REPORT

Machine Learning

California Housing Price Prediction

Subodh Wasnik

Summary

Using the available dataset, we intend to develop a home price model to forecast median house values in California. Given all the other criteria, this algorithm ought to be able to learn from the data and forecast the median housing price in any district. We used python libraries like numpy, pandas, matplotlib, and sklearn. The different types of the model built are linear regression, Decision tree regression, Random Forest regression, Lasso Regression, Ridge regression, and ElasaticNet Regression.

Problem Statement and Tasks Performed

The project aims at building a model of housing prices to predict median house values in California using the provided dataset. This model should learn from the data and be able to predict the median housing price in any district, given all the other metrics.

Districts or block groups are the smallest geographical units for which the US Census Bureau publishes sample data (a block group typically has a population of 600 to 3,000 people). There are 20,640 districts in the project dataset.

Analysis Tasks to be performed:

1. Build a model of housing prices to predict median house values in California using the provided dataset.

2. Train the model to learn from the data to predict the median housing price in any district, given all the other metrics.

3. Predict housing prices based on median_income and plot the regression chart for it.

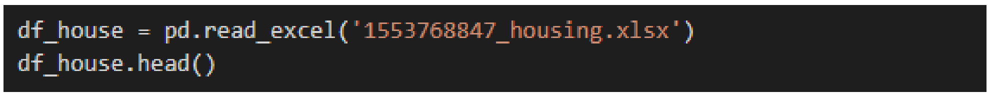
Dataset Description

Field	Description
longitude	(Signed numeric - float): Longitude value for the block in California, USA
latitude	(Numeric - float): Latitude value for the block in California, USA
housing_median_age	(Numeric - int): Median age of the house in the block
total_rooms	(Numeric - int): Count of the total number of rooms (excluding bedrooms) in all houses in the block
total_bedrooms	(Numeric - float: Count of the total number of bedrooms in all houses in the block
population	(Numeric - int): Count of the total number of populations in the block
households	(Numeric - int): Count of the total number of households in the block
median_income	(Numeric - float): Median of the total household income of all the houses in the block
ocean_proximity	(Numeric - categorical): Type of the landscape of the block [Unique Values: 'NEAR BAY', '<1H OCEAN', 'INLAND', 'NEAR OCEAN', 'ISLAND']
median_house_value	(Numeric - int): Median of the household prices of all the houses in the block

Programming

1. **Load the data**

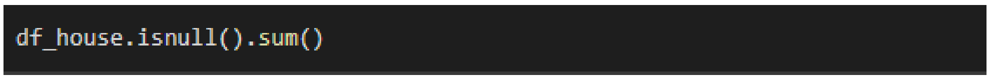
   Read the "**housing.csv**" file from the folder into the program.
   Print the first few rows of this data.



```
df_house = pd.read_excel('1553768847_housing.xlsx')
df_house.head()
```

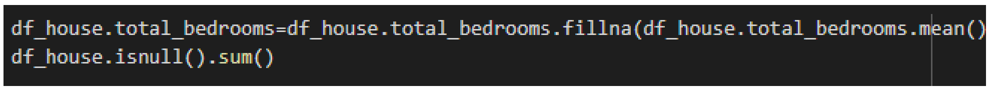
2. **Handle Missing Values**

   We first check what values are missing in the dataset.



```
df_house.isnull().sum()
```

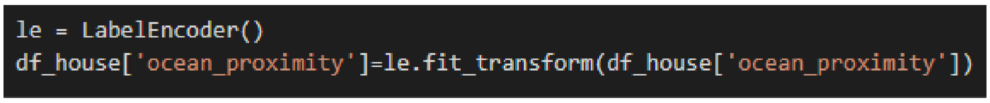
It is observed that there are 207 null values in Column total_bedrooms. We replace the null values with the mean and check for nulls again.



```
df_house.total_bedrooms=df_house.total_bedrooms.fillna(df_house.total_bedrooms.mean()
df_house.isnull().sum()
```

3. **Encode Categorical Data**

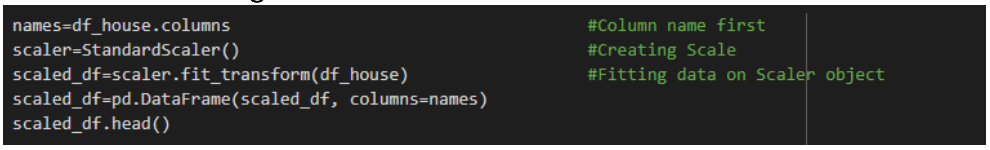
   We convert the categorical column in the dataset to numerical data.



```
le = LabelEncoder()
df_house['ocean_proximity']=le.fit_transform(df_house['ocean_proximity'])
```

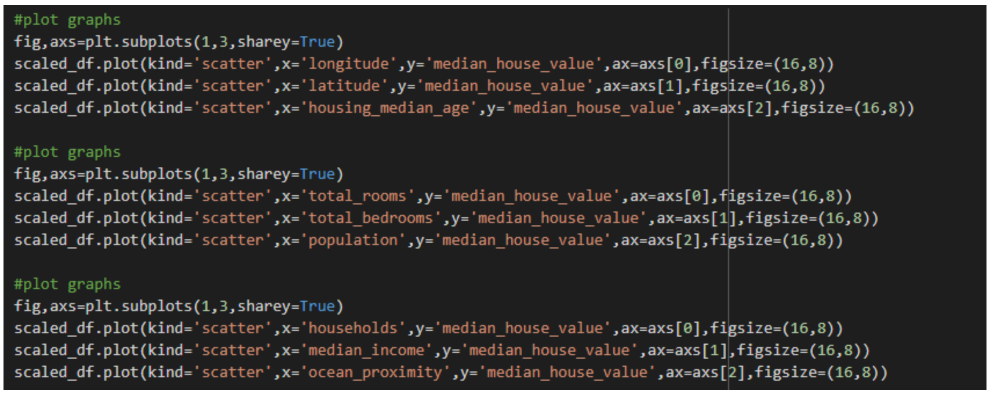
4. **Standardize data**

   Standardize training and test datasets.

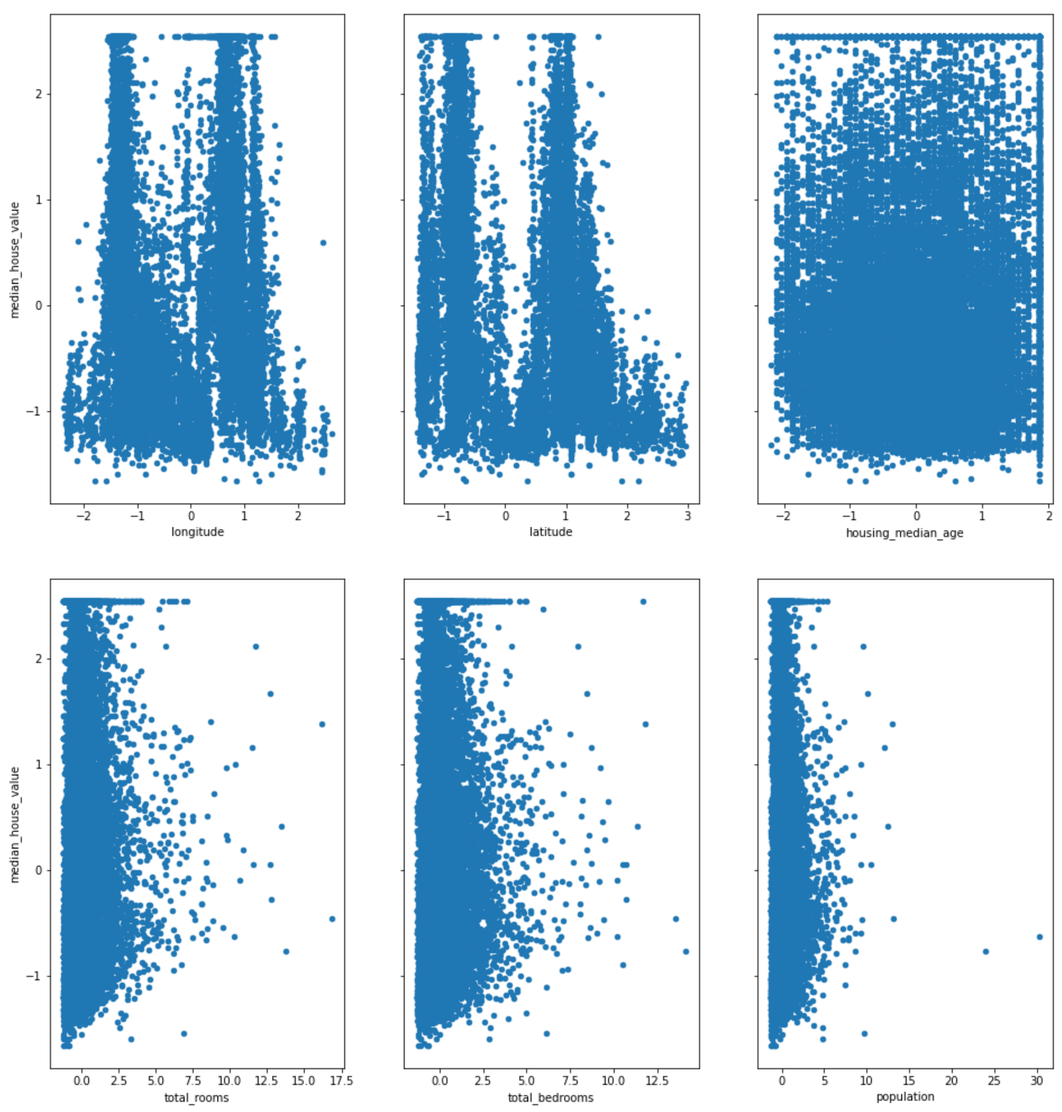


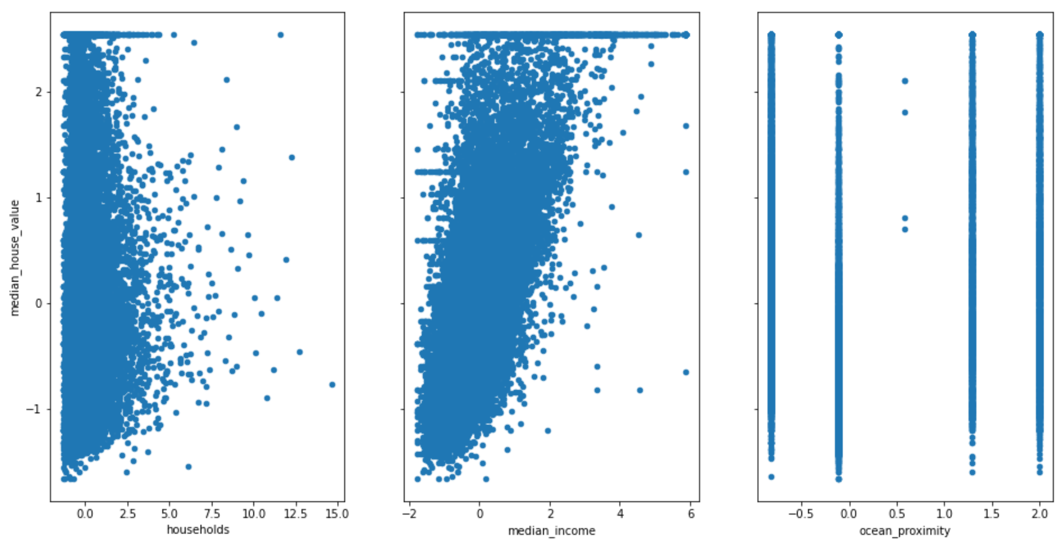
```
names=df_house.columns                                   #Column name first
scaler=StandardScaler()                                  #Creating Scale
scaled_df=scaler.fit_transform(df_house)                 #Fitting data on Scaler object
scaled_df=pd.DataFrame(scaled_df, columns=names)
scaled_df.head()
```

5. **Check for linearity**



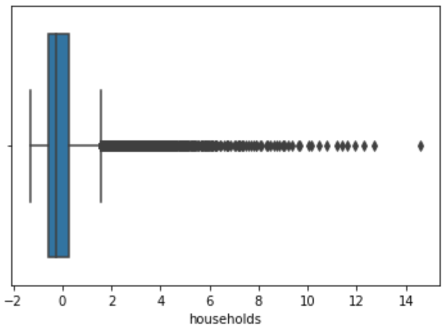
```
#plot graphs
fig,axs=plt.subplots(1,3,sharey=True)
scaled_df.plot(kind='scatter',x='longitude',y='median_house_value',ax=axs[0],figsize=(16,8))
scaled_df.plot(kind='scatter',x='latitude',y='median_house_value',ax=axs[1],figsize=(16,8))
scaled_df.plot(kind='scatter',x='housing_median_age',y='median_house_value',ax=axs[2],figsize=(16,8))

#plot graphs
fig,axs=plt.subplots(1,3,sharey=True)
scaled_df.plot(kind='scatter',x='total_rooms',y='median_house_value',ax=axs[0],figsize=(16,8))
scaled_df.plot(kind='scatter',x='total_bedrooms',y='median_house_value',ax=axs[1],figsize=(16,8))
scaled_df.plot(kind='scatter',x='population',y='median_house_value',ax=axs[2],figsize=(16,8))

#plot graphs
fig,axs=plt.subplots(1,3,sharey=True)
scaled_df.plot(kind='scatter',x='households',y='median_house_value',ax=axs[0],figsize=(16,8))
scaled_df.plot(kind='scatter',x='median_income',y='median_house_value',ax=axs[1],figsize=(16,8))
scaled_df.plot(kind='scatter',x='ocean_proximity',y='median_house_value',ax=axs[2],figsize=(16,8))
```

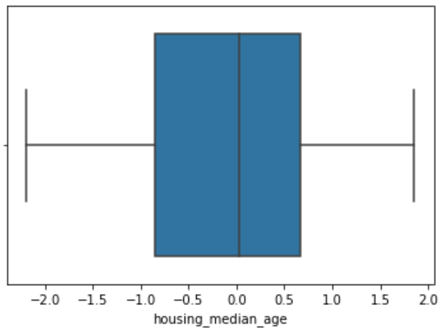


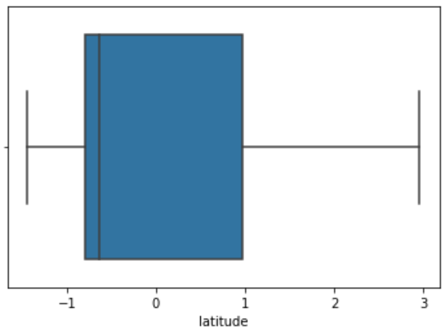


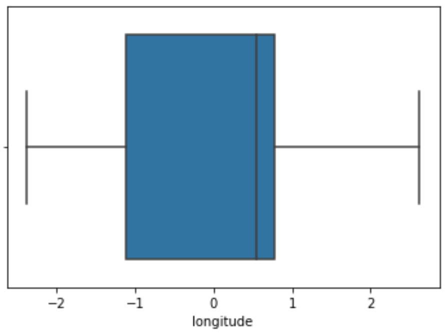
Check for Outliers

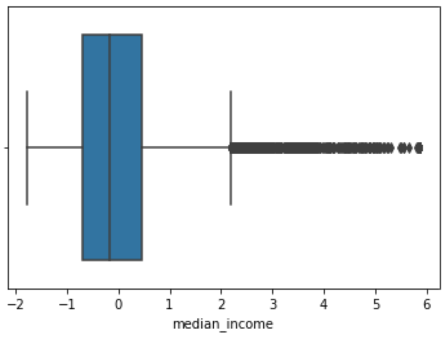
```python
for column in scaled_df:                                #Check for outliers
  plt.figure()
  sns.boxplot(x=scaled_df[column])
```

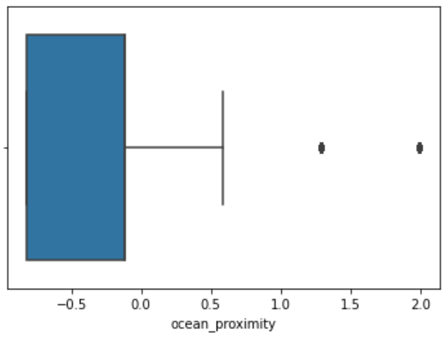


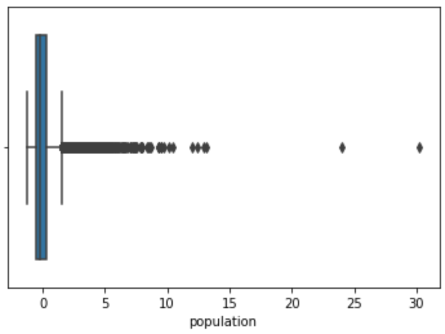


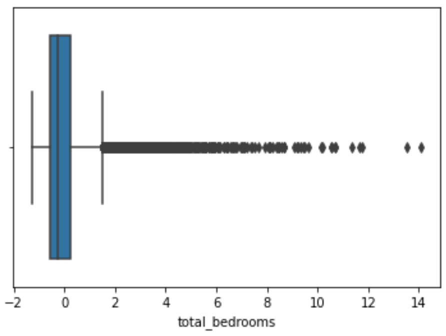


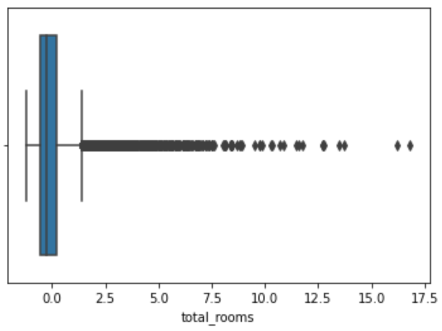


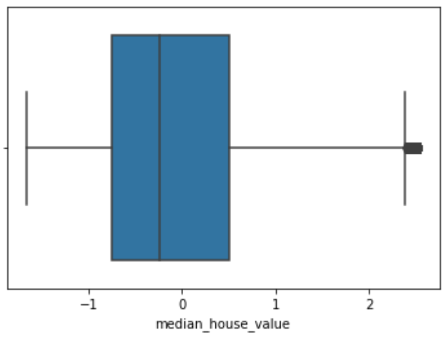




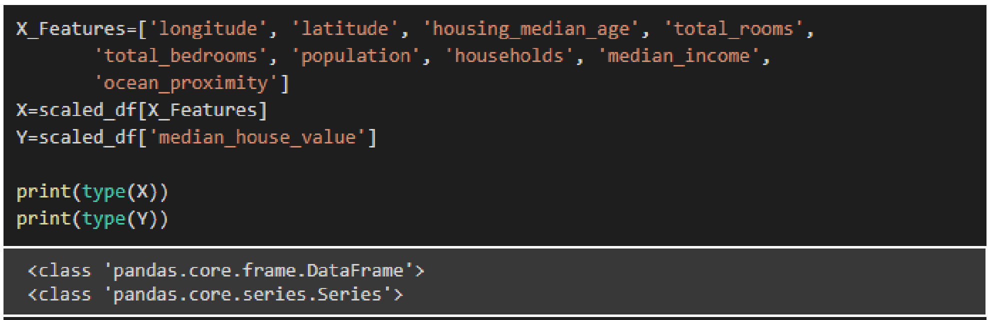






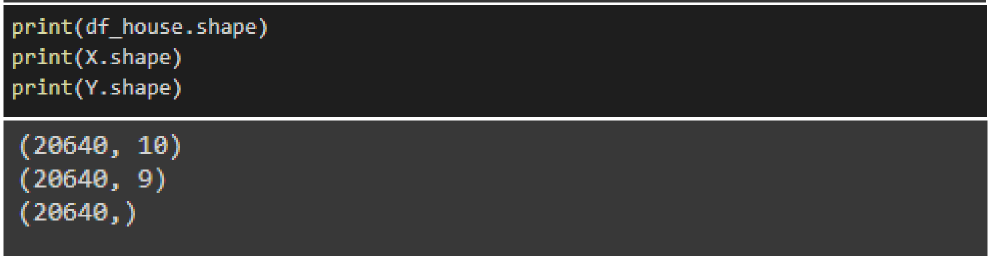


Extract input (X) and output (Y) from the dataset.



```
X_Features=['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'ocean_proximity']
X=scaled_df[X_Features]
Y=scaled_df['median_house_value']

print(type(X))
print(type(Y))
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
```

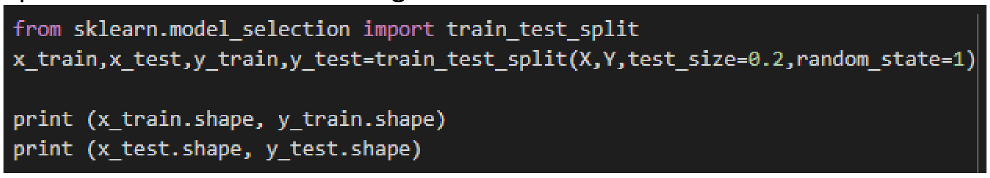


```
print(df_house.shape)
print(X.shape)
print(Y.shape)
```

```
(20640, 10)
(20640, 9)
(20640,)
```

6. Split the Dataset

Split the data into 80% training dataset and 20% test dataset.

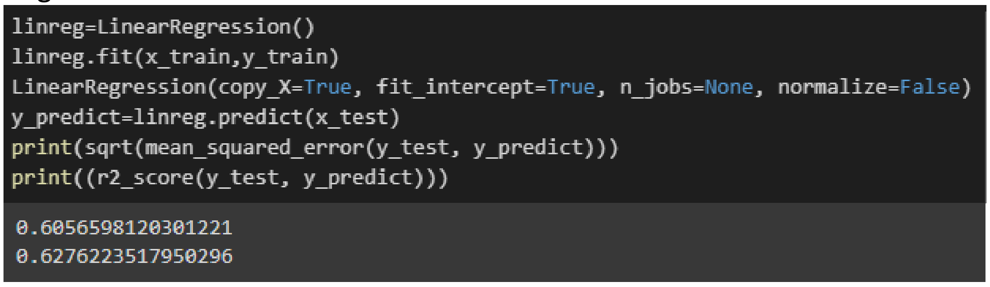


```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.2,random_state=1)

print (x_train.shape, y_train.shape)
print (x_test.shape, y_test.shape)
```

7. Applying the Regression Models

a. Linear Regression

We perform Linear Regression on training data. Predict output for test dataset using the fitted model and Print root mean squared error (RMSE) from Linear Regression.

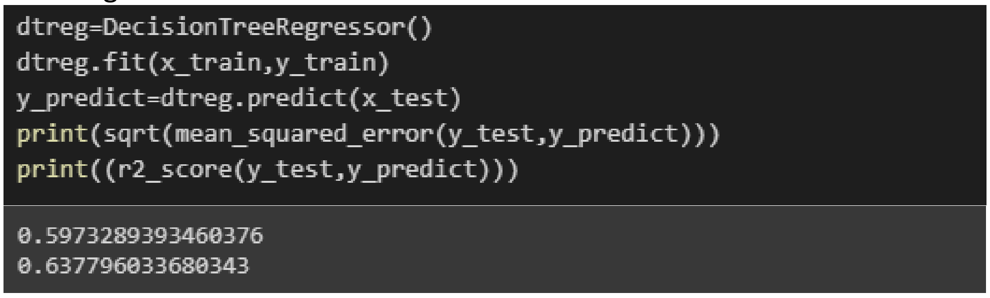


```
linreg=LinearRegression()
linreg.fit(x_train,y_train)
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
y_predict=linreg.predict(x_test)
print(sqrt(mean_squared_error(y_test, y_predict)))
print((r2_score(y_test, y_predict)))
```

```
0.6056598120301221
0.6276223517950296
```

b. Decision Tree Regression

We perform Decision Tree Regression on training data. Predict output for test dataset using the fitted model and print root mean squared error from Decision Tree Regression.



```
dtreg=DecisionTreeRegressor()
dtreg.fit(x_train,y_train)
y_predict=dtreg.predict(x_test)
print(sqrt(mean_squared_error(y_test,y_predict)))
print((r2_score(y_test,y_predict)))
```

```
0.5973289393460376
0.637796033680343
```

c. **Random Forrest Regression**
We perform Random Forest Regression on training data. Predict output for test dataset using the fitted model and print RMSE (root mean squared error) from Random Forest Regression.

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression
rfreg=RandomForestRegressor()
rfreg.fit(x_train,y_train)
y_predict=rfreg.predict(x_test)
print(sqrt(mean_squared_error(y_test,y_predict)))
print(r2_score(y_test,y_predict))
```

```
0.42551589952989105
0.8161948505025981
```

d. **Linear Regression with one independent variable**
Extract just the median_income column from the independent variables (from **X_train** and **X_test**). Perform Linear Regression to predict housing values based on **median_income**. And predicts output for the test dataset using the fitted model. Plot the fitted model for training data as well as for test data to check if the fitted model satisfies the test data.

```
x_train_Income=x_train[['median_income']]
x_test_Income=x_test[['median_income']]
print(x_train_Income.shape)
print(y_train.shape)
```

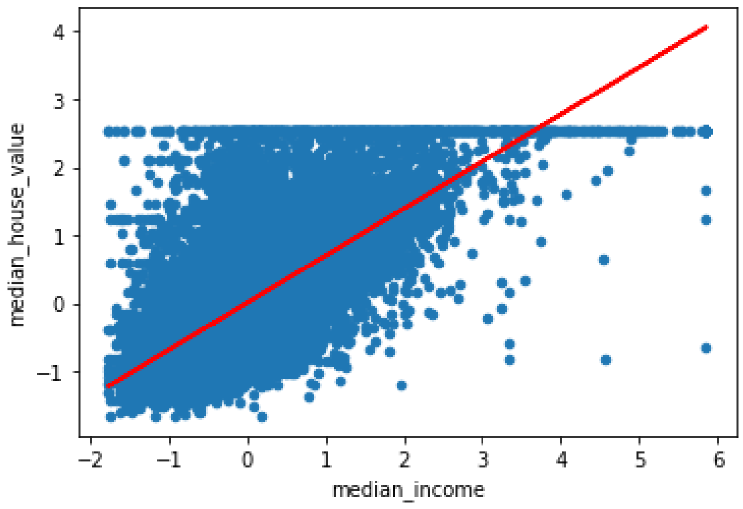
```
(16512, 1)
(16512,)
```

```
linreg=LinearRegression()
linreg.fit(x_train_Income,y_train)
y_predict = linreg.predict(x_test_Income)
#print intercept and coefficient of the linear equation
print(linreg.intercept_, linreg.coef_)
print(sqrt(mean_squared_error(y_test,y_predict)))
print((r2_score(y_test,y_predict)))
```

```
0.005623019866893164 [0.69238221]
0.7212595914243148
0.47190835934467734
```

Looking at the above values we can say that coefficient: a unit increase in median_income increases the median_house_value by 0.692 units.

```
scaled_df.plot(kind='scatter',x='median_income',y='median_house_value')
plt.plot(x_test_Income,y_predict,c='red',linewidth=2)
```


median_house_value
median_income