Importing the libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Importing the datasets

```
df_housing = pd.read_csv('housing.csv')
df_housing.shape
```

```
(20640, 10)
```

```
type(df_housing)
```

```
pandas.core.frame.DataFrame
```

```
df_housing.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms
0	-122.23	37.88	41	880	129.0
1	-122.22	37.86	21	7099	1106.0
2	-122.24	37.85	52	1467	190.0
3	-122.25	37.85	52	1274	235.0
4	-122.25	37.85	52	1627	280.0

```
df_housing.isna().sum()
```

```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms      207
population            0
households            0
median_income         0
ocean_proximity       0
```

```
median_house_value      0
dtype: int64
```

```
df_housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  int64  
 3   total_rooms         20640 non-null  int64  
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  int64  
 6   households          20640 non-null  int64  
 7   median_income       20640 non-null  float64
 8   ocean_proximity     20640 non-null  object 
 9   median_house_value  20640 non-null  int64  
dtypes: float64(4), int64(5), object(1)
memory usage: 1.6+ MB
```

```
df_housing.describe().T
```

	count	mean	std	min	
longitude	20640.0	-119.569704	2.003532	-124.3500	-
latitude	20640.0	35.631861	2.135952	32.5400	
housing_median_age	20640.0	28.639486	12.585558	1.0000	
total_rooms	20640.0	2635.763081	2181.615252	2.0000	1
total_bedrooms	20433.0	537.870553	421.385070	1.0000	
population	20640.0	1425.476744	1132.462122	3.0000	
households	20640.0	499.539680	382.329753	1.0000	
median_income	20640.0	3.870671	1.899822	0.4999	

Fixing missing values, taking mean of 'total_bedrooms'

```
df_housing['total_bedrooms'].fillna(435,inplace=True)
```

```
df_housing.isna().sum()
```

```
longitude           0
latitude            0
```

```
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
ocean_proximity       0
median_house_value    0
dtype: int64
```

Encoding categorical columns

```
1 from sklearn.preprocessing import LabelEncoder
```

```
1 le = LabelEncoder()
```

```
1 df_housing['ocean_proximity'] = le.fit_transform(df_housing['ocean_proximity'])
```

```
1 df_housing.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	ho
0	-122.23	37.88	41	880	129.0	322	
1	-122.22	37.86	21	7099	1106.0	2401	
2	-122.24	37.85	52	1467	190.0	496	
3	-122.25	37.85	52	1274	235.0	558	
4	-122.25	37.85	52	1627	280.0	565	

Extracting X and y features

X values

```
1 X = df_housing.iloc[:,:9].values
2 X
```

```
array([[-1.2223e+02,  3.7880e+01,  4.1000e+01, ...,  1.2600e+02,
         8.3252e+00,  3.0000e+00],
       [-1.2222e+02,  3.7860e+01,  2.1000e+01, ...,  1.1380e+03,
         8.3014e+00,  3.0000e+00],
       [-1.2224e+02,  3.7850e+01,  5.2000e+01, ...,  1.7700e+02,
         7.2574e+00,  3.0000e+00],
       ...,
```

```
[-1.2122e+02,  3.9430e+01,  1.7000e+01, ...,  4.3300e+02,
  1.7000e+00,  1.0000e+00],
[-1.2132e+02,  3.9430e+01,  1.8000e+01, ...,  3.4900e+02,
  1.8672e+00,  1.0000e+00],
[-1.2124e+02,  3.9370e+01,  1.6000e+01, ...,  5.3000e+02,
  2.3886e+00,  1.0000e+00]])
```

y values

```
1 y = df_housing.iloc[:,-1].values
2 y = y.reshape(len(y),1)
3 y
```

```
array([[452600],
       [358500],
       [352100],
       ...,
       [ 92300],
       [ 84700],
       [ 89400]])
```

Splitting the dataset

```
1 from sklearn.model_selection import train_test_split
```

```
1 X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2)
```

```
1 print('X_train: ', X_train.shape)
2 print('X_test: ', X_test.shape)
3 print('y_train: ', y_train.shape)
4 print('y_test: ', y_test.shape)
```

```
X_train:  (16512, 9)
X_test:  (4128, 9)
y_train:  (16512, 1)
y_test:  (4128, 1)
```

Standardizing the data

```
1 from sklearn.preprocessing import StandardScaler
```

```
1 scaler = StandardScaler()
```

Linear Regression

```
1 from sklearn.linear_model import LinearRegression
```

```
1 linreg = LinearRegression()
```

```
1 linreg.fit(X_train,y_train)
```

```
    LinearRegression()
```

```
1 y_pred_linreg = linreg.predict(X_test)
```

```
1 from sklearn.metrics import mean_squared_error, r2_score
```

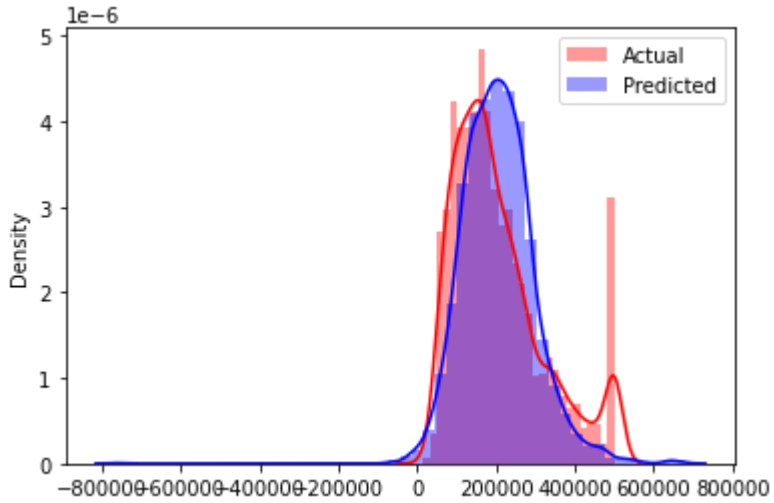
```
1 r2_score(y_pred=y_pred_linreg, y_true = y_test)
```

```
    0.6278917263635644
```

```
1 mean_squared_error(y_pred=y_pred_linreg, y_true = y_test)
```

```
    5046769733.634418
```

```
1 sns.distplot(y_test, color='red', label='Actual')
2 sns.distplot(y_pred_linreg, color='blue',label='Predicted')
3 plt.legend()
4 plt.show()
```



Decision Tree Regression

```
1 from sklearn.tree import DecisionTreeRegressor
```

```
dtregressor = DecisionTreeRegressor()
```

```
dtregressor.fit(X=X_train, y=y_train)
```

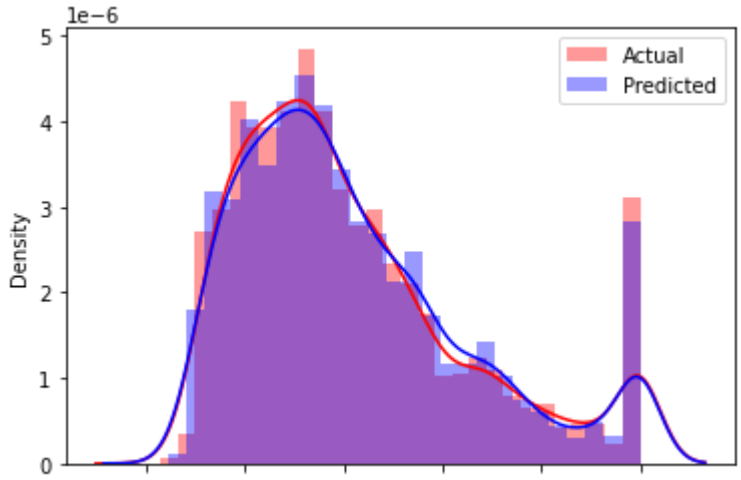
```
DecisionTreeRegressor()
```

```
y_pred_dtr = dtregressor.predict(X_test)
```

```
print('Mean squared error: ',mean_squared_error(y_pred=y_pred_dtr, y_true = y_test))
print('r2 score: ',r2_score(y_pred=y_pred_dtr, y_true = y_test))
```

```
Mean squared error:  4938335006.7286825
r2 score:  0.6358868323740925
```

```
sns.distplot(y_test, color='red', label='Actual')
sns.distplot(y_pred_dtr, color='blue', label='Predicted')
plt.legend()
plt.show()
```



Random Forest Regression

```
from sklearn.ensemble import RandomForestRegressor
```

```
rfrregressor = RandomForestRegressor(n_estimators = 100)
```

```
rfrregressor.fit(X_train, y_train)
```

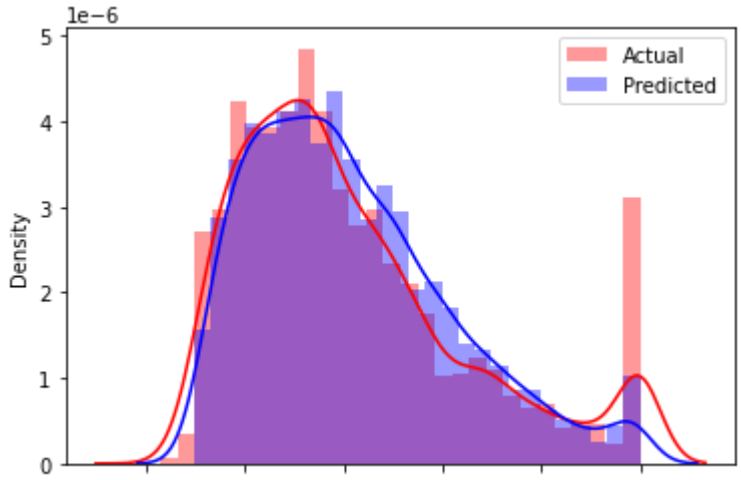
```
RandomForestRegressor()
```

```
y_pred_rfr = rfrregressor.predict(X_test)
```

```
print('Mean squared error: ',mean_squared_error(y_pred=y_pred_rfr, y_true = y_test))
print('r2 score: ',r2_score(y_pred=y_pred_rfr, y_true = y_test))
```

```
Mean squared error:  2468793791.857087
r2 score:  0.8179709706726166
```

```
sns.distplot(y_test, color='red',label='Actual')
sns.distplot(y_pred_rfr, color='blue',label='Predicted')
plt.legend()
plt.show()
```



Performing Linear Regression with just one variable, i.e medium_income

```
X = df_housing['median_income'].values
X = X.reshape(len(X),1)
X
```

```
array([[8.3252],
       [8.3014],
       [7.2574],
       ...,
       [1.7   ],
       [1.8672],
       [2.3886]])
```

```
y
```

```
array([[452600],
       [358500],
       [352100],
       ...,
       [ 92300],
```

```
          [ 84700],
          [ 89400]])
```

```
1 X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2)
```

```
1 scaler.fit_transform(X_train, y_train)
```

```
   array([[ 0.11076219],
          [-0.14682281],
          [-0.42640383],
          ...,
          [ 0.19848317],
          [ 0.57120524],
          [-0.17150256]])
```

```
1 print('X_train: ', X_train.shape)
2 print('X_test: ', X_test.shape)
3 print('y_train: ', y_train.shape)
4 print('y_test: ', y_test.shape)
```

```
   X_train:  (16512, 1)
   X_test:  (4128, 1)
   y_train:  (16512, 1)
   y_test:  (4128, 1)
```

```
1 linreg.fit(X_train,y_train)
```

```
   LinearRegression()
```

```
1 y_pred_onevarlr = linreg.predict(X_test)
```

```
1 print('Mean squared error: ',mean_squared_error(y_pred=y_pred_onevarlr, y_true = y_test))
2 print('r2 score: ',r2_score(y_pred=y_pred_onevarlr, y_true = y_test))
```

```
   Mean squared error:  6707032036.902624
   r2 score:  0.492381528867627
```

```
1 sns.distplot(y_test, color='red', label='Actual')
2 sns.distplot(y_pred_onevarlr, color='blue',label='Predicted')
3 plt.legend()
4 plt.show()
```