# Predicting Price for houses

## Machine Learning - Project 1

## Group 10

In [1]:
```
import pandas as pd
import numpy as np
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.neighbors import NearestNeighbors,KNeighborsClassifier
import matplotlib.pylab as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn import linear_model
```

## Importing Housing Dataset file

In [2]:
```
Housing = pd.read_csv("HousingDataSet.csv")
```

In [3]:
```
Housing.head()
```

Out[3]:

| | id | date | price | bedrooms | bathrooms | sqft_living | sqft_lot | floors | waterfront |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 7129300520 | 20141013T000000 | 221900.0 | 3 | 1.00 | 1180 | 5650 | 1.0 | 0 |
| **1** | 6414100192 | 20141209T000000 | 538000.0 | 3 | 2.25 | 2570 | 7242 | 2.0 | 0 |
| **2** | 5631500400 | 20150225T000000 | 180000.0 | 2 | 1.00 | 770 | 10000 | 1.0 | 0 |
| **3** | 2487200875 | 20141209T000000 | 604000.0 | 4 | 3.00 | 1960 | 5000 | 1.0 | 0 |
| **4** | 1954400510 | 20150218T000000 | 510000.0 | 3 | 2.00 | 1680 | 8080 | 1.0 | 0 |

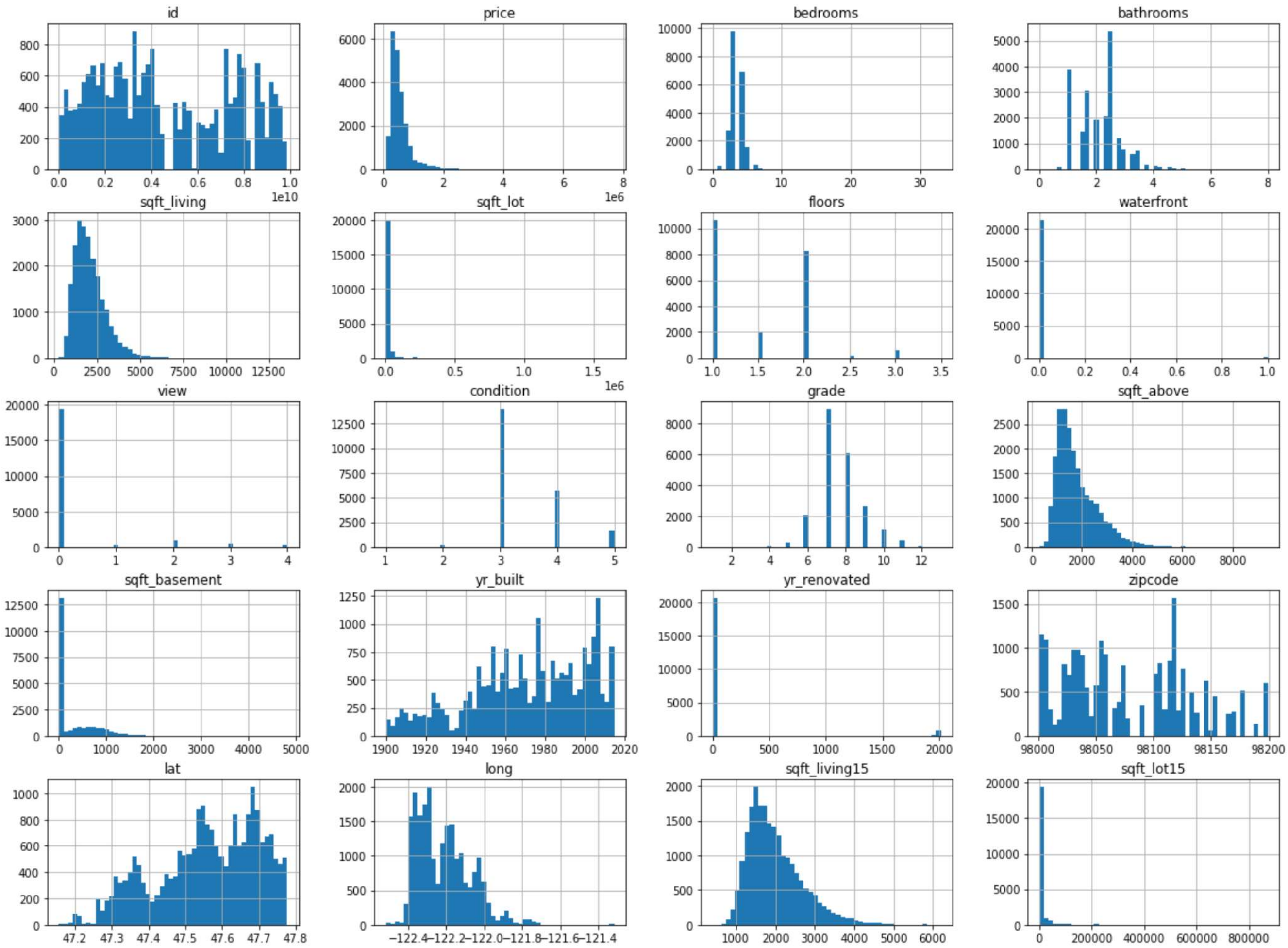
5 rows × 21 columns

In [4]:
```
Housing.shape
Housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             21613 non-null  int64  
 1   date           21613 non-null  object 
 2   price          21613 non-null  float64
 3   bedrooms       21613 non-null  int64  
 4   bathrooms      21613 non-null  float64
 5   sqft_living    21613 non-null  int64  
 6   sqft_lot       21613 non-null  int64  
 7   floors         21613 non-null  float64
 8   waterfront     21613 non-null  int64  
 9   view           21613 non-null  int64  
 10  condition      21613 non-null  int64  
 11  grade          21613 non-null  int64  
 12  sqft_above     21613 non-null  int64  
 13  sqft_basement  21613 non-null  int64  
 14  yr_built       21613 non-null  int64  
 15  yr_renovated   21613 non-null  int64  
 16  zipcode        21613 non-null  int64  
 17  lat            21613 non-null  float64
 18  long           21613 non-null  float64
 19  sqft_living15  21613 non-null  int64  
 20  sqft_lot15     21613 non-null  int64  
dtypes: float64(5), int64(15), object(1)
memory usage: 3.5+ MB
```

In [5]:
```
Housing.hist(bins = 50, figsize =(20,15))
```

Out[5]:
```
array([[<AxesSubplot:title={'center':'id'}>,
        <AxesSubplot:title={'center':'price'}>,
        <AxesSubplot:title={'center':'bedrooms'}>,
        <AxesSubplot:title={'center':'bathrooms'}>],
       [<AxesSubplot:title={'center':'sqft_living'}>,
        <AxesSubplot:title={'center':'sqft_lot'}>,
        <AxesSubplot:title={'center':'floors'}>,
        <AxesSubplot:title={'center':'waterfront'}>],
       [<AxesSubplot:title={'center':'view'}>,
        <AxesSubplot:title={'center':'condition'}>,
        <AxesSubplot:title={'center':'grade'}>,
        <AxesSubplot:title={'center':'sqft_above'}>],
       [<AxesSubplot:title={'center':'sqft_basement'}>,
        <AxesSubplot:title={'center':'yr_built'}>,
        <AxesSubplot:title={'center':'yr_renovated'}>,
        <AxesSubplot:title={'center':'zipcode'}>],
       [<AxesSubplot:title={'center':'lat'}>,
        <AxesSubplot:title={'center':'long'}>,
        <AxesSubplot:title={'center':'sqft_living15'}>,
        <AxesSubplot:title={'center':'sqft_lot15'}>]], dtype=object)
```

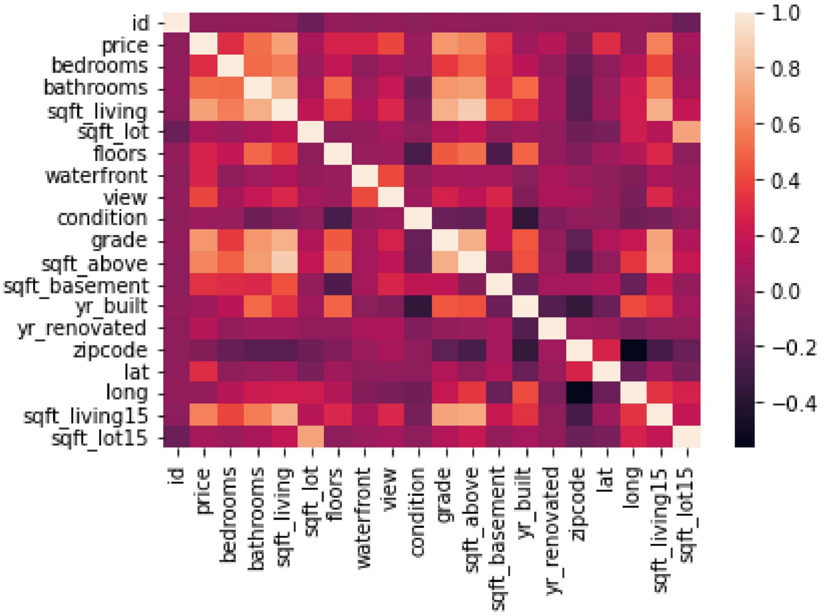


In [6]:
```python
Housing['bedrooms'].max()
```
Out[6]:
```
33
```

In [7]:
```python
Corr_Matrix = Housing.corr()
```

In [8]:
```python
Corr_Matrix['price'].sort_values(ascending = False)
```
Out[8]:
```
price            1.000000
sqft_living      0.702035
grade            0.667434
sqft_above       0.605567
sqft_living15    0.585379
bathrooms        0.525138
view             0.397293
sqft_basement    0.323816
bedrooms         0.308350
lat              0.307003
waterfront       0.266369
floors           0.256794
yr_renovated     0.126434
sqft_lot         0.089661
sqft_lot15       0.082447
yr_built         0.054012
condition        0.036362
long             0.021626
id              -0.016762
zipcode         -0.053203
Name: price, dtype: float64
```

In [9]:
```python
sns.heatmap(Housing.corr())
```
Out[9]:
```
<AxesSubplot:>
```



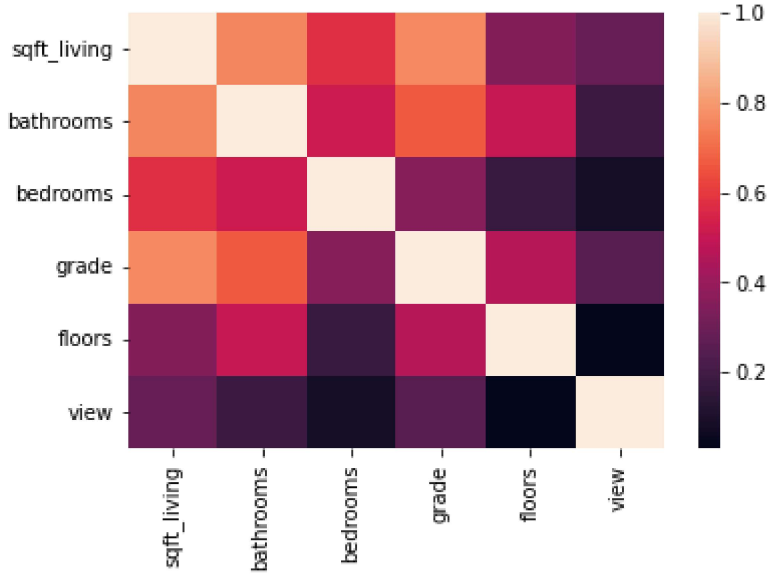

## Selecting the highly correlated features with price for prediction

In [11]:
```python
X = Housing[['sqft_living','bathrooms','bedrooms', 'grade','floors','view']]
y = Housing['price']
```

## Correlation between selected set of variables

In [71]:
```
sns.heatmap(X.corr())
```

Out[71]: <AxesSubplot:>




## Multiple Linear Regression

In [13]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

In [14]:
```
lm = LinearRegression()
```

In [15]:
```
lm.fit(X_train, y_train)
```

Out[15]: LinearRegression()

In [16]:
```
coeff_df = pd.DataFrame(lm.coef_,X.columns, columns=['Coefficient'])
```

In [17]:
```
coeff_df
```

Out[17]:

| | Coefficient |
|---|---|
| **sqft_living** | 203.161613 |
| **bathrooms** | -14607.393883 |
| **bedrooms** | -31084.966289 |
| **grade** | 92334.117869 |
| **floors** | -21272.496541 |
| **view** | 95120.225965 |

In [18]:
```
predictions = lm.predict(X_test)
```

## Predicting House Price using Multiple Linear Regression

In [39]:
```
per_error = 100*(predictions-y_test)/y_test

df_prd_tst = pd.DataFrame({'Predicted Price':predictions.astype('int64'), 'Actual Price
df_prd_tst.head(2)
```

Out[39]:

| | Predicted Price | Actual Price | % Error |
|---|---|---|---|
| **17384** | 346470 | 297000.0 | 16.656819 |
| **722** | 1399923 | 1578000.0 | -11.284953 |

## Average Accuracy Score for Multiple Linear Regression

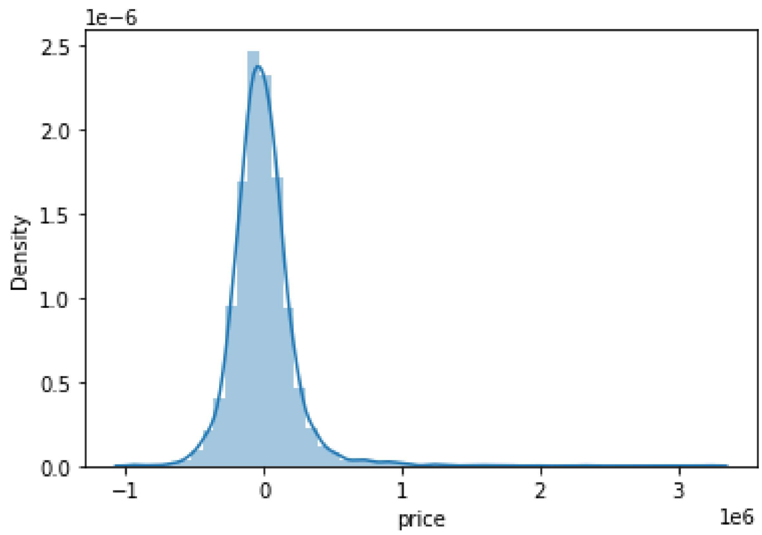
In [19]:
```
r2_score = lm.score(X_test,y_test)
print(r2_score*100,'%')
```

```
57.6709013625585 %
```

In [20]:
```
sns.distplot((y_test-predictions), bins = 50)
```

```
C:\Users\sanja\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please
adapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[20]:
```
<AxesSubplot:xlabel='price', ylabel='Density'>
```




## KNN Classification Model using K=10

In [21]:
```
dataset = Housing[['sqft_living','bathrooms','bedrooms', 'grade','floors','view','price
```

In [22]:
```
X = dataset.iloc[:, :6]
y = dataset.iloc[:, 6]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0
```

In [23]:
```
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.fit_transform(X_test)
```

In [24]:
```
classifier = KNeighborsClassifier(n_neighbors=10)
classifier.fit(X_train, y_train)
```

Out[24]:
```
KNeighborsClassifier(n_neighbors=10)
```

In [25]:
```
y_pred = classifier.predict(X_test)
```

In [26]:
```
per_error = 100*(y_pred-y_test)/y_test
Knn10Error = per_error.mean()
```

## Average Accuracy Score for K=10

In [27]:
```
print(100+Knn10Error,'%')
```

```
72.03732430437984 %
```

## KNN Classification Model using K=5

In [28]:
```
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
```

Out[28]:
```
KNeighborsClassifier()
```

In [29]:
```
y_pred = classifier.predict(X_test)
```

In [30]:
```
per_error = 100*(y_pred-y_test)/y_test
Knn5Error = per_error.mean()
```

## Average Accuracy Score for K=5

In [31]:
```
print(100+Knn5Error,'%')
```

```
75.91960151848336 %
```

## K=5 gives better prediction accuracy than K=10

## Predicting prices of 2 houses using KNN (K=5) and comparing with actual price of house

In [32]:
```
per_error = 100*(y_pred-y_test)/y_test

df_prd_tst = pd.DataFrame({'Predicted Price':y_pred.astype('int64'), 'Actual Price':y_te
df_prd_tst.head(2)
```

Out[32]:

| | Predicted Price | Actual Price | % Error |
|---|---|---|---|
| **17384** | 277500 | 297000.0 | -6.565657 |
| **722** | 1450000 | 1578000.0 | -8.111534 |

## Lasso Regression Model

In [64]:
```
X = Housing[['sqft_living','bathrooms','bedrooms', 'grade','floors','view']]
y = Housing['price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
clf = linear_model.Lasso(alpha=0.1)
```

In [65]:
```
clf.fit(X_train, y_train)
```

Out[65]:
```
Lasso(alpha=0.1)
```

In [66]:
```
clfpredictions = clf.predict(X_test)
```

## Predicting Price of houses using Lasso Regression Model

In [67]:
```
per_error = 100*(clfpredictions-y_test)/y_test

df_prd_tst = pd.DataFrame({'Predicted Price':clfpredictions.astype('int64'), 'Actual Pri
df_prd_tst.head(2)
```

Out[67]:

| | Predicted Price | Actual Price | % Error |
|---|---|---|---|
| **17384** | 346470 | 297000.0 | 16.656819 |
| **722** | 1399923 | 1578000.0 | -11.284953 |

## Average Accuracy Score for Lasso Regression Model

In [68]:
```
predictions = clf.predict(X_test)
r2_score = clf.score(X_test,y_test)
print(r2_score*100,'%')
```

```
57.67089906135141 %
```

## After comparing the accuracy score for all the models used, KNN (K=5) has the highest accuracy, that is 75.92%

In [ ]:
```
```