PROJECT REPORT

TOPIC: ***<u>RENT CALCULATOR</u>***

<u>SUBMITTED BY:</u>

BHUVAN CHOPRA

HARSH BHATT

KAUSTUBH SAHU

MANVI GOEL

TABLE OF CONTENTS

SNo.	TOPIC	PAGENO.
1	Acknowledgement	3
2	Introduction	4
3	Languages and Frameworks Used	5
4	Dataset Analysis	6
5	Source Code and Its Explanation	9-17
6	Front End	18-20
7	Conclusion	21

ACKNOWLEDGEMENT

First of all, we thank lord, almighty to help us in our project keeping us healthy and serving us in all the endeavours.

We would also like to express our special thanks of gratitude to our teachers, Mr. Peeyush Jain, Mrs. Himani , Ms.Priya Gaba and Mrs. Sakshi Sharma who gave us the golden opportunity to do this wonderful project under Machine Learning and Its Applications, on the topic Rent Calculator, which helped us in doing research and we came to know about various new aspects of Machine Learning and Python. We appreciate the guidance given by the supervisors as well as the panels.

Thus, we are grateful to all those who helped us continuously to put our persistent efforts into the making and completion of this project.

INTRODUCTION

Our project – Rent Calculator is an application which serves the **purpose** of providing the customer an easy way to estimate the rent of the property they are aspiring to purchase without intervention of the broker in the process. The customer can easily estimate the rent of the property with all his desired facilities in any area by sitting anywhere. Also, the can flexibly change his preferences of facilities in order to obtain a property on rent in his budget. Thus, this interface aims to simplify the work of customers by predicting the rent of the property they wish to purchase and set their budget approximately.

Thus, our **problem** is to predict the rent of property given various features like locality, it's geographical location, property size, property age, type of house, structure and interior, lease type, various amenities like gym, lift, furnishing status etc.

Now, the question arises that **how we have developed this application**?
This application is based on the Machine Learning Models using Python. We train our machine on the basis of previous data of that region and then predict the rent of the property on the basis of it.

Next question which pops up is whether **our problem statement is a Regression problem or a classification problem**?
So, here given various features by the customer, our Machine Learning Model has to predict the rent of the property, thus it is a Regression problem. It is a problem with multiple input variables and hence, is a Multivariate Regression Problem.

LANGUAGES AND FRAMEWORK USED

-Programming Language Used: Python

Python is an interpreted, high-level, general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. It has filter, map, and reduce functions; list comprehensions, dictionaries, sets and generator expressions. Python's large standard library, commonly cited as one of its greatest strengths.

-Framework Used: Anaconda 2019.03 Python 3.7 Version

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, macOS, and Linux. The command line program conda is both a package manager and an environment manager, to help data scientists ensure that each version of each package has all the dependencies it requires and works correctly. Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages and update them, all inside Navigator.

We have used **Jupyter Notebook** in Anaconda Navigator. The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

DATASET ANALYSIS

The dataset which we collected is a CSV file from a company-NoBroker which deals in the selling and purchasing of property.

The dataset contains 25,000 training examples. There are 26 features in the dataset which include id, type, locality, activation date, latitude, longitude, lease type, gym, lift, swimming pool, negotiable, furnishing, parking, property size, property age, bathroom, facing, cupboard, floor, total floors, amenities, water supply, building type, balconies, rent and deposit.

Most of the data in these features is categorical and for Machine Learning Models to work they must be converted into numeric data.

Further, the feature 'amenities' consists within it a dictionary of 19 features like lift, gym, Internet, Servant, Intercom, HK, RWH, STP. These features in dictionary are in a random order and not necessarily present for each training example.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
id	type	locality	activation_	latitude	longitude	lease_type	gym	lift	swimming_	negotiable	furnishing	parking	property_s	property_a	bathroom
ff8081815	BHK2	Jayanagara	########	12.9366	77.57691	FAMILY	0	0	0	0	SEMI_FUR	TWO_WHI	1000	5	2
ff8081815	BHK2	Basaveshw	########	12.99799	77.54522	FAMILY	0	0	0	1	SEMI_FUR	BOTH	1218	20	3
ff8081815	BHK3	Jaya Nagar	########	12.9357	77.58764	FAMILY	0	1	0	0	SEMI_FUR	BOTH	1820	5	3
ff8081815	BHK2	Murugeshp	########	12.95351	77.65612	FAMILY	0	1	0	1	SEMI_FUR	BOTH	1100	5	2
ff8081815	BHK2	Whitefield	########	12.96852	77.74244	ANYONE	1	1	1	1	SEMI_FUR	BOTH	1475	0	2
ff8081816	BHK2	HSR Layou	########	12.922	77.64624	FAMILY	0	0	0	1	FULLY_FU	NONE	1180	6	2
ff8081816	BHK3	Harlur	########	12.90996	77.67475	FAMILY	0	0	0	0	SEMI_FUR	BOTH	2600	5	4
ff8081815	BHK2	Vijaya Nag	########	12.96416	77.51894	FAMILY	0	0	0	1	SEMI_FUR	BOTH	900	15	2
ff8081816	BHK1	Doddanekl	########	12.97034	77.6888	FAMILY	0	0	0	0	SEMI_FUR	NONE	400	5	1
ff8081815	BHK2	BTM 2nd S	########	12.90946	77.60935	ANYONE	0	0	0	1	SEMI_FUR	TWO_WHI	720	10	1

Q	R	S	T	U	V	W	X	Y	Z
facing	cup_board	floor	total_floo	amenities	water_sup	building_ty	balconies	rent	deposit
S	2	2	2	{"LIFT":false,"GYM":false,"INTERNET":false,"A	CORP_BOI	IF	1	22000	220000
W	2	0	1	{"LIFT":false,"GYM":false,"INTERNET":false,"A	CORPORA	IH	0	20000	200000
E	3	4	9	{"LIFT":true,"GYM":false,"INTERNET":true,"AC	CORPORA	AP	2	38000	250000
E	2	4	4	{"LIFT":true,"GYM":false,"INTERNET":true,"AC	BOREWEL	AP	1	30000	300000
E	2	1	9	{"LIFT":true,"GYM":true,"INTERNET":true,"AC"	CORP_BOI	AP	2	26500	150000
N	2	1	4	{"LIFT":false,"GYM":false,"INTERNET":true,"AC	CORP_BOI	AP	3	30000	175000
W	6	0	2	{"LIFT":false,"GYM":false,"INTERNET":false,"A	BOREWEL	GC	2	35500	216000
W	2	0	3	{"LIFT":false,"GYM":false,"INTERNET":true,"AC	CORP_BOI	AP	1	15000	150000
E	1	1	3	{"PARK":false,"HK":false,"LIFT":false,"PB":false	CORPORA	IH	1	11000	50000
E	2	1	2	{"LIFT":false,"GYM":false,"INTERNET":true,"AC	CORPORA	IF	1	14000	60000

SOURCE CODE AND ITS EXPLANATION

```
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as pt
import seaborn as sns

#libraries for label encoding
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder

#for feature selection
from sklearn.ensemble import ExtraTreesClassifier

#linear regression model
from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split
from sklearn import metrics

from scipy import stats

#Random Forest Regressor model
from sklearn.ensemble import RandomForestRegressor

from sklearn.preprocessing import StandardScaler
```

This imports all the necessary libraries for our code.

```
In [5]: #loading csv file
        data =pd.read_csv("Downloads/hackathon_rentomter_nobroker.csv")
        data
```

Out[5]:

	id	type	locality	activation_date	latitude	longitude	lease_type	gym	lift	swimming_pool	...	facin
0	ff8081815917971401591af8895032d0	BHK2	Jayanagara	2018-07-05 17:09:49	12.936601	77.576914	FAMILY	0	0	0	...	
1	ff80818157dbe1fb0157dc4aa07513dc	BHK2	Basaveshwar Nagar	2017-12-30 17:22:12	12.997989	77.545219	FAMILY	0	0	0	...	\
2	ff8081815d304406015d30edcd7c4f44	BHK3	Jaya Nagar East,Jayanagar	2017-07-11 15:42:05	12.935696	77.587642	FAMILY	0	1	0	...	
3	ff8081815b106986015b154d908f2ac2	BHK2	Murugeshpalya	2018-07-07 14:26:14	12.953507	77.656118	FAMILY	0	1	0	...	
4	ff8081815e80b789015e84a7a42a62a3	BHK2	Whitefield	2017-09-27 12:19:28	12.968520	77.742436	ANYONE	1	1	1	...	
5	ff80818162537f2d01625cfb76f37165	BHK2	HSR Layout	2018-03-25 18:15:19	12.921999	77.646238	FAMILY	0	0	0	...	
6	ff80818160792cc001608ec7b922154a	BHK3	Harlur	2018-07-26 21:01:47	12.909958	77.674755	FAMILY	0	0	0	...	\

Then, we load our dataset in CSV file using pandas.

-DATA ANALYSIS

```
In [6]: #determining count of each feature for training examples
        data.count()

Out[6]: id                 25000
        type               25000
        locality           25000
        activation_date    25000
        latitude           25000
        longitude          25000
        lease_type         25000
        gym                25000
        lift               25000
        swimming_pool      25000
        negotiable         25000
        furnishing         25000
        parking            25000
        property_size      25000
        property_age       25000
        bathroom           25000
        facing             25000
        cup_board          25000
        floor              25000
        total_floor        25000
        amenities          25000
        water_supply       25000
        building_type      25000
        balconies          25000
        rent               25000
        deposit            25000
        dtype: int64
```

Here, we count the number of training examples for each feature.

```
#check for null values in dataset
data.isnull().sum()

id                 0
type               0
locality           0
activation_date    0
latitude           0
longitude          0
lease_type         0
gym                0
lift               0
swimming_pool      0
negotiable         0
furnishing         0
parking            0
property_size      0
property_age       0
bathroom           0
facing             0
cup_board          0
floor              0
total_floor        0
amenities          0
water_supply       0
building_type      0
balconies          0
rent               0
deposit            0
dtype: int64
```

Here, we check for null values in our dataset.

```
#determine the type of data for each feature
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 26 columns):
id                  25000 non-null object
type                25000 non-null object
locality            25000 non-null object
activation_date     25000 non-null object
latitude            25000 non-null float64
longitude           25000 non-null float64
lease_type          25000 non-null object
gym                 25000 non-null int64
lift                25000 non-null int64
swimming_pool       25000 non-null int64
negotiable          25000 non-null int64
furnishing          25000 non-null object
parking             25000 non-null object
property_size       25000 non-null int64
property_age        25000 non-null float64
bathroom            25000 non-null int64
facing              25000 non-null object
cup_board           25000 non-null float64
floor               25000 non-null int64
total_floor         25000 non-null float64
amenities           25000 non-null object
water_supply        25000 non-null object
building_type       25000 non-null object
balconies           25000 non-null float64
rent                25000 non-null int64
deposit             25000 non-null float64
dtypes: float64(7), int64(8), object(11)
```

data.info() gives us the information of each data type so that we can estimate the categorical data in our dataset.

-DATA CLEANING

```
#HANDLING CATEGORICAL VARIABLES
```

```
#Using label encoder to transform categorical data of locality feature into integer
le=preprocessing.LabelEncoder()
data['locality']=le.fit_transform(data['locality'])
data['locality'].head()
```

```
0    1061
1     419
2    1027
3    1443
4    2084
Name: locality, dtype: int32
```

```
#determining number of unique values in locality
a=np.unique(data['locality'])
a.shape
```

```
(2177,)
```

```
#applying label encoder to other features containing categorical data
data['type']=le.fit_transform(data['type'])
data['lease_type']=le.fit_transform(data['lease_type'])
data['furnishing']=le.fit_transform(data['furnishing'])
data['parking']=le.fit_transform(data['parking'])
data['facing']=le.fit_transform(data['facing'])
data['water_supply']=le.fit_transform(data['water_supply'])
data['building_type']=le.fit_transform(data['building_type'])
```

Here, we transform all the categorical variables using Label Encoding.

```
#replacing true and false in amenities feature with 1 and 0 respectively
data['amenities']=data['amenities'].str.replace('false','0')
data['amenities']=data['amenities'].str.replace('true','1')
data['amenities'].head()
```

```
0    {"LIFT":0,"GYM":0,"INTERNET":0,"AC":0,"CLUB":0...
1    {"LIFT":0,"GYM":0,"INTERNET":0,"AC":0,"CLUB":0...
2    {"LIFT":1,"GYM":0,"INTERNET":1,"AC":0,"CLUB":1...
3    {"LIFT":1,"GYM":0,"INTERNET":1,"AC":0,"CLUB":0...
4    {"LIFT":1,"GYM":1,"INTERNET":1,"AC":0,"CLUB":1...
Name: amenities, dtype: object
```

Here, we replace the true and false values in the amenities feature by 1 and 0 respectively.

The below code splits the dictionary in amenities columns into different columns and then concatenates it with the rest of the dataset.

```
#splitting the amenities feature into various columns
df1 = data
df1["amenities"] = df1["amenities"].apply(lambda x : dict(eval(str(x)))) #parses the string
df2 = df1["amenities"].apply(pd.Series )
df2.head()
```

	LIFT	GYM	INTERNET	AC	CLUB	INTERCOM	POOL	CPA	FS	SERVANT	SECURITY	SC	GP	PARK	RWH	STP	HK	PB	VP
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
2	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0
3	1.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0
4	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0

```
#concatenating the above columns in the original dataset
data = pd.concat([data, df2], axis=1)
```

Next, we drop the repeated columns in the dataset.

```
#dropping repeated columns
data.drop(['amenities','GYM','LIFT','POOL'],axis=1,inplace=True)
```

Then, we check for the null values in the added columns.

```
#checking null values in the dataset for the added columns
data.isnull().sum()
```

```
id                   0
type                 0
locality             0
activation_date      0
latitude             0
longitude            0
lease_type           0
gym                  0
lift                 0
swimming_pool        0
negotiable           0
furnishing           0
parking              0
property_size        0
property_age         0
bathroom             0
facing               0
cup_board            0
floor                0
total_floor          0
water_supply         0
building_type        0
balconies            0
rent                 0
deposit              0
INTERNET             0
AC                   0
CLUB              1283
INTERCOM             0
CPA               1283
FS                   0
SERVANT           1283
SECURITY             0
SC                   0
GP                1283
PARK                 0
RWH               1283
STP               1283
HK                   0
PB                   0
VP                1283
dtype: int64
```

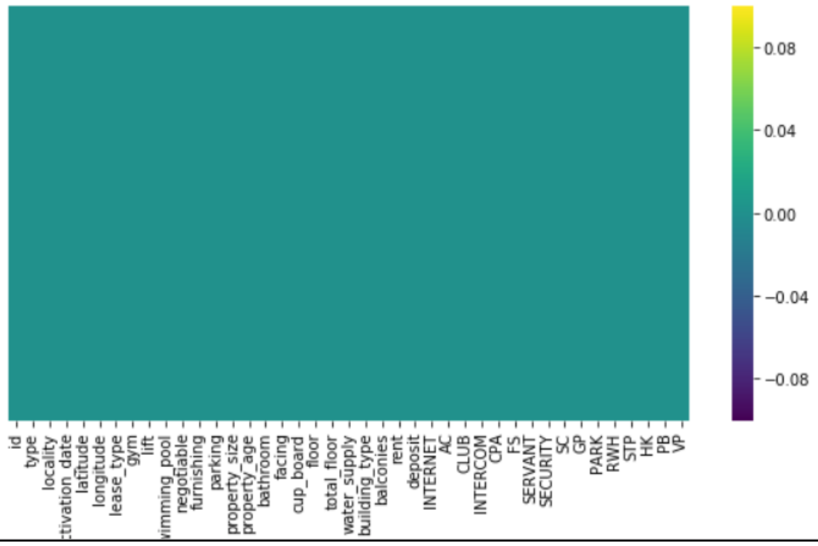
Next, we fill in these missing values by 0 as if these amenities would have been present then it would have been explicitly stated.

```
#filling null values in columns with 0 as no information about these amenities is given
data[['CLUB','CPA','GP','RWH','SERVANT','STP','VP']] = data[['CLUB','CPA','GP','RWH','SERVANT','STP','VP']].fillna(0,axis = 1,in
```

Next, we verify the same by plotting heatmap.

```
pt.figure(figsize=(10,5))
sns.heatmap(data.isnull(),yticklabels=False,cmap = 'viridis')
```

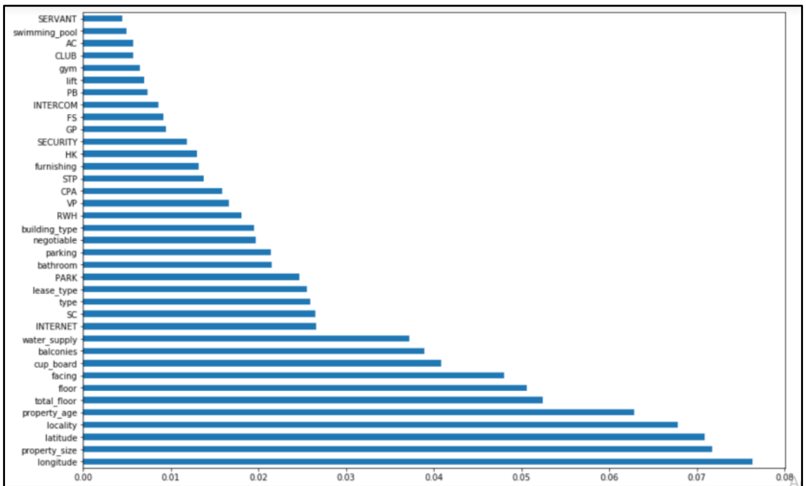
```
<matplotlib.axes._subplots.AxesSubplot at 0x1dae7c93f60>
```



-FEATURE ENGINEERING

```
#dropping id as rent is not dependent on it
#dropping activation_date as we already have property age feature
data.drop(["id","activation_date"],axis=1,inplace=True)
```

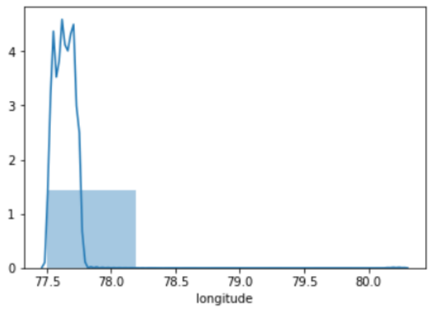
```
#Using ExtraTreesClassifier for predicting the importance of features for rent determination
pt.figure(figsize=(15,10))
x=data.drop(['rent','deposit'],axis=1)
y=data['rent']
model=ExtraTreesClassifier()
model.fit(x,y)
print(model.feature_importances_)
feat_importances=pd.Series(model.feature_importances_,index=x.columns)
feat_importances.nlargest(37).plot(kind='barh') #37 columns in total in x
pt.show()
```



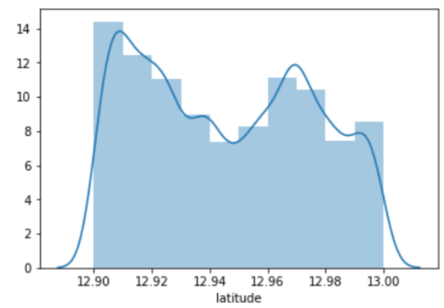
Extra Trees Classifier implements a meta estimator that fits a number of randomized decision trees and predicts importance of features.

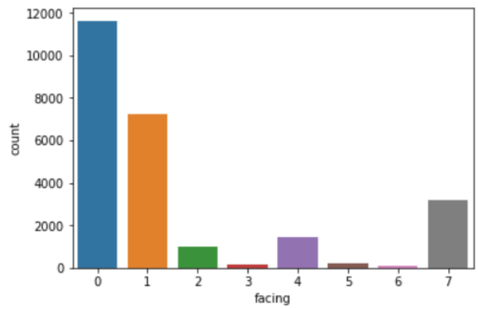
```
#plotting various plots for understanding features
```

```
sns.distplot(data['longitude'], bins=4)
pt.show()
```



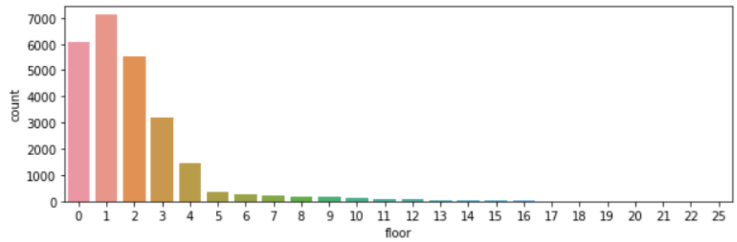

```
sns.distplot(data['latitude'], bins=10)
pt.show()
```





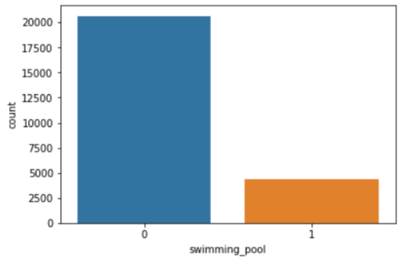

```
pt.figure(figsize=(10,3))
sns.countplot(x='floor',data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1daecc58978>
```




```
sns.countplot(x='swimming_pool',data=data)
```

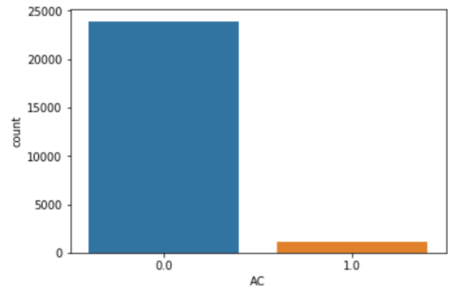
```
<matplotlib.axes._subplots.AxesSubplot at 0x1daf2872860>
```




```
#swimming pool present in approx 16% of property
```

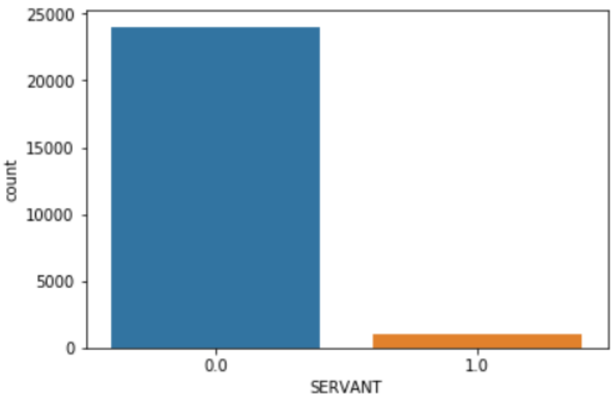
```
sns.countplot(x='AC',data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1daea2b1dd8>
```




```
sns.countplot(x='SERVANT',data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1daeb35fd30>
```



Likewise, we plot graphs for every feature but we found we could not drop it because each feature has its own significance. A user may require it so outliers play a very important role in it. But we can drop Servant because only 4% of values were 1 and 4% values were also previously null.

-DEVELOPING MODEL

LINEAR REGRESSION

```
#Linear Regression
```

```
#implementing Linear Regression without dropping any feature
X=data.drop(['rent','deposit'],axis=1)
Y=data['rent']
```

```
X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size = 0.1,random_state =10)
```

```
reg=LinearRegression()
```

```
reg.fit(X_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

```
pdt = reg.predict(X_test)
```

```
score = reg.score(X_test,y_test)
print(score*100)
```

```
65.90802103858617
```

```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, pdt))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, pdt))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, pdt)))
```

```
Mean Absolute Error: 3709.6939929122022
Mean Squared Error: 25679547.911282428
Root Mean Squared Error: 5067.499177235496
```

IMPROVING LINEAR REGRESSION BY DROPPING SERVANT

```
#IMPROVING MODEL BY DROPPING SERVANT

X=data.drop(['SERVANT','rent','deposit'],axis=1)
Y=data['rent']

X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size = 0.1,random_state =10)

reg=LinearRegression()

reg.fit(X_test,y_test)

LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)

pdt = reg.predict(X_test)

score = reg.score(X_test,y_test)
print(score*100)

70.50020156853101

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, pdt))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, pdt))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, pdt)))

Mean Absolute Error: 3421.903400028017
Mean Squared Error: 22220519.614056043
Root Mean Squared Error: 4713.8646155841225
```

USING Z-SCORE FOR REMOVING OUTLIERS

```
In [734]: data= data[(z < 3).all(axis=1)] #to remove or filter the outliers and get the clean data.

In [735]: data.shape

Out[735]: (20373, 39)

In [736]: #Applying Linear Regression after removing outliers
          x=data.drop(['rent','deposit','SERVANT'],axis=1)
          y=data['rent']
          x_train, x_test, y_train, y_test= train_test_split(x, y,test_size = 0.1, random_state = 10)

In [737]: lr=LinearRegression()
          lr.fit(x_train,y_train)

Out[737]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                   normalize=False)

In [738]: pred=lr.predict(x_test)

In [739]: print(metrics.mean_absolute_error(y_test,pred))
          print(metrics.mean_squared_error(y_test,pred))
          print(np.sqrt(metrics.mean_squared_error(y_test,pred)))
          from sklearn.metrics import mean_squared_error, r2_score
          print('Variance score: %.2f' % r2_score(y_test, pred))# r2 score should be greater

          3086.061224502846
          16944233.899638116
          4116.337437533288
          Variance score: 0.63
```

RANDOM FOREST REGRESSOR

```
#dropping servant
x=data.drop(['rent','deposit','SERVANT'],axis=1)
y=data['rent']
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)
```

```
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.fit_transform(x_test)
```

```
C:\Users\hp\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py:645: DataConversionWarning: Data with input dty
int64, float64 were all converted to float64 by StandardScaler.
  return self.partial_fit(X, y)
C:\Users\hp\Anaconda3\lib\site-packages\sklearn\base.py:464: DataConversionWarning: Data with input dtype int32, inte
were all converted to float64 by StandardScaler.
  return self.fit(X, **fit_params).transform(X)
C:\Users\hp\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py:645: DataConversionWarning: Data with input dty
int64, float64 were all converted to float64 by StandardScaler.
  return self.partial_fit(X, y)
C:\Users\hp\Anaconda3\lib\site-packages\sklearn\base.py:464: DataConversionWarning: Data with input dtype int32, inte
were all converted to float64 by StandardScaler.
  return self.fit(X, **fit_params).transform(X)
```

```
clf = RandomForestRegressor(n_estimators=45)
clf.fit(x_train, y_train)
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=45, n_jobs=None,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

```
score=clf.score(x_test,y_test)
print(score*100)
```

```
81.41518278727118
```

```
pred=clf.predict(x_test)
```

```
print(metrics.mean_absolute_error(y_test,pred))
print(metrics.mean_squared_error(y_test,pred))
print(np.sqrt(metrics.mean_squared_error(y_test,pred)))
```

```
2555.9221199999997
13377407.714760792
3657.5138707543942
```

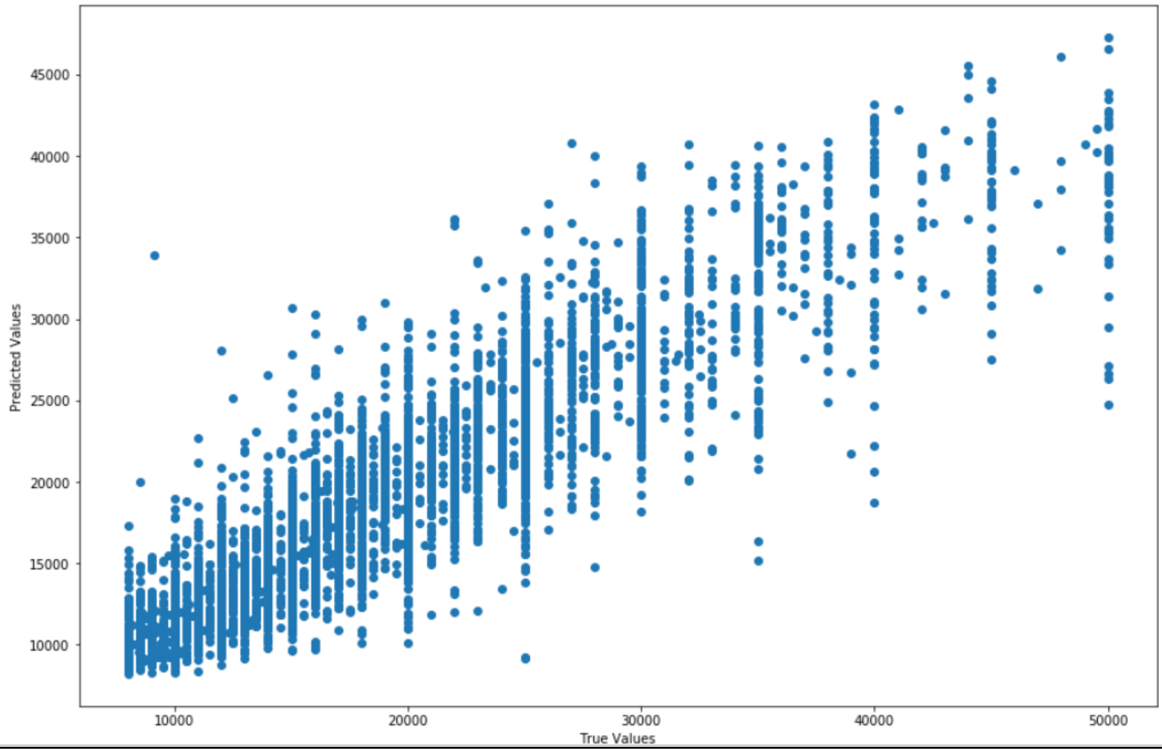
```
df=pd.DataFrame({'Actual':y_test,'predicted':pred})
df.head(10)
```

	Actual	predicted
18339	24000	21377.777778
6182	18000	15406.666667
3788	20000	19233.333333
6958	40000	39244.444444
23107	10000	9988.888889
20367	40000	20644.444444
7218	42000	38488.888889
17931	10000	9922.222222
5828	8000	8377.777778

Thus, we got maximum accuracy by Random Forest Regressor of 81.41%.

PLOTTING GRAPH BETWEEN TRUE VALUES AND PREDICTED VALUES

```
pt.scatter(y_test,pred)
pt.xlabel('True Values')
pt.ylabel('Predicted Values')
pt.show()
```



SERIALISE THE DATA AND SAVE IN DISK USING PICKLE

```
import pickle

pickle.dump(clf,open('model.pkl','wb'))

model=pickle.load(open('model.pkl','rb'))
```

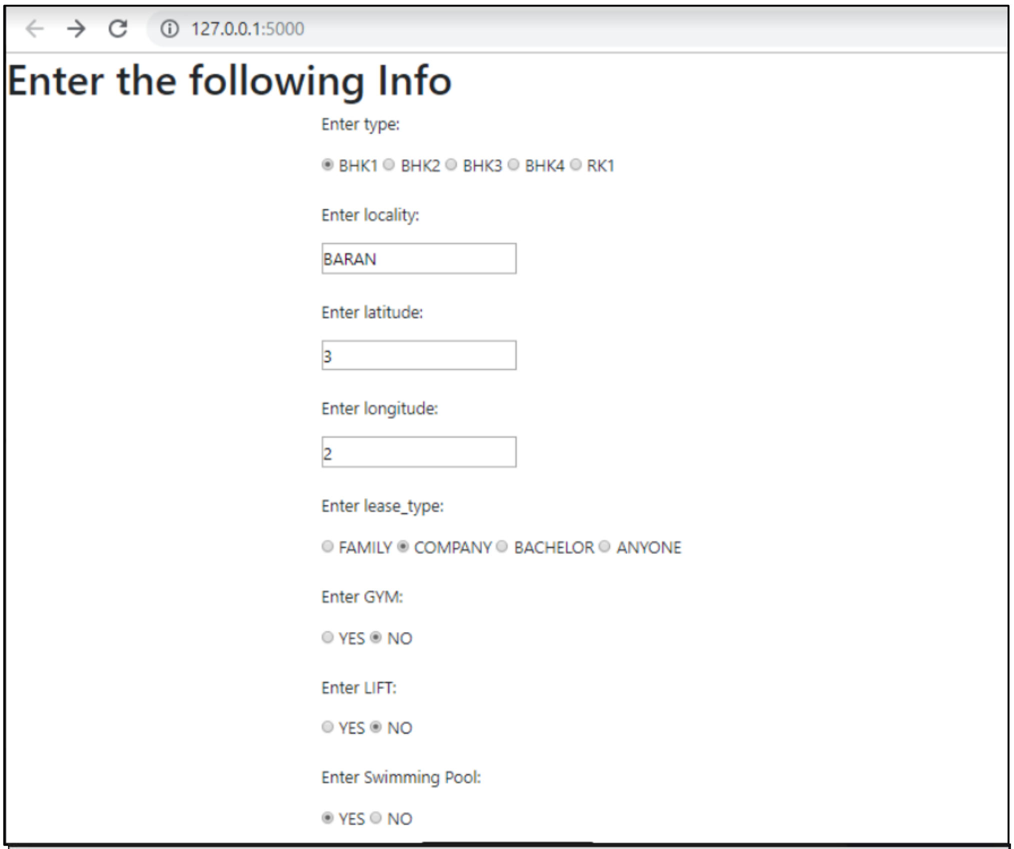
FRONT END

CODE:

```
from flask import Flask, render_template, request
import numpy as np
import pickle
app = Flask(__name__)

@app.route('/', methods=['GET','POST'])

def send():
    if request.method == 'POST':
        typ = int(request.form['type'])
        locality = request.form['locality']
        latitude = float(request.form['l  longitude: str
        longitude = float(request.form['longitude'])
        lease_type = int(request.form['lease_type'])
        gym = int(request.form['gym'])
        lift = int(request.form['lift'])
        swimming_pool = int(request.form['swimming_pool'])
        furnishing = int(request.form['furnishing'])
        Parking = int(request.form['Parking'])
        size = int(request.form['size'])
        age = int(request.form['age'])
        floor = int(request.form['floor'])
        bathroom = int(request.form['bathrooms'])
        INTERNET = int(request.form['INTERNET'])
        AC = int(request.form['AC'])
        CLUB = int(request.form['CLUB'])
        INTERCOM = int(request.form['INTERCOM'])
        CPA = int(request.form['CPA'])
        FS = int(request.form['FS'])
        SERVANT = int(request.form['SERVANT'])
```

```
        SECURITY = int(request.form['SECURITY'])
        SC = int(request.form['SC'])
        GP = int(request.form['GP'])
        PARK = int(request.form['PARK'])
        RWH = int(request.form['RWH'])
        STP = int(request.form['STP'])
        HK = int(request.form['HK'])
        PB = int(request.form['PB'])
        VP = int(request.form['VP'])
        Water_supply = int(request.form['Water_supply'])
        building_type = int(request.form['building_type'])
        balconies = int(request.form['balconies'])

        w = np.array([typ,4,lease_type,gym,lift,swimming_pool,furnishing,Parking,size,age,floor,bathroom, INTERNET,
                     AC,CLUB,INTERCOM,CPA,FS,SERVANT,SECURITY,SC,GP,PARK,RWH,STP,HK,PB,
                     VP,Water_supply,building_type,size,floor,bathroom,balconies,1,0,1])

        model = pickle.load(open('model.pkl','rb'))
        rent = model.predict(np.array(w).reshape(1,37))
        import math
        res=math.floor(rent)

        return render_template("out.html", outcome=res)

    return render_template("index.html")


if __name__ == "__main__":
    app.run()
```

FORM:



127.0.0.1:5000

Enter the following Info

Enter type:

BHK1 BHK2 BHK3 BHK4 RK1

Enter locality:

BARAN

Enter latitude:

3

Enter longitude:

2

Enter lease_type:

FAMILY COMPANY BACHELOR ANYONE

Enter GYM:

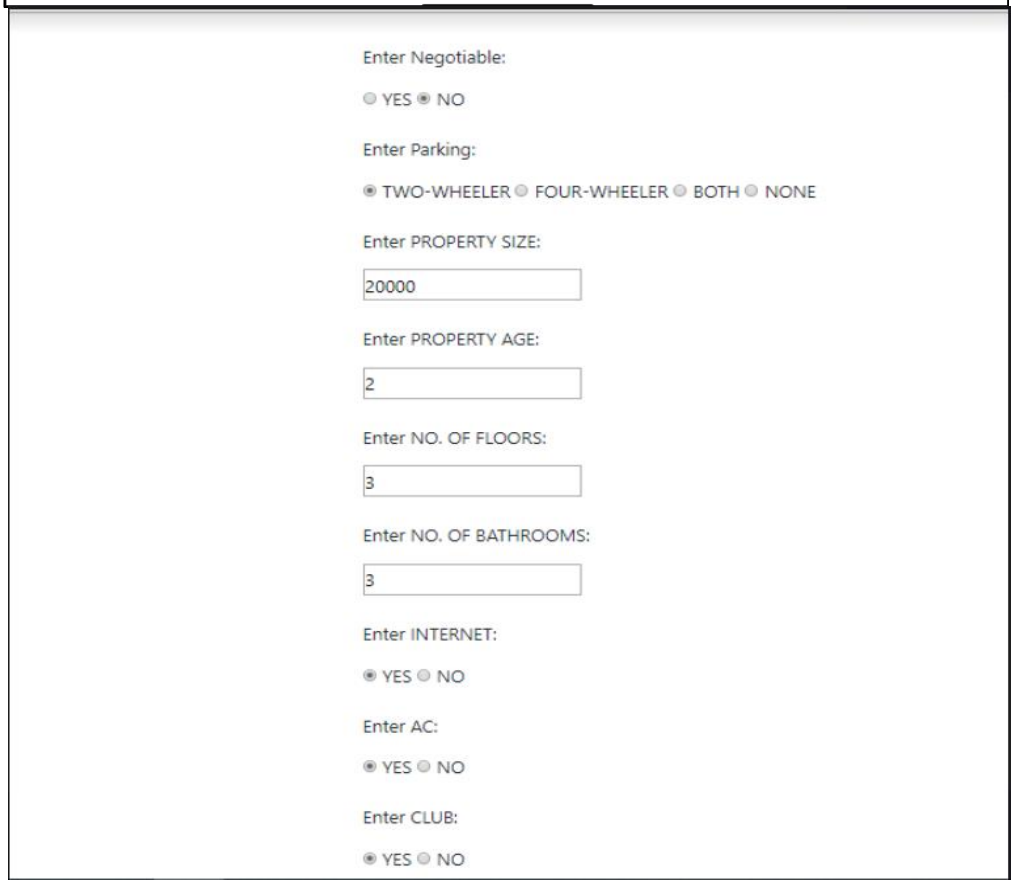
YES NO

Enter LIFT:

YES NO

Enter Swimming Pool:

YES NO



Enter Negotiable:

YES NO

Enter Parking:

TWO-WHEELER FOUR-WHEELER BOTH NONE

Enter PROPERTY SIZE:

20000

Enter PROPERTY AGE:

2

Enter NO. OF FLOORS:

3

Enter NO. OF BATHROOMS:

3

Enter INTERNET:

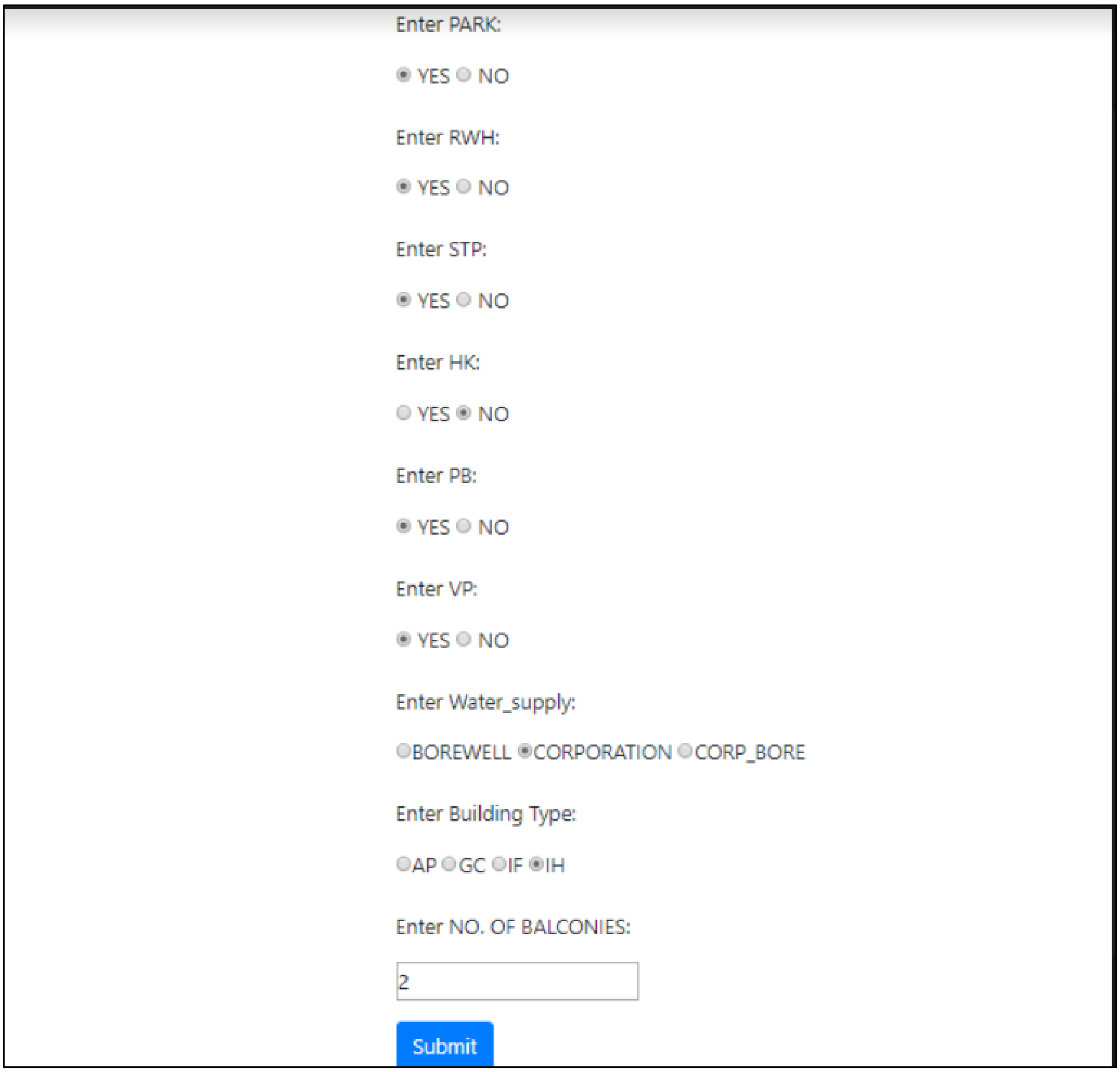
YES NO

Enter AC:

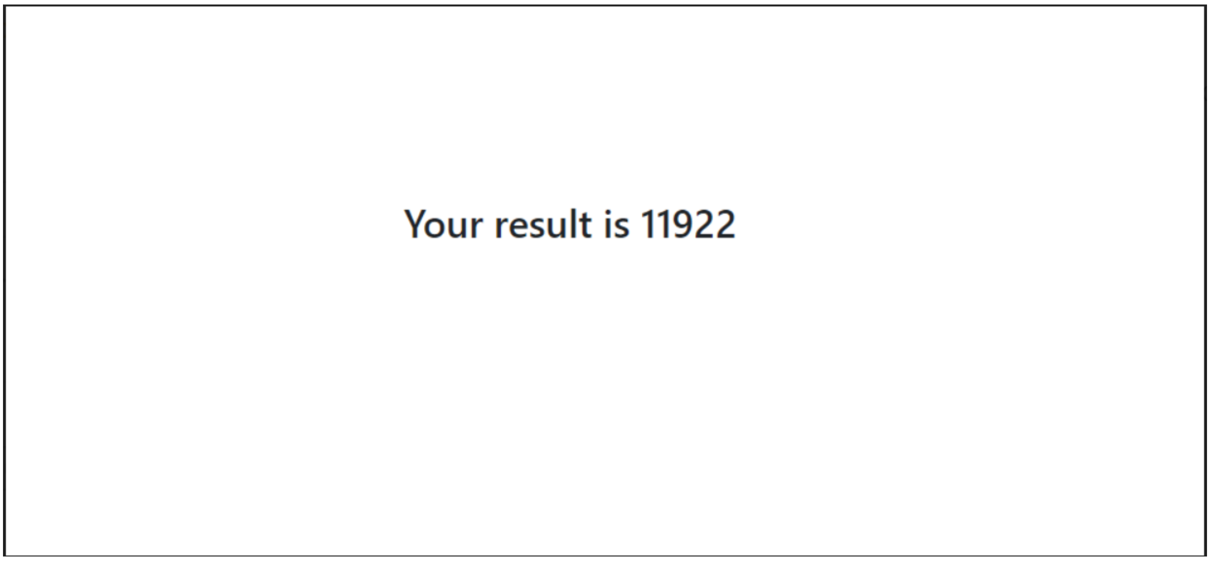
YES NO

Enter CLUB:

YES NO


Enter PARK:
YES NO
Enter RWH:
YES NO
Enter STP:
YES NO
Enter HK:
YES NO
Enter PB:
YES NO
Enter VP:
YES NO
Enter Water_supply:
BOREWELL CORPORATION CORP_BORE
Enter Building Type:
AP GC IF IH
Enter NO. OF BALCONIES:
2
Submit

OUTPUT:


Your result is 11922

CONCLUSION

Thus, our model is developed using Linear Forest Regressor which gives us the maximum accuracy of 81.18% and mean absolute error of 2568.25. We have developed the GUI for it using Pickle library and on the front end the user fills out his priorities of features and the rent is predicted using the following model and is displayed to the viewer.

Thus, our model meets the expectations of customers by displaying an estimated rent where the error varies between 9%-21%.