HDB_Prediction_Single_Variable

September 26, 2025

1 Import libraries and data

[1]:
```
import pandas as pd
import numpy as np
import plotly.express as px
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import datetime
from sklearn.model_selection import train_test_split
```

[3]:
```
housing_df = pd.read_csv(r"F:\Data Science Grand Track\Machine␣
↪Learning\datasets\housing\hdb\hdb_resale_data_cleaned.csv")
```

[4]:
```
display(housing_df)
```

	Unnamed: 0	floor_area_sqm	room_count	age	resale_price
0	0	44.0	2	46	232000.0
1	1	67.0	3	47	250000.0
2	2	67.0	3	45	262000.0
3	3	68.0	3	45	265000.0
4	4	67.0	3	45	265000.0
…	…	…	… …	…	
201129	216822	122.0	5	37	735000.0
201130	216823	122.0	5	38	755000.0
201131	216824	127.0	5	37	735000.0
201132	216825	127.0	5	37	795000.0
201133	216826	127.0	5	37	790000.0

[201134 rows x 5 columns]

2 Extract features

Say, for the purpose of this snippet, we want to demo the correlation of flat size and price.

2.1 Creating X, y

[7]:
```
X = housing_df['floor_area_sqm'].values
y = housing_df['resale_price'].values
```

2.2 Check shape:

[8]:
```
print(X.shape)
print(y.shape)
```

```
(201134,)
(201134,)
```

2.3 Reshape X

[9]:
```
X = X.reshape(-1,1)
```

2.4 Testing X

[12]:
```
display(X)
print(X.shape)
```

```
array([[ 44.],
       [ 67.],
       [ 67.],
       …,
       [127.],
       [127.],
       [127.]])

(201134, 1)
```

3 Create model:

3.1 Creating an object

[14]:
```
# Creating an object
predict_price = LinearRegression()
```

3.2 Split data into train and test

[13]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
↪random_state=42)
```

3.3 Fit testing data to model (train the model)

[16]:
```
predict_price.fit(X_train, y_train)
```

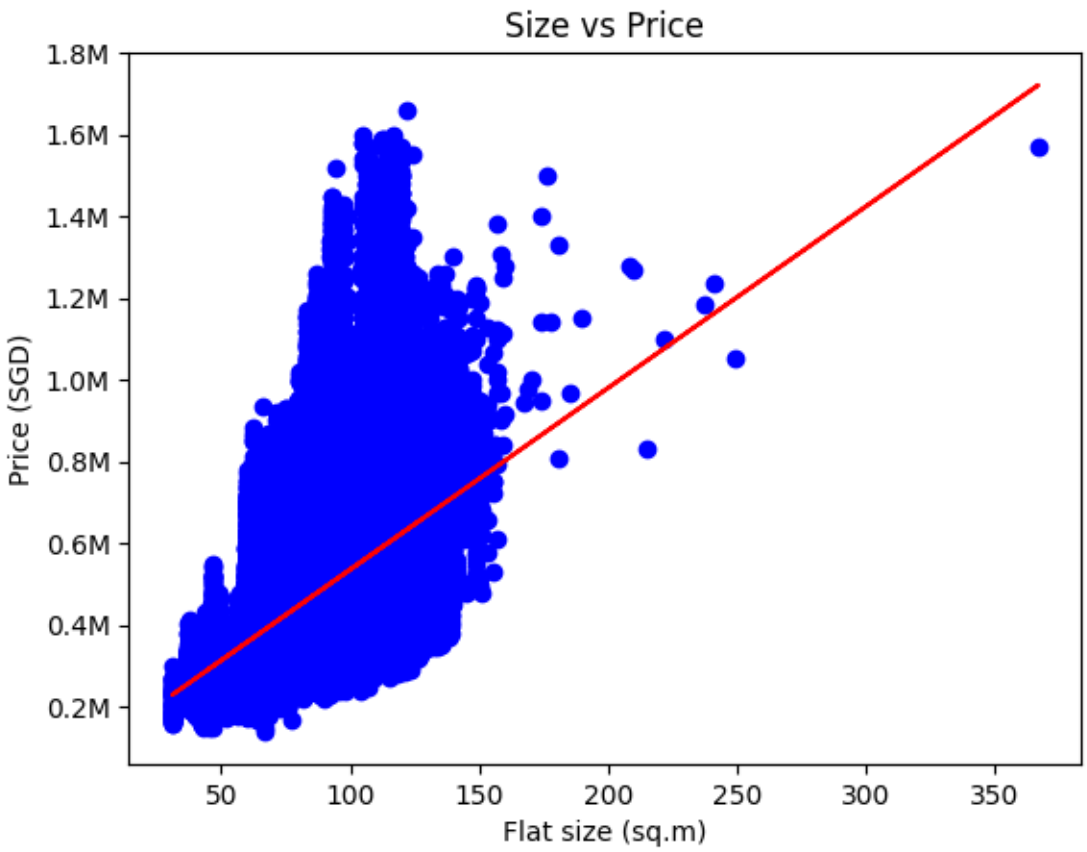
[16]:
```
LinearRegression()
```

3.4 Using model on all of X

[17]:
```
predicted_price = predict_price.predict(X)
```

4 Plotting:

[32]:
```
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import numpy as np # Assuming X, y, and predicted_price are NumPy arrays

# --- Setup the Figure and Axes ---
# Create the figure (fig) and the axes (ax)
fig, ax = plt.subplots()

# --- Your Plotting Code ---
# Scatter plot of raw data:
ax.scatter(X, y, color="blue")

# Prediction line:
ax.plot(X, predicted_price, color="red")

# Labeling:
ax.set_title("Size vs Price")
ax.set_xlabel("Flat size (sq.m)")
ax.set_ylabel("Price (SGD)")

# --- Scaling/Formatting the Y-Axis Ticks ---
def millions_formatter(x, pos):
    'The two args are the value and tick position'
    # Format the number to one decimal place and append 'M' for million
    return f'{x/1000000:1.1f}M'

# Create the formatter
formatter = ticker.FuncFormatter(millions_formatter)

# Apply the formatter to the y-axis
ax.yaxis.set_major_formatter(formatter)

# --- Display the Plot ---
plt.show()
```


Size vs Price
1.8M
1.6M
1.4M
1.2M
1.0M
0.8M
0.6M
0.4M
0.2M
Price (SGD)
50
100
150
200
250
300
350
Flat size (sq.m)

[ ]:

[ ]: