**Simple Linear Rergression on Real Estate Dataset**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

estate\_data = pd.read\_excel("Linear Regression.xlsx", sheet\_name= "Linear Regression")

estate\_data.head(2)

Out[130]:

price sqft\_living bedrooms bathrooms floors

0 221900 1180 3 1.00 1.0

1 538000 2570 3 2.25 2.0

estate\_data.isnull().sum()

Out[131]:

price 0

sqft\_living 0

bedrooms 0

bathrooms 0

floors 0

dtype: int64

**Model 1: price vs sqft\_living**

data = estate\_data[["price", "sqft\_living"]]

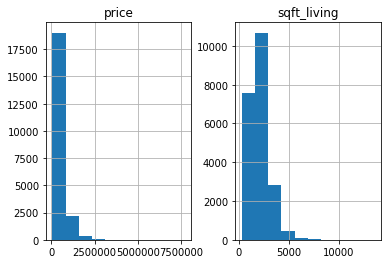
data.hist()

Out[133]:

array([[<matplotlib.axes.\_subplots.AxesSubplot object at 0x0000020D71F53A08>,

<matplotlib.axes.\_subplots.AxesSubplot object at 0x0000020D728AFB88>]],

dtype=object)



data.corr()

Out[134]:

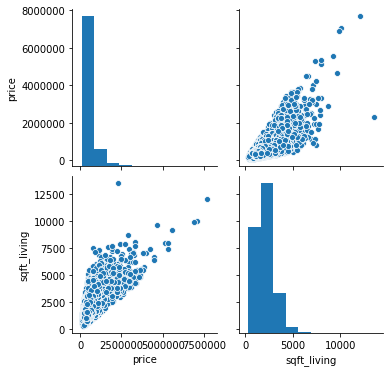
price sqft\_living

price 1.000000 0.702035

sqft\_living 0.702035 1.000000

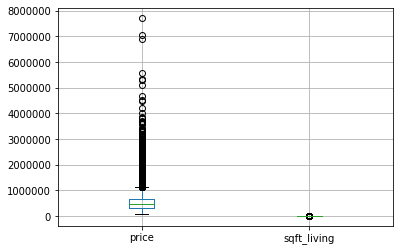
sns.pairplot(data)

Out[135]: <seaborn.axisgrid.PairGrid at 0x20d72605548>



data.boxplot()

Out[136]: <matplotlib.axes.\_subplots.AxesSubplot at 0x20d72a56c48>



Y = data.iloc[:,:1]

X = data.iloc[:,1:]

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 =2)

from sklearn.linear\_model import LinearRegression

lin\_reg = LinearRegression()

lin\_reg.fit(X\_train,y\_train)

Out[144]: LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=None, normalize=False)

print(lin\_reg.coef\_, lin\_reg.intercept\_)

**[[280.67382569]] [-42568.70358496]**

from sklearn.metrics import mean\_squared\_error,r2\_score

y\_pred= lin\_reg.predict(X\_test)

RMSE = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

r2 = r2\_score(y\_test,y\_pred)

print("RMSE:",RMSE)

**RMSE: 263380.00189817196**

print("r2:", r2)

**r2: 0.5031163723285275**

estate\_data.columns

Out[152]: Index(['price', 'sqft\_living', 'bedrooms', 'bathrooms', 'floors'], dtype='object')

**Model 2: price vs 'bedrooms'**

data = estate\_data[['price','bedrooms']]

data.corr()

Out[154]:

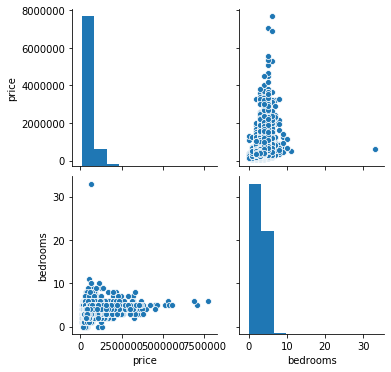
price bedrooms

price 1.00000 0.30835

bedrooms 0.30835 1.00000

sns.pairplot(data)

Out[155]: <seaborn.axisgrid.PairGrid at 0x20d72baa208>



Y = data.iloc[:,:1]

X = data.iloc[:,1:]

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,Y, test\_size = 0.2, random\_state =2)

lin\_reg.fit(X\_train,y\_train)

Out[159]: LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=None, normalize=False)

print(lin\_reg.coef\_, lin\_reg.intercept\_)

**[[118660.62797868]] [139952.87593386]**

y\_pred= lin\_reg.predict(X\_test)

RMSE = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

r2 = r2\_score(y\_test,y\_pred)

print("RMSE:",RMSE)

**RMSE: 352717.9654187645**

print("r2:", r2)

**r2: 0.10886345250291574**

**Model 3: price vs 'bathrooms'**

data = estate\_data[['price','bathrooms']]

data.corr()

Out[167]:

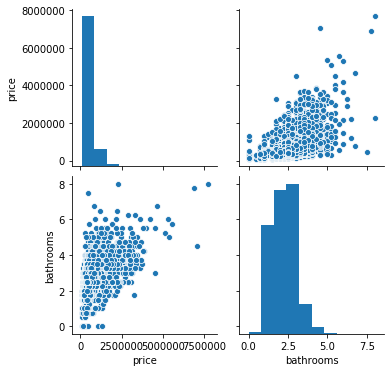
price bathrooms

price 1.000000 0.525138

bathrooms 0.525138 1.000000

sns.pairplot(data)

Out[168]: <seaborn.axisgrid.PairGrid at 0x20d72db3988>



Y = data.iloc[:,:1]

X = data.iloc[:,1:]

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,Y, test\_size = 0.2, random\_state =2)

lin\_reg.fit(X\_train,y\_train)

Out[172]: LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=None, normalize=False)

print(lin\_reg.coef\_, lin\_reg.intercept\_)

**[[249143.95803858]] [13073.99575289]**

y\_pred = lin\_reg.predict(X\_test)

RMSE = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

r2 = r2\_score(y\_test,y\_pred)

print("RMSE:",RMSE)

**RMSE: 316774.90190998075**

print("r2:", r2)

**r2: 0.28122887124177365**

**Model 4: price vs 'floors'**

data = estate\_data[['price','floors']]

data.corr()

Out[180]:

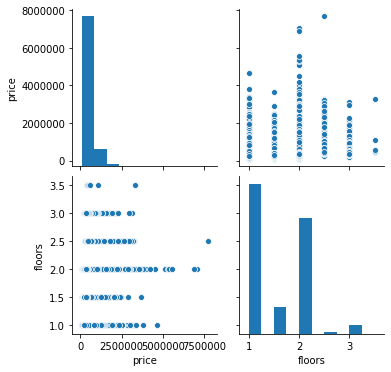
price floors

price 1.000000 0.256794

floors 0.256794 1.000000

sns.pairplot(data)

Out[181]: <seaborn.axisgrid.PairGrid at 0x20d727e9188>



Y = data.iloc[:,:1]

X = data.iloc[:,1:]

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,Y, test\_size = 0.2, random\_state =2)

lin\_reg.fit(X\_train,y\_train)

Out[185]: LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=None, normalize=False)

print(lin\_reg.coef\_, lin\_reg.intercept\_)

**[[171376.44562902]] [283309.93245029]**

y\_pred = lin\_reg.predict(X\_test)

RMSE = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

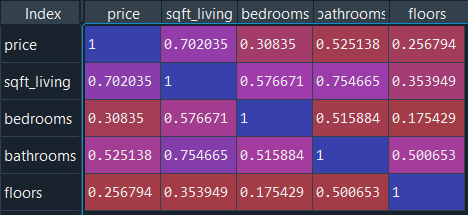
r2 = r2\_score(y\_test,y\_pred)

print("RMSE:",RMSE)

**RMSE: 359677.77234107786**

print("r2:", r2)

**r2: 0.0733487976687478**



**Inference:**

|  |  |  |  |
| --- | --- | --- | --- |
| **DV** | **IDV** | **RMSE** | **R2** |
| price | sqft\_living | 263380.0019 | 0.503116 |
| price | bedrooms | 352717.9654 | 0.108863 |
| price | bathrooms | 316774.9019 | 0.281229 |
| price | floors | 359677.7723 | 0.073349 |

Large RMSE values are observed in all the models as there many outliers in the target(price) feature.

Also the model accuracy is low for the price vs bedrooms , price vs floors as the have a low positive correlation

The model accuracy for price vs sqft\_living, price vs bathrooms is higher than the other two models as the have moderate positive correlation.

These models are not efficient enough to accurately predict the price.