```
# Importing required libraries
from google.colab import drive
import pandas as pd
import matplotlib.pyplot as plt
import os
import numpy as np
drive.mount('/content/grive')
```

```
Drive already mounted at /content/grive; to attempt to forcibly remount, call drive.mount("/content/grive", force_remount=True).
```

```
#cd grive/My\ Drive/2Year1Term/CS490DB/CS490DB-Project/data
```

```
data = pd.read_csv("kc_house_data.csv")
```

```
data.isnull().sum()
```

```
id               0
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
grade            0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
zipcode          0
lat              0
long             0
sqft_living15    0
sqft_lot15       0
dtype: int64
```

```
data.dtypes
```

```
id                int64
date             object
price           float64
bedrooms          int64
bathrooms       float64
sqft_living       int64
sqft_lot          int64
floors          float64
waterfront        int64
view              int64
condition         int64
grade             int64
sqft_above        int64
sqft_basement     int64
yr_built          int64
yr_renovated      int64
zipcode           int64
lat             float64
long            float64
sqft_living15     int64
sqft_lot15        int64
dtype: object
```

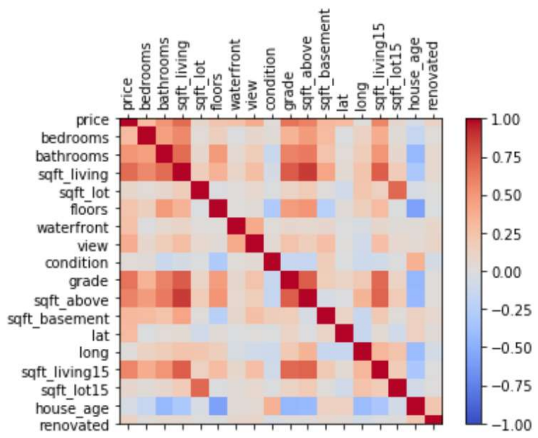
```
data['date'] = pd.to_datetime(data['date'])
data['bathrooms'] = data['bathrooms'].astype('int')
data['floors'] = data['floors'].astype('int')
data['price'] = data['price'].astype('int')
```

```
#getting signifficant data
data["house_age"] = data["date"].dt.year - data['yr_built']
data['renovated'] = data['yr_renovated'].apply(lambda yr: 0 if yr == 0 else 1)

data.drop('date', axis=1, inplace=True)
```

```
data.drop('yr_renovated', axis=1, inplace=True)
data.drop('yr_built', axis=1, inplace=True)
data.drop('zipcode', axis=1, inplace=True)

data.set_index(data['id'], inplace=True)
data.drop('id', axis=1, inplace=True)
data.head()
```

id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basem
7129300520	221900	3	1	1180	5650	1	0	0	3	7	1180	
6414100192	538000	3	2	2570	7242	2	0	0	3	7	2170	
5631500400	180000	2	1	770	10000	1	0	0	3	6	770	
2487200875	604000	4	3	1960	5000	1	0	0	5	7	1050	
1954400510	510000	3	2	1680	8080	1	0	0	3	8	1680	

```
corr = data.corr()
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(corr,cmap='coolwarm', vmin=-1, vmax=1)
fig.colorbar(cax)
ticks = np.arange(0,len(data.columns),1)
ax.set_xticks(ticks)
plt.xticks(rotation=90)
ax.set_yticks(ticks)
ax.set_xticklabels(data.columns)
ax.set_yticklabels(data.columns)
plt.show()
```




```
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor


from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

y = data['price']
X = data.drop('price', axis = 1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

model = RandomForestRegressor()
model.fit(X_train, y_train)

pred = model.predict(X_test)
```

```
pred = model.predict(X_test)
print("The R2 square value of Random Forest is :", r2_score(y_test, pred)*100)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of n_estimators will chan
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
The R2 square value of Lasso is : 83.20787946457914
```

```
model = LinearRegression()
model.fit(X_train, y_train)

pred = model.predict(X_test)
print("The R2 square value of Linear is :", r2_score(y_test, pred)*100)
```

```
The R2 square value of Linear is : 69.57877895978987
```

```
# Define model. Specify a number for random_state to ensure same results each run
model = DecisionTreeRegressor(random_state=1)
model.fit(X_train, y_train)

pred = model.predict(X_test)
print("The R2 square value of Linear is :", r2_score(y_test, pred)*100)
```

```
The R2 square value of Linear is : 74.01705600896047
```

```
from sklearn.linear_model import ElasticNet
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import GradientBoostingRegressor


model = ElasticNet()
model.fit(X_train, y_train)

pred = model.predict(X_test)
print("The R2 square value of Elastic is :", r2_score(y_test, pred)*100)
```

```
The R2 square value of Elastic is : 61.23281139311318
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/coordinate_descent.py:475: ConvergenceWarning: Objective did not conv
  positive)
```

```
model = KNeighborsRegressor()
model.fit(X_train, y_train)

pred = model.predict(X_test)
print("The R2 square value of Elastic is :", r2_score(y_test, pred)*100)
```

```
The R2 square value of Elastic is : 48.98028161459251
```

```
model = GradientBoostingRegressor()
model.fit(X_train, y_train)

pred = model.predict(X_test)
print("The R2 square value of Elastic is :", r2_score(y_test, pred)*100)
```

```
The R2 square value of Elastic is : 85.72645694697634
```

```
from sklearn.linear_model import LogisticRegression


from sklearn import linear_model
reg = linear_model.Ridge(alpha=.5)
reg.fit(X_train, y_train)

pred = model.predict(X_test)
print("The R2 square value of Ridge is :", r2_score(y_test, pred)*100)


from sklearn import svm
clf = svm.SVR()
clf.fit(X, y)
reg.fit(X_train, y_train)

pred = model.predict(X_test)
```

```
print("The R2 square value of SVM is :", r2_score(y_test, pred)*100)
```