Naive Baye's- Classifier

Author - Dev

We will do Knn classifier on our same housing market data.

Importing all the needed modules:

```
import pandas as pd
import numpy as np
from sklearn import preprocessing, naive_bayes, model_selection, metrics
import matplotlib.pyplot as plt
import seaborn as sb
```

Reading Data

```
#Importing our training and testing house prices data set.
df=pd.read_excel("/Users/devmarwah/Downloads/BA-Predict-2.xlsx")
```

```
#Printing head of training dataset:
df.head()
```

	LotArea	OverallQual	YearBuilt	YearRemodAdd	BsmtFinSF1	FullBath	HalfBath	BedroomAbvGr	TotRmsAbvGrd	Fireplaces	GarageArea	YrSold	SalePrice
0	7340	4	1971	1971	322	1	0	2	4	0	684	2007	110000
1	8712	5	1957	2000	860	1	0	2	5	0	756	2009	153000
2	7875	7	2003	2003	0	2	1	3	8	1	393	2006	180000
3	14859	7	2006	2006	0	2	0	3	7	1	690	2006	240000
4	6173	5	1967	1967	599	1	0	3	6	0	288	2007	125500

Data cleaning:

Checking for any missing values in our dataset:

```
df.isna().sum()
```

```
LotArea         0
OverallQual     0
YearBuilt       0
YearRemodAdd    0
BsmtFinSF1      0
FullBath        0
HalfBath        0
BedroomAbvGr    0
TotRmsAbvGrd    0
Fireplaces      0
GarageArea      0
YrSold          0
SalePrice       0
dtype: int64
```

Hence, there are no missing values in our dataset.

Now, checking for duplicates in the dataset.

```
df.duplicated().sum()
```

```
0
```

Hence, there are no dubplicate values in our dataset.

Data prepration:

```
df.head()
```

	LotArea	OverallQual	YearBuilt	YearRemodAdd	BsmtFinSF1	FullBath	HalfBath	BedroomAbvGr	TotRmsAbvGrd	Fireplaces	GarageArea	YrSold	SalePrice
0	7340	4	1971	1971	322	1	0	2	4	0	684	2007	110000
1	8712	5	1957	2000	860	1	0	2	5	0	756	2009	153000
2	7875	7	2003	2003	0	2	1	3	8	1	393	2006	180000
3	14859	7	2006	2006	0	2	0	3	7	1	690	2006	240000
4	6173	5	1967	1967	599	1	0	3	6	0	288	2007	125500

In our dataset, following variables are categorical: "OverallQual","FullBath","HalfBath","BedroomAbvGr","TotRmsAbvGrd","Fireplaces","YrSold"

Hence, we need to convert them to category dtype.

```
df[["OverallQual","FullBath","HalfBath","BedroomAbvGr","TotRmsAbvGrd","Fireplaces","YrSold"]]=df[["OverallQual","FullBath","HalfBath","BedroomAbvGr","To
#Printing dtypes of our data to verify:
df.dtypes
```

```
LotArea            int64
OverallQual     category
YearBuilt          int64
YearRemodAdd       int64
BsmtFinSF1         int64
FullBath        category
HalfBath        category
BedroomAbvGr    category
TotRmsAbvGrd    category
Fireplaces      category
GarageArea         int64
YrSold          category
SalePrice          int64
dtype: object
```

Hence, all categorical variables have been converted to category dtype now.

```
#Variable OverallQual just rates overall quality and is not necessary in our analysis so we will drop it
df.drop('OverallQual',axis=1,inplace=True)
```

```
#Printing count of distinct categories in variables to find variables which should be converted as dummy variables:
print(
df['FullBath'].unique(),
df['HalfBath'].unique(),
df['BedroomAbvGr'].unique(),
df['TotRmsAbvGrd'].unique(),
df['Fireplaces'].unique(),
df['YrSold'].unique())
```

```
[1, 2, 0]
Categories (3, int64): [0, 1, 2] [0, 1, 2]
Categories (3, int64): [0, 1, 2] [2, 3, 5, 4, 1]
Categories (5, int64): [1, 2, 3, 4, 5] [4, 5, 8, 7, 6, 12, 9, 10, 11]
Categories (9, int64): [4, 5, 6, 7, ..., 9, 10, 11, 12] [0, 1, 2]
Categories (3, int64): [0, 1, 2] [2007, 2009, 2006, 2010, 2008]
Categories (5, int64): [2006, 2007, 2008, 2009, 2010]
```

```
#Here totrmsabvgrd has 9 categories and can make our model complex hence we are dropping it:
df.drop('TotRmsAbvGrd',axis=1,inplace=True)
```

Naive Baye's classification is applied on classes. Hence, we will convert Saleprice into two categories ('High' and 'Low')

```
#Coverting SalePrice into 'High' and 'Low' class:
df['Price']=pd.DataFrame(np.where(df['SalePrice']>=df['SalePrice'].mean(),0,1))
df.head()
```

	LotArea	YearBuilt	YearRemodAdd	BsmtFinSF1	FullBath	HalfBath	BedroomAbvGr	Fireplaces	GarageArea	YrSold	SalePrice	Price
0	7340	1971	1971	322	1	0	2	0	684	2007	110000	1
1	8712	1957	2000	860	1	0	2	0	756	2009	153000	1
2	7875	2003	2003	0	2	1	3	1	393	2006	180000	0
3	14859	2006	2006	0	2	0	3	1	690	2006	240000	0
4	6173	1967	1967	599	1	0	3	0	288	2007	125500	1

Making training and testing data sets:

```
#Dropping SalePrice first
df.drop('SalePrice',axis=1,inplace=True)
```

Dropping all the numeric variables as Naive Baye's works best on categorical data:

```
df.drop(['LotArea','YearBuilt','YearRemodAdd','BsmtFinSF1','GarageArea'],axis=1,inplace=True)
df.head()
```

	FullBath	HalfBath	BedroomAbvGr	Fireplaces	YrSold	Price
0	1	0	2	0	2007	1
1	1	0	2	0	2009	1
2	2	1	3	1	2006	0
3	2	0	3	1	2006	0
4	1	0	3	0	2007	1

```
# Examining data type of the data:
# df=df.cat.codes


#Making training and testing dataset now:
x=np.array(df.drop('Price',axis=1))
y=np.array(df['Price'])
t=model_selection.train_test_split
x_train, x_test, y_train, y_test = t(x,y,test_size=0.1,random_state=2)
```

Model Construction

Our data is finally prepared and we can apply knn-classifier model :

```
clf=naive_bayes.CategoricalNB()
clf.fit(x_train,y_train)
```

```
▾ CategoricalNB
CategoricalNB()
```

Printing accuracy of our model:

```
print(clf.score(x_test,y_test))
```

```
0.5555555555555556
```

Having a look at predictions made:

```
p=clf.predict(x_test)
p
```

```
array([0, 1, 0, 1, 1, 1, 0, 1, 1])
```

Making confusion matrix of our model:

```
cm=metrics.confusion_matrix
c=cm(y_test,p)
plt.figure()
sb.heatmap(c,annot=True,cmap="Blues",xticklabels=['Low','High'],yticklabels=['Low','High'])
plt.xlabel('Actual values')
```

```
plt.ylabel('Predicted Values')
```

```
Text(50.722222222222214, 0.5, 'Predicted Values')
```