2 180 24

[2]:
```
from glob import glob

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
```

[3]:
```
#Prepare Data
#import data
```

[4]:
```
df = pd.read_csv(r"C:\Users\LENOVO\Desktop\kaggle practice data\real_estate_dataset.csv")
```

[5]:
```
df.shape
```

[5]:
```
(500, 12)
```

[6]:
```
df.info()
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   ID                  500 non-null    int64
 1   Square_Feet         500 non-null    float64
 2   Num_Bedrooms        500 non-null    int64
 3   Num_Bathrooms       500 non-null    int64
 4   Num_Floors          500 non-null    int64
 5   Year_Built          500 non-null    int64
 6   Has_Garden          500 non-null    int64
 7   Has_Pool            500 non-null    int64
 8   Garage_Size         500 non-null    int64
 9   Location_Score      500 non-null    float64
 10  Distance_to_Center  500 non-null    float64
 11  Price               500 non-null    float64
dtypes: float64(4), int64(8)
memory usage: 47.0 KB
```

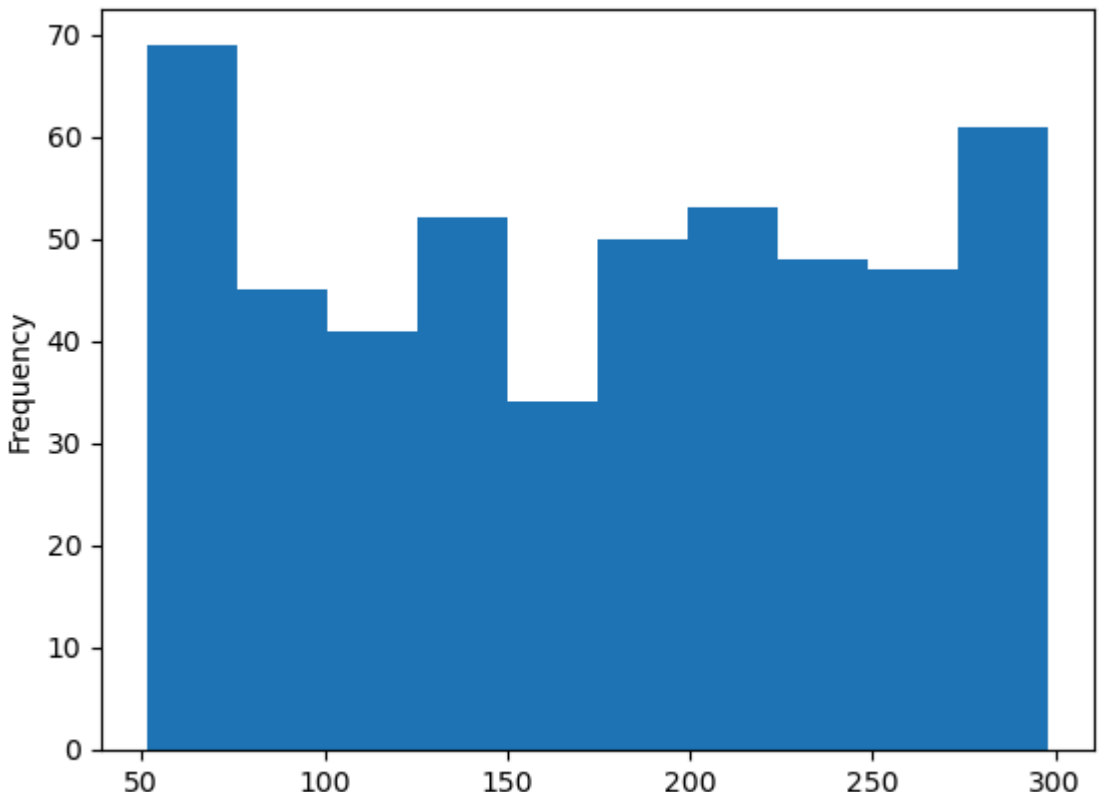
[6]:

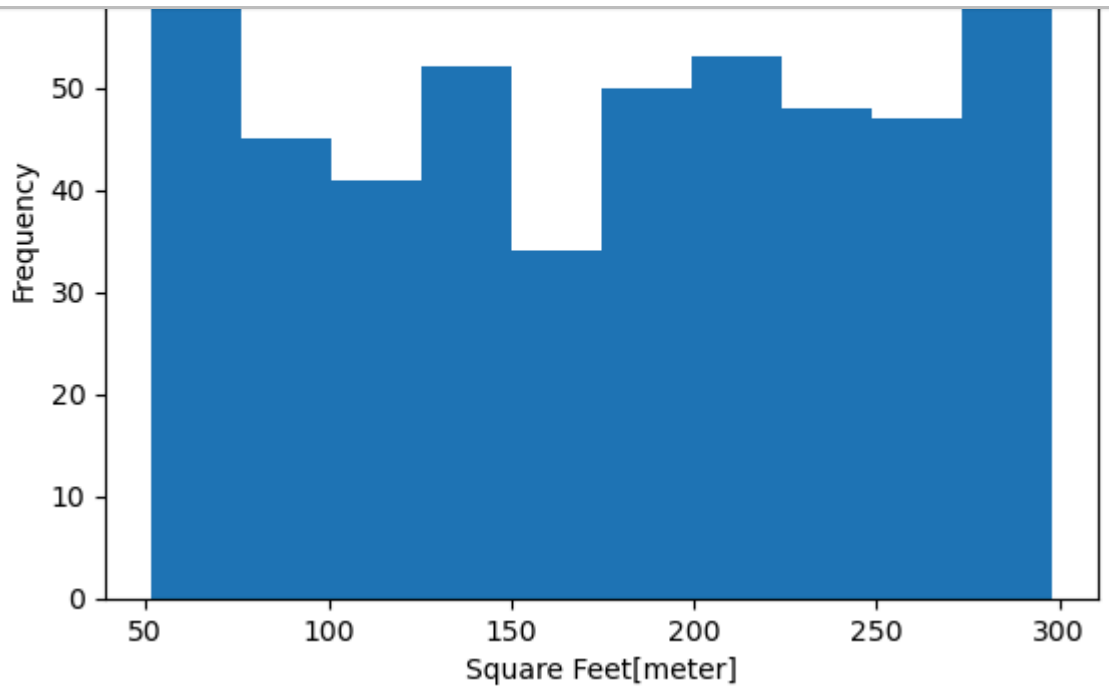
	ID	Square_Feet	Num_Bedrooms	Num_Bathrooms	Num_Floors	Year_Built	Has_Garden	Has_Pool	Garage_Size	Location_Score	Distance_t
0	1	143.635030	1	3	3	1967	1	1	48	8.297631	
1	2	287.678577	1	2	1	1949	0	1	37	6.061466	
2	3	232.998485	1	3	2	1923	1	0	14	2.911442	
3	4	199.664621	5	2	2	1918	0	0	17	2.070949	
4	5	89.004660	4	3	3	1999	1	0	34	1.523278	

[7]:
```
#data Explore
#Histogram
```

[8]:
```
plt.hist(df["Square_Feet"])
plt.xlabel("Square Feet[meter]")
plt.ylabel("Frequency")
plt.title("Distribution of Apartment size")
```

[8]:
```
Text(0.5, 1.0, 'Distribution of Apartment size')
```





[9]:
```
#Looking at the histogram above we can see the plot has no outliers
```

[10]:
```
df.describe()["Square_Feet"]
```

[10]:
```
count    500.000000
mean     174.640428
std       74.672102
min       51.265396
25%      110.319923
50%      178.290937
75%      239.031220
max      298.241199
Name: Square_Feet, dtype: float64
```

[11]:
```
#summary statistics also confirm no outliers in the "Square Feet column"
```

[12]:
```
#Scatter plot
```

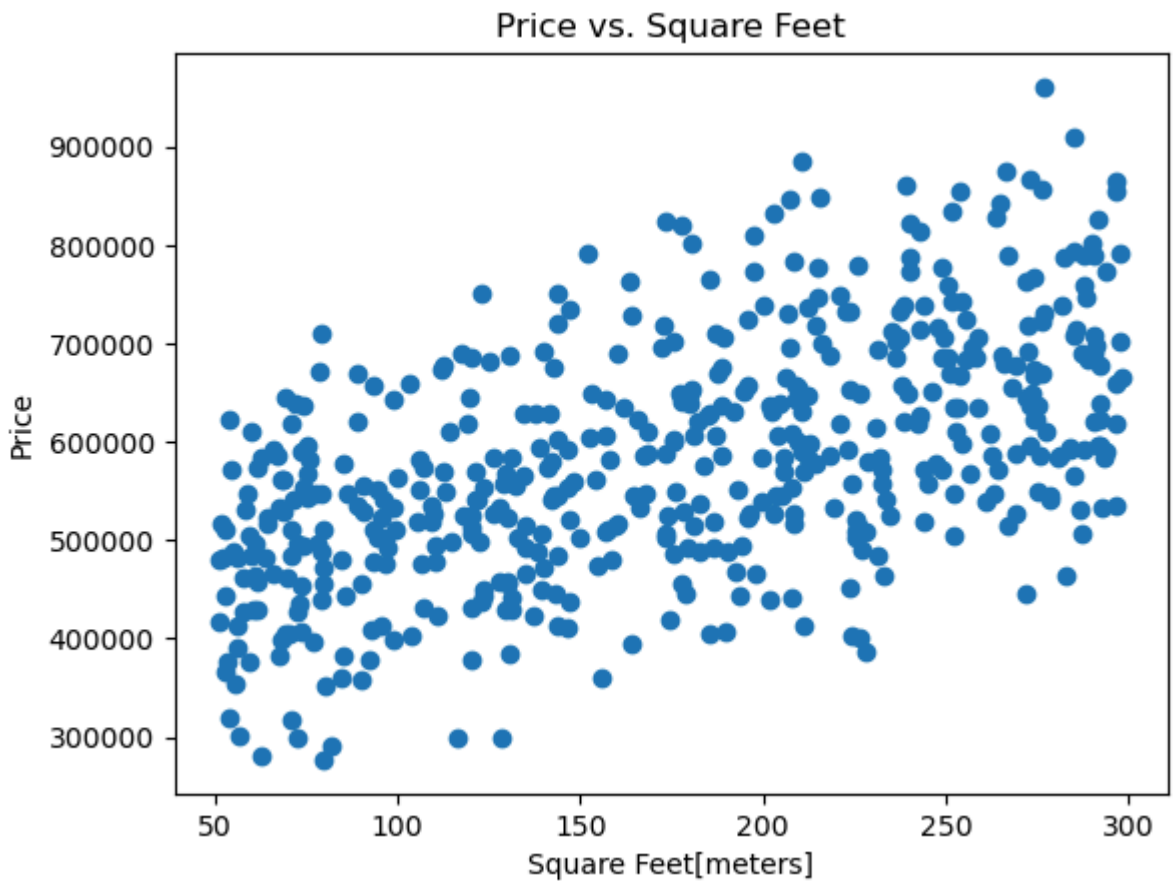
[13]:
```
plt.scatter(x = df["Square_Feet"], y = df["Price"])
plt.xlabel("Square Feet[meters]")
plt.ylabel("Price")
plt.title("Price vs. Square Feet");
```

jupyter real estate data Price vs Square Feet Last Checkpoint: 3 seconds ago

File Edit View Run Kernel Settings Help

Code JupyterLab Pytho

[13]:
```
plt.scatter(x = df["Square_Feet"], y = df["Price"])
plt.xlabel("Square Feet[meters]")
plt.ylabel("Price")
plt.title("Price vs. Square Feet");
```



[14]:
```
a moderate positive correlation between  the price and the size of the apartment, meaning, if there is an increase in the square f
```

[15]:
```
p_correlation = df["Square_Feet"].corr(df["Price"])
p_correlation
```

[15]:
```
0.5586036660836262
```

[16]:
```
#Split
#seprating feature matrix X_train and target vector y_train from our dataset
```

[16]:
```
#Split
#seprating feature matrix X_train and target vector y_train from our dataset
```

[17]:
```
features = ["Square_Feet"]
X_train = df[features]
X_train.head()
```

[17]:

	Square_Feet
0	143.635030
1	287.678577
2	232.998485
3	199.664621
4	89.004660

[18]:
```
target = "Price"
y_train = df[target]
y_train.head()
```

[18]:
```
0    602134.816747
1    591425.135386
2    464478.696880
3    583105.655996
4    619879.142523
Name: Price, dtype: float64
```

[19]:
```
#Model Building
#Baseline
```

[20]:
```
y_mean = y_train.mean()
y_mean
```

[20]:
```
582209.6295285609
```

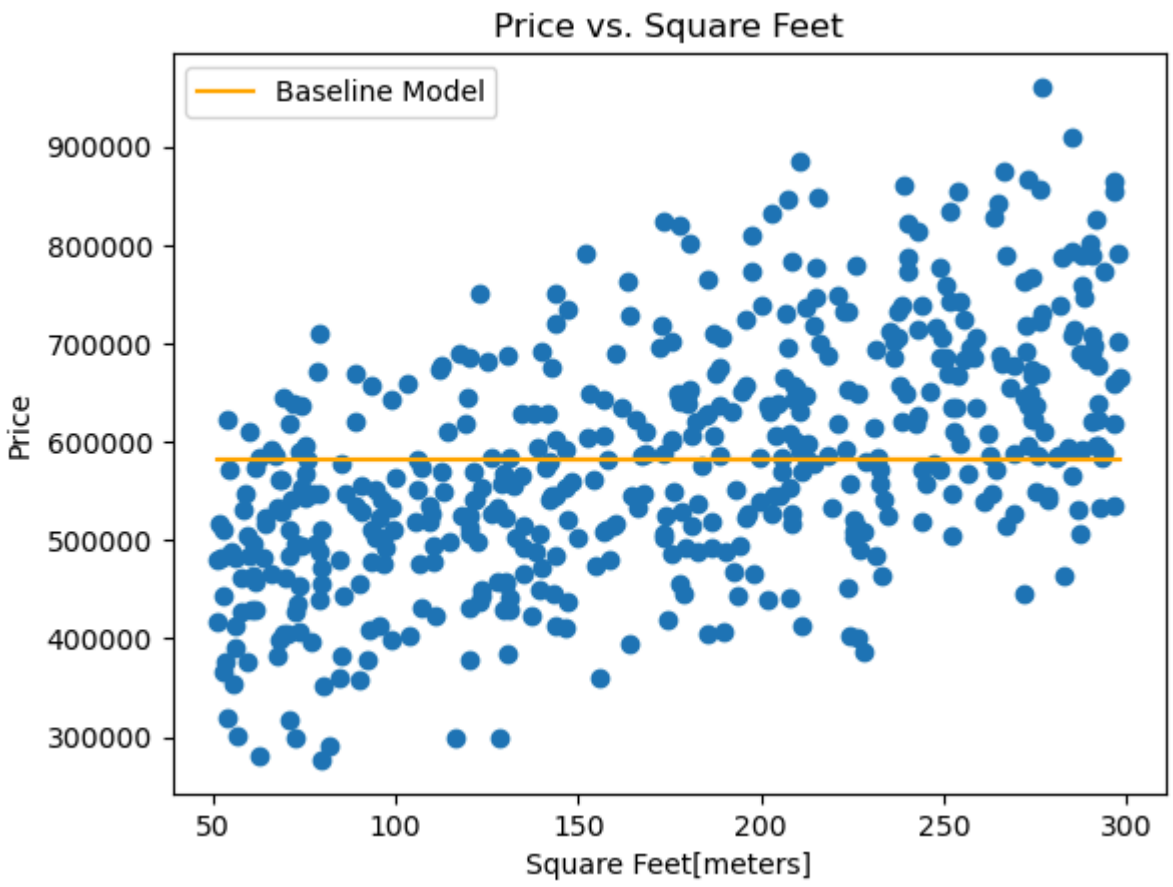
[21]:
```
y_pred_baseline = [y_mean]*len(y_train)
y_pred_baseline[:5]
```

[21]:
```
[582209.6295285609,
 582209.6295285609,
 582209.6295285609,
 582209.6295285609,
```

[21]:
```
y_pred_baseline = [y_mean]*len(y_train)
y_pred_baseline[:5]
```

[21]:
```
[582209.6295285609,
 582209.6295285609,
 582209.6295285609,
 582209.6295285609,
 582209.6295285609]
```

[22]:
```
plt.plot(X_train.values, y_pred_baseline,color="orange", label = "Baseline Model")
plt.scatter(X_train, y_train)
plt.xlabel("Square Feet[meters]")
plt.ylabel("Price")
plt.title("Price vs. Square Feet")
plt.legend();
```



[ ]:

[23]:
```
mae_baseline = mean_absolute_error(y_train,y_pred_baseline)
print("Mean Price:", round(y_mean,2))
print("Baseline MAE:", round(mae_baseline,2))
```

```
Mean Price: 582209.63
Baseline MAE: 97778.29
```

[24]:
```
#The information above tells us that if we always predicted that our price would be 582209.63, our prediction would be off by an
```

[25]:
```
#iterate
```

[26]:
```
model = LinearRegression()
```

[27]:
```
model.fit(X_train,y_train)
```

[27]:
```
LinearRegression()
```

[28]:
```
#Evaluate
```

[29]:
```
y_pred_training = model.predict(X_train)
y_pred_training[:5]
```

[29]:
```
array([553849.07789484, 685605.30961363, 635589.58231844, 605099.1892585 ,
       503878.83076167])
```

[30]:
```
#Evaluation
```

[31]:
```
mae_training = mean_absolute_error(y_train,y_pred_training)
print("Training MAE:", round(mae_training,2))
```

```
Training MAE: 81415.24
```

[32]:
```
#wow! Our model beat the baseline by 16000, that's a good indicator that it will help predict prices
```

[33]:
```
#Communication Results
```

[34]:
```
#y = mx + b, where y = price, m = coefficient, x = square feet, b = intercept
```

[35]:
```
intercept = round(model.intercept_,2)
```

[29]:
```
y_pred_training = model.predict(X_train)
y_pred_training[:5]
```

[29]:
```
array([553849.07789484, 685605.30961363, 635589.58231844, 605099.1892585 ,
       503878.83076167])
```

[30]:
```
#Evaluation
```

[31]:
```
mae_training = mean_absolute_error(y_train,y_pred_training)
print("Training MAE:", round(mae_training,2))
```

```
Training MAE: 81415.24
```

[32]:
```
#wow! Our model beat the baseline by 16000, that's a good indicator that it will help predict prices
```

[33]:
```
#Communication Results
```

[34]:
```
#y = mx + b, where y = price, m = coefficient, x = square feet, b = intercept
```

[35]:
```
intercept = round(model.intercept_,2)
print("Model Intercept:", intercept)
```

```
Model Intercept: 422466.52
```

[36]:
```
coefficient = round(model.coef_[0],2)
print("Model coefficient for Square_Feet:", coefficient)
```

```
Model coefficient for Square_Feet: 914.7
```

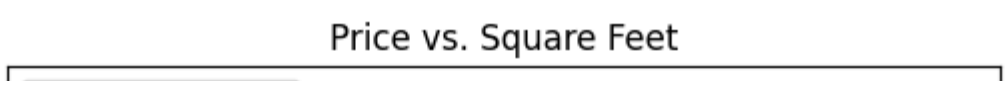
[37]:
```
#From above, for every square feet meter, we add to our apartment a model predict an additinal amount of 914.7 in price
```

[38]:
```
#Model Equation
```

[39]:
```
print(f"Price = {intercept} + {coefficient}* Square_Feet")
```

```
Price = 422466.52 + 914.7* Square_Feet
```

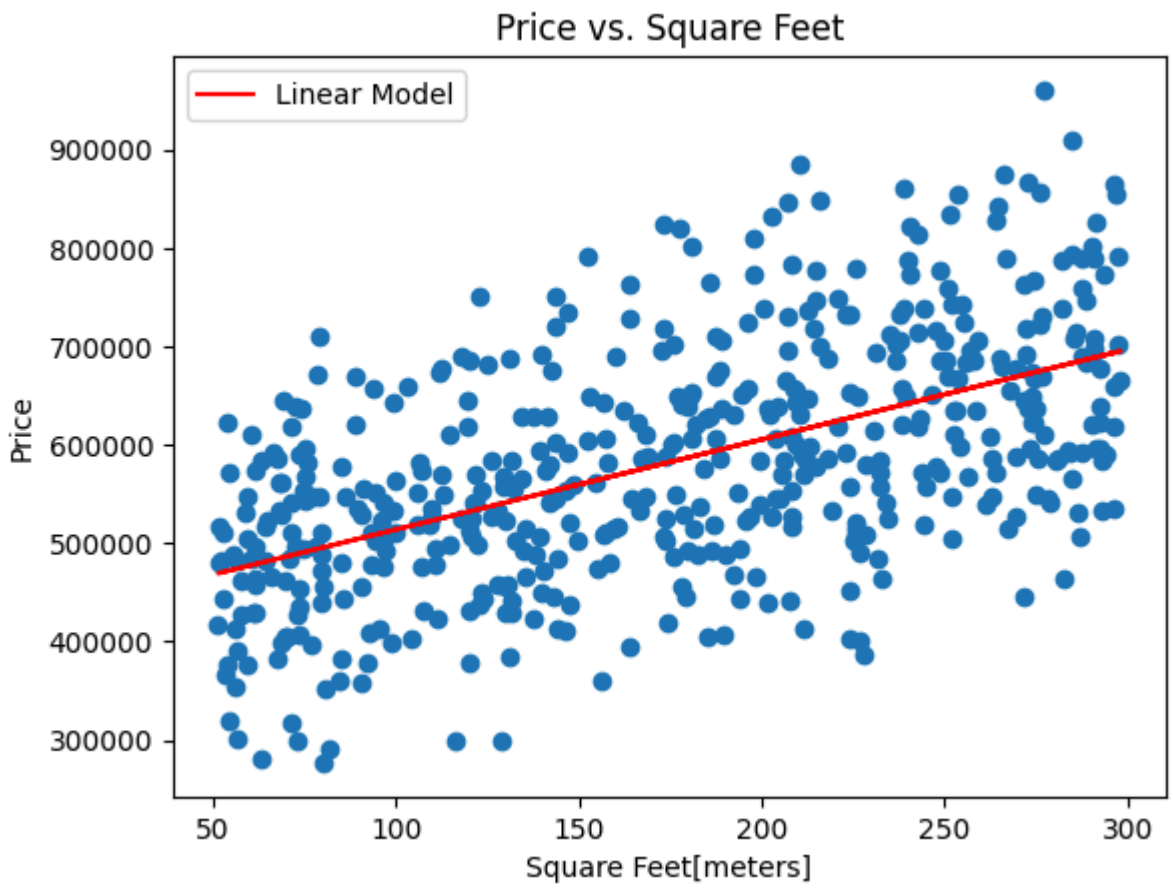
[37]:
```
plt.plot(X_train.values, model.predict(X_train),color="red", label = "Linear Model")
plt.scatter(X_train, y_train)
plt.xlabel("Square Feet[meters]")
plt.ylabel("Price")
plt.title("Price vs. Square Feet")
plt.legend();
```



[39]:
```
print(f"Price = {intercept} + {coefficient}* Square_Feet")
```

```
Price = 422466.52 + 914.7* Square_Feet
```

[37]:
```
plt.plot(X_train.values, model.predict(X_train),color="red", label = "Linear Model")
plt.scatter(X_train, y_train)
plt.xlabel("Square Feet[meters]")
plt.ylabel("Price")
plt.title("Price vs. Square Feet")
plt.legend();
```



[ ]: