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"]]
```

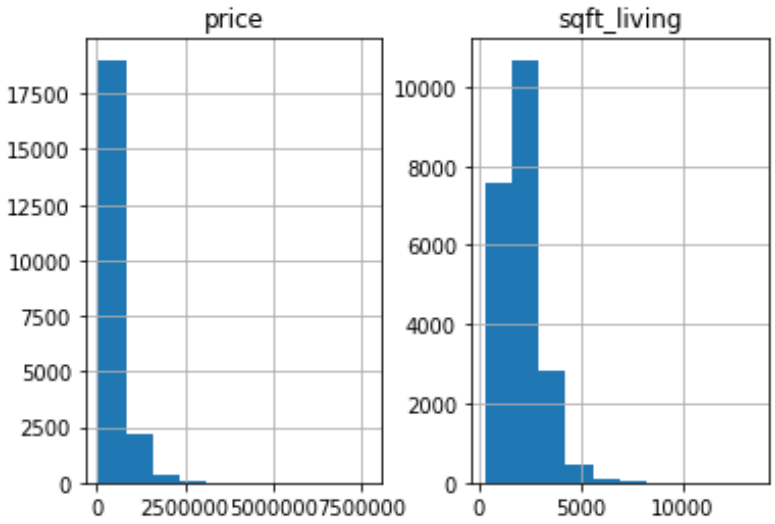
data.hist()

Out[133]:

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

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

dtype=object)



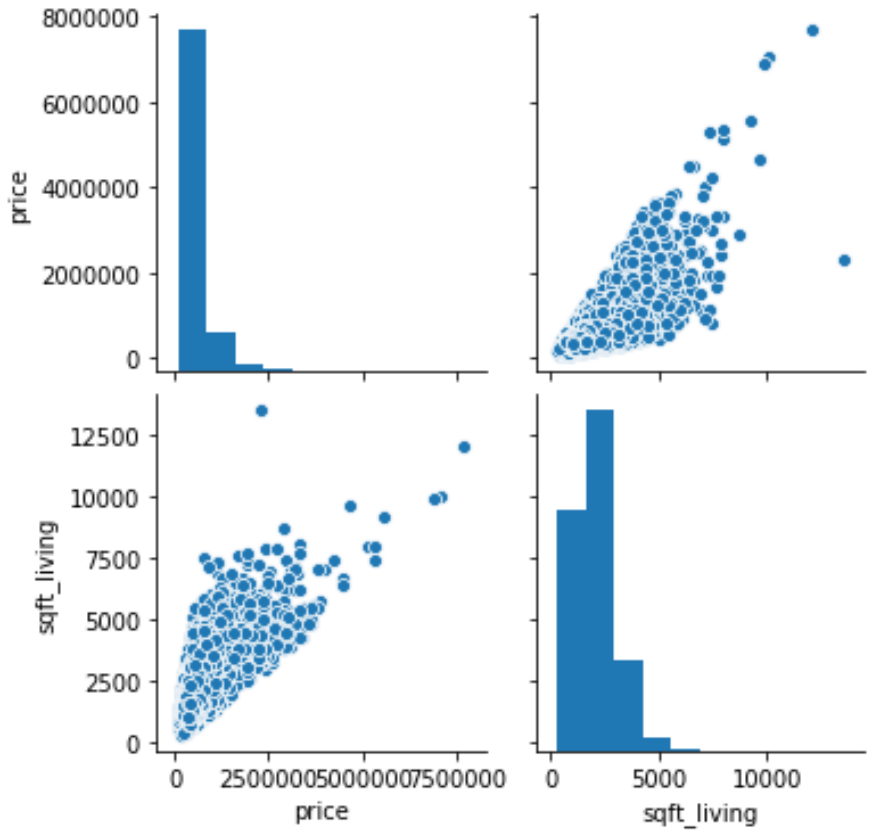
data.corr()

Out[134]:

	price	sqft_living
price	1.000000	0.702035
sqft_living	0.702035	1.000000

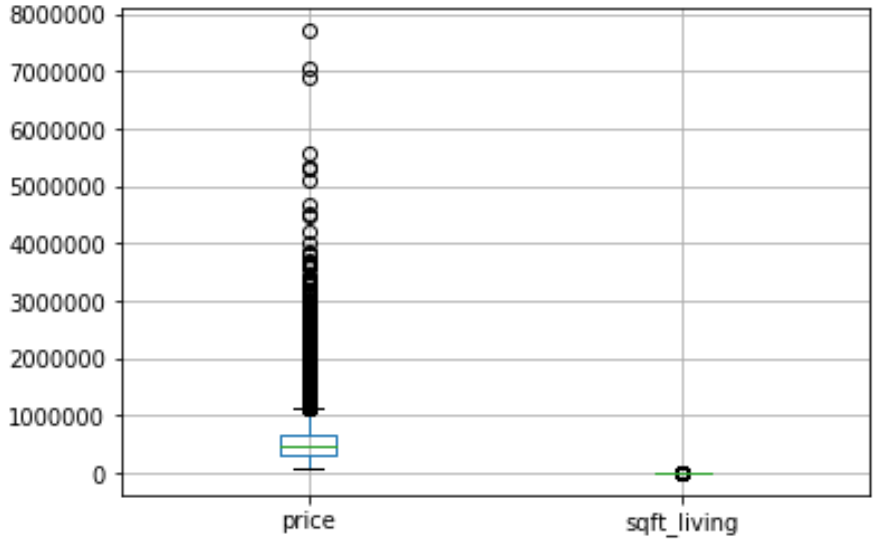
sns.pairplot(data)

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']]

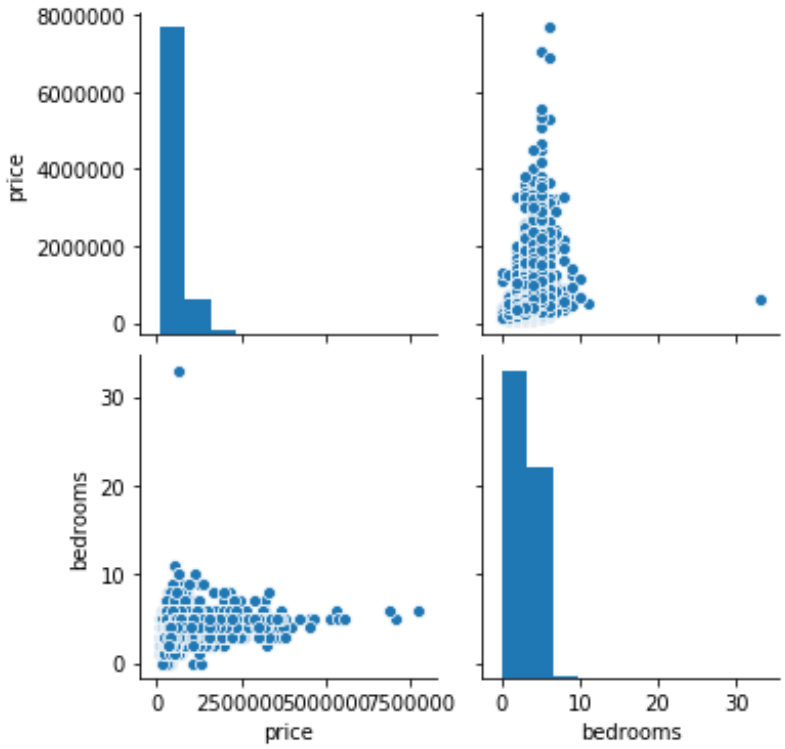
data.corr()

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

<u>Model 3: price vs 'bathrooms'</u>

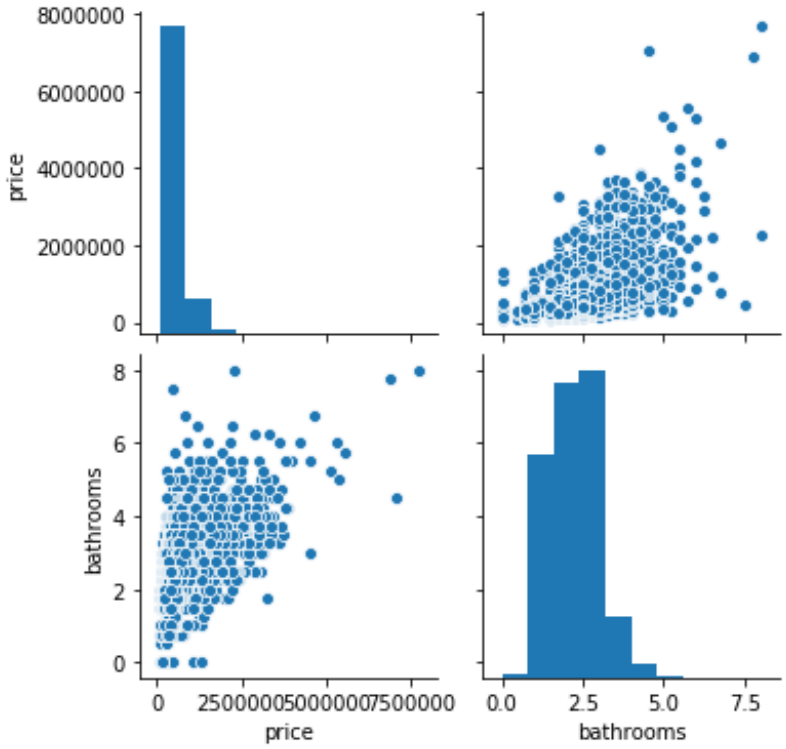
```
data = estate_data[['price','bathrooms']]

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']]

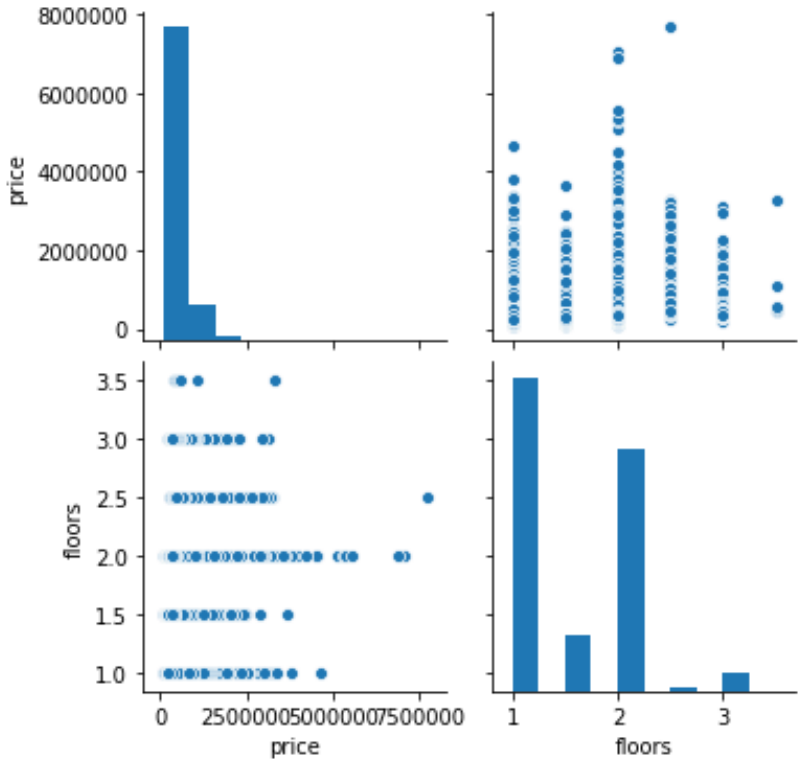
data.corr()

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

Index	price	sqft_living	bedrooms	bathrooms	floors
price	1	0.702035	0.30835	0.525138	0.256794
sqft_living	0.702035	1	0.576671	0.754665	0.353949
bedrooms	0.30835	0.576671	1	0.515884	0.175429
bathrooms	0.525138	0.754665	0.515884	1	0.500653
floors	0.256794	0.353949	0.175429	0.500653	1

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.