

HOUSE RENT PRICE PREDICTION USING IBM WATSON STUDIO MACHINE LEARNING

1. INTRODUCTION

a. OVERVIEW

Determining the house rent is very important nowadays as the price of the land and price of the house increases frequently. So, it becomes challenging to estimate the house rent. The price of a rental house helps the people to know the rent of the house based on their preferred location and also, they'll be able to choose the right place based on their budget. There are several factors that affect the price of the rental house such as the physical condition, location, landmarks, etc.

b. PURPOSE

In this project, we present a house rent prediction technique that utilizes historical data to train simple machine learning models which are more accurate and can help us predict the rent of the house. The evaluation results show that the accuracy of the models is good.

enough to be used alongside the current state-of-the-art techniques.

2. LITERATURE SURVEY

a. EXISTING PROBLEM

Nowadays as the price of the land and price of the house increases frequently So, it becomes challenging to estimate the house rent.

House Price prediction, is important to drive Real Estate efficiency. As earlier, House prices were determined by calculating the acquiring and selling price in a locality. Therefore, the House Price prediction model is very essential in filling the information gap and improve Real Estate efficiency. However, the disadvantage of this practice is that these evaluators could be biased because buyers, sellers or mortgages have bestowed interest. We therefore need an automated model of prediction that can help to predict property values without bias

b. PROPOSED SOLUTION

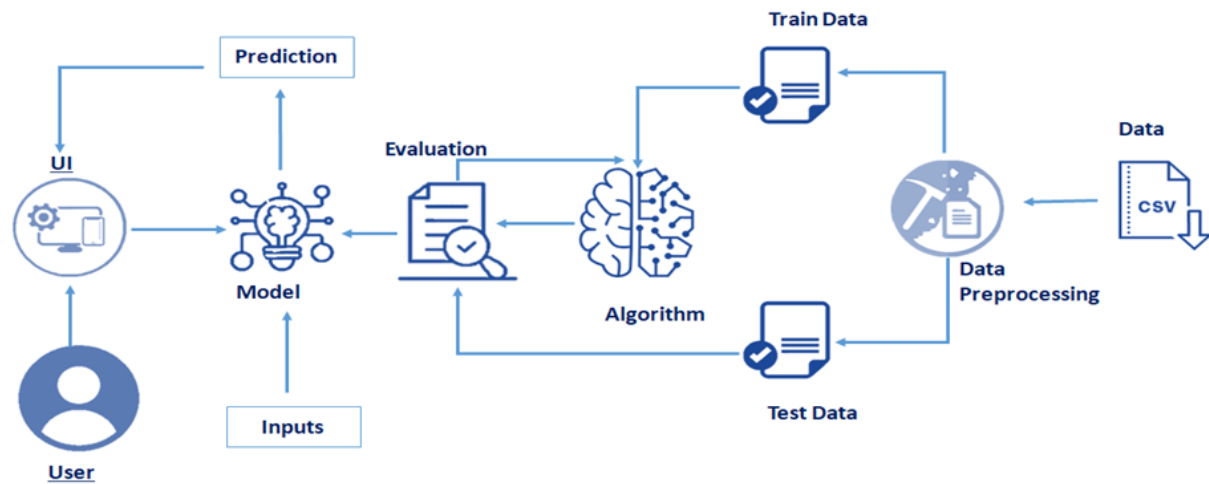
we present a house rent prediction technique that utilizes

historical data to train simple machine learning models which are more accurate and can help us predict the rent of the house. The evaluation results show that the accuracy of the models is good enough to be used alongside the current state-of-the-art techniques.

This project uses various regression techniques to predict the house rent such as Decision tree, Random Forest techniques, etc. We will train and test the data with these algorithms. From this best model is selected and saved in pkl format. We will be doing flask integration and IBM deployment. This automated model can help first - time buyers and less experienced customers to see if property rates are overrated or underrated.

3. THEORATICAL ANALYSIS

a. BLOCK DIAGRAM



b. HARDWARE /SOFTWARE DESIGNING

Hardware:

1. Processor : intel core i3
2. Ram : 4GB
3. Hard disk : 30GB
4. Operating System : Windows
5. Programming : python 3.6+

Software:

1. Anaconda Navigator
2. Jupyter Notebook
3. Spyder

1. IBM Watson studio

BM Watson Studio - IBM Watson Studio helps data scientists and analysts prepare data and build models at scale across any cloud. IBM Watson Machine Learning - IBM Watson Machine Learning helps data scientists and developers accelerate AI and machine-learning deployment. IBM Cloud Object Storage - IBM Cloud Object Storage makes it possible to store practically limitless amounts of data, simply and cost effectively.

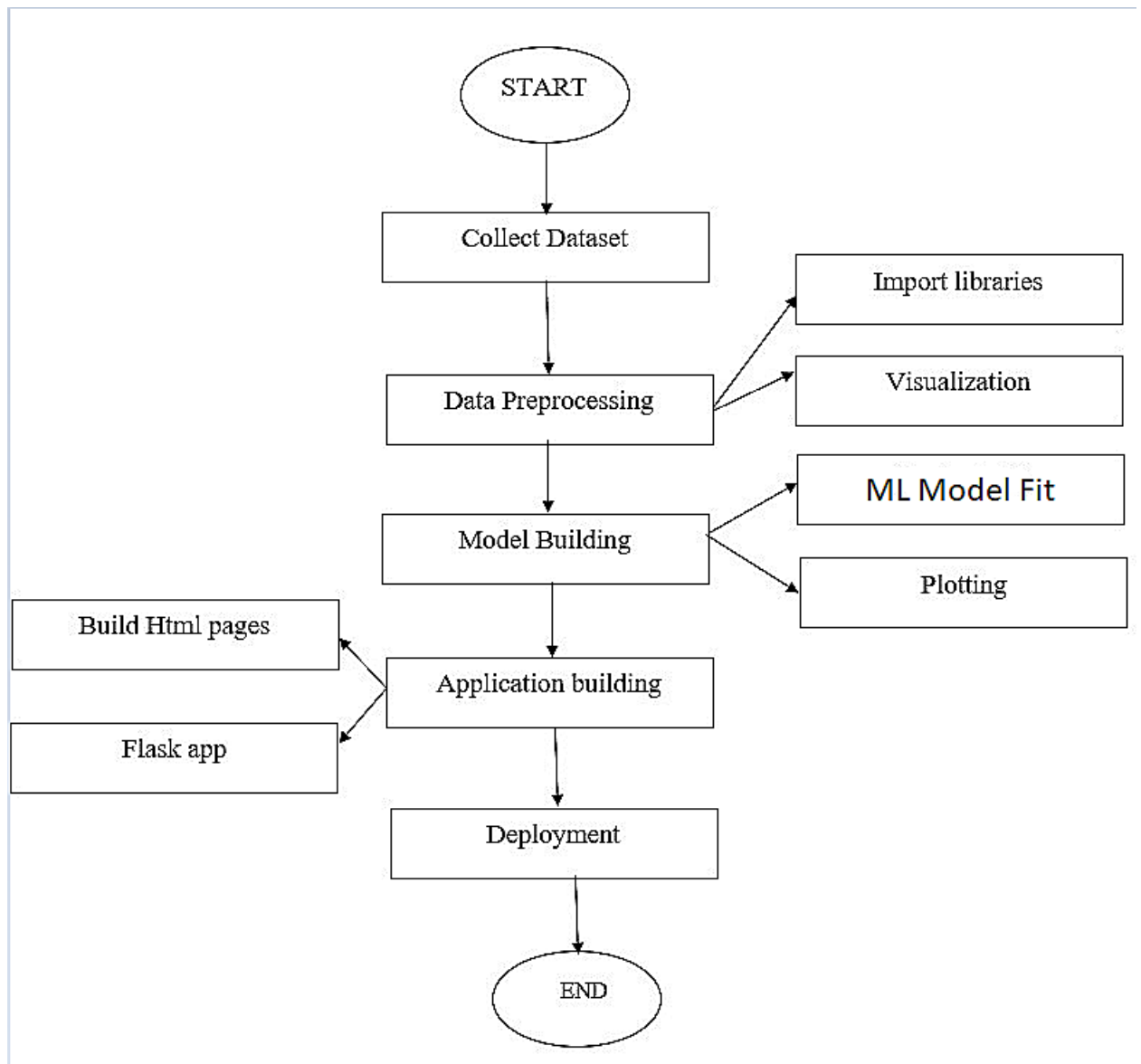
4. EXPERIMENTAL INVESTIGATIONS

In this section, we will be creating and training our model for predicting house rent price. Since there are multiple algorithms, we can use to build our model, we will compare the accuracy scores after testing and pick the most accurate algorithm.

From this list, we are using DecisionTree, RandomForest to perform our predictions. We then see which algorithm produces the highest accuracy and select it as our algorithm of choice for future use.

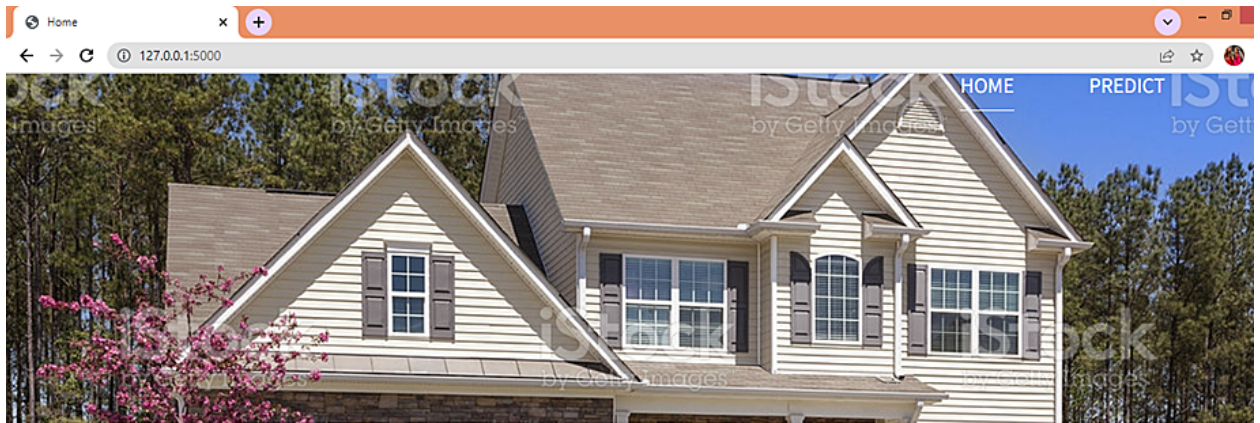
On the results of the following algorithms, we have done the conclusion the Random Forest model is the most accurate out of all the models which we have tested.

5. FLOWCHART



6. RESULT

OUTPUT:



CITY WISE HOUSE RENT PREDICTION

House rents changed with respect to city so in this project data is taken based on different-different city's so that the peoples can understand based on different-different city's what is the leaving expences in that particular city.

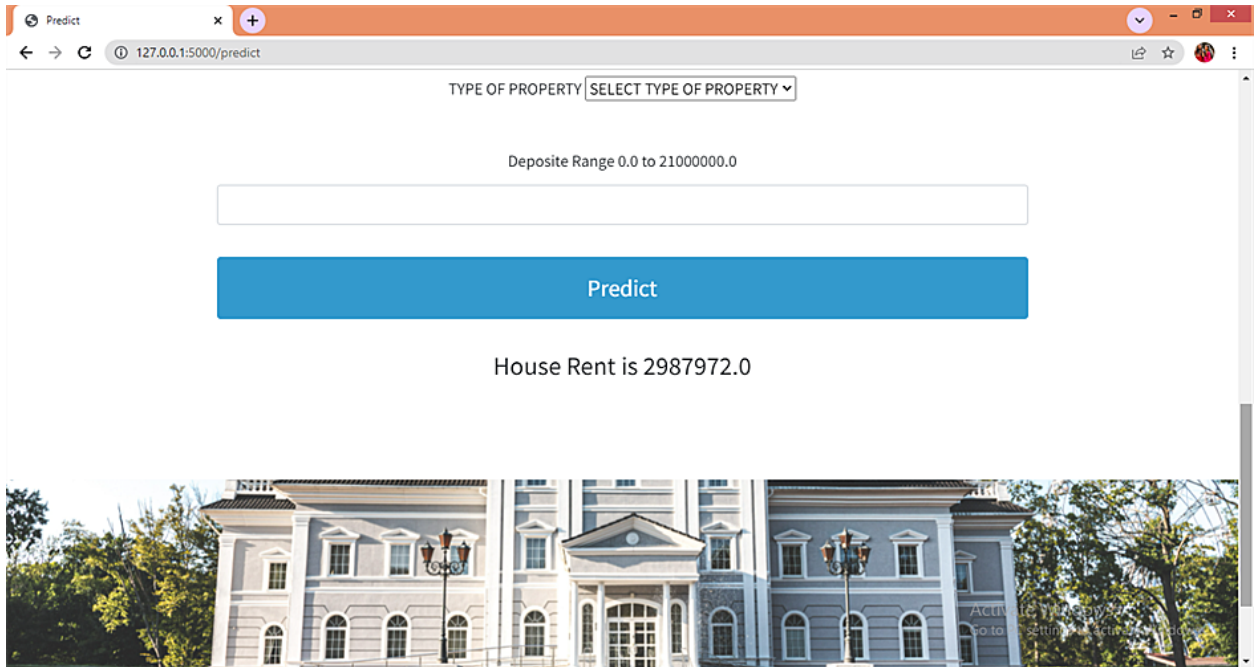
Activate Windows
Go to PC settings to activate Windows.

The main aim of this project is to create a model based on historical data and estimating the leaving expences on some

Let's click on predict button on right side top corner. Now enter the input. And click predict.

A screenshot of the 'Predict' page of the web application. The browser's address bar shows '127.0.0.1:5000/pred'. The page has a light blue header with the text 'Enter the values to predict the House Rent:'. Below this, there are several input fields and dropdown menus. The 'City' dropdown is set to 'Bangalore'. The 'SELECT NO OF BHKs (Range 1.0 to 32.0)' input field contains the value '24'. The 'House Size in sqft_per_inch: Range 1.0 to 2717.0' input field contains the value '1500'. The 'TYPE OF HOUSE' dropdown is set to 'Built-up Area'. The 'TYPE OF PROPERTY' dropdown is set to 'Residential'. The 'Deposit Range 0.0 to 21000000.0' input field contains the value '10000'. In the bottom right corner, there is a small text overlay that says 'Activate Windows Go to PC settings to activate Windows.'

Now predicted house rent will be displayed.



The screenshot shows a web browser window with the title 'Predict'. The address bar displays '127.0.0.1:5000/predict'. The main content area features a form with a dropdown menu labeled 'TYPE OF PROPERTY' and a placeholder 'SELECT TYPE OF PROPERTY'. Below this is a text input field with the label 'Deposit Range 0.0 to 21000000.0'. A large blue button labeled 'Predict' is positioned below the input field. The result of the prediction is displayed as 'House Rent is 2987972.0'. At the bottom of the page, there is a wide image of a large, ornate, light blue building with white columns and a central entrance.

7. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- High accuracy rate
- Relatively inexpensive and fast

DISADVANTAGES:

- Difficulty getting a panel experts
- Longer time for getting consensus
- Uncertain environment

8. APPLICATIONS

There are many sites in the present date that provide house rental services. Customers can rent a place from owners directly from the website. The challenge for such companies is to decide the perfect price for a place. These companies use ML to predict the price of place based on the information provided.

9. CONCLUSION

To use various machine learning algorithms for solving this problem. Out of that the Random forest is predict better accuracy than other models. “House Price Prediction Using Machine Learning” has presented to predict house price based on various features on given data. It helps people to buy house in budget and reduce loss of money.

10. FUTURE SCOPE

House rent price prediction allows to predict the rent efficiently . it will predict the rent accurately based on some factors, There are several factors that affect the price of the rental house such as the physical condition, location, landmark etc The right price of the rental house helps the people's to select the house and go for that areas to bargaining of that house.

11. BIBLIOGRAPHY

1. <https://www.sciencedirect.com/science/article/pii/S1877050920316318>
2. <https://www.tandfonline.com/doi/full/10.1080/09599916.2020.1832558>

12. APPENDIX

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import r2_score, mean_squared_error
import pickle
import seaborn as sns
from scipy import stats
plt.style.use('fivethirtyeight')
import warnings
warnings.filterwarnings('ignore')
```

- `df=pd.read_csv(r'C:\Users\Sreelakshmi
gopinathan\houserentpriceprediction\images\99acres_data.csv')`

```
df.head()
```

- `print("Numbers Of Area Type :")`

```
print()
print(df['build_up_area'].value_counts())
sns.set(rc = {'figure.figsize':(8,4)})
```

```
sns.countplot(x='build_up_area', data=df, palette = 'Set2')
```

- print("Types of Property :")

```
print()
print(df['Type_of_property'].value_counts())
sns.set(rc = {'figure.figsize':(15,4)})
sns.countplot(x='Type_of_property', data=df, palette = 'Set2')
```

- print("Property Located Based On City :- ")

```
print()
print(df['city'].value_counts())
sns.set(rc = {'figure.figsize':(10,2)})
sns.countplot(x='city', data=df, palette = 'Set2')
```

- sns.boxplot(df['monthly_rant'])
- sns.heatmap(df.corr(),annot=True)
- df.describe(include='all')
- df.info()
- df.shape
- df.isnull().sum()
- df.drop(['Baths','location_of_the_property'],axis=1,inplace=True)

```
df.Type_of_property.unique()
```

- `df = df[df.Type_of_property!='for']`

```
df = df[df.Type_of_property!='Serviced']
df = df[df.Type_of_property!='Floor']
df = df[df.Type_of_property!='for']
df.shape
```

- `df = df[df.Type_of_property!='Serviced']`

```
df.shape
```

- `df = df[df.Type_of_property!='Floor']`

```
df.shape
```

- `df.Type_of_property.unique()`

- `df['monthly_rant']=np.log1p(df['monthly_rant'])`

```
sns.distplot(df['monthly_rant'])
```

- `cty = LabelEncoder()`

```
b_u_a = LabelEncoder()
T_o_p = LabelEncoder()
df['city'] = cty.fit_transform(df['city'])
df['build_up_area']=b_u_a.fit_transform(df['build_up_area'])
df['Type_of_property']=T_o_p.fit_transform(df['Type_of_property'])
print("city",df['city'].unique())
print(cty.inverse_transform(list(df['city'].unique())))
print()
```

```

print("build_up_area:",df['build_up_area'].unique())
print(b_u_a.inverse_transform(list(df['build_up_area'].
unique()))))
print()
print("Type_of_property",
df['Type_of_property'].unique())
print(T_o_p.inverse_transform(list(df['Type_of_proper
t_y'].unique()))))
print()

```

- `x = df.drop('monthly_rant',axis=1)`

```

y = df.monthly_rant
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.
2,random_state=10)
print('Shape of xtrain {}'.format(xtrain.shape))
print('Shape of xtest {}'.format(xtest.shape))
print('Shape of ytrain {}'.format(ytrain.shape))
print('Shape of ytest {}'.format(ytest.shape))

```

- `st=StandardScaler()`

```

xtrain_scaled=st.fit_transform(xtrain)
xtest_scaled=st.transform(xtest)

```

- `def linear_reg(xtrain_scaled,xtest_scaled,ytrain,ytest):`

```

    lr=LinearRegression()
    lr.fit(xtrain_scaled,ytrain)
    ypred=lr.predict(xtest_scaled)
    score=r2_score(ytest,ypred)
    rmse=np.sqrt(mean_squared_error(ytest,ypred))
    print('***Linear Regression model***')

```

```
        print('Score for Linear Regression model is  
{0}'.format(score))  
        print('RMSE for Linear Regression model is  
{0}'.format(rmse))
```

- def

```
random_forest_regressor(xtrain_scaled,xtest_scaled,ytrain,ytest):
```

```
    rf=RandomForestRegressor()  
    rf.fit(xtrain_scaled,ytrain)  
    ypred=(rf.predict(xtest_scaled))  
    score=r2_score(ytest,ypred)  
    rmse=np.sqrt(mean_squared_error(ytest,ypred))  
    print('***Random Forest Regressor Model***')  
    print('Score for Random Forest Regressor Model is  
{0}'.format(score))  
    print('RMSE for Random Forest Regressor Model is  
{0}'.format(rmse))
```

- def

```
gradient_boosting_regressor(xtrain_scaled,xtest_scaled,ytrain,ytest  
):
```

```
    gb=GradientBoostingRegressor()  
    gb.fit(xtrain_scaled,ytrain)  
    ypred=(gb.predict(xtest_scaled))  
    score=r2_score(ytest,ypred)  
    rmse=np.sqrt(mean_squared_error(ytest,ypred))  
    print('***Gradient Boosting Regressor Model***')  
    print('Score for Gradient Boosting Regressor Model is  
{0}'.format(score))  
    print('RMSE for Gradient Boosting Regressor Model is
```

```
{}'.format(rmse))
```

- ```
def model_compare(xtrain_scaled,xtest_scaled,ytrain,ytest):

 linear_reg(xtrain_scaled,xtest_scaled,ytrain,ytest)
 print('-'*100)

 random_forest_regressor(xtrain_scaled,xtest_scaled,ytrain,ytest)
 print('-'*100)

 gradient_boosting_regressor(xtrain_scaled,xtest_scaled,ytrain,ytest)
 print('-'*100)
```
- ```
model_compare(xtrain_scaled,xtest_scaled,ytrain,ytest)
```
- ```
rf=RandomForestRegressor()

rf.fit(xtrain_scaled,ytrain)
ypred=(rf.predict(xtest_scaled))
```
- ```
pickle.dump(rf,open('model.pkl','wb'))
```