vv4lv2mas

May 30, 2025

1 House Rent Prediction

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
```

[2]:
```python
data=pd.read_csv('House_Rent_Dataset.csv')
```

[3]:
```python
data.head(5)
```

[3]:
```
    Posted On  BHK   Rent  Size            Floor    Area Type  \
0  2022-05-18    2  10000  1100  Ground out of 2   Super Area
1  2022-05-13    2  20000   800       1 out of 3   Super Area
2  2022-05-16    2  17000  1000       1 out of 3   Super Area
3  2022-07-04    2  10000   800       1 out of 2   Super Area
4  2022-05-09    2   7500   850       1 out of 2  Carpet Area

             Area Locality     City Furnishing Status  Tenant Preferred  \
0                   Bandel  Kolkata       Unfurnished  Bachelors/Family
1  Phool Bagan, Kankurgachi  Kolkata    Semi-Furnished  Bachelors/Family
2   Salt Lake City Sector 2  Kolkata    Semi-Furnished  Bachelors/Family
3              Dumdum Park  Kolkata       Unfurnished  Bachelors/Family
4            South Dum Dum  Kolkata       Unfurnished         Bachelors

   Bathroom Point of Contact
0         2    Contact Owner
1         1    Contact Owner
2         1    Contact Owner
3         1    Contact Owner
4         1    Contact Owner
```

[4]:
```python
data.isnull().sum()
```

[4]:
```
Posted On            0
BHK                  0
```

```
Rent                 0
Size                 0
Floor                0
Area Type            0
Area Locality        0
City                 0
Furnishing Status    0
Tenant Preferred     0
Bathroom             0
Point of Contact     0
dtype: int64
```

[5]:
```
data.describe().round(2)
```

[5]:

	BHK	Rent	Size	Bathroom
count	4746.00	4746.00	4746.00	4746.00
mean	2.08	34993.45	967.49	1.97
std	0.83	78106.41	634.20	0.88
min	1.00	1200.00	10.00	1.00
25%	2.00	10000.00	550.00	1.00
50%	2.00	16000.00	850.00	2.00
75%	3.00	33000.00	1200.00	2.00
max	6.00	3500000.00	8000.00	10.00

1.1 looking at the mean, median, highest, and lowest rent of the houses.

[6]:
```
print(f"Mean Rent: {data.Rent.mean().round(2)}")
print(f"Median Rent: {data.Rent.median()}")
print(f"Highest Rent: {data.Rent.max()}")
print(f"Lowest Rent: {data.Rent.min()}")
```

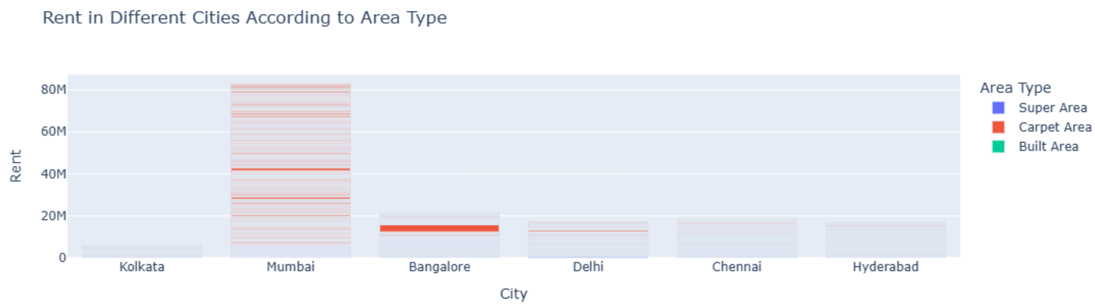
```
Mean Rent: 34993.45
Median Rent: 16000.0
Highest Rent: 3500000
Lowest Rent: 1200
```

1.1.1 Q. Now let's have a look at the rent of the houses in different cities according to the number of bedrooms, halls, and kitchens:

[7]:
```
plt = px.bar(data, x=data['City'],
      y=data['Rent'],
      color = data['BHK'],
      title = 'Rent in Different Cities According to BHK')
plt.show()
```

1.1.2 Now let's have a look at the rent of the houses in different cities according to the area type:

[8]:
```
figure = px.bar(data, x=data['City'],
               y=data['Rent'],
               color = data['Area Type'],
               title = "Rent in Different Cities According to Area Type")
figure.show()
```



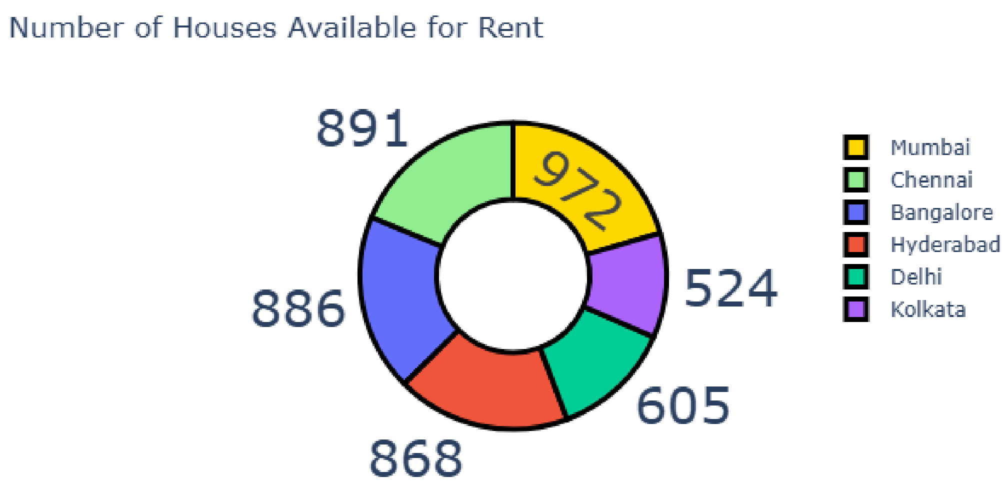
1.1.3 Q.Now let's have a look at the rent of the houses in different cities according to the size of the house.

[9]:
```
figure = px.bar(data, x=data["City"],
                y = data["Rent"],
                color = data["Size"],
            title="Rent in Different Cities According to Size")
figure.show()
```

1.1.4 Q. Now let's have a look at the number of houses available for rent in different cities according to the dataset?

[13]:
```
cities = data['City'].value_counts()
label = cities.index
counts = cities.values
colors = ['gold','lightgreen']

fig = go.Figure(data=[go.Pie(labels=label,values=counts, hole=0.5)])
fig.update_layout(title_text='Number of Houses Available for Rent')
fig.update_traces(hoverinfo='label+percent', textinfo='value', textfont_size=30,
                  marker=dict(colors=colors, line=dict(color='black', width=3)))
fig.show()
```

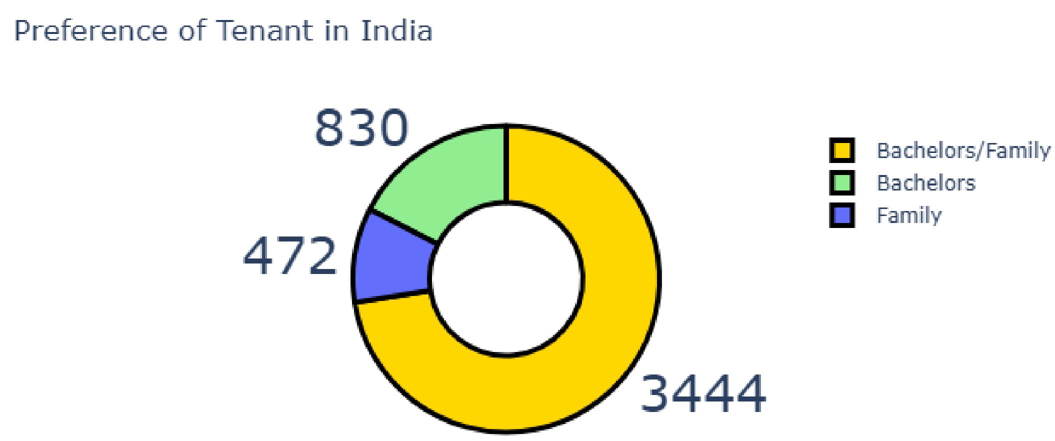


1.1.5 Q.Now let's have a look at the number of houses available for different types of tenants

[15]:
```
# Preference of Tenant
tenant = data["Tenant Preferred"].value_counts()
label = tenant.index
counts = tenant.values
colors = ['gold','lightgreen']

fig = go.Figure(data=[go.Pie(labels=label, values=counts, hole=0.5)])
fig.update_layout(title_text='Preference of Tenant in India')
fig.update_traces(hoverinfo='label+percent', textinfo='value', textfont_size=30,
                  marker=dict(colors=colors, line=dict(color='black', width=3)))
```

```
fig.show()
```



1.1.6 Now we will convert all the categorical features into numerical features that we need to train a house rent prediction model:

```
[ ]: # # Those three terms-Super Area, Carpet Area, and Built-up Area-are standard␣
     ↪ways of measuring how big an apartment or house is. Here's what each one␣
     ↪means:

     # # | Term                 | What it Measures                                                                                                                                   ␣
     ↪                                                                                                                                                                               ␣
     ↪                          |
     # # | ----------------- |␣
     ↪-------------------------------------------------------------------------------------------
     ↪|
     # # | **Carpet Area**    | The actual "usable" floor space inside your walls-
     where
     #                          |  you can lay a carpet. It excludes the thickness of␣
     ↪inner walls, balconies, common areas,
     # etc.  |
     # # | **Built-up Area** | Carpet area + the area occupied by the walls␣
     ↪themselves.
     #                          | So it's a bit larger than the carpet area, because it␣
     ↪includes wall thickness.                    |
     # # | **Super Area**     | Built-up area + a proportionate share of common␣
     ↪spaces in the
```

```
# building complex (like lobbies, staircases, lifts, corridors, clubhouse).
 ↪          /
```

[16]:
```python
data["Area Type"] = data["Area Type"].map({"Super Area": 1,
                                           "Carpet Area": 2,
                                           "Built Area": 3})
data["City"] = data["City"].map({"Mumbai": 4000, "Chennai": 6000,
                                 "Bangalore": 5600, "Hyderabad": 5000,
                                 "Delhi": 1100, "Kolkata": 7000})
data["Furnishing Status"] = data["Furnishing Status"].map({"Unfurnished": 0,
                                                           "Semi-Furnished": 1,
                                                           "Furnished": 2})
data["Tenant Preferred"] = data["Tenant Preferred"].map({"Bachelors/Family": 2,
                                                         "Bachelors": 1,
                                                         "Family": 3})
print(data.head())
```

```
   Posted On  BHK   Rent  Size            Floor  Area Type  \
0  2022-05-18    2  10000  1100  Ground out of 2          1
1  2022-05-13    2  20000   800       1 out of 3          1
2  2022-05-16    2  17000  1000       1 out of 3          1
3  2022-07-04    2  10000   800       1 out of 2          1
4  2022-05-09    2   7500   850       1 out of 2          2

             Area Locality  City  Furnishing Status  Tenant Preferred  \
0                   Bandel  7000                  0                 2
1  Phool Bagan, Kankurgachi  7000                  1                 2
2   Salt Lake City Sector 2  7000                  1                 2
3              Dumdum Park  7000                  0                 2
4            South Dum Dum  7000                  0                 1

   Bathroom Point of Contact
0         2     Contact Owner
1         1     Contact Owner
2         1     Contact Owner
3         1     Contact Owner
4         1     Contact Owner
```

[17]:
```python
#splitting data
from sklearn.model_selection import train_test_split
x = np.array(data[["BHK", "Size", "Area Type", "City",
                   "Furnishing Status", "Tenant Preferred",
                   "Bathroom"]])
y = np.array(data[["Rent"]])

xtrain, xtest, ytrain, ytest = train_test_split(x, y,
                                                test_size=0.10,
```

```
                                              random_state=42)
```

[18]:
```
from keras.models import Sequential
from keras.layers import Dense, LSTM
model = Sequential()
model.add(LSTM(128, return_sequences=True,
               input_shape= (xtrain.shape[1], 1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
model.summary()
```

```
C:\Users\Asus\AppData\Roaming\Python\Python312\site-
packages\keras\src\layers\rnn\rnn.py:200: UserWarning:

Do not pass an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first layer in
the model instead.
```

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 7, 128)	66,560
lstm_1 (LSTM)	(None, 64)	49,408
dense (Dense)	(None, 25)	1,625
dense_1 (Dense)	(None, 1)	26

Total params: 117,619 (459.45 KB)

Trainable params: 117,619 (459.45 KB)

Non-trainable params: 0 (0.00 B)

[19]:
```
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(xtrain, ytrain, batch_size=1, epochs=21)
```

```
Epoch 1/21
4271/4271            22s 5ms/step -
```

```
loss: 4960848384.0000
Epoch 2/21
4271/4271            20s 5ms/step -
loss: 4217724672.0000
Epoch 3/21
4271/4271            19s 5ms/step -
loss: 3740038912.0000
Epoch 4/21
4271/4271            20s 5ms/step -
loss: 3404541696.0000
Epoch 5/21
4271/4271            20s 5ms/step -
loss: 9108176896.0000
Epoch 6/21
4271/4271            20s 5ms/step -
loss: 3232720896.0000
Epoch 7/21
4271/4271            19s 5ms/step -
loss: 2969470976.0000
Epoch 8/21
4271/4271            19s 5ms/step -
loss: 7200079872.0000
Epoch 9/21
4271/4271            30s 7ms/step -
loss: 2943646464.0000
Epoch 10/21
4271/4271            19s 4ms/step -
loss: 3040873472.0000
Epoch 11/21
4271/4271            18s 4ms/step -
loss: 3584683008.0000
Epoch 12/21
4271/4271            20s 5ms/step -
loss: 3207720960.0000
Epoch 13/21
4271/4271            17s 4ms/step -
loss: 6307440128.0000
Epoch 14/21
4271/4271            18s 4ms/step -
loss: 19707887616.0000
Epoch 15/21
4271/4271            18s 4ms/step -
loss: 2618063360.0000
Epoch 16/21
4271/4271            17s 4ms/step -
loss: 4373302272.0000
Epoch 17/21
4271/4271            17s 4ms/step -
```

```
loss: 3855800064.0000
Epoch 18/21
4271/4271            17s 4ms/step -
loss: 6564930560.0000
Epoch 19/21
4271/4271            18s 4ms/step -
loss: 3646164992.0000
Epoch 20/21
4271/4271            17s 4ms/step -
loss: 4346147328.0000
Epoch 21/21
4271/4271            17s 4ms/step -
loss: 3692125696.0000
```

[19]: <keras.src.callbacks.history.History at 0x174ebb8ec00>

[22]:
```
print("Enter House Details to Predict Rent")
a = int(input("Number of BHK: "))
b = int(input("Size of the House(Sq. km): "))
c = int(input("Area Type (Super Area = 1, Carpet Area = 2, Built Area = 3): "))
d = int(input("Pin Code of the City: "))
e = int(input("Furnishing Status of the House (Unfurnished = 0, Semi-Furnished␣
 ↪= 1, Furnished = 2): "))
f = int(input("Tenant Type (Bachelors = 1, Bachelors/Family = 2, Only Family =␣
 ↪3): "))
g = int(input("Number of bathrooms: "))
features = np.array([[a, b, c, d, e, f, g]])
print("Predicted House Price(in Rs) = ", model.predict(features))
```

```
Enter House Details to Predict Rent

Number of BHK:  3
Size of the House(Sq. km):  1500
Area Type (Super Area = 1, Carpet Area = 2, Built Area = 3):  2
Pin Code of the City:  4000
Furnishing Status of the House (Unfurnished = 0, Semi-Furnished = 1, Furnished =
2):  1
Tenant Type (Bachelors = 1, Bachelors/Family = 2, Only Family = 3):  2
Number of bathrooms:  2

1/1            0s 35ms/step
Predicted House Price(in Rs) =  [[27642.043]]
```

2 Conclusion

2.0.1 This is how we will predict the rent of a housing property. With appropriate data and Machine Learning techniques, many real estate platforms find housing options according to the customer's budget.

[ ]:
```

```