House Rent Prediction Machine Learning Project

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go

data = pd.read_csv("/content/House_Rent_Dataset.csv")
print(data.head())
```

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

```
print(data.isnull().sum())
```

```
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
```

```
print(data.describe())
```

```
                BHK          Rent         Size     Bathroom
count  4746.000000  4.746000e+03  4746.000000  4746.000000
mean      2.083860  3.499345e+04   967.490729     1.965866
std       0.832256  7.810641e+04   634.202328     0.884532
min       1.000000  1.200000e+03    10.000000     1.000000
25%       2.000000  1.000000e+04   550.000000     1.000000
50%       2.000000  1.600000e+04   850.000000     2.000000
75%       3.000000  3.300000e+04  1200.000000     2.000000
max       6.000000  3.500000e+06  8000.000000    10.000000
```

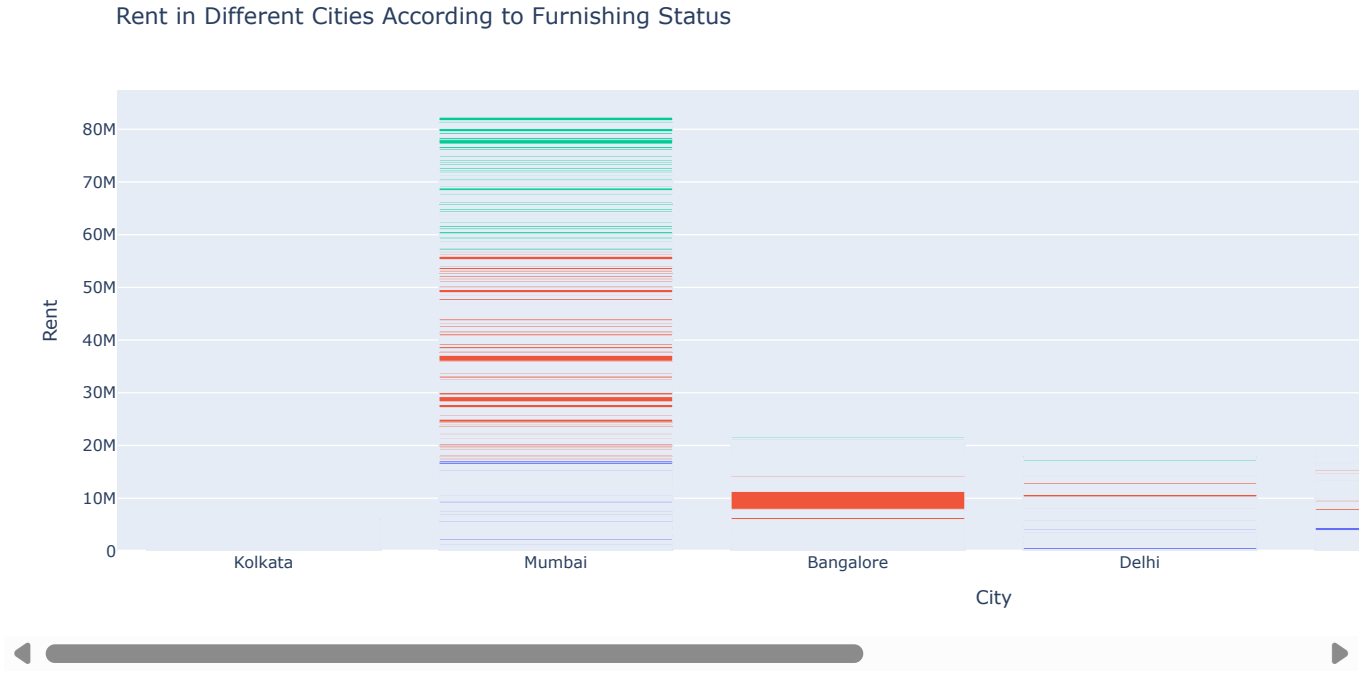
```
print(f"Mean Rent: {data.Rent.mean()}")
print(f"Median Rent: {data.Rent.median()}")
print(f"Highest Rent: {data.Rent.max()}")
print(f"Lowest Rent: {data.Rent.min()}")
```

```
Mean Rent: 34993.45132743363
Median Rent: 16000.0
Highest Rent: 3500000
Lowest Rent: 1200
```

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


```
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()
```


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



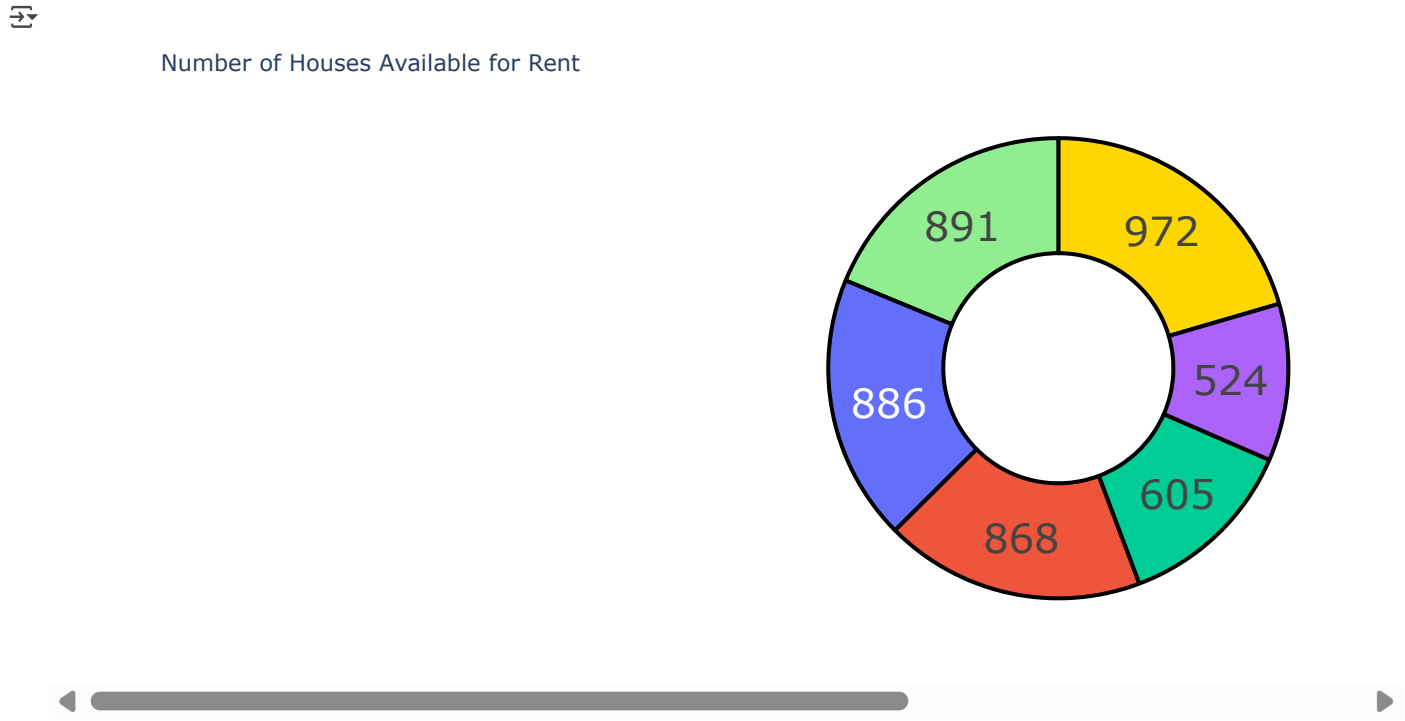

```
figure = px.bar(data, x=data["City"],
                y = data["Rent"]
                color = data["Size"],
            title="Rent in Different Cities According to Size")
figure.show()
```

```
    File "<ipython-input-8-279ee621c7c5>", line 2
      y = data["Rent"]
        ^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

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

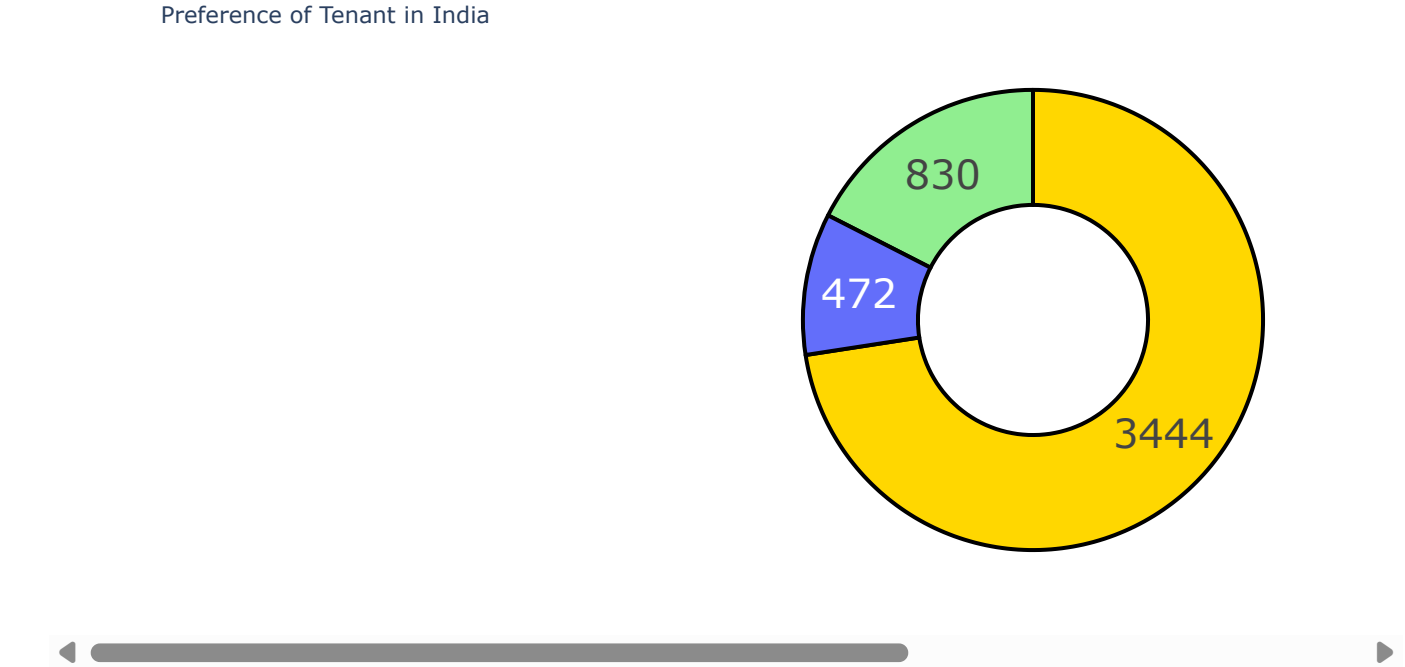

```
cities = data["City"].value_counts()
label = cities.index
counts = cities.values
colors = ['gold','lightgreen']
```

```
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()
```




```
# 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()
```




```
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  18-05-2022    2  10000  1100  Ground out of 2        NaN
1  13-05-2022    2  20000   800       1 out of 3        NaN
2  16-05-2022    2  17000  1000       1 out of 3        NaN
3  04-07-2022    2  10000   800       1 out of 2        NaN
4  09-05-2022    2   7500   850       1 out of 2        NaN

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

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

```
#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)
```

```
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()
```

```
/usr/local/lib/python3.11/dist-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
```

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)

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

```
Epoch 1/21
4271/4271 ━━━━━━━━━━━━━━━━━━━━ 53s 11ms/step - loss: nan
Epoch 2/21
4271/4271 ━━━━━━━━━━━━━━━━━━━━ 81s 11ms/step - loss: nan
Epoch 3/21
4271/4271 ━━━━━━━━━━━━━━━━━━━━ 48s 11ms/step - loss: nan
Epoch 4/21
4271/4271 ━━━━━━━━━━━━━━━━━━━━ 48s 11ms/step - loss: nan
Epoch 5/21
4271/4271 ━━━━━━━━━━━━━━━━━━━━ 81s 11ms/step - loss: nan
Epoch 6/21
4271/4271 ━━━━━━━━━━━━━━━━━━━━ 81s 11ms/step - loss: nan
Epoch 7/21
```

```
4271/4271 ━━━━━━━━━━━━━━━━━━━━ 47s 11ms/step - loss: nan
Epoch 8/21
4271/4271 ━━━━━━━━━━━━━━━━━━━━ 82s 11ms/step - loss: nan
Epoch 9/21
4271/4271 ━━━━━━━━━━━━━━━━━━━━ 81s 11ms/step - loss: nan
Epoch 10/21
4271/4271 ━━━━━━━━━━━━━━━━━━━━ 49s 11ms/step - loss: nan
Epoch 11/21
4271/4271 ━━━━━━━━━━━━━━━━━━━━ 80s 11ms/step - loss: nan
Epoch 12/21
4271/4271 ━━━━━━━━━━━━━━━━━━━━ 83s 11ms/step - loss: nan
Epoch 13/21
4271/4271 ━━━━━━━━━━━━━━━━━━━━ 81s 11ms/step - loss: nan
Epoch 14/21
4271/4271 ━━━━━━━━━━━━━━━━━━━━ 49s 11ms/step - loss: nan
Epoch 15/21
4271/4271 ━━━━━━━━━━━━━━━━━━━━ 80s 11ms/step - loss: nan
Epoch 16/21
4271/4271 ━━━━━━━━━━━━━━━━━━━━ 82s 11ms/step - loss: nan
Epoch 17/21
4271/4271 ━━━━━━━━━━━━━━━━━━━━ 82s 11ms/step - loss: nan
Epoch 18/21
4271/4271 ━━━━━━━━━━━━━━━━━━━━ 82s 11ms/step - loss: nan
Epoch 19/21
4271/4271 ━━━━━━━━━━━━━━━━━━━━ 83s 11ms/step - loss: nan
Epoch 20/21
4271/4271 ━━━━━━━━━━━━━━━━━━━━ 81s 11ms/step - loss: nan
Epoch 21/21
4271/4271 ━━━━━━━━━━━━━━━━━━━━ 48s 11ms/step - loss: nan
<keras.src.callbacks.history.History at 0x79ca4e77bb50>
```

```
print("Enter House Details to Predict Rent")
a = int(input("Number of BHK: "))
b = int(input("Size of the House: "))
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 = ", model.predict(features))
```

```
Enter House Details to Predict Rent
Number of BHK: 3
Size of the House: 1100
Area Type (Super Area = 1, Carpet Area = 2, Built Area = 3): 2
Pin Code of the City: 1100
Furnishing Status of the House (Unfurnished = 0, Semi-Furnished = 1, Furnished = 2): 1
Tenant Type (Bachelors = 1, Bachelors/Family = 2, Only Family = 3): 3
Number of bathrooms: 2
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
Predicted House Price =  [[nan]]
```

Start coding or generate with AI.