Real Estate Valuation Using ML

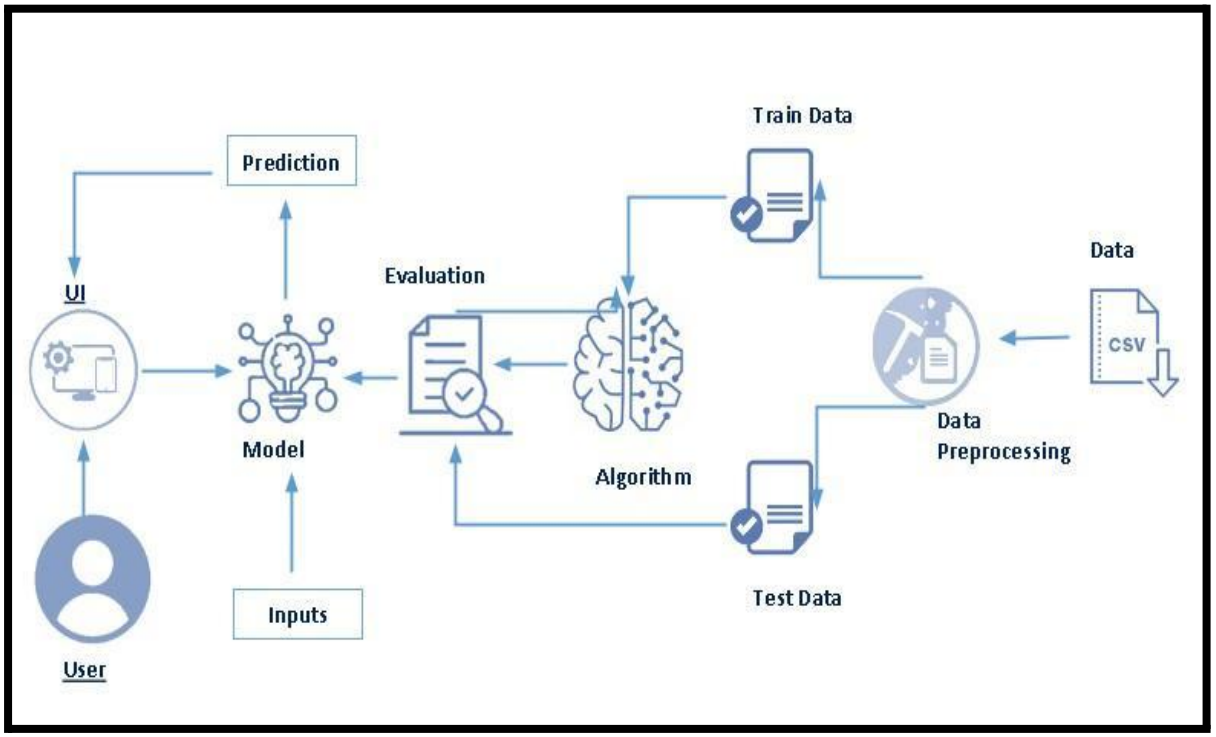
Project Description:

Real estate prices are observed when properties change hands. The high costs and large average volume of a typical transaction (as compared to equity markets for example), lead to infrequent observations for the same asset. In between transactions, real estate professionals and investors need to rely on valuations — the most likely price to be obtained in the market, had the property been put up for sale. It's a hypothetical value, not the actual registered price.

The AI wave brings unique business and ethical challenges, challenges to which real estate will not remain immune. More importantly, it indicates the increased role played by high purity timely data.. Future business models should therefore design new products and services not by obfuscating the value of this input to avoid compensating users for their data contribution, but by formally recognizing its worth and organizing the proper market needed to transact it.

We will be using classification algorithms such as Decision tree, Random forest, GradientBoost, AdaBoost and xgboost. 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.

Technical Architecture:



Pre requisites:

To complete this project, you must required following software's, concepts and packages

- **Anaconda navigator :**
  - Refer the link below to download anaconda navigator
  - Link : https://youtu.be/1ra4zH2G4o0
- **Python packages:**
  - Open anaconda prompt as administrator
  - Type "pip install numpy" and click enter.
  - Type "pip install pandas" and click enter.
  - Type "pip install sklearn" and click enter.
  - Type "pip install matplotlib" and click enter.
  - Type "pip install xgboost" and click enter.
  - Type "pip install pickle" and click enter.
  - Type "pip install seaborn" and click enter.
  - Type "pip install Flask" and click enter.

Prior Knowledge:

You must have prior knowledge of following topics to complete this project.

- **ML Concepts**
  - Supervised learning: https://www.javatpoint.com/supervised-machine-learning
  - Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning

    - Regression and classification
    - Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm
    - Random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm
    - KNN: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
    - Xgboost: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/
    - Evaluation metrics: https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/
- **Flask Basics** : https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Objectives:

By the end of this project you will:

- Know fundamental concepts and techniques used for machine learning.
- Gain a broad understanding about data.
- Have knowledge on pre-processing the data/transformation techniques on outlier and some visualization concepts.

Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

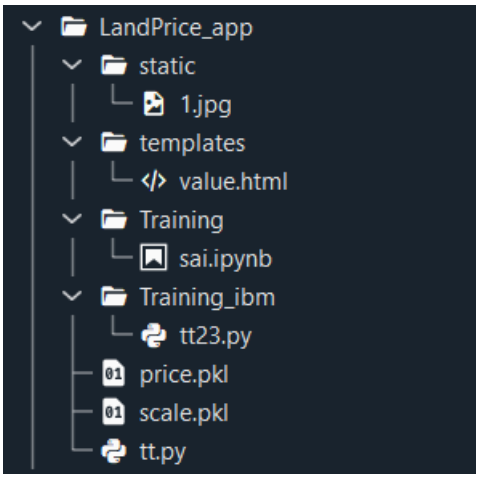
To accomplish this, we have to complete all the activities listed below,

- Data collection
    - Collect the dataset or create the dataset
- Visualizing and analyzing data
    - Univariate analysis
    - Bivariate analysis
    - Multivariate analysis
    - Descriptive analysis
- Data pre-processing
    - Checking for null values
    - Handling outlier
    - Handling categorical data

    - Splitting data into train and test
- Model building
    - Import the model building libraries
    - Initializing the model
    - Training and testing the model
    - Evaluating performance of model
    - Save the model
- Application Building
    - Create an HTML file
    - Build python code

Project Structure:

Create the Project folder which contains files as shown below



- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- price.pkl is our saved model and scale.pkl is our standard scaler file. Further we will use this model for flask integration.
- Training folder contains model training files and Training_ibm folder contains IBM deployment files.

Milestone 1: Data Collection

ML depends heavily on data, It is most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

Activity 1: Download the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used drug200.csv data. This data is downloaded from archive.ics.uci.edu. Please refer the link given below to download the dataset.

Link: https://archive.ics.uci.edu/ml/datasets/Real+estate+valuation+data+set

Milestone 2: Visualizing and analysing the data

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analysing techniques.

Note: There is n number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1: Importing the libraries

Import the necessary libraries as shown in the image(optional) .

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report, confusion_m
import warnings
import pickle
from scipy import stats
warnings.filterwarnings('ignore')
plt.style.use('fivethirtyeight')
```

Activity 2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read_excel() to read the dataset. As a parameter we have to give the directory of excel file.

```
import pandas as pd
import numpy as np
```

```
dt = pd.read_excel('realstate_dataset.xlsx')
```

```
dt.head()
```

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number c convenienc store
0	1	2012.916667	32.0	84.87882	1
1	2	2012.916667	19.5	306.59470	

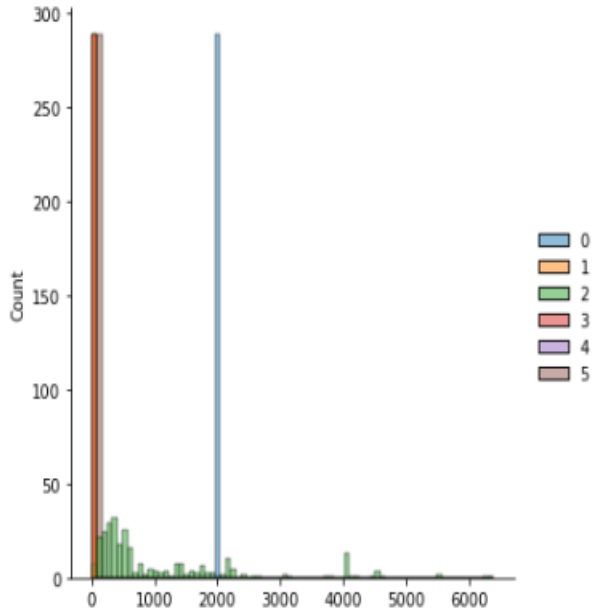
Activity 3: Univariate analysis

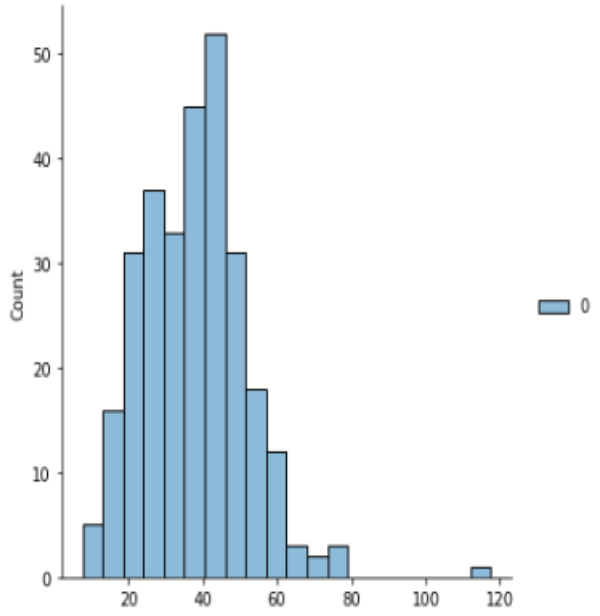
In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as distplot .

- Seaborn package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.

```
sns.displot(x_train)
sns.displot(y_train)
```

```
<seaborn.axisgrid.FacetGrid at 0x174e6089a30>
```





Activity 4: Bivariate analysis

To find the relation between two features we use bivariate analysis. Here we are visualizing the relationship between transaction date ,house age, distance to the nearest MRT station, number of convenience stores in the living circle on foot, geographic coordinate, latitude, geographic coordinate, longitude.

```
dt.head()
```

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	1	2012.916667	32.0	84.87882	10	24.98298	121.54024	37.9
1	2	2012.916667	19.5	306.59470	9	24.98034	121.53951	42.2
2	3	2013.583333	13.3	561.98450	5	24.98746	121.54391	47.3
3	4	2013.500000	13.3	561.98450	5	24.98746	121.54391	54.8
4	5	2012.833333	5.0	390.56840	5	24.97937	121.54245	43.1

Activity 5: Multivariate analysis

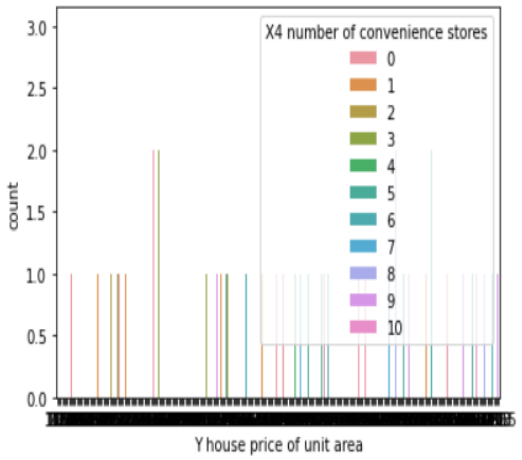
In simple words, multivariate analysis is to find the relation between multiple features. Here we have used countplot from seaborn package.

- From the below image, we came to a conclusion that most of houses has 3 or 0 number of convenience stores.

```
sns.countplot(dt['Y house price of unit area'], hue = dt['X4 number of convenience stores'])
```

```
C:\Users\k Sai Shashank\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

```
<AxesSubplot:xlabel='Y house price of unit area', ylabel='count'>
```



Activity 6: Descriptive analysis

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

```
dt.describe()
```

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
count	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000
mean	207.500000	2013.148953	17.712560	1083.885689	4.094203	24.969030	121.533361	37.980193
std	119.655756	0.281995	11.392485	1262.109595	2.945562	0.012410	0.015347	13.606488
min	1.000000	2012.666667	0.000000	23.382840	0.000000	24.932070	121.473530	7.600000
25%	104.250000	2012.916667	9.025000	289.324800	1.000000	24.963000	121.528085	27.700000
50%	207.500000	2013.166667	16.100000	492.231300	4.000000	24.971100	121.538630	38.450000
75%	310.750000	2013.416667	28.150000	1454.279000	6.000000	24.977455	121.543305	46.600000
max	414.000000	2013.583333	43.800000	6488.021000	10.000000	25.014590	121.566270	117.500000

Milestone 3: Data Pre-processing

As we have understood how the data is lets pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much of randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling categorical data
- Handling outliers
- Scaling Techniques
- Splitting dataset into training and test set

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Activity 1: Checking for null values

- Let’s find the shape of our dataset first, To find the shape of our data, df.shape method

is used. To find the data type, df.info() function is used.

```
dt.shape
```

```
(414, 8)
```

```
dt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 414 entries, 0 to 413
Data columns (total 8 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   No                                      414 non-null    int64
 1   X1 transaction date                     414 non-null    float64
 2   X2 house age                            414 non-null    float64
 3   X3 distance to the nearest MRT station  414 non-null    float64
 4   X4 number of convenience stores         414 non-null    int64
 5   X5 latitude                             414 non-null    float64
 6   X6 longitude                            414 non-null    float64
 7   Y house price of unit area              414 non-null    float64
dtypes: float64(6), int64(2)
memory usage: 26.0 KB
```

- For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function to it. From the below image we found that there are no null values present in our dataset. So we can skip handling of missing values step.

```
dt.isnull().any()
```

```
No                                      False
X1 transaction date                     False
X2 house age                            False
X3 distance to the nearest MRT station  False
X4 number of convenience stores         False
X5 latitude                             False
X6 longitude                            False
Y house price of unit area              False
dtype: bool
```

Let's look for any outliers in the dataset

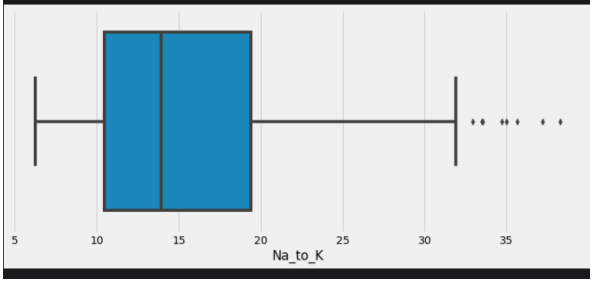
Activity 2: Handling outliers

With the help of boxplot, outliers are visualized. And here we are going to find upper bound and lower bound of Na_to_K feature with some mathematical formula.

- From the below diagram, we could visualize that Na_to_K feature has outliers. Boxplot from seaborn library is used here.

```
# Finding outliers

plt.figure(figsize=(12,5))
sns.boxplot(df['Na_to_K'])
plt.show()
```



- To find upper bound we have to multiply IQR (Interquartile range) with 1.5 and add it with 3rd quantile. To find lower bound instead of adding, subtract it with 1st quantile. Take image attached below as your reference.

```
lw=dt['X3 distance to the nearest MRT station'].quantile(0.05)
up=dt['X3 distance to the nearest MRT station'].quantile(0.95)
dt=(dt['X3 distance to the nearest MRT station'] >lw) & (dt['X3 distance to the nearest MRT station'] < up)
```

Activity 3: Handling Categorical Values

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding. But no categorical values in our dataset.

Activity 4: Splitting data into train and test

Now let's split the Dataset into train and test sets

Changes: first split the dataset into x and y and then split the data set

Here x and y variables are created. On x variable, dt is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=20)
```

Milestone 4: Model Building

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. for this project we are applying four classification algorithms. The best model is saved based on its performance.

Activity 1: Decision tree model

A function named decisionTree is created and train and test data are passed as the parameters. Inside the function, DecisionTreeClassifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```
from sklearn.tree import DecisionTreeRegressor
x_train,x_test,y_train,y_test = train_test_split(x,y
```

```
dtr = DecisionTreeRegressor(random_state=40)
dtr.fit(x_train,y_train)
```

```
DecisionTreeRegressor(random_state=40)
```

```
y_pred=dtr.predict(x_test)
```

Activity 2: Random forest model

A function named randomForest is created and train and test data are passed as the parameters. Inside the function, RandomForestClassifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```
from sklearn.ensemble import RandomForestRegressor
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=
```

```
rf=RandomForestRegressor(n_estimators=20,random_state=80)
rf.fit(x_train,y_train)
```

```
C:\Users\k Sai Shashank\AppData\Local\Temp\ipykernel_97176\1814452946.py:2: Dat
aConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ravel().
  rf.fit(x_train,y_train)
```

```
RandomForestRegressor(n_estimators=20, random_state=80)
```

```
y_pred=rf.predict(x_test)
```

Activity 3 : Multilinear Regression

A function named MR is created and train and test data are passed as the parameters. Inside the function, LinearRegression algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```
: from sklearn.linear_model import LinearRegression
: lr = LinearRegression()
: from sklearn.model_selection import train_test_split
: x_train,x_test,y_train,y_test = train_test_split(x,y,tes
: lr.fit(x_train,y_train)
: LinearRegression()
: y_pred=lr.predict(x_test)
```

Activity 4: Xgboost model

A function named xgboost is created and train and test data are passed as the parameters. Inside the function, XGBRegressor algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```
import xgboost
```

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

```
xg = xgboost.XGBRegressor(objective='reg:linear',n_estimators=50,seed=23)
```

```
xg.fit(x_train,y_train)
```

```
[19:38:25] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.6.
0/src/objective/regression_obj.cu:203: reg:linear is now deprecated in favor of
reg:squarederror.

XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=50, n_jobs=0,
             num_parallel_tree=1, objective='reg:linear', predictor='auto',
             random_state=23, reg_alpha=0, ...)
```

```
y_pred = xg.predict(x_test)
```

Activity 5: GradientBoost model

A function named xgboost is created and train and test data are passed as the parameters. Inside the function, GradientBoostingRegressor algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```
from sklearn.ensemble import GradientBoostingRegressor
```

```
gbr = GradientBoostingRegressor(max_depth=3,n_estimators=18,learning_rate=1)
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=
```

```
gbr.fit(x_train,y_train)
```

```
C:\Users\k Sai Shashank\anaconda3\lib\site-packages\sklearn\ensemble\_gb.py:49
4: DataConversionWarning: A column-vector y was passed when a 1d array was expe
cted. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

GradientBoostingRegressor(learning_rate=1, n_estimators=18)
```

Activity 5: ADABoost model

A function named ADAboost is created and train and test data are passed as the parameters. Inside the function, ADARegressor algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```
from sklearn.ensemble import AdaBoostRegressor
```

```
adr = AdaBoostRegressor(n_estimators=10,learning_rate=1,random_state=20)
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=
adr.fit(x_train,y_train)
```

```
C:\Users\k Sai Shashank\anaconda3\lib\site-packages\sklearn\utils\validation.p
y:993: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using rave
l().
  y = column_or_1d(y, warn=True)
```

```
AdaBoostRegressor(learning_rate=1, n_estimators=10, random_state=20)
```

```
y_pred=adr.predict(x_test)
```

Now let's see the performance of all the models and save the best model

Activity 5: Compare the model

For comparing the above four models compareModel function is defined.

After calling the function, the results of models are displayed as output. From the Six model random forest is performing well. From the below image, We can see the accuracy of the 78% to the model..

Compare the models

```
print("The accuracy of Muli linear Regression" ,{r2_score(y_pred,y_testm)})
print("The accuracy of Decision Tree Regression" ,{r2_score(y_predd,y_testd)})
print("The accuracy of Random Forest Regression" ,{r2_score(y_predr,y_testr)})
print("The accuracy of AdaBoost Regression" ,{r2_score(y_preda,y_testa)})
print("The accuracy of Gradient Boost Regression" ,{r2_score(y_predg,y_testg)})
print("The accuracy of XGBoost Regression" ,{r2_score(y_predx,y_testx)})
```

```
The accuracy of Muli linear Regression {0.4145698019981675}
The accuracy of Decision Tree Regression {0.6228546701807869}
The accuracy of Random Forest Regression {0.7823834718984976}
The accuracy of AdaBoost Regression {0.6429258588012791}
The accuracy of Gradient Boost Regression {0.6268913804382098}
The accuracy of XGBoost Regression {0.7340603736466449}
```

Activity 6: Evaluating performance of the model and saving the model

From sklearn , metrics is used to evaluate the score of the model. On the parameters, we have given rf (model name), x, y, cv (as 5 folds). Our model is performing well. So, we are saving the model by pickle.dump().

Note: To understand cross validation, refer this link. https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85.

```
RandomForestRegressor(n_estimators=20, random_state=80)
```

```
y_pred=rf.predict(x_test)
```

```
ac = r2_score(y_pred,y_test)
rm=mean_absolute_error(y_pred,y_test)
ms=mean_squared_error(y_pred,y_test)
```

```
ac
```

```
0.7823834718984976
```

```
rm
```

```
4.294445333333332
```

```
ms
```

```
31.404086267111108
```

Milestone 5: Application Building

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building serverside script

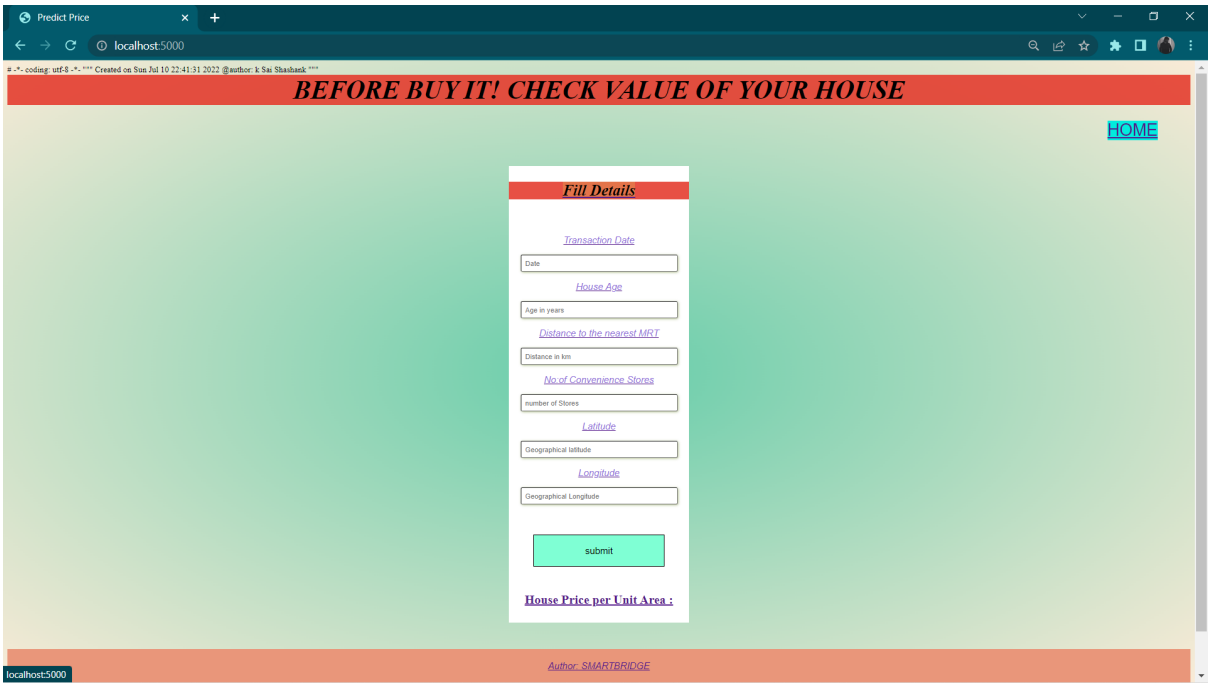
Activity1: Building Html Pages:

For this project create three HTML files namely

- Value.html

and save them in templates folder.

Let's see how our value.html page looks like:



Now when you click on submit button from left bottom button you will show output

Activity 2: Build Python code:

Import the libraries

```
"""
from flask import Flask,render_template,request
import pickle
#import numpy as np
```

Load the saved model. Importing flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.

```
app=Flask(__name__)

md = pickle.load(open("price.pkl","rb"))
sc = pickle.load(open("scale.pkl","rb"))
```

Render HTML page:

```
@app.route('/')
def hello_world():
    return render_template("value.html")
```

Here we will be using declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with value.html function. Hence, when the home page of the web server is opened in browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

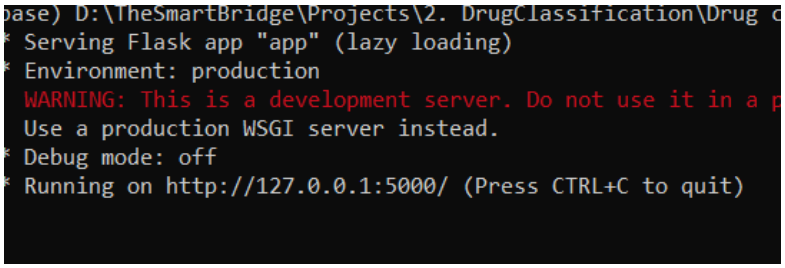
Retrieves the value from UI:

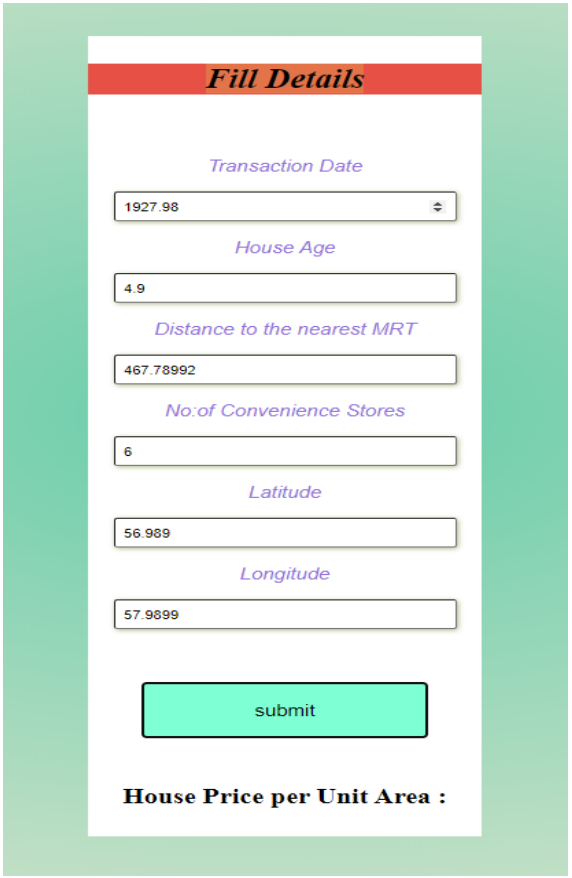
```
@app.route('/value',methods = ["POST"])
def value():
    p = request.form["a"]
    q= request.form["b"]
    r = request.form["c"]
    s = request.form["d"]
    t = request.form["e"]
    u = request.form["f"]
    data = [[p,q,r,s,t,u]]
    dt=(sc.fit_transform(data))
    prd=md.predict(dt)
    return render_template("value.html",y=prd)
app.run(debug = True)
```

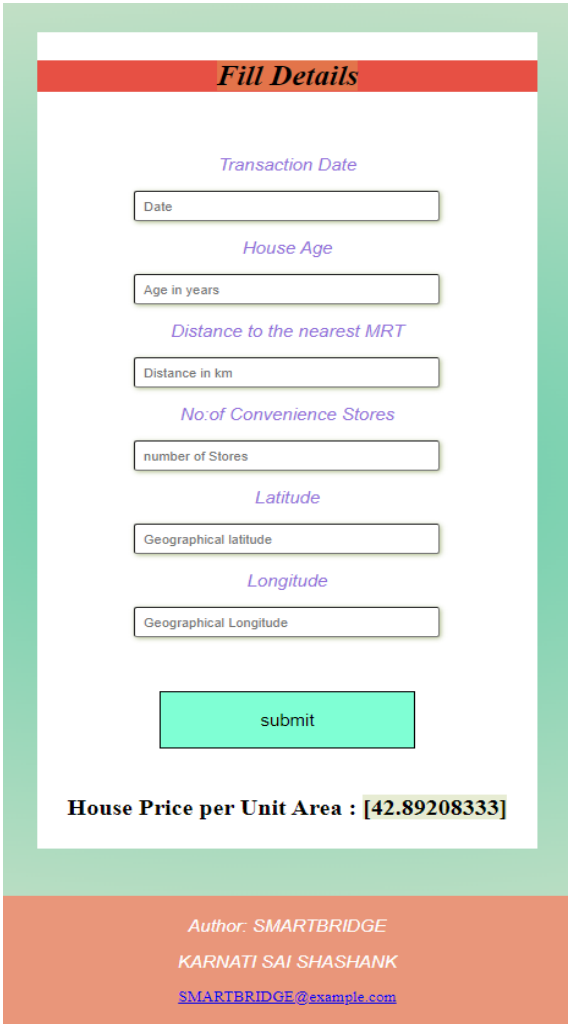
Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the md.predict() function. This function returns the prediction. And this prediction value will rendered to the text that we have mentioned in the value.html page .

Activity 3: Run the application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python tt.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top right corner, enter the inputs, click on the submit button, and see the result/prediction on the web.




Fill Details
Transaction Date
1927.98
House Age
4.9
Distance to the nearest MRT
467.78992
No:of Convenience Stores
6
Latitude
56.989
Longitude
57.9899
submit
House Price per Unit Area :


Fill Details
Transaction Date
Date
House Age
Age in years
Distance to the nearest MRT
Distance in km
No:of Convenience Stores
number of Stores
Latitude
Geographical latitude
Longitude
Geographical Longitude
submit
House Price per Unit Area : [42.89208333]
Author: SMARTBRIDGE
KARNATI SAI SHASHANK
SMARTBRIDGE@example.com