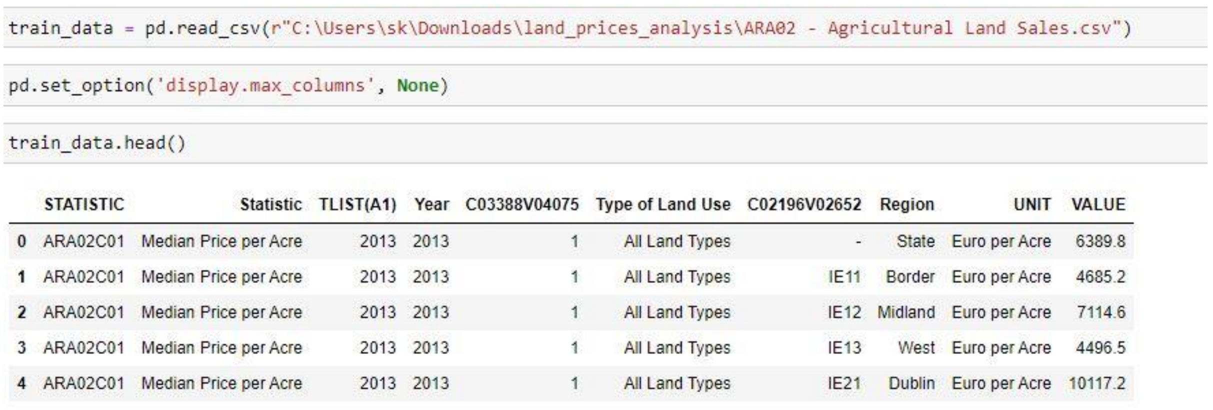
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

1)**pandas**:
pandas is used for data crunching.
2)**numpy** :
It is used for scientific computing it provides some mathematics and statistics like functions.
3)**Matplotlib:**
It is used for data Visualization For example to make Charts and graphs.
4)**Seaborn**:
It is also used for data Visualization For example to make Charts and graphs.

so these are four major python libraries which I have imported to use.

Import csv file and create a data frame

Now here I am loading data. As data is in excel file so I am importing Excel File with .csv file format.



```
train_data = pd.read_csv(r"C:\Users\sk\Downloads\land_prices_analysis\ARA02 - Agricultural Land Sales.csv")
```

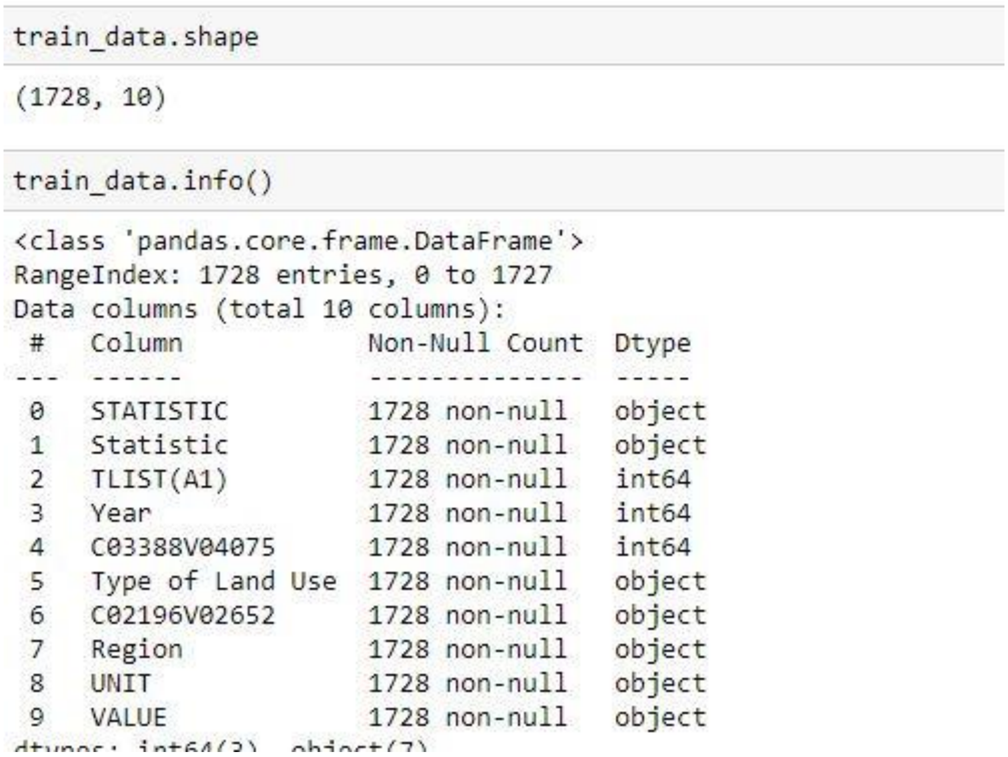
```
pd.set_option('display.max_columns', None)
```

```
train_data.head()
```

	STATISTIC	Statistic	TLIST(A1)	Year	C03388V04075	Type of Land Use	C02196V02652	Region	UNIT	VALUE
0	ARA02C01	Median Price per Acre	2013	2013	1	All Land Types	-	State	Euro per Acre	6389.8
1	ARA02C01	Median Price per Acre	2013	2013	1	All Land Types	IE11	Border	Euro per Acre	4685.2
2	ARA02C01	Median Price per Acre	2013	2013	1	All Land Types	IE12	Midland	Euro per Acre	7114.6
3	ARA02C01	Median Price per Acre	2013	2013	1	All Land Types	IE13	West	Euro per Acre	4496.5
4	ARA02C01	Median Price per Acre	2013	2013	1	All Land Types	IE21	Dublin	Euro per Acre	10117.2

As here excel data is loaded into a data frame 'train data'.
Data Frame is also like a table.
At the end first 5 records are displayed.

Size/shape of data and general information about data



```
train_data.shape
```

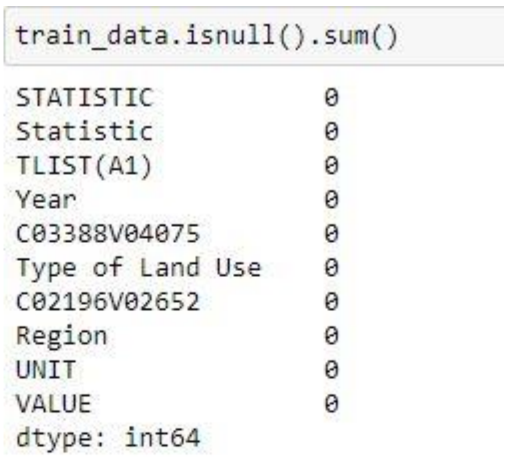
```
(1728, 10)
```

```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1728 entries, 0 to 1727
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   STATISTIC         1728 non-null   object
 1   Statistic         1728 non-null   object
 2   TLIST(A1)         1728 non-null   int64
 3   Year              1728 non-null   int64
 4   C03388V04075      1728 non-null   int64
 5   Type of Land Use  1728 non-null   object
 6   C02196V02652      1728 non-null   object
 7   Region            1728 non-null   object
 8   UNIT              1728 non-null   object
 9   VALUE             1728 non-null   object
dtypes: int64(3), object(7)
```

There are 10 features or columns and 1728 rows or records in dataset.

Checking for any missing values



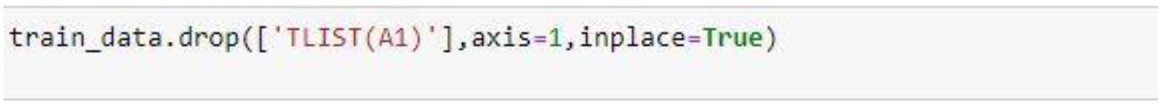
```
train_data.isnull().sum()
```

```
STATISTIC           0
Statistic           0
TLIST(A1)           0
Year                0
C03388V04075        0
Type of Land Use    0
C02196V02652        0
Region              0
UNIT                0
VALUE               0
dtype: int64
```

As shown above there are no null or missing values in data.

Feature Engineering

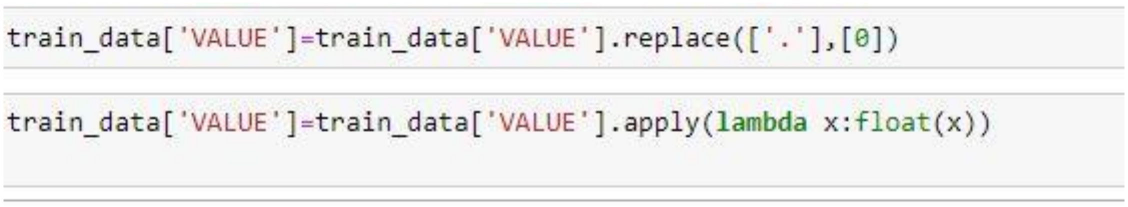
Removing 'LIST(A1)' feature



```
train_data.drop(['TLIST(A1)'],axis=1,inplace=True)
```

As ''LIST(A1)'' is similar to year feature so it I removed.

Converting value feature into float type



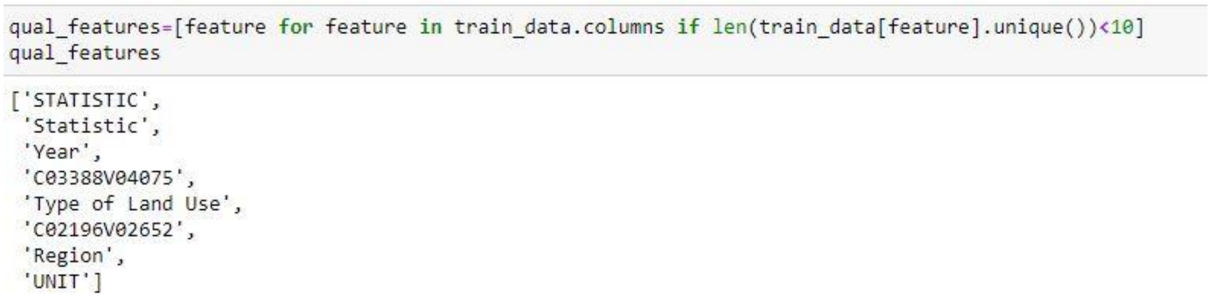
```
train_data['VALUE']=train_data['VALUE'].replace(['.'],[0])
```

```
train_data['VALUE']=train_data['VALUE'].apply(lambda x:float(x))
```

As value feature has data type of object so it is converted to integer but problem is that there is dot character is exists so at first dot character is converted to zero than type of Value feature is changed by using lambda function

Exploratory data analysis

Extracting qualitative features



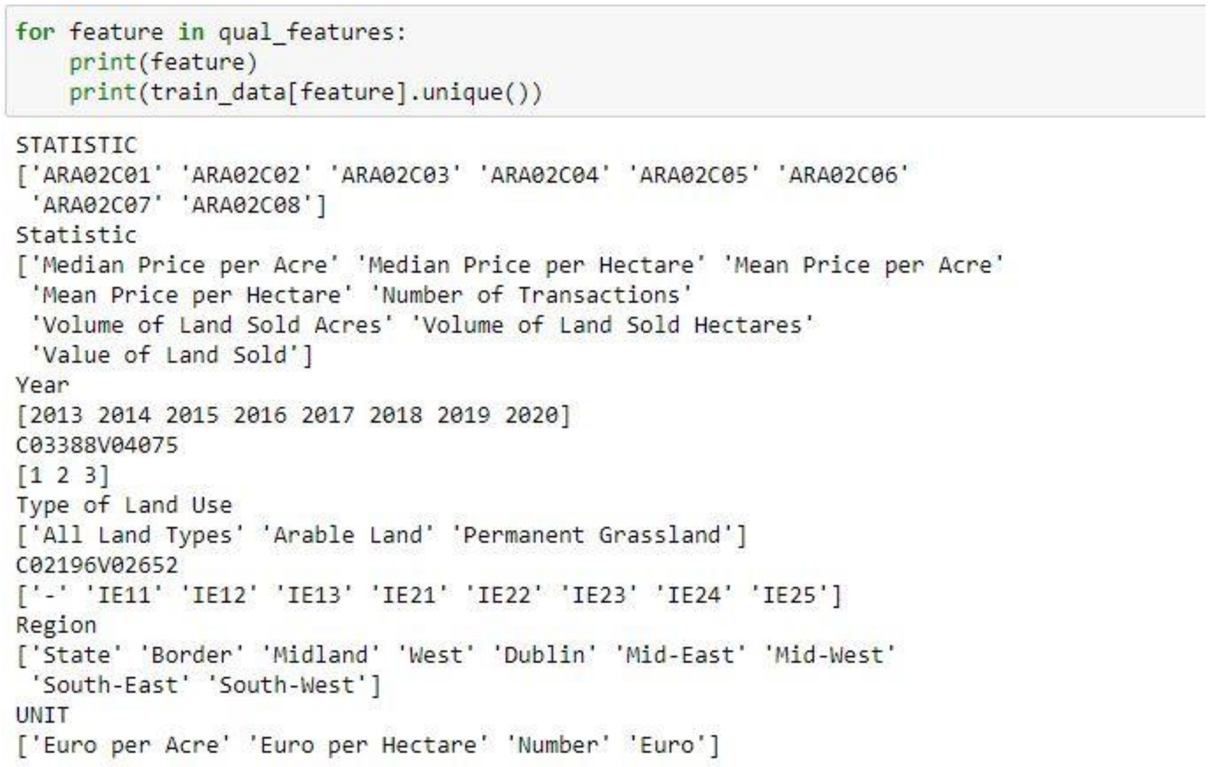
```
qual_features=[feature for feature in train_data.columns if len(train_data[feature].unique())<10]
qual_features
```

```
['STATISTIC',
 'Statistic',
 'Year',
 'C03388V04075',
 'Type of Land Use',
 'C02196V02652',
 'Region',
 'UNIT']
```

Here above qualitative features are those features which are non-numerical features having less than 10 categories.

These features are also called categorical features.

Unique values of categorical features

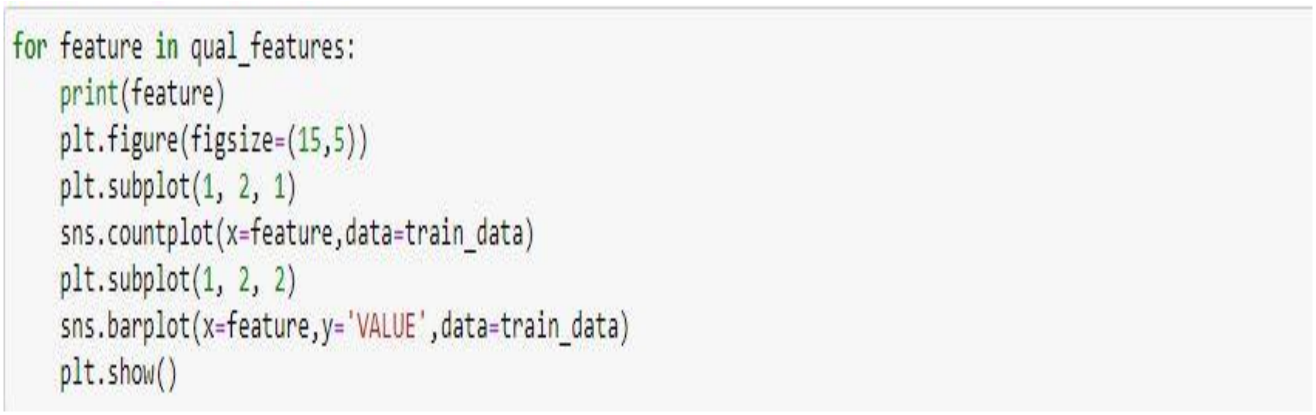


```
for feature in qual_features:
    print(feature)
    print(train_data[feature].unique())
```

```
STATISTIC
['ARA02C01' 'ARA02C02' 'ARA02C03' 'ARA02C04' 'ARA02C05' 'ARA02C06'
 'ARA02C07' 'ARA02C08']
Statistic
['Median Price per Acre' 'Median Price per Hectare' 'Mean Price per Acre'
 'Mean Price per Hectare' 'Number of Transactions'
 'Volume of Land Sold Acres' 'Volume of Land Sold Hectares'
 'Value of Land Sold']
Year
[2013 2014 2015 2016 2017 2018 2019 2020]
C03388V04075
[1 2 3]
Type of Land Use
['All Land Types' 'Arable Land' 'Permanent Grassland']
C02196V02652
['-' 'IE11' 'IE12' 'IE13' 'IE21' 'IE22' 'IE23' 'IE24' 'IE25']
Region
['State' 'Border' 'Midland' 'West' 'Dublin' 'Mid-East' 'Mid-West'
 'South-East' 'South-West']
UNIT
['Euro per Acre' 'Euro per Hectare' 'Number' 'Euro']
```

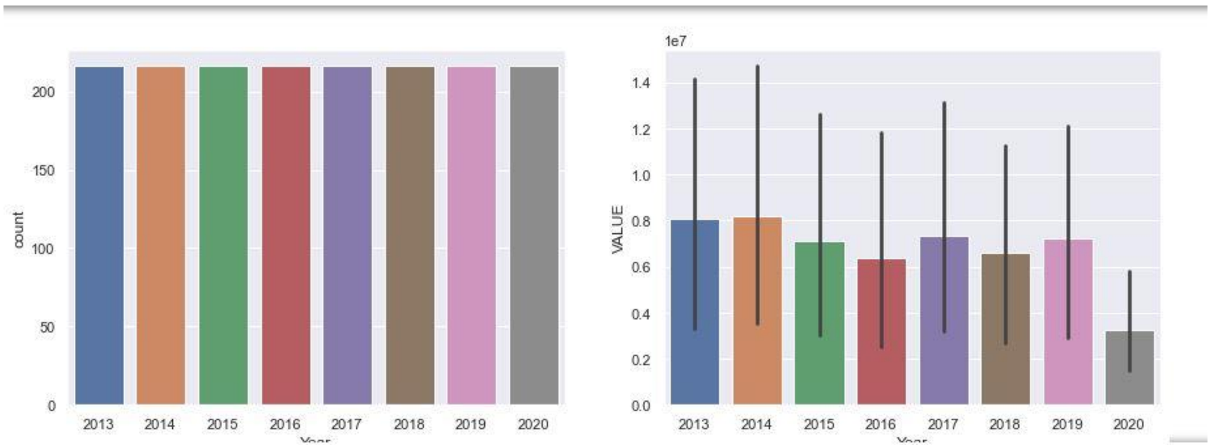
Here above are unique values of all categorical features .

Exploring categorical features using bar charts



```
for feature in qual_features:
    print(feature)
    plt.figure(figsize=(15,5))
    plt.subplot(1, 2, 1)
    sns.countplot(x=feature,data=train_data)
    plt.subplot(1, 2, 2)
    sns.barplot(x=feature,y='VALUE',data=train_data)
    plt.show()
```

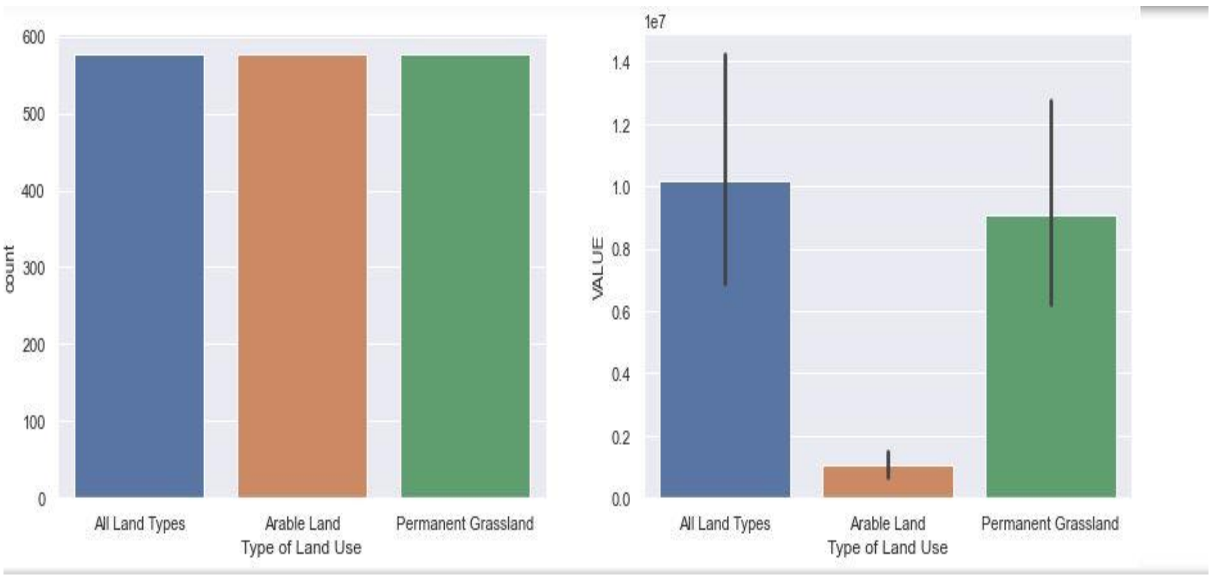
Relationship of years with sales



Left bar chart shows all years has same frequency.

Right bar charts shows that 2013 and 2014 are years with high sales

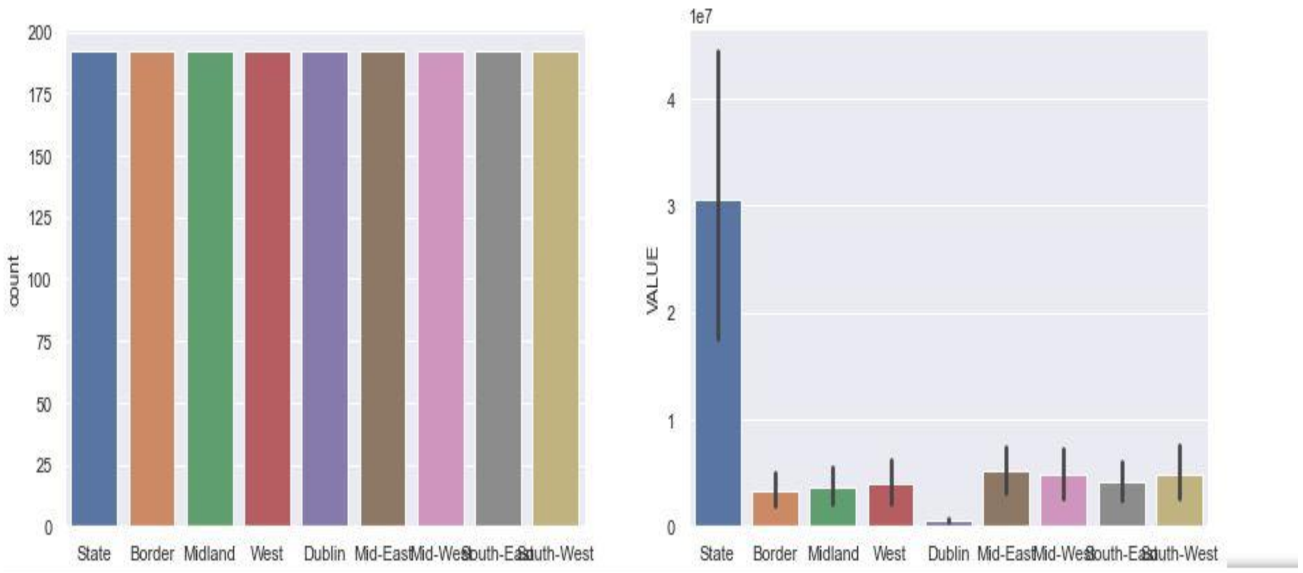
Relationship of land use with sales



Left bar chart shows that all types of lands have same frequency.

right chart shows all types of land and grassland are lands with high sales

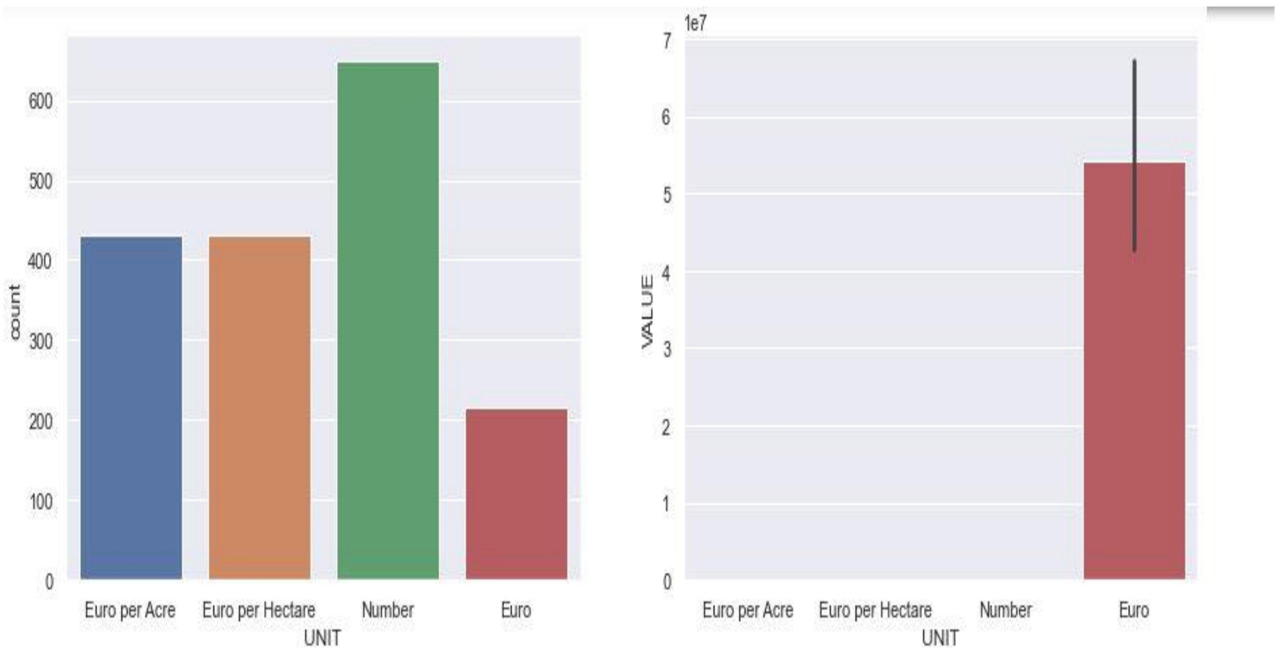
Relationship of region with sales



Left bar chart shows that all region have same frequency

Right bar chart shows that state is region with high sales of land

Relationship of unit with sales



Left bar chart shows that Number has high frequency than other

Right bar chart shows that Euro is currency with high sales.

Quantitative features:

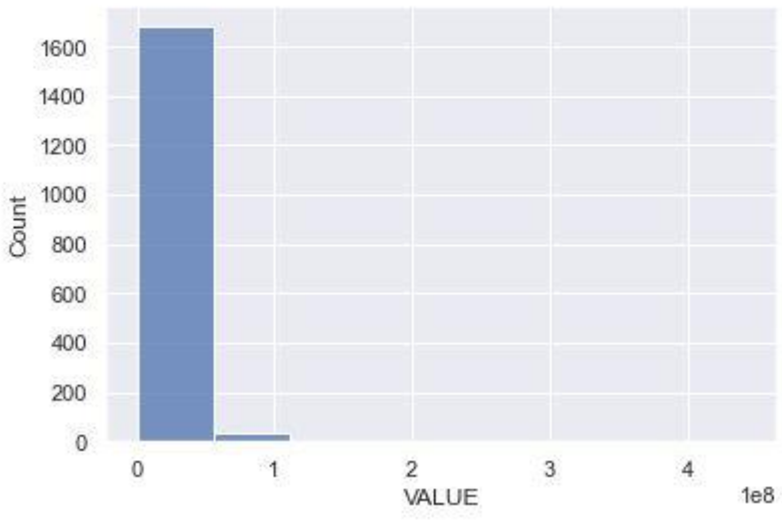
```
quant_features=[feature for feature in train_data.columns if train_data[feature].dtype!='O' and
                len(train_data[feature].unique())>10 ]
quant_features
```

```
['VALUE']
```

There is only one feature which is numerical and have more than 10 unique values.

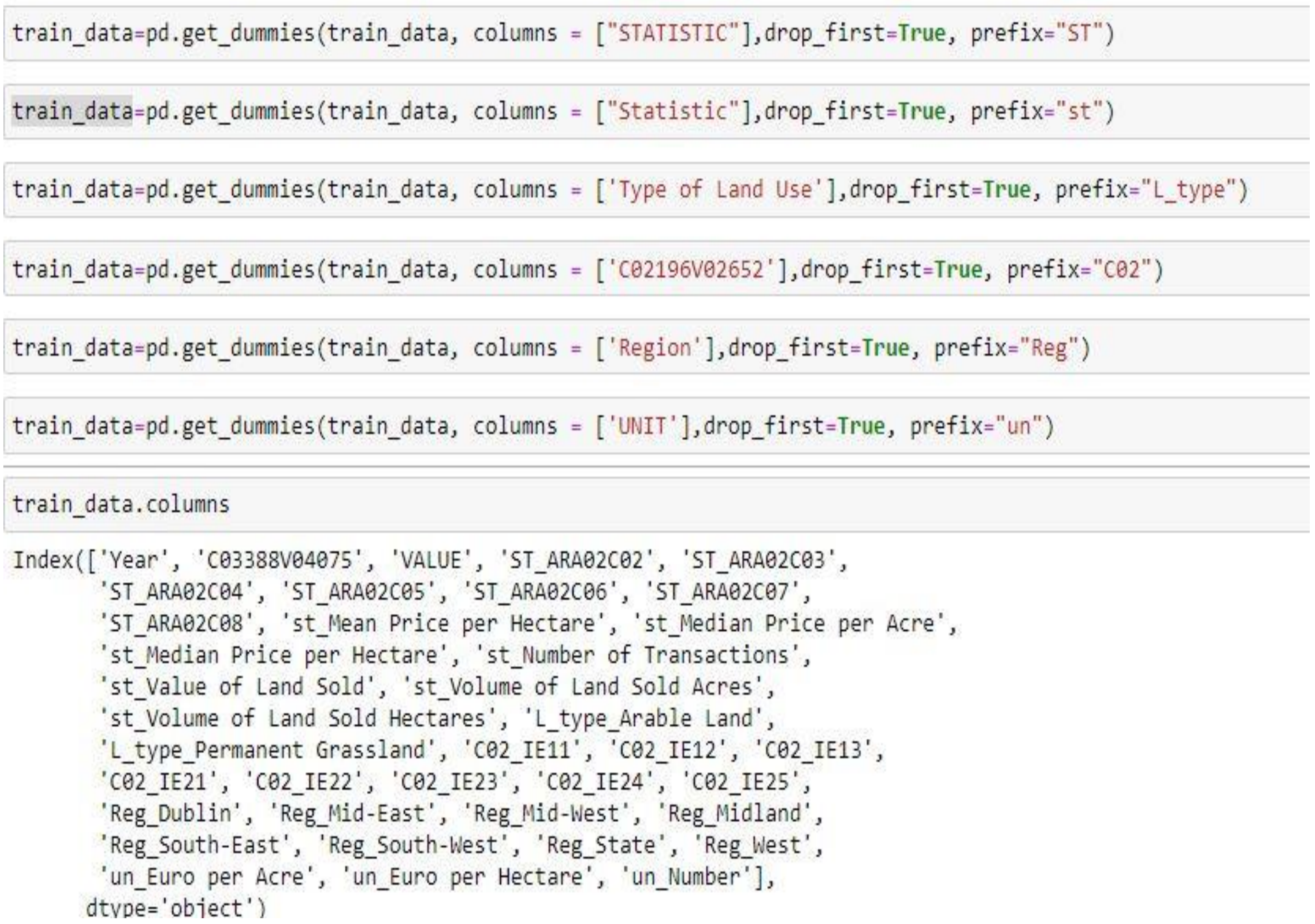
Showing distribution of values in quantitative features using histogram

```
for feature in quant_features:
    sns.histplot(train_data[feature],bins=8)
    plt.show()
```



So this is distribution of VALUE so as by looking on histogram it is clear that 0-50% has highest sales VALUE

Handling categorical features using One hot encoding



```
train_data=pd.get_dummies(train_data, columns = ["STATISTIC"],drop_first=True, prefix="ST")
```

```
train_data=pd.get_dummies(train_data, columns = ["Statistic"],drop_first=True, prefix="st")
```

```
train_data=pd.get_dummies(train_data, columns = ['Type of Land Use'],drop_first=True, prefix="L_type")
```

```
train_data=pd.get_dummies(train_data, columns = ['C02196V02652'],drop_first=True, prefix="C02")
```

```
train_data=pd.get_dummies(train_data, columns = ['Region'],drop_first=True, prefix="Reg")
```

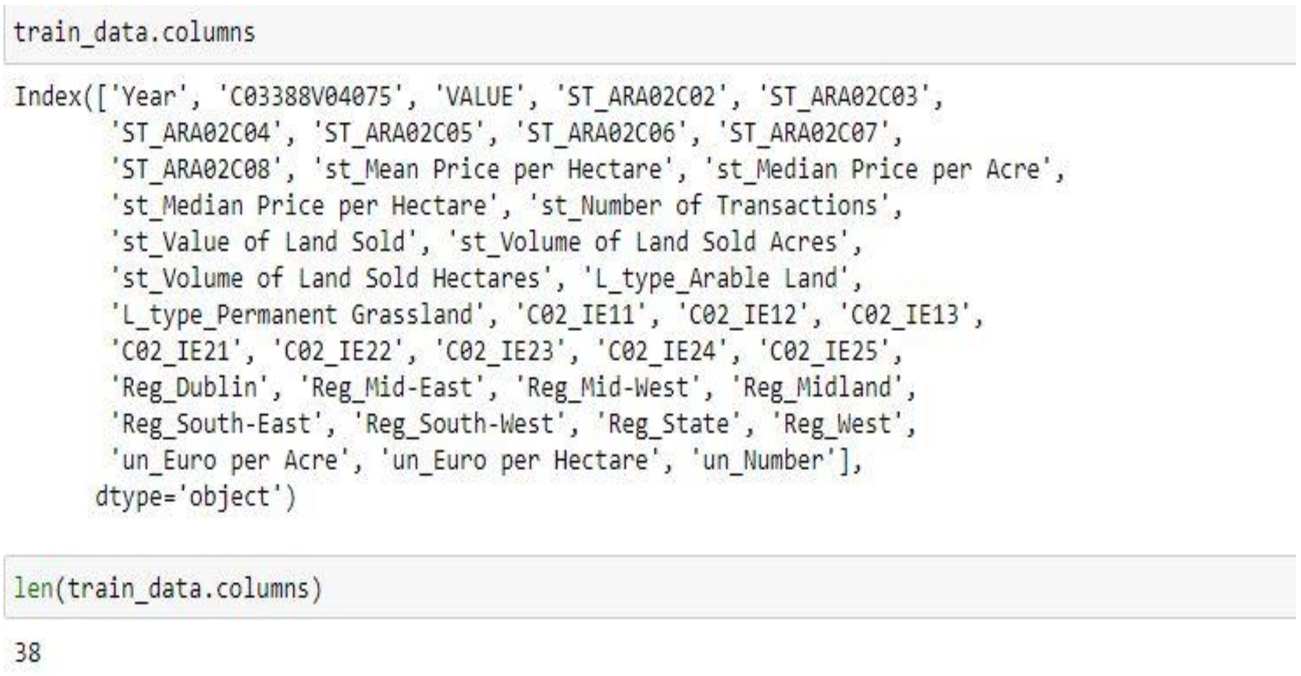
```
train_data=pd.get_dummies(train_data, columns = ['UNIT'],drop_first=True, prefix="un")
```

```
train_data.columns
```

```
Index(['Year', 'C03388V04075', 'VALUE', 'ST_ARA02C02', 'ST_ARA02C03',
       'ST_ARA02C04', 'ST_ARA02C05', 'ST_ARA02C06', 'ST_ARA02C07',
       'ST_ARA02C08', 'st_Mean Price per Hectare', 'st_Median Price per Acre',
       'st_Median Price per Hectare', 'st_Number of Transactions',
       'st_Value of Land Sold', 'st_Volume of Land Sold Acres',
       'st_Volume of Land Sold Hectares', 'L_type_Arable Land',
       'L_type_Permanent Grassland', 'C02_IE11', 'C02_IE12', 'C02_IE13',
       'C02_IE21', 'C02_IE22', 'C02_IE23', 'C02_IE24', 'C02_IE25',
       'Reg_Dublin', 'Reg_Mid-East', 'Reg_Mid-West', 'Reg_Midland',
       'Reg_South-East', 'Reg_South-West', 'Reg_State', 'Reg_West',
       'un_Euro per Acre', 'un_Euro per Hectare', 'un_Number'],
      dtype='object')
```

Here six categorical features are converted to numerical features.

So that can be used as inputs in training.



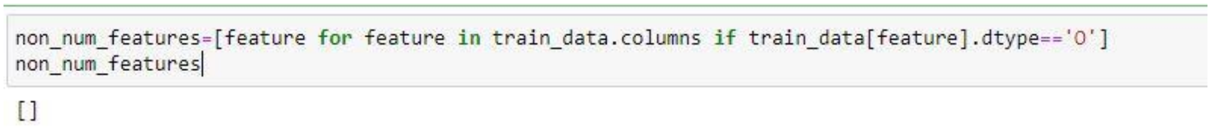
```
train_data.columns
```

```
Index(['Year', 'C03388V04075', 'VALUE', 'ST_ARA02C02', 'ST_ARA02C03',
       'ST_ARA02C04', 'ST_ARA02C05', 'ST_ARA02C06', 'ST_ARA02C07',
       'ST_ARA02C08', 'st_Mean Price per Hectare', 'st_Median Price per Acre',
       'st_Median Price per Hectare', 'st_Number of Transactions',
       'st_Value of Land Sold', 'st_Volume of Land Sold Acres',
       'st_Volume of Land Sold Hectares', 'L_type_Arable Land',
       'L_type_Permanent Grassland', 'C02_IE11', 'C02_IE12', 'C02_IE13',
       'C02_IE21', 'C02_IE22', 'C02_IE23', 'C02_IE24', 'C02_IE25',
       'Reg_Dublin', 'Reg_Mid-East', 'Reg_Mid-West', 'Reg_Midland',
       'Reg_South-East', 'Reg_South-West', 'Reg_State', 'Reg_West',
       'un_Euro per Acre', 'un_Euro per Hectare', 'un_Number'],
      dtype='object')
```

```
len(train_data.columns)
```

```
38
```

So there are total 38 features after performing one hot encoding so at beginning there are 10 features. After one hot encoding it becomes 38 features.



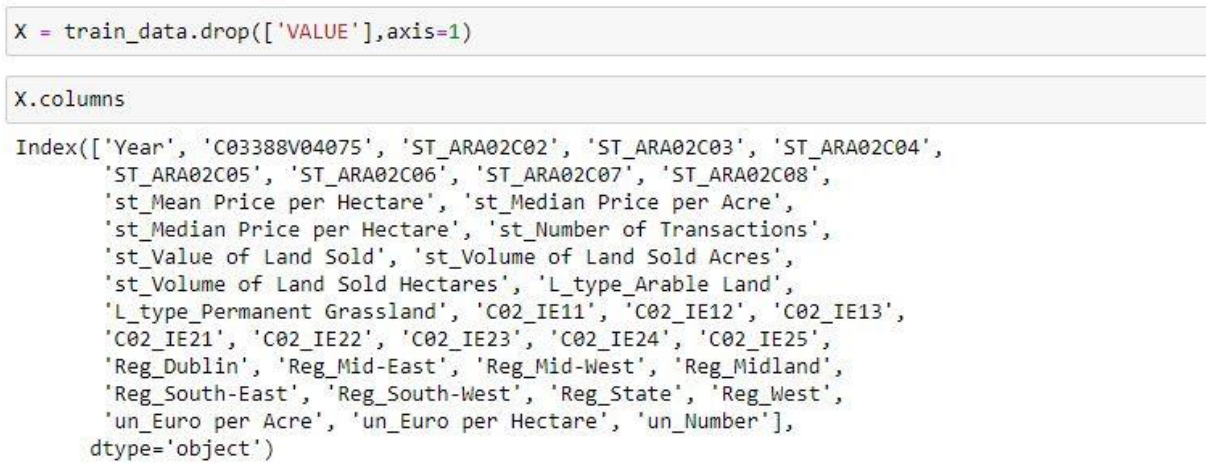
```
non_num_features=[feature for feature in train_data.columns if train_data[feature].dtype=='O']
non_num_features
```

```
[]
```

As after doing one hot encoding there is no categorical features

Preparing dataset for regression

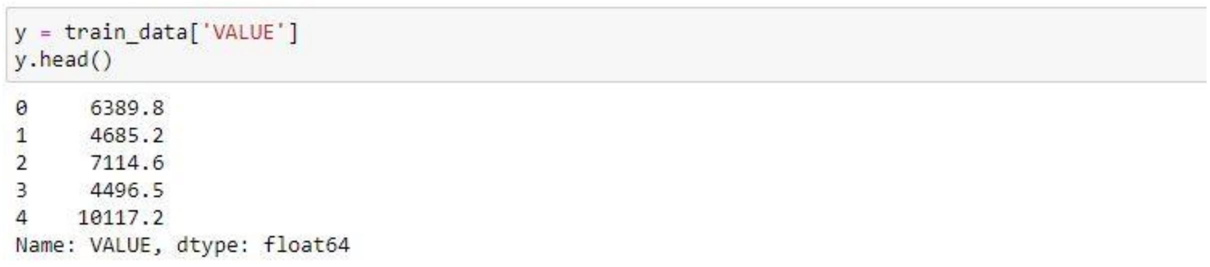
Splitting of data into dependant and independent



```
X = train_data.drop(['VALUE'],axis=1)
```

```
X.columns
```

```
Index(['Year', 'C03388V04075', 'ST_ARA02C02', 'ST_ARA02C03', 'ST_ARA02C04',
       'ST_ARA02C05', 'ST_ARA02C06', 'ST_ARA02C07', 'ST_ARA02C08',
       'st_Mean Price per Hectare', 'st_Median Price per Acre',
       'st_Median Price per Hectare', 'st_Number of Transactions',
       'st_Value of Land Sold', 'st_Volume of Land Sold Acres',
       'st_Volume of Land Sold Hectares', 'L_type_Arable Land',
       'L_type_Permanent Grassland', 'C02_IE11', 'C02_IE12', 'C02_IE13',
       'C02_IE21', 'C02_IE22', 'C02_IE23', 'C02_IE24', 'C02_IE25',
       'Reg_Dublin', 'Reg_Mid-East', 'Reg_Mid-West', 'Reg_Midland',
       'Reg_South-East', 'Reg_South-West', 'Reg_State', 'Reg_West',
       'un_Euro per Acre', 'un_Euro per Hectare', 'un_Number'],
      dtype='object')
```

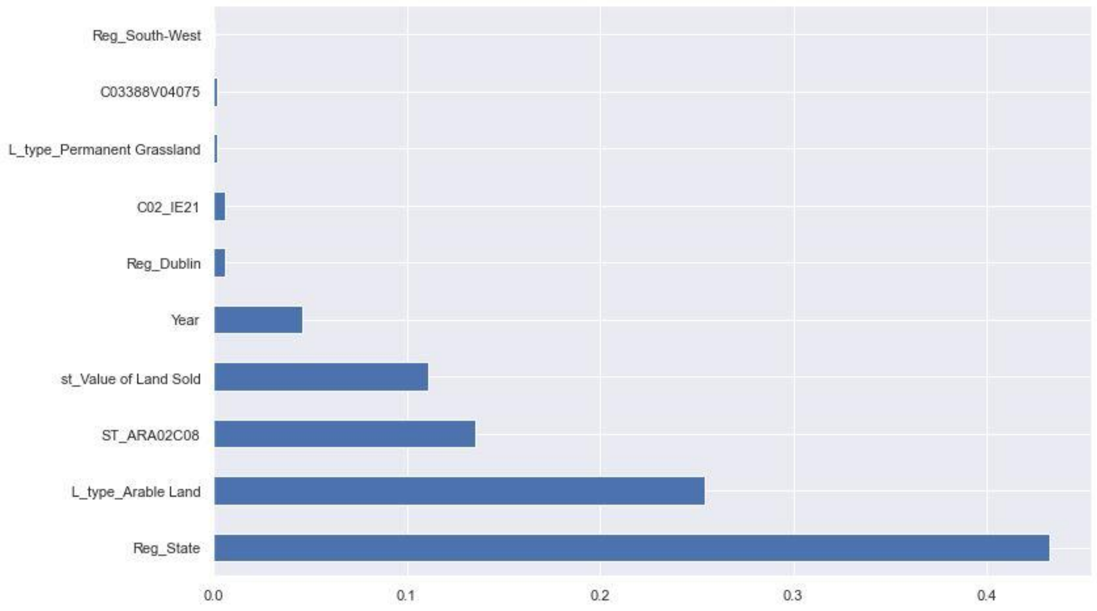


```
y = train_data['VALUE']
y.head()
```

```
0     6389.8
1     4685.2
2     7114.6
3     4496.5
4    10117.2
Name: VALUE, dtype: float64
```

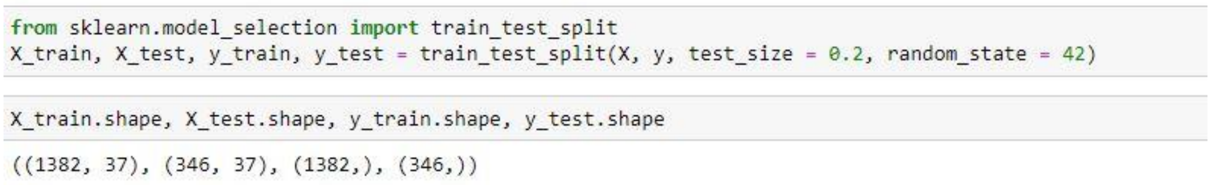
As value is targeted feature so it cannot be used as inputs so that is why it is removed from independent or inputs feature. Here inputs are represented by X

Visualizing feature importance



This bar chart is showing importance of each feature with respect to target feature sales. As by looking here it is shown that region of state and Arable land has high feature importance.

Splitting of data into train and test



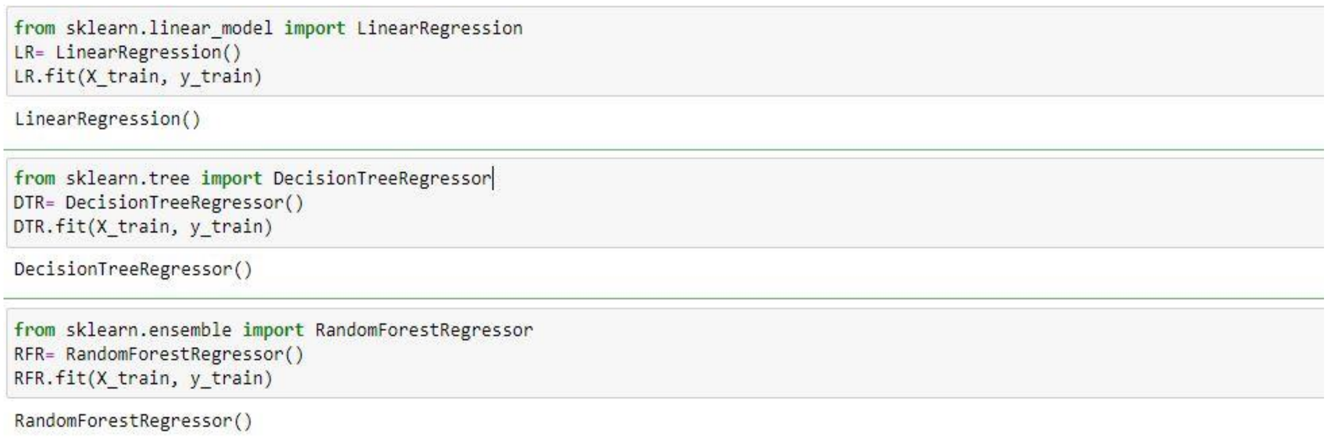
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((1382, 37), (346, 37), (1382,), (346,))
```

Here dataset is divided into train and test part at the ratio of 20% .test part is used for evaluation of model.

Machine learning Model Building



```
from sklearn.linear_model import LinearRegression
LR= LinearRegression()
LR.fit(X_train, y_train)
```

```
LinearRegression()
```

```
from sklearn.tree import DecisionTreeRegressor
DTR= DecisionTreeRegressor()
DTR.fit(X_train, y_train)
```

```
DecisionTreeRegressor()
```

```
from sklearn.ensemble import RandomForestRegressor
RFR= RandomForestRegressor()
RFR.fit(X_train, y_train)
```

```
RandomForestRegressor()
```

Here following three regression algorithms are used:

- Linear Regression

- Decision Tree Regression
- Random Forest Regression

Data training phase is now done here

Predictions

Exploratory Data Analysis or EDA is used to take insights from the data. Data Scientists and Analysts try to find different patterns, relations, and anomalies in the data using some statistical graphs and other visualization techniques

1.Get maximum insights from a data set <br>

2.Uncover underlying structure <br>

3.Extract important variables from the dataset <br>

4.Detect outliers and anomalies(if any) <br>

5.Test underlying assumptions <br>

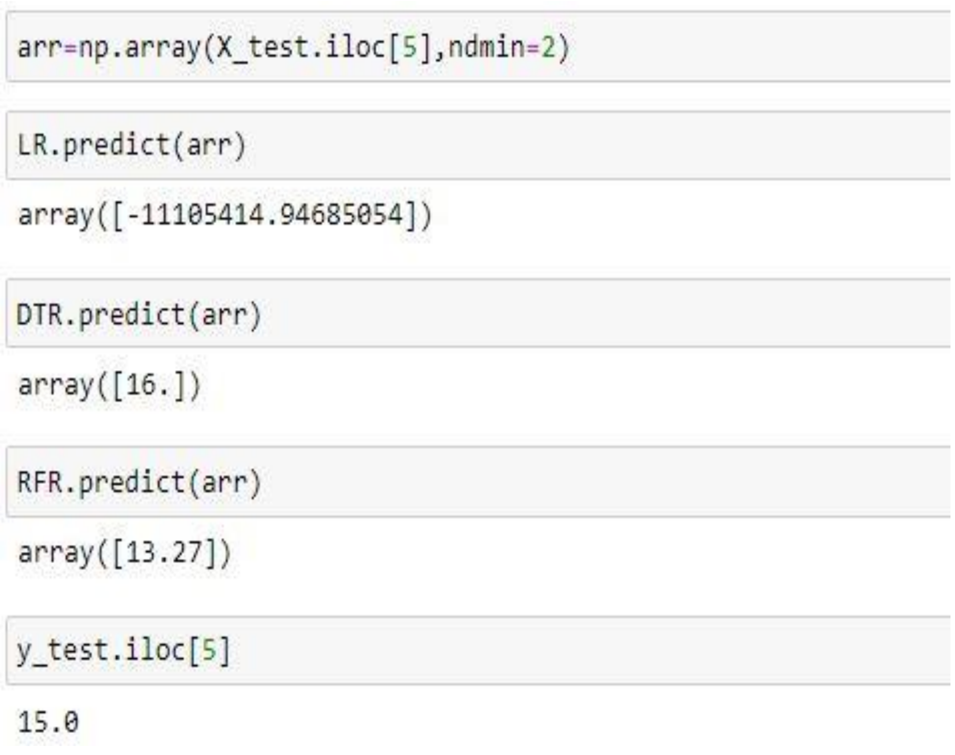
6.Determine the optimal factor settings <br>

Building predictions

Here below I am going to perform predictions so 5th record of test data is selected as inputs.

As row data is one dimensional data .So that is why it is converted to two dimensional data

As predict function requires inputs to be two dimensional.



```
arr=np.array(X_test.iloc[5],ndmin=2)
```

```
LR.predict(arr)
```

```
array([-11105414.94685054])
```

```
DTR.predict(arr)
```

```
array([16.])
```

```
RFR.predict(arr)
```

```
array([13.27])
```

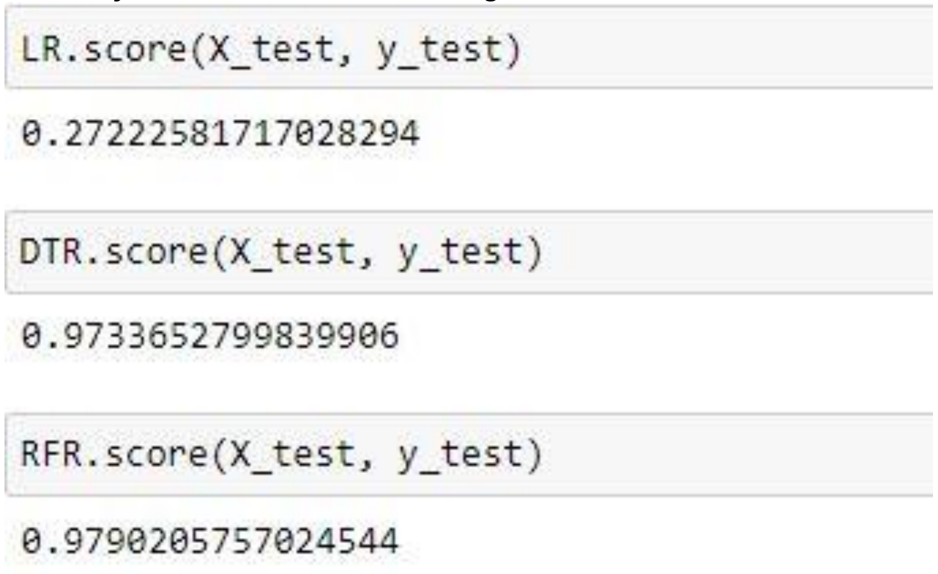
```
y_test.iloc[5]
```

```
15.0
```

Hare it the difference between predictions and actual value can be compared.

Decision Tree Regression and Random Forest regression have predictions near than actual value.

Accuracy scores of machine learning models



```
LR.score(X_test, y_test)
```

```
0.27222581717028294
```

```
DTR.score(X_test, y_test)
```

```
0.9733652799839906
```

```
RFR.score(X_test, y_test)
```

```
0.9790205757024544
```

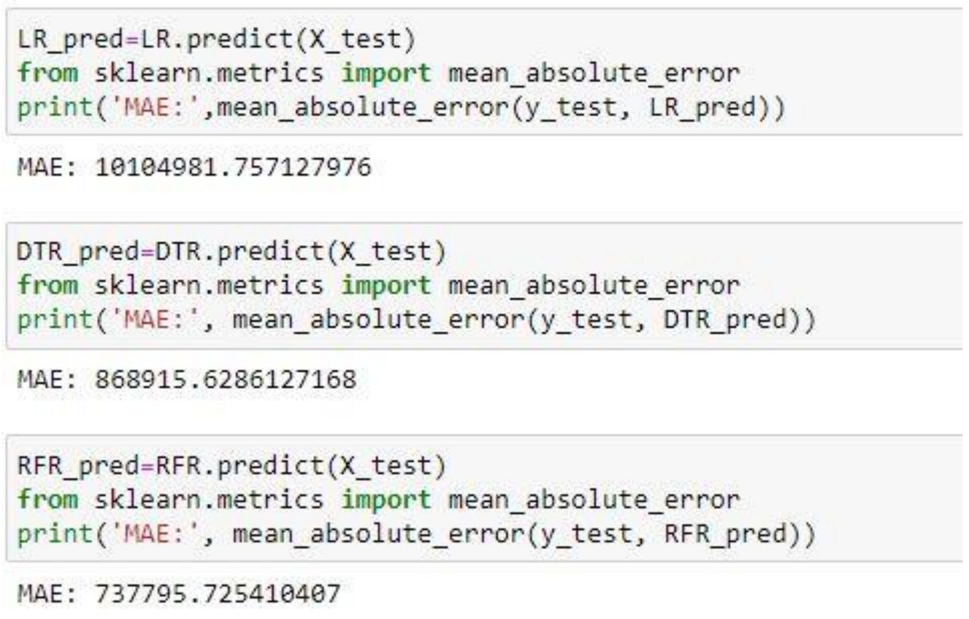
Here are accuracy scores of three different regression models.

Random Forest and Decision Tree has high accuracy scores but linear regression has very less accuracy score

Mean Absolute Error

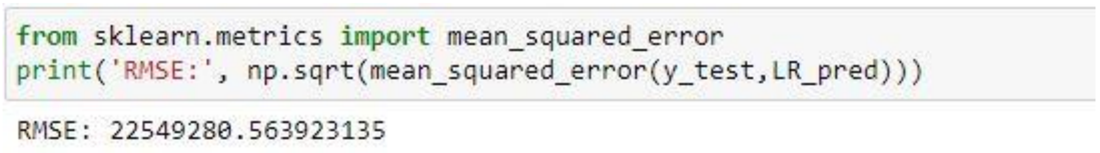
The Mean Absolute Error(MAE) is the average of all absolute errors. The formula is:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |x_i - x|$$


```
LR_pred=LR.predict(X_test)
from sklearn.metrics import mean_absolute_error
print('MAE:',mean_absolute_error(y_test, LR_pred))
```
```
MAE: 10104981.757127976
```
```
DTR_pred=DTR.predict(X_test)
from sklearn.metrics import mean_absolute_error
print('MAE:', mean_absolute_error(y_test, DTR_pred))
```
```
MAE: 868915.6286127168
```
```
RFR_pred=RFR.predict(X_test)
from sklearn.metrics import mean_absolute_error
print('MAE:', mean_absolute_error(y_test, RFR_pred))
```
```
MAE: 737795.725410407
```

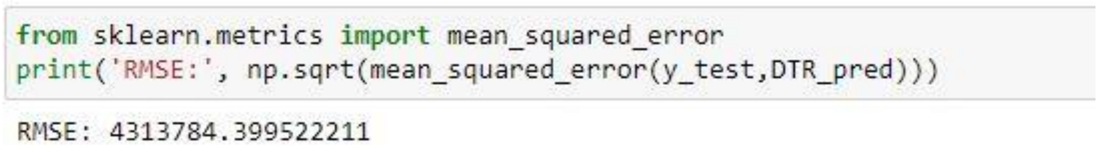
Here is Mean Absolute Error of three models .It shows that Random Forest Regression has very less MAE error which shows that it is very accurate.

Root Mean Square Error



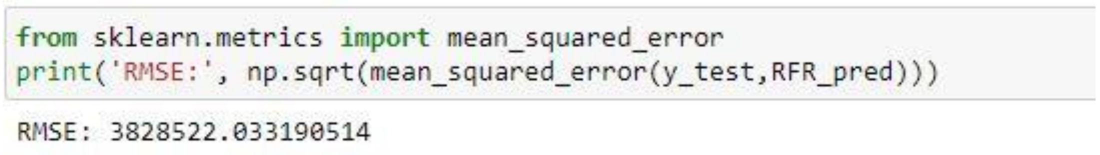
```
from sklearn.metrics import mean_squared_error
print('RMSE:', np.sqrt(mean_squared_error(y_test,LR_pred)))
```

```
RMSE: 22549280.563923135
```



```
from sklearn.metrics import mean_squared_error
print('RMSE:', np.sqrt(mean_squared_error(y_test,DTR_pred)))
```

```
RMSE: 4313784.399522211
```



```
from sklearn.metrics import mean_squared_error
print('RMSE:', np.sqrt(mean_squared_error(y_test,RFR_pred)))
```

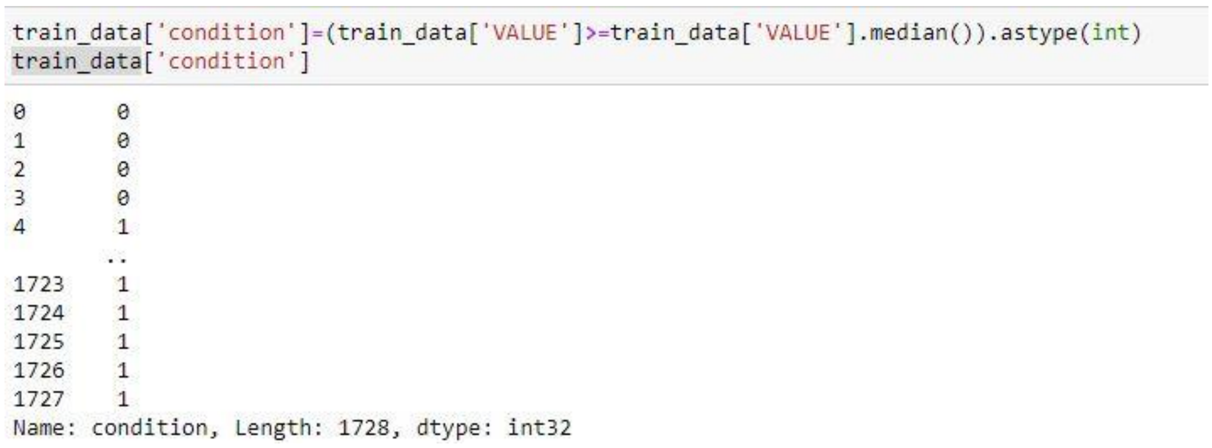
```
RMSE: 3828522.033190514
```

Here are root mean square errors of three models.

As by looking above it is clear that Random Forest regression is machine learning model very less Root mean Square Error root mean square error.

Preparing dataset for classification

Making a new categorical feature for performing classification

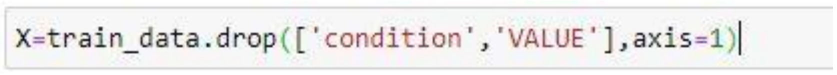


```
train_data['condition']=(train_data['VALUE']>=train_data['VALUE'].median()).astype(int)
train_data['condition']
```

```
0       0
1       0
2       0
3       0
4       1
       ..
1723    1
1724    1
1725    1
1726    1
1727    1
Name: condition, Length: 1728, dtype: int32
```

Here a new feature is developed by taking median as threshold of VALUE so above median is considered as expensive and below median is considered as cheaper than above median is converted to 1 and below median is converted to 0.

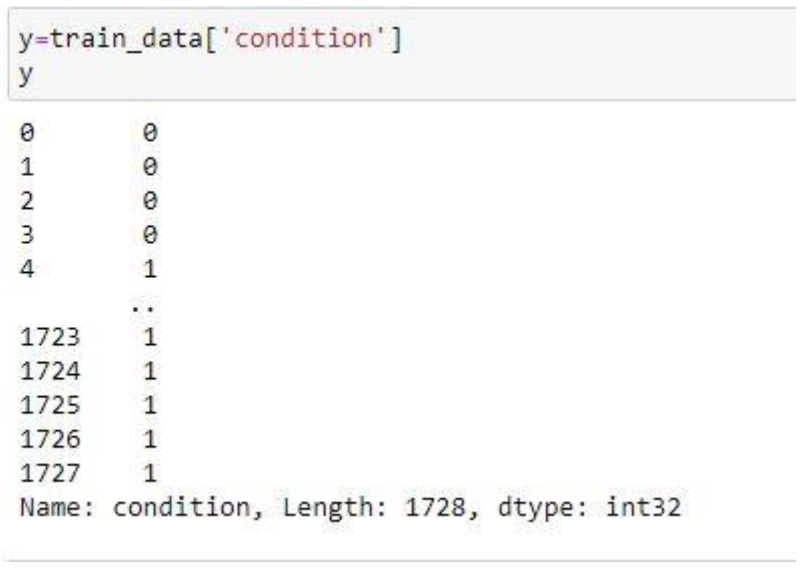
Preparing independent data by removing two features



```
X=train_data.drop(['condition','VALUE'],axis=1)
```

As condition is targeted feature so it is removed with its parent feature VALUE so here X is data frame which have inputs features

Preparing dependant feature



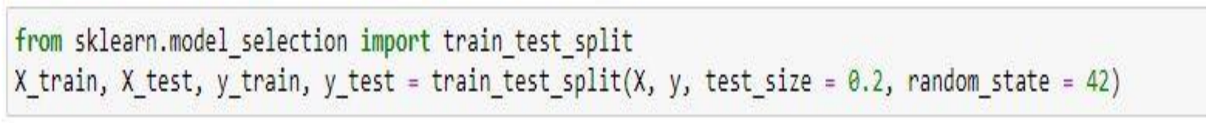
```
y=train_data['condition']
y
```

```
0       0
1       0
2       0
3       0
4       1
       ..
1723    1
1724    1
1725    1
1726    1
1727    1
Name: condition, Length: 1728, dtype: int32
```

Here condition is that feature which is to be predicted so it is targeted feature or dependant feature.

It has values of zeroes and ones.

Splitting data into train and test

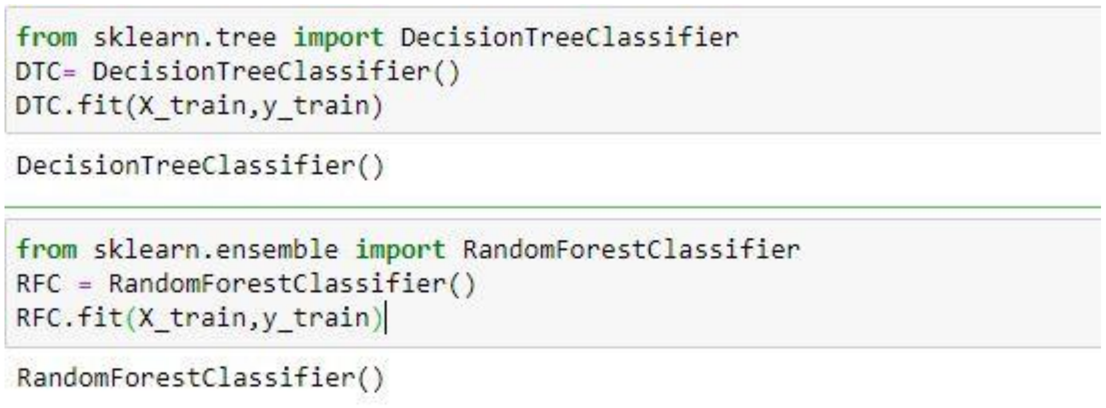


```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

Here X_train and y_train is 80 % of data which is giving as inputs to model

Here X_test and y_test is covering 20% of data which is used to evaluate models.

Building classification models.



```
from sklearn.tree import DecisionTreeClassifier
DTC= DecisionTreeClassifier()
DTC.fit(X_train,y_train)
```

```
DecisionTreeClassifier()
```

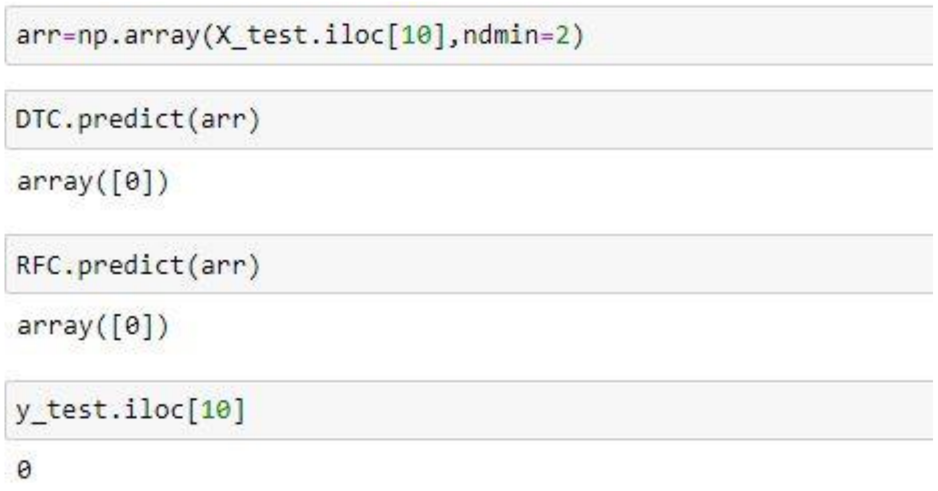
```
from sklearn.ensemble import RandomForestClassifier
RFC = RandomForestClassifier()
RFC.fit(X_train,y_train)
```

```
RandomForestClassifier()
```

Here two classification models are used which are following:

- Decision Tree Classification
- Random Forest Classification

Doing predictions



```
arr=np.array(X_test.iloc[10],ndmin=2)
```

```
DTC.predict(arr)
```

```
array([0])
```

```
RFC.predict(arr)
```

```
array([0])
```

```
y_test.iloc[10]
```

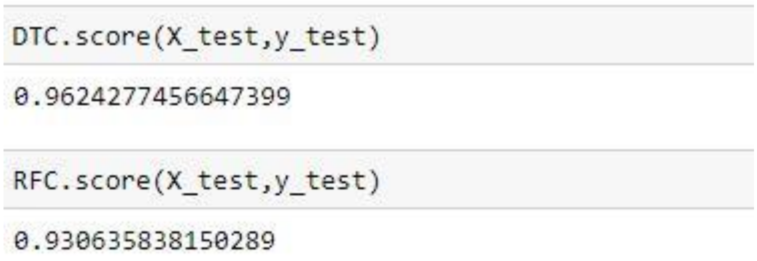
```
0
```

Here are predictions of two classification models on the basis of row 10 of test data.so all are giving same result.

Evaluating classification models

Accuracy score

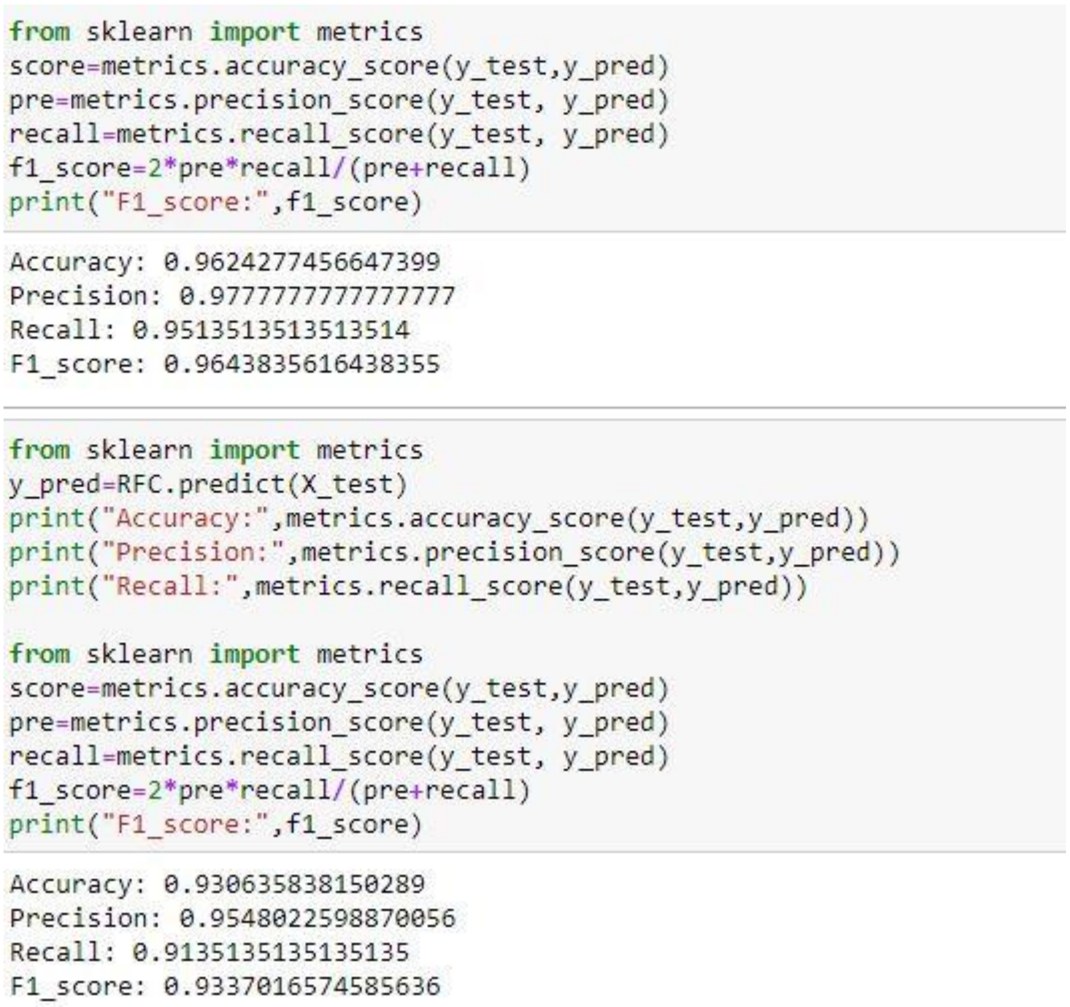
dividing the number of correct predictions (the corresponding diagonal in the matrix) by the total number of samples is called accuracy


```
DTC.score(X_test,y_test)
```
```
0.9624277456647399
```
```
RFC.score(X_test,y_test)
```
```
0.930635838150289
```

Here Decision Tree classification has high accuracy than Random forest regression

F1_score

The F1 score is defined as the harmonic mean of precision and recall.

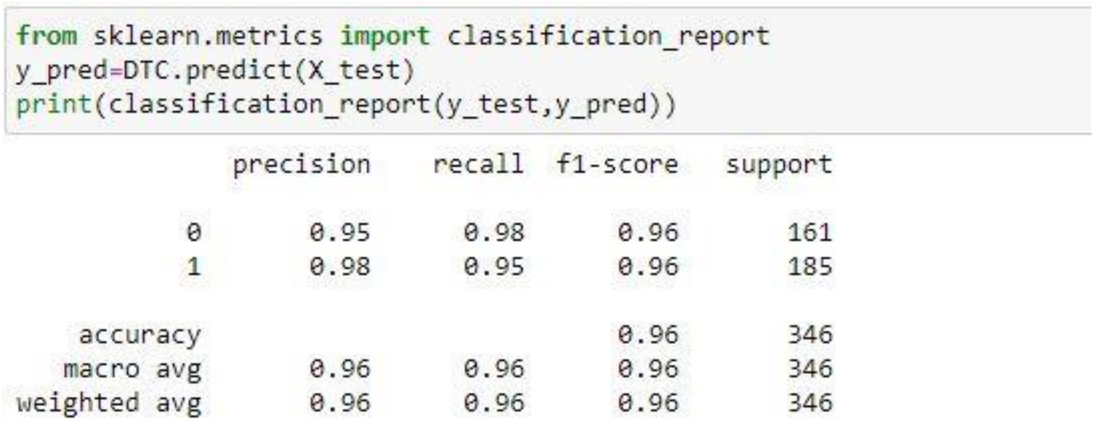

```
from sklearn import metrics
score=metrics.accuracy_score(y_test,y_pred)
pre=metrics.precision_score(y_test, y_pred)
recall=metrics.recall_score(y_test, y_pred)
f1_score=2*pre*recall/(pre+recall)
print("F1_score:",f1_score)
```
```
Accuracy: 0.9624277456647399
Precision: 0.9777777777777777
Recall: 0.9513513513513514
F1_score: 0.9643835616438355
```
```
from sklearn import metrics
y_pred=RFC.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test,y_pred))
print("Precision:",metrics.precision_score(y_test,y_pred))
print("Recall:",metrics.recall_score(y_test,y_pred))

from sklearn import metrics
score=metrics.accuracy_score(y_test,y_pred)
pre=metrics.precision_score(y_test, y_pred)
recall=metrics.recall_score(y_test, y_pred)
f1_score=2*pre*recall/(pre+recall)
print("F1_score:",f1_score)
```
```
Accuracy: 0.930635838150289
Precision: 0.9548022598870056
Recall: 0.9135135135135135
F1_score: 0.9337016574585636
```
As by examining F1 score it is clear that Decision Tree Classification has high F1_score

Precision ( ) is defined as the number of true positives ( ) over the number of true positives plus the number of false positives ( ). Recall ( ) is defined as the number of true positives ( ) over the number of true positives plus the number of false negatives ( ).

So it has high performance.

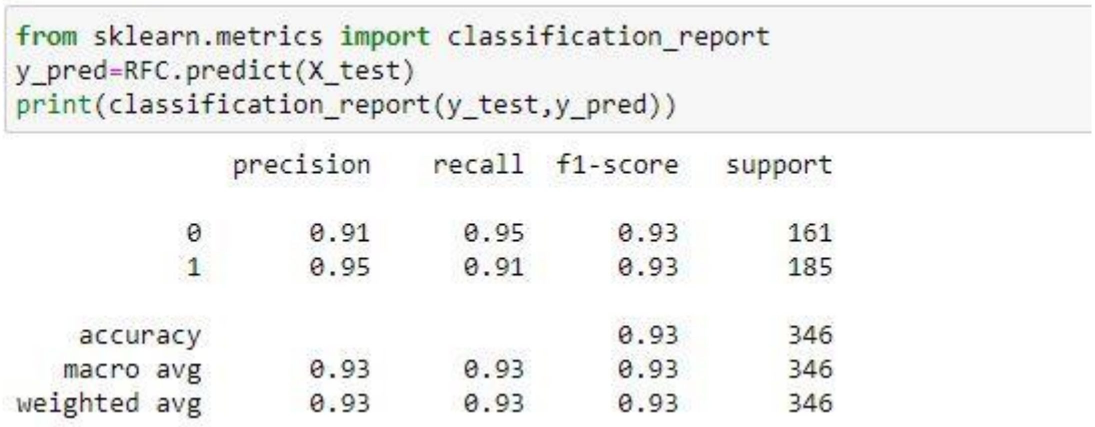
Classification report

What is a classification report?
A classification report is a performance evaluation metric in machine learning. It is used to show the precision, recall, F1 Score, and support of your trained classification model.



```
from sklearn.metrics import classification_report
y_pred=DTC.predict(X_test)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.98      0.96       161
           1       0.98      0.95      0.96       185

    accuracy                           0.96       346
   macro avg       0.96      0.96      0.96       346
weighted avg       0.96      0.96      0.96       346
```



```
from sklearn.metrics import classification_report
y_pred=RFC.predict(X_test)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.91      0.95      0.93       161
           1       0.95      0.91      0.93       185

    accuracy                           0.93       346
   macro avg       0.93      0.93      0.93       346
weighted avg       0.93      0.93      0.93       346
```

Here this classification report shows precision recall and F1_score of each category

So in both models category of one and zero has same F1_score.

Conclusions:

On the basis of analysis and visualization these are conclusions.

Missing values

There are no missing values in dataset so it is not needed to clean dataset.

Feature Engineering:

Here one feature is removed as this is similar to year

Here a feature ''VALUE" IS Converted to 1 and zero by replacement

Best model:

As Random Forest regression has high accuracy in regression but decision tree classifier is best model with high accuracy rate in classification.

Reference:

Dataset at https://data.gov.ie/:

https://data.gov.ie/dataset/ara02-agricultural-land-sales?package_type=dataset

Article on this topic ''How To Perform Exploratory Data Analysis -A Guide for Beginners''

Machine learning tasks:

https://machinelearningmastery.com/machine-learning-in-python-step-by-step/

here is step by step method of building machine learning model

Exploratory data analysis:

https://medium.com/@hammad173216/my-first-exploratory-data-analysis-project-in-python-on-sales-analysis-2e48880d4094