PROJECT: Predicting house prices using machine learning
Phase 5: - Project Documentation & Submission

1. **DOCUMENTATION: -**

- **Problem Statement:**

The housing market is an important and complex sector that impacts people's lives in many ways. For many individuals and families, buying a house is one of the biggest investments they will make in their lifetime. Therefore, it is essential to accurately predict the prices of houses so that buyers and sellers can make informed decisions. This project aims to use machine learning techniques to predict house prices based on various features such as location, square footage, number of bedrooms and bathrooms, and other relevant factors.

- **Design Thinking Process:**

Data Source: Choose a dataset containing information about houses, including features like location, square footage, bedrooms, bathrooms, and price.
DATASET link:
(https://github.com/Tahira77/project1/files/12777164/HousePricePrediction.xlsx)
Data Preprocessing: Clean and preprocess the data, handle missing values, and convert categorical features into numerical representations.
Feature Selection: Select the most relevant features for predicting house prices.
Model Selection: Choose a suitable regression algorithm (e.g., Linear Regression, Random Forest Regressor) for predicting house prices.
Here we are choosing Linear regression for predicting house price.
Model Training: Train the selected model using the preprocessed data.
Evaluation: Evaluate the model's performance using metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared.
TECHNOLOGY USED:
Machine Learning:
"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if it performance at tasks in T, as measured by P, improves with experience E."
Example: predicting whether the given object is pen or pencil?
E=the experience or predicting many pens and pencil
T=the task of predict pen or pencil
P=the probability that of whether it is a pen or pencil.
In general, any machine learning problem can be assigned to one of the two broad classifications:
Supervised learning .
Unsupervised learning.

TOOLS USED:

- PYTHON
- TENSOR FLOW
- Android Studio
- Anaconda

HOW IT WORKS?

Collecting the data: First step was to collect data we collected data from different sources and merged them together to form our training data set.
Then we trained the model using machine learning algorithm which in this case is multiple linear regression.
Based on the generated graphs we predict the cost of the house.

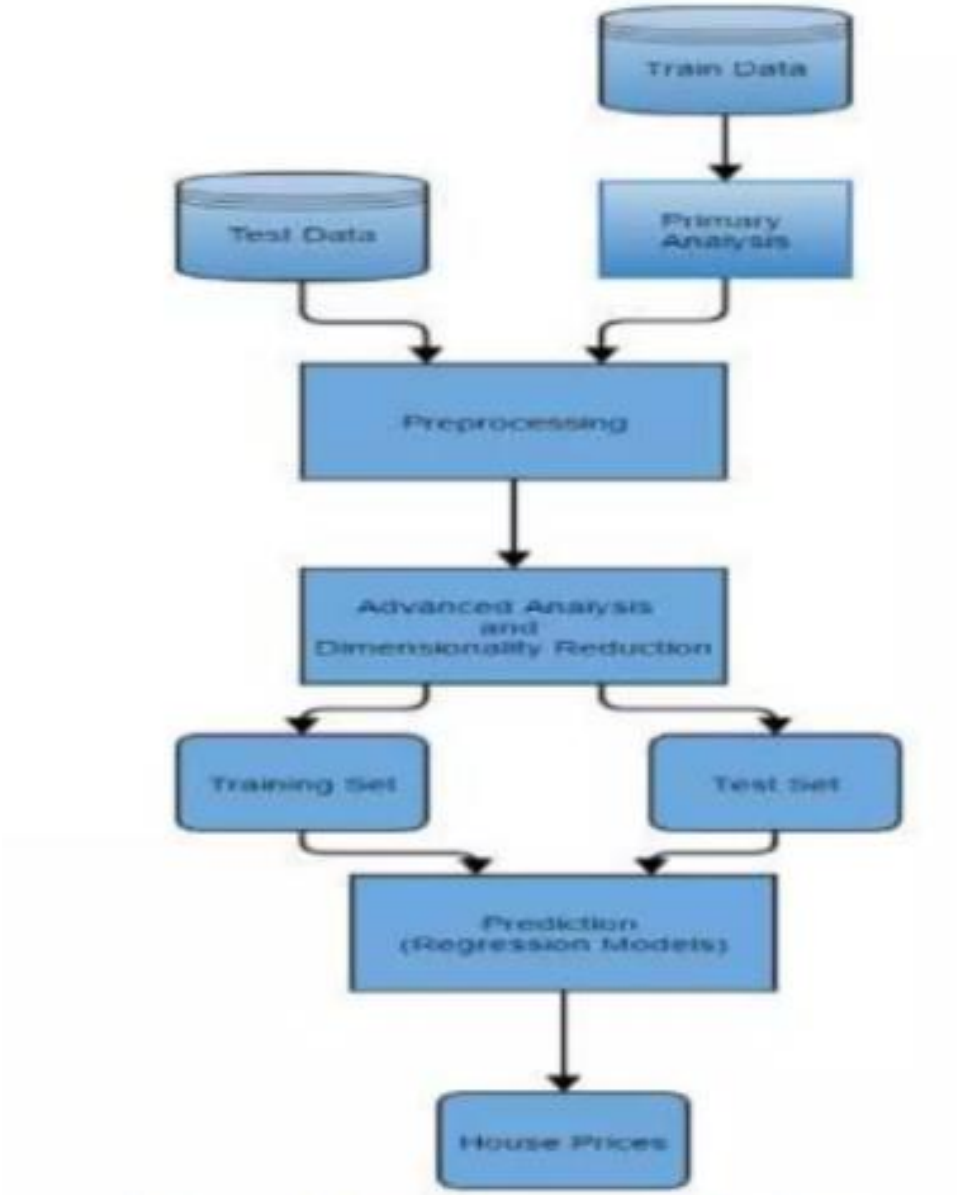


Figure 1. Architecture Diagram

FUTURE WORK:

Our model had a low rmse score, but there is still room for improvement. In a real world scenario, we can use such a model to predict house prices. This model should check for new data, once In a month, and incorporate them to expand the dataset and produce better results.
We can try out other dimensionality reduction techniques like Univariate feature selection and Recursive feature elimination in the initial stage.

We can try out other advanced regression techniques, like random forest and Bayesian ridge algorithm, for prediction. Since the data is highly correlated, we should also try Elastic Net regression technique.

PROPOSED SYSTEM: -

System includes set of codes that processes on the available dataset to effectively predicts the value of outcome depending upon user input using the concept of Linear regression.
Efficient and proper use of the system can eradicate the cases where the customers get cheated by the real estate agents in terms of house prices.
Proper usage of model is beneficial to both the customer as well as agents guiding customers.

ADVANTAGES: -

Good interpretability
Its very simple to understand
Space complexity is very low it just needs to save the weights at the end of training. It's a high latency algorithm.

- **Phases Of Development:**

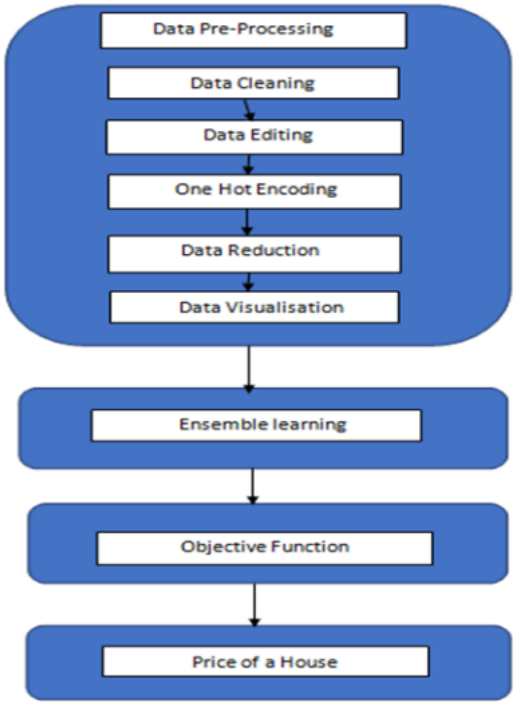


Fig:Different phases of development

Dataset Used:

Here we are using Kaggle dataset of USA_Housing for predicting house prices using machine learning:

file:///C:\Users\Thahira\Desktop\MyProject\USA_housing.csv

Following are the parameters in dataset for house price prediction of 'USA'

- Avg. Area Income
- House Age

- Number of Rooms
- Avg. Area Number of Bedrooms
- Area Population
- Price
- Address
- 24*7 security

Data Preprocessing:

Data pre-processing is a process used for refining data before fed into model. Data pre-processing is vaguely divided into four stages called a) Data cleaning b) Data Editing c) Data reduction d) Data wrangling. Data cleaning is process where inaccurate data or if a data field is empty, then value is filled using mean or median or entire record is deleted from data. If data is recorded manually these problems tend to happen. Calculate the mean value considering the value of attributes and number of records in the data. Data editing is process where outliers are picked from data and eradicated. Outliers are mainly recorded in data mainly due to experimental errors produced by machined due to malfunctioning or due to some other parameters. Data reduction is termed as the process of reducing data using some kind of normalisation for easy process of data. Z score is one of processes used for normalisation. Data wrangling is termed as a process where data is transformed or mapped. Data munging, data visualisation and data aggregation comes under this process. Data visualisation is process where statistics are used for producing graphs. Data aggregation is process where data is filtered before fed into model. During data pre-processing all machine learning algorithms are able to learn from categorical data and converted into numbers using a process called one hot encoding. By using this method categorical values are converted into numbers for both in input and output. Categorical values are converted into binary vector using one hot encoding. Categorical values are converted into integers by mapping the values from binary vector. Then each data value can be represented as a integer in a binary vector where 1 is used to represent data value and all others are represented as zeroes. For Example, take a attribute called colour with values red, red, yellow, green Prediction of House Price Using XGBoost Regression Algorithm 2153 where binary vector is represented as [1,0,0], [1,0,0], [0,1,0], [0,0,1] then fed into model which is easier to understand for most machine learning algorithms. XGBoost regression also uses one hot encoding for understanding categorical values.

Now, we categorize the features depending on their datatype (int, float, object) and then calculate the number of them.

```
obj = (dataset.dtypes == 'object') object_cols = list(obj[obj].index)
print("Categorical variables:",len(object_cols))

int_ = (dataset.dtypes == 'int') num_cols = list(int_[int_].index) print("Integer
variables:",len(num_cols))

fl = (dataset.dtypes == 'float') fl_cols = list(fl[fl].index) print("Float variables:",len(fl_cols))
```

Output:

Categorical variables : 4

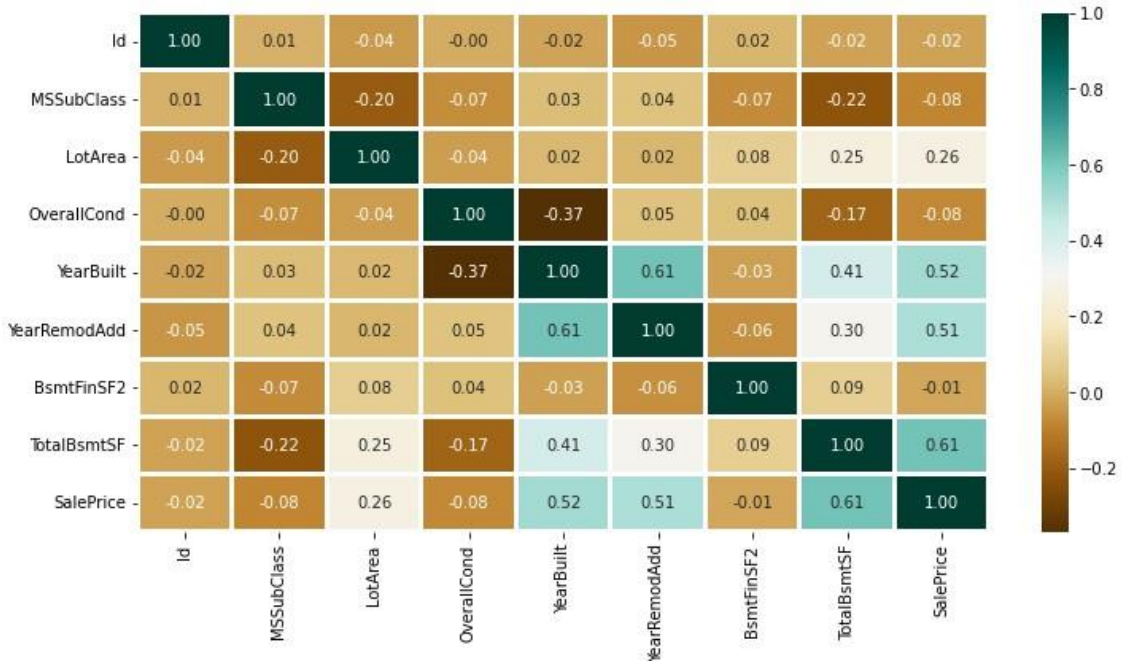
Integer variables : 6 Float variables : 3

Exploratory Data Analysis:

EDA refers to the deep analysis of data so as to discover different patterns and spot anomalies. Before making inferences from data it is essential to examine all your variables.
So here let's make a heatmap using seaborn library.

```
plt.figure(figsize=(12, 6)) sns.heatmap(dataset.corr(),  cmap = 'BrBG',  fmt = '.2f',
linewidths = 2,
      annot = True)
```

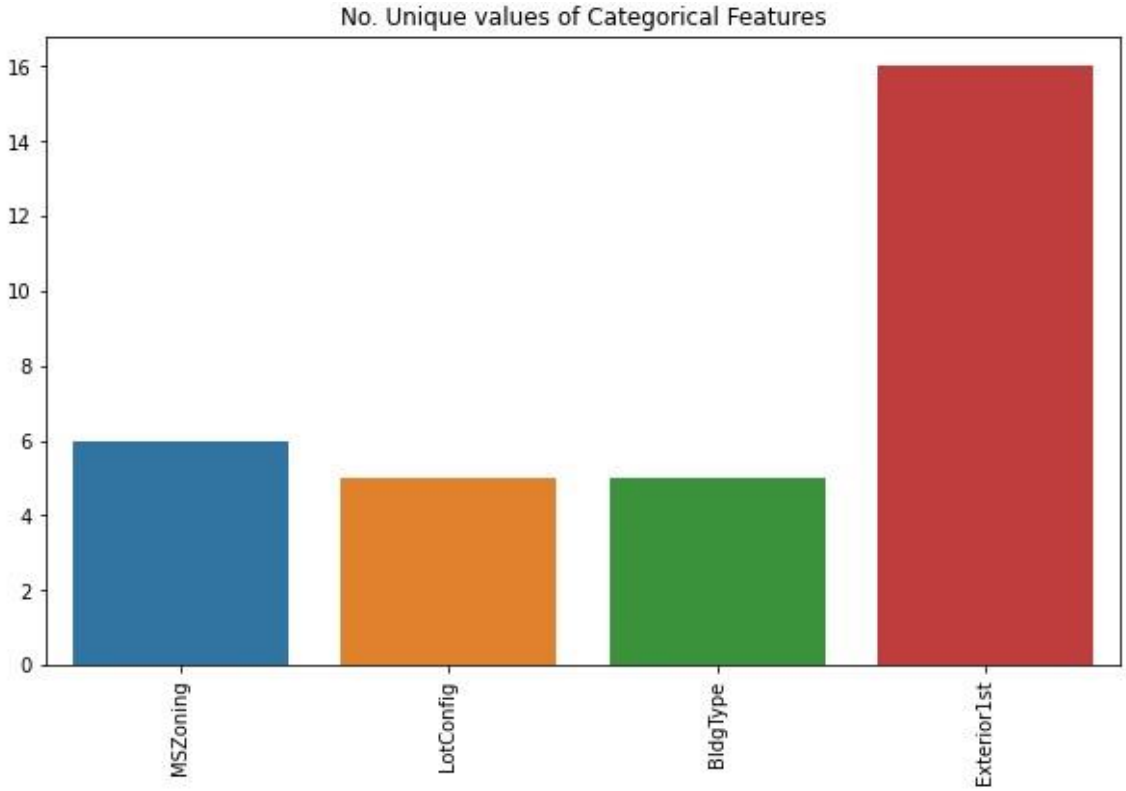
Output:



To analyze the different categorical features. Let's draw the barplot.

```
unique_values = [] for col in object_cols: unique_values.append(dataset[col].unique().size)
plt.figure(figsize=(10,6))
plt.title('No. Unique values of Categorical Features') plt.xticks(rotation=90)
sns.barplot(x=object_cols,y=unique_values)
```

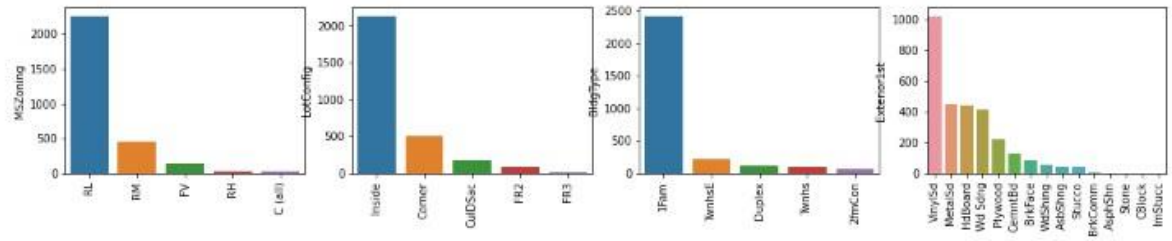
Output:



The plot shows that Exterior1st has around 16 unique categories and other features have around 6 unique categories. To findout the actual count of each category we can plot the bargraph of each four features separately.

```
plt.figure(figsize=(18, 36))
plt.title('Categorical Features: Distribution') plt.xticks(rotation=90) index = 1

for col in object_cols:
        y = dataset[col].value_counts()   plt.subplot(11, 4, index)          plt.xticks(rotation=90)
 sns.barplot(x=list(y.index), y=y)  index += 1
```

Output:



Data Cleaning:

Data cleaning is the way to improvise the data or remove incorrect, corrupted or irrelevant data. As in our dataset, there are some columns that are not important and irrelevant for the model training. So, we can drop that column before training. There are 2 approaches to dealing with empty/null values

We can easily delete the column/row (if the feature or record is not much important).

Filling the empty slots with mean/mode/0/NA/etc. (depending on the dataset requirement).

As Id Column will not be participating in any prediction. So we can Drop it.

```
dataset.drop(['Id'],
                    axis=1,
                    inplace=True)
```

Replacing SalePrice empty values with their mean values to make the data distribution symmetric.

```
dataset['SalePrice'] = dataset['SalePrice'].fillna(
dataset['SalePrice'].mean())
```

Drop records with null values (as the empty records are very less).

```
new_dataset = dataset.dropna()
```

Checking features which have null values in the new dataframe (if there are still any).

```
new_dataset.isnull().sum()
```

Output:

```
MSSubClass      0
MSZoning        0
LotArea         0
LotConfig       0
BldgType        0
OverallCond     0
YearBuilt       0
YearRemodAdd    0
Exterior1st     0
BsmtFinSF2      0
TotalBsmtSF     0
SalePrice       0
dtype: int64
```

Splitting Dataset into Training and Testing:

X and Y splitting (i.e. Y is the SalePrice column and the rest of the other columns are X)

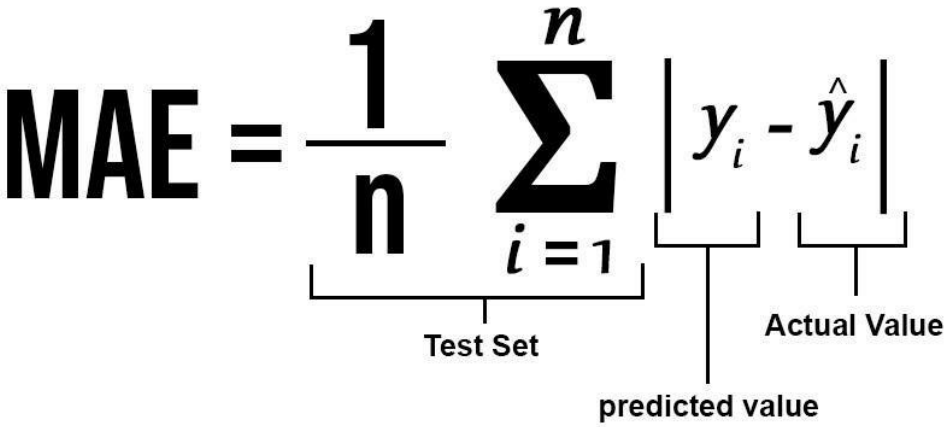
```
from sklearn.metrics import mean_absolute_error from sklearn.model_selection import
train_test_split

= df_final.drop(['SalePrice'], axis=1)
= df_final['SalePrice']

# Split the training set into
# training and validation set
X_train, X_valid, Y_train, Y_valid = train_test_split(
        X, Y, train_size=0.8, test_size=0.2, random_state=0)
```

Model and Accuracy:

As we have to train the model to determine the continuous values, so we will be using these regression models.

- SVM-Support Vector Machine
- Random Forest Regressor
- Linear Regressor

And To calculate loss we will be using the mean_absolute_percentage_error module. It can easily be imported by using sklearn library. The formula for Mean Absolute Error :



SVM – Support vector Machine:

SVM can be used for both regression and classification model. It finds the hyperplane in the n-dimensional plane.
from sklearn import svm from sklearn.svm import SVC from sklearn.metrics import mean_absolute_percentage_error

model_SVR = svm.SVR() model_SVR.fit(X_train,Y_train)
Y_pred = model_SVR.predict(X_valid)

print(mean_absolute_percentage_error(Y_valid, Y_pred))
Output :
0.18705129

REGRESSION:

Random Forest Regression:

Random Forest is an ensemble technique that uses multiple of decision trees and can be used for both regression and classification tasks.
from sklearn.ensemble import RandomForestRegressor

model_RFR = RandomForestRegressor(n_estimators=10)
model_RFR.fit(X_train, Y_train)
Y_pred = model_RFR.predict(X_valid)

mean_absolute_percentage_error(Y_valid, Y_pred)
Output :
0.1929469

Linear Regression:

Linear Regression predicts the final output-dependent value based on the given independent features. Like, here we have to predict SalePrice depending on features like MSSubClass, YearBuilt, BldgType, Exterior1st etc.

```
from sklearn.linear_model import LinearRegression

model_LR = LinearRegression()
model_LR.fit(X_train, Y_train)
Y_pred = model_LR.predict(X_valid)

print(mean_absolute_percentage_error(Y_valid, Y_pred))
```

Output :

0.187416838

CatBoost Classifier

CatBoost is a machine learning algorithm implemented by Yandex and is opensource. It is simple to interface with deep learning frameworks such as Apple's Core ML and Google's TensorFlow. Performance, ease-of-use, and robustness are the main advantages of the CatBoost library. from catboost import CatBoostRegressor cb_model = CatBoostRegressor() cb_model.fit(X_train, y_train) preds = cb_model.predict(X_valid)

```
cb_r2_score=r2_score(Y_valid, preds) cb_r2_score
```

output

0.893643437976127

XGBOOST REGRESSION:-

XGBoost regression is short form for extreme gradient boost regression. It works well compared to others machine learning algorithms.XGBoost is one of the best supervised learning algorithms which can inferred by the way it flows, it consists of objective function and base learners.Loss function is present in objective function which shows the difference between actual values and predicted values whereas regularisation term is used for showing how far is actual value away from predicted value. Ensemble learning used in XGBoost considers many models which are known as base learners for predicting a single value. Not all base learners are expected to have bad prediction so that after summing up all of them bad prediction cancelled out by good prediction. A regressor is the one that fits a model using given features and predicts the unknown output value. Dataset for processing is taken from a Kaggle competition and fed into the model after pre-processing as specified.The algorithm for prediction of price of house using XG Boost Regression is specified below.

3.1 XG Boost Regression Algorithm for House Price Prediction Input: House attributes dataset. Output: Price of house.

1. Check input dataset for missing values and calculate d mean is replaced in place of missing value.
2. Divide attributes based on values in data fields as categorical and non-categorical rows.
3. Check Non categorical rows for outliers using outlier detection techniques and remove all outliers.
4. Convert categorical rows into binary vectors using one hot encoding.
5. Divide dataset for cross validation using train test split.

6. Apply Ensemble learning through training and combining individual models termed as base learners in order to derive a single prediction.

a) Calculate Mean Squared Error (MSE) with true values to predicted values.

b) Classify independent models as weak-learners and strong-learners using error detection.

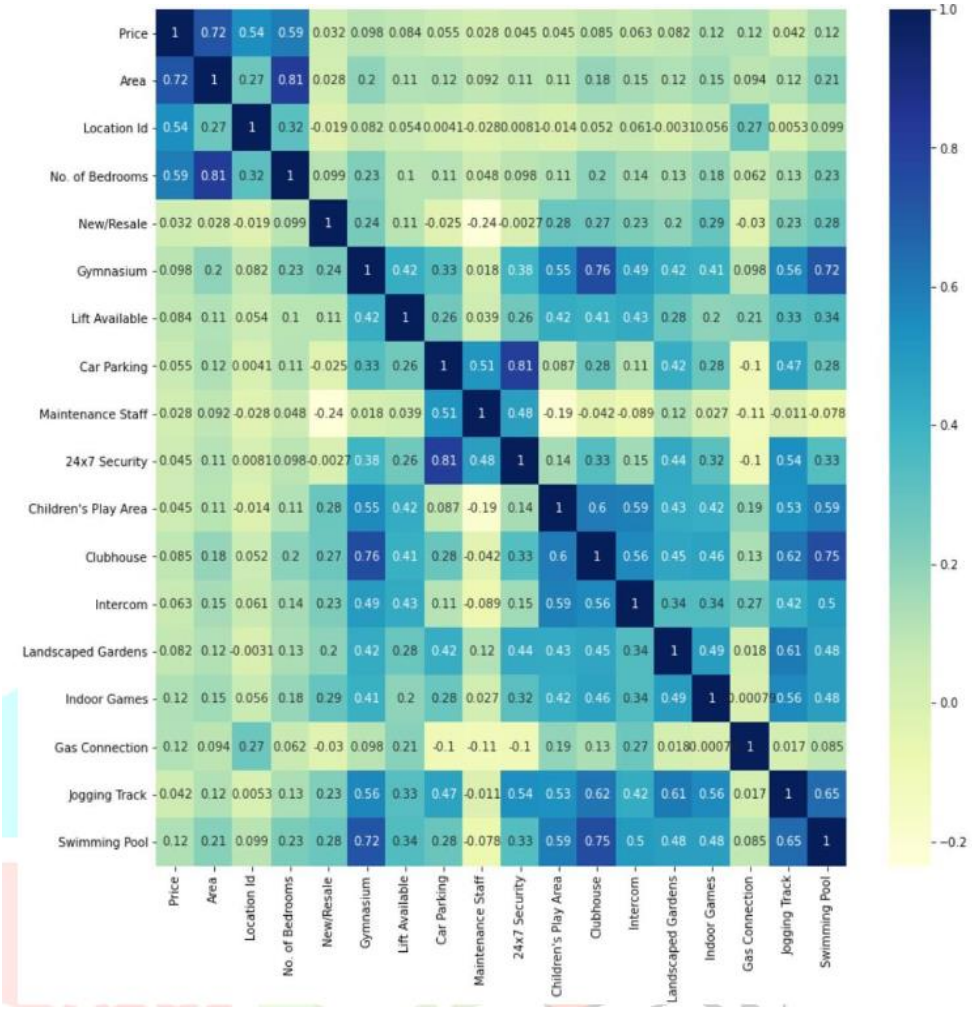
c) Total mean cancels bad prediction with good prediction.

7. Objective function contains the loss function and regularisation term to calculate difference between actual value and predicted value.

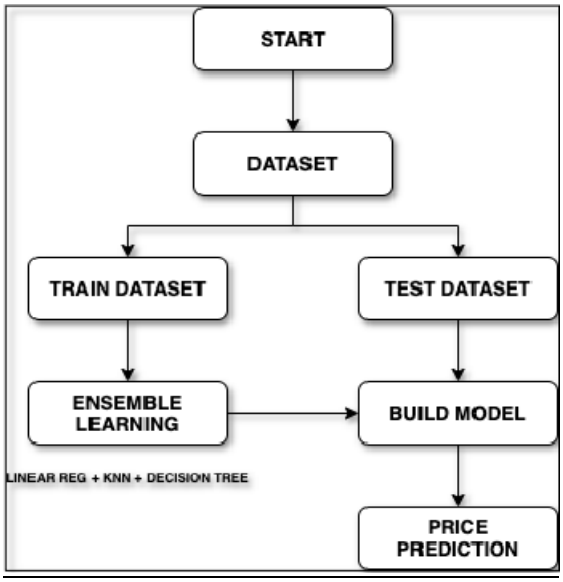
ENSEMBLE LEARNING:-

The ensemble learning model is a combination of algorithms or models which help to improve predictions
depending on the features of the dataset. We could conclude from our experimental results that a weighted average of predictions
from Linear Regression, KNN, and Decision Tree models provided the lowest error values compared to predictions from individual
algorithms from Fig.6 and Table.4 below. The weights assigned to the predictions of models are based on its performance and
features of our dataset exclusively.

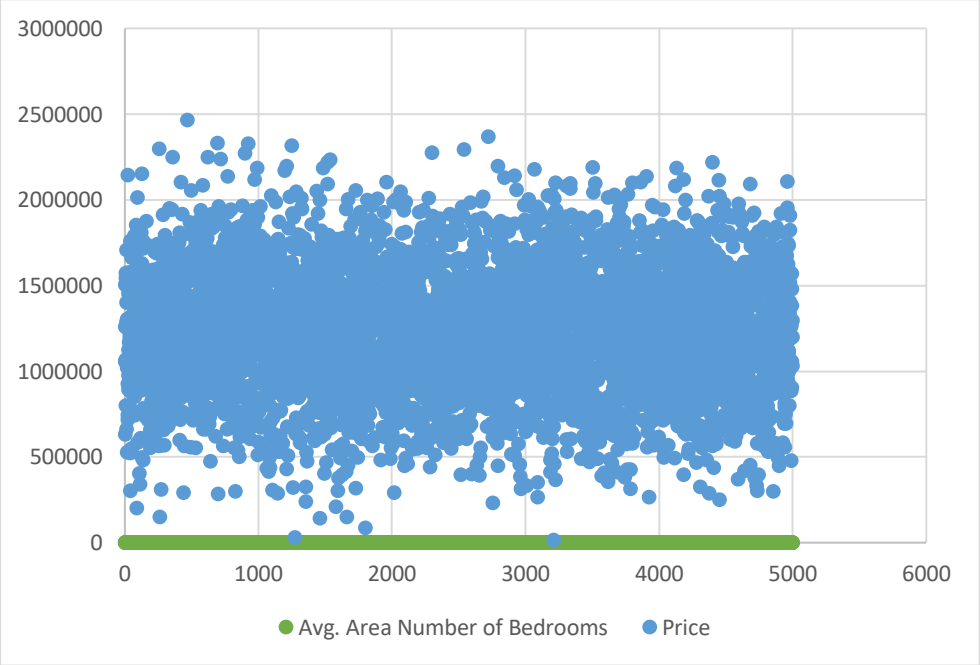
Below given Table.4 mentions the error metrics of the model: Mean Absolute Error (MAE), Mean Squared Error (MSE), Root
Mean Squared Error (RMSE), R2 Score, and Weight Assigned (W).



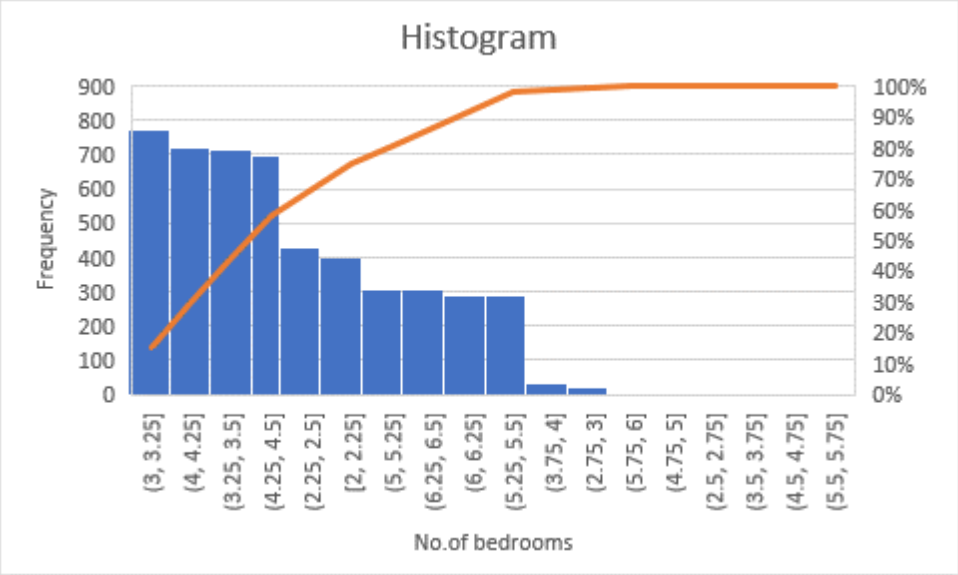
METHODOLOGY FLOW CHART:-



VISUALISATION(SCATTER PLOT):-



DATA VISUALISATION(HISTOGRAM):-

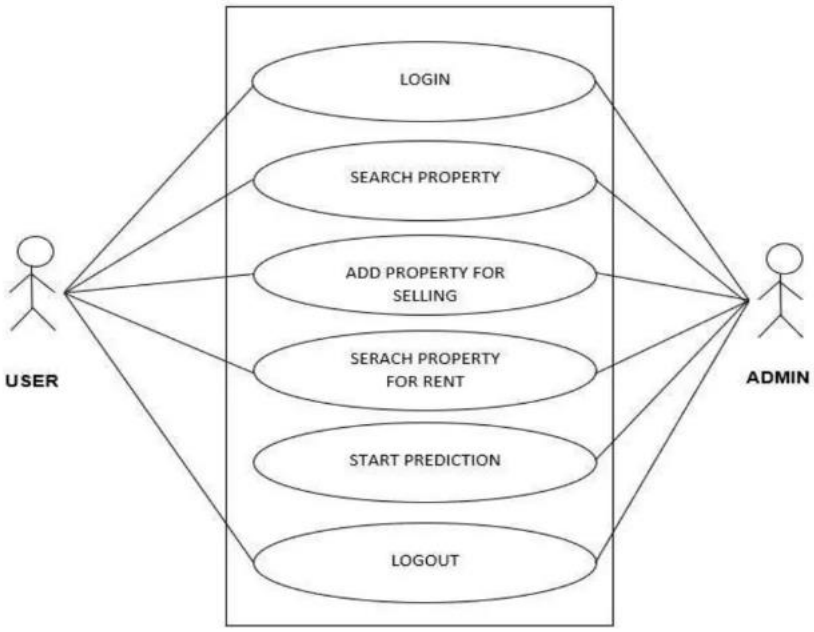


APPLYING MACHINE LEARNING ALGORITHM:-

We set independent and target variables as x and y respectively.
Split the dataset into training and testing in 70:30 ratio.
Fitting the train set to multiple linear regression and getting the score of training model.
Fitting the train set to decision tree and getting the score of the model.
Fitting the train set to random forest and getting the score of the model.
Calculate the model score to understand how our model performed along with the explained variance score.

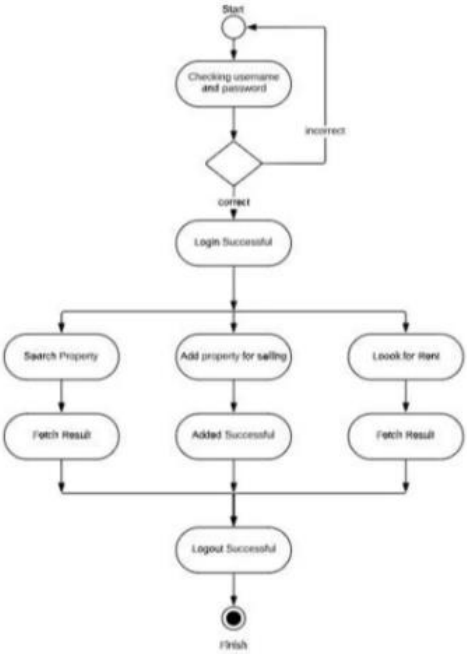
UML DIAGRAMS:-

Use case diagrams are used to gather the requirements of a system including internal and external influences.

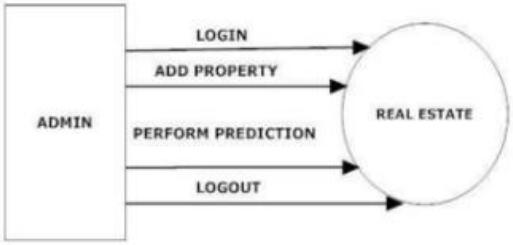


USE CASE DIAGRAM

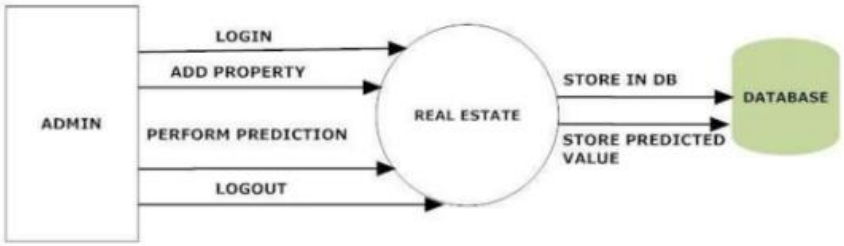
Activity diagram is basically a flowchart to represent the flow from one activity to another activity



A data-flow diagram(DFD) is a way of representing a flow of a data of a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself.

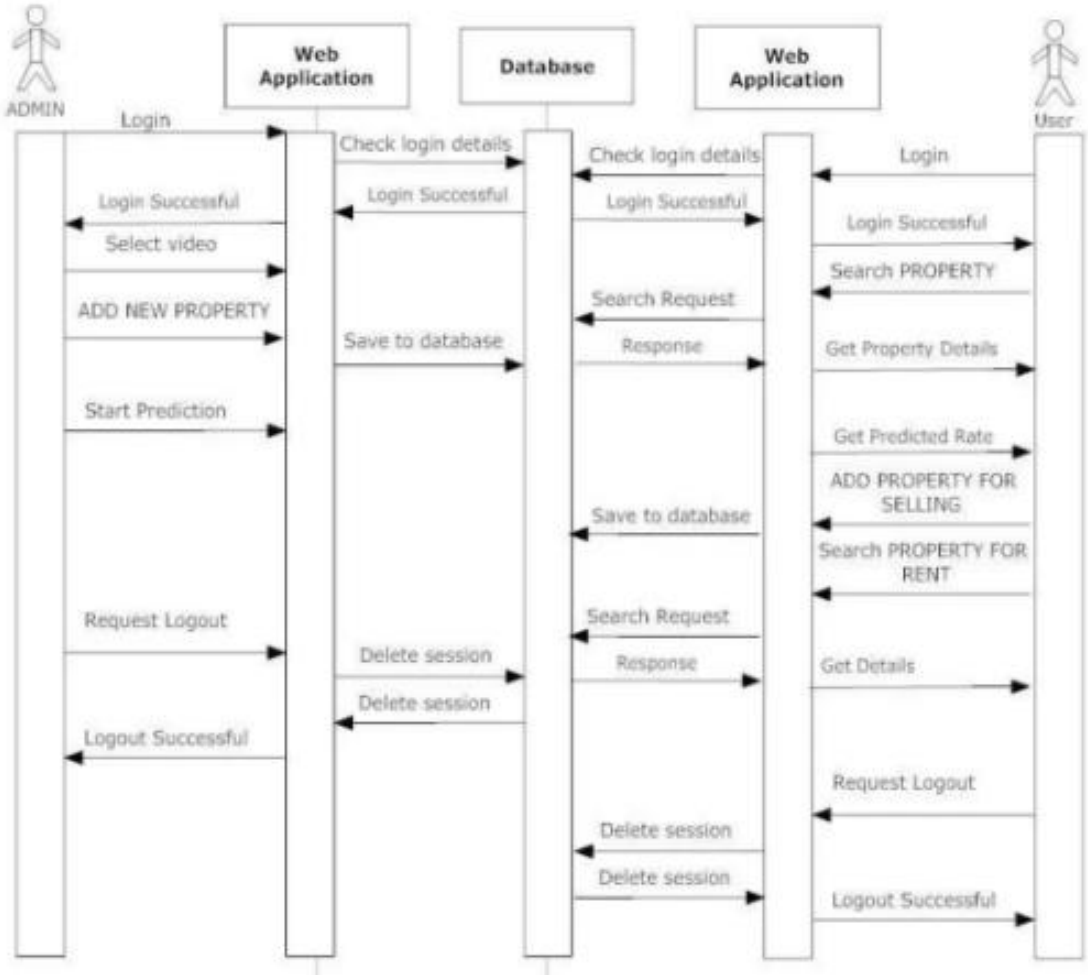


DFD level 0(admin)



DFD level 1(admin)

Sequence diagram emphasizes on sequence of messages.



METHOD OF ANALYSIS-RANDOM FOREST

Variables
Using the same variable used in the linear regression.
Trees and Nodes
Checking various combinations of number of trees and maximum number of nodes to get the best result.
Using number of trees=100 and maximum nodes=10 for best fitted model.
Model Accuracy
Checking model accuracy using error rate and MAPE.
Decision
Drop the variable if the model accuracy falls or remains same.

FEASIBILITY STUDY:-

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. The feasibility study of the proposed system is carried out. It is carried out to ensure that the proposed system is not a burden to the company.
Economic feasibility
Technical feasibility
Social feasibility

Hardware and Software Requirements

Hardware specification

Processor: i3

RAM: 4 GB or more

Hard disk: 16 GB or more

GPU: 2 GB

Software Specification

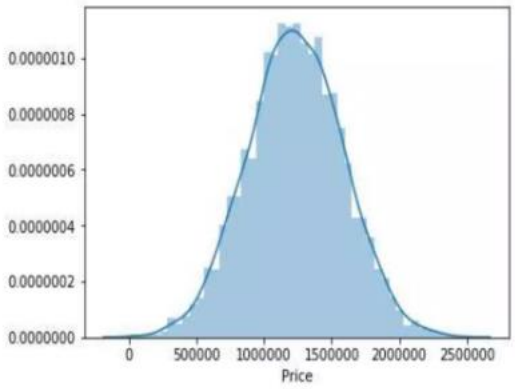
Platform: Windows operating system

JupyterLab

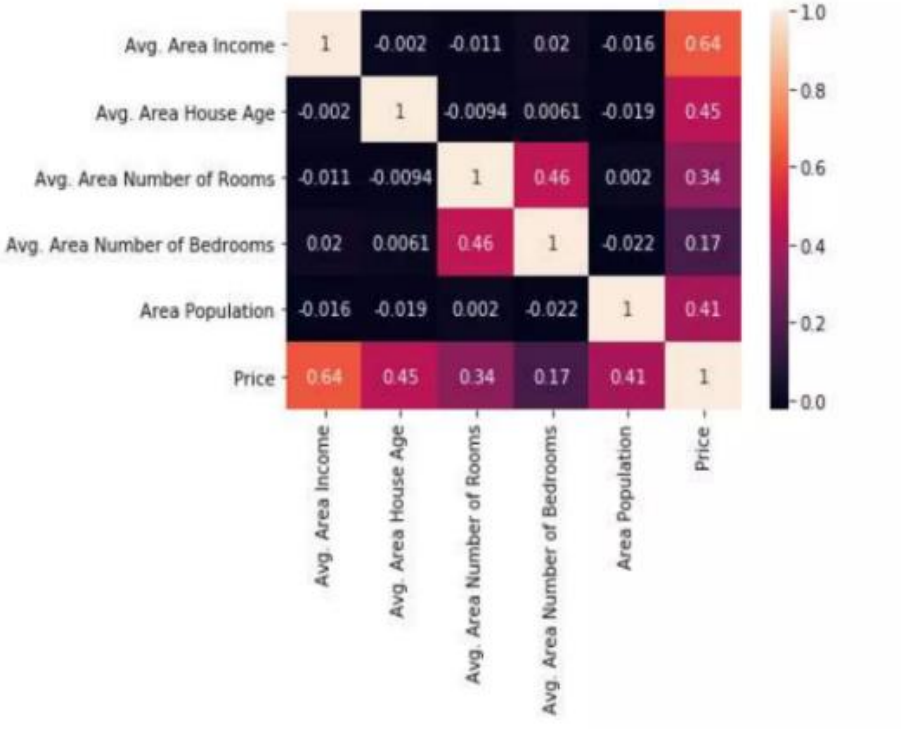
Python 3

OUTPUT

- Distribution plot: (Data)



- Heat map plot:



Feature Selection:

Feature selection is a crucial step in building a predictive model. It involves choosing the most relevant features (attributes) that are likely to influence the house prices. Common techniques for feature selection include:
Correlation analysis to identify features strongly correlated with the target variable (e.g., price).
Recursive Feature Elimination (RFE) using techniques like linear regression or tree-based models to rank and select features.
Domain knowledge and intuition to include or exclude specific features based on their relevance.
Model Selection and Training:

Choose an appropriate machine learning algorithm or model for the task. Common models for predicting house prices include:
Linear Regression
Decision Trees
Random Forest
Gradient Boosting (e.g., XGBoost or LightGBM)
Neural Networks
Split your dataset into training and testing sets to train and evaluate your model. A common split is 80% for training and 20% for testing.
Model Training:

Train the selected model on the training data. This involves fitting the model to learn the relationships between the features and the target variable (house prices).
Hyperparameter tuning can be performed to optimize the model's performance.
Evaluation:

After training the model, it's essential to evaluate its performance to assess its predictive accuracy. Common evaluation metrics for regression problems (like predicting house prices) include:
Mean Absolute Error (MAE)
Mean Squared Error (MSE)
Root Mean Squared Error (RMSE)
R-squared (R^2) or coefficient of determination
Additionally, you can use visualization techniques like scatter plots to visually assess the model's predictions.

Cross-Validation:

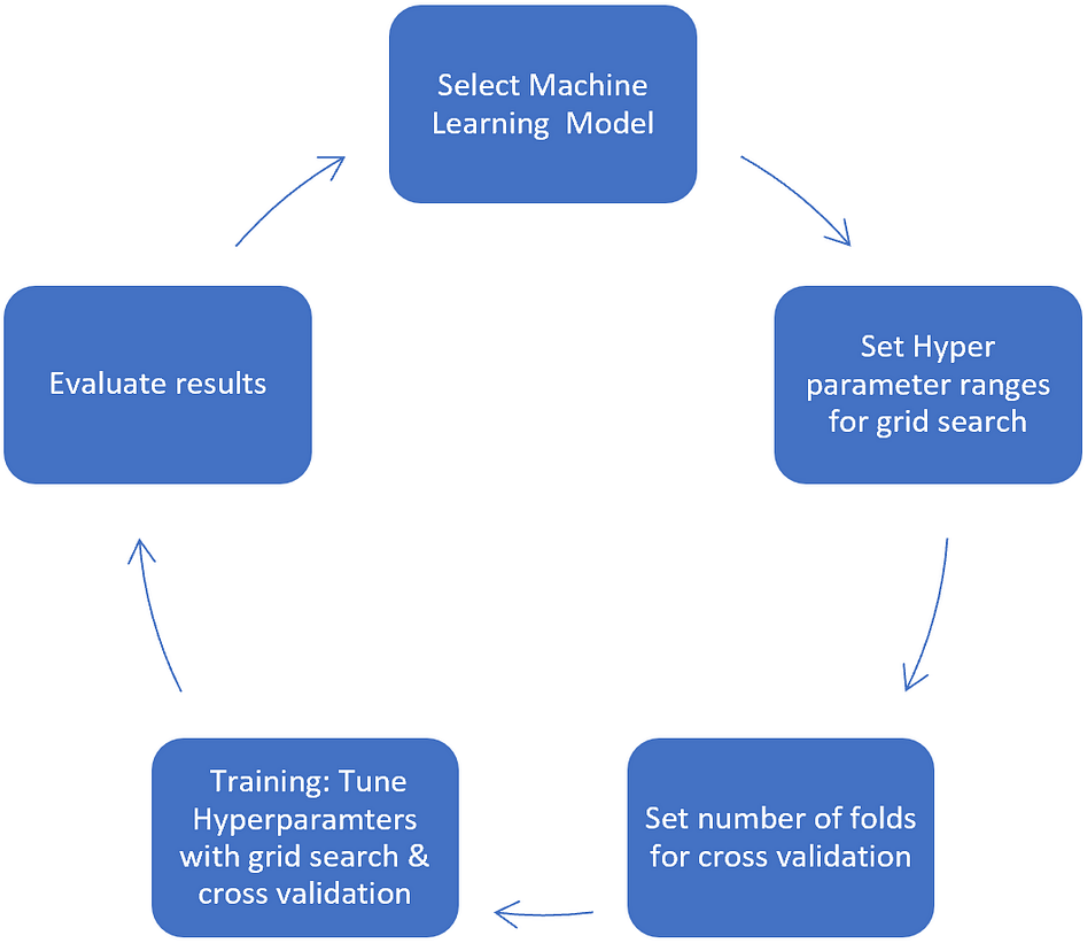
To ensure that your model's performance is robust and not overfitting to the training data, consider using cross-validation techniques like k-fold cross-validation. This involves splitting the data into multiple folds, training and testing the model on different subsets, and calculating average evaluation metrics.

Fine-Tuning:

If the model's performance is not satisfactory, you can iterate by adjusting hyperparameters, trying different algorithms, or collecting more data if possible.
Deployment:
Once you have a well-performing model, you can deploy it for making predictions on new data, such as estimating house prices for real estate listings.

Machine learning

I follow a standard development cycle for machine learning. As a beginner or even a pro, you'll likely have to go through many iterations of the cycle before you are able to get your models working to a high standard.



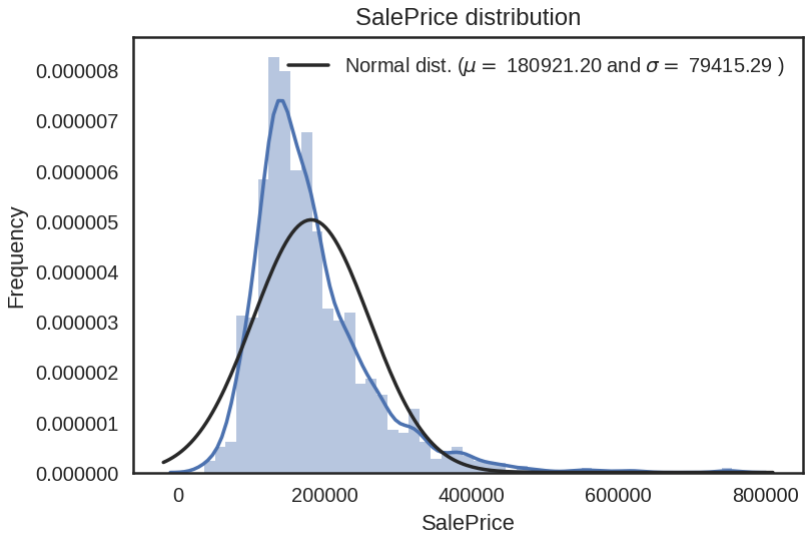
APPLYING MACHINE LEARNING ALGORITHM:-

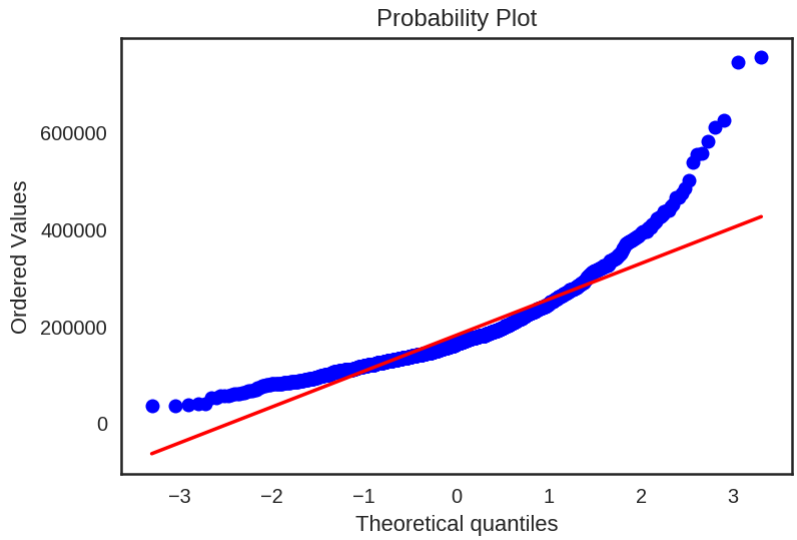
We set independent and target variables as x and y respectively.
Split the dataset into training and testing in 70:30 ratio.
Fitting the train set to multiple linear regression and getting the score of training model.
Fitting the train set to decision tree and getting the score of the model.
Fitting the train set to random forest and getting the score of the model.
Calculate the model score to understand how our model performed along with the explained variance score.

Analyzing the Test Variable (Sale Price)

Let's check out the most interesting feature in this study: Sale Price. Important Note: This data is from Ames, Iowa. The location is extremely correlated with Sale Price. (I had to take a double-take at a point, since I consider myself a house-browsing enthusiast)
In [6]:
Getting Description

```
train['SalePrice'].describe()
```
Out[6]:
```
count     1460.000000
mean    180921.195890
std      79442.502883
min      34900.000000
25%     129975.000000
50%     163000.000000
75%     214000.000000
max     755000.000000
Name: SalePrice, dtype: float64
```
With an average house price of $180921, it seems like I should relocated to Iowa!
In [7]:
```
# Plot Histogram
sns.distplot(train['SalePrice'] , fit=norm);

# Get the fitted parameters used by the function
(mu, sigma) = norm.fit(train['SalePrice'])
print( '\n mu = {:.2f} and sigma = {:.2f}\n'.format(mu, sigma))
plt.legend(['Normal dist. ($\mu=$ {:.2f} and $\sigma=$ {:.2f} )'.format(mu, sigma)],
           loc='best')
plt.ylabel('Frequency')
plt.title('SalePrice distribution')

fig = plt.figure()
res = stats.probplot(train['SalePrice'], plot=plt)
plt.show()

print("Skewness: %f" % train['SalePrice'].skew())
print("Kurtosis: %f" % train['SalePrice'].kurt())
```
```
 mu = 180921.20 and sigma = 79415.29
```





Skewness: 1.882876

Kurtosis: 6.536282

Looks like a normal distribution? Not quite! Looking at the kurtosis score, we can see that there is a very nice peak. However, looking at the skewness score, we can see that the sale prices deviate from the normal distribution. Going to have to fix this later! We want our data to be as "normal" as possible.

FUTURE WORK:

Our model had a low rmse score, but there is still room for improvement. In a real world scenario, we can use such a model to predict house prices. This model should check for new data, once In a month, and incorporate them to expand the dataset and produce better results.

We can try out other dimensionality reduction techniques like Univariate feature selection and Recursive feature elimination in the initial stage.

We can try out other advanced regression techniques, like random forest and Bayesian ridge algorithm, for prediction. Since the data is highly correlated, we should also try Elastic Net regression technique.

PROPOSED SYSTEM: -

System includes set of codes that processes on the available dataset to effectively predicts the value of outcome depending upon user input using the concept of Linear regression.

Efficient and proper use of the system can eradicate the cases where the customers get cheated by the real estate agents in terms of house prices.

Proper usage of model is beneficial to both the customer as well as agents guiding customers.

ADVANTAGES: -

- Good interpretability
- Its very simple to understand
- Space complexity is very low it just needs to save the weights at the end of training. It's a high latency algorithm.
-

Understanding the Client and their Problem:-

A benefit to this study is that we can have two clients at the same time! (Think of being a divorce lawyer for both interested parties) However, in this case, we can have both clients with no conflict of interest!

Client Housebuyer: This client wants to find their next dream home with a reasonable price tag. They have their locations of interest ready. Now, they want to know if the house price matches the house value. With this study, they can understand which features (ex. Number of bathrooms, location, etc.) influence the final price of the house. If all matches, they can ensure that they are getting a fair price.

Client Houseseller: Think of the average house-flipper. This client wants to take advantage of the features that influence a house price the most. They typically want to buy a house at a low price and invest on the features that will give the highest return. For example, buying a house at a good location but small square footage. The client will invest on making rooms at a small cost to get a large return.

Loading Data and Packages

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
import warnings
import xgboost as xgb
import lightgbm as lgb
from scipy.stats import skew
from scipy import stats
from scipy.stats.stats import pearsonr
from scipy.stats import norm
from collections import Counter
from sklearn.linear_model import LinearRegression,LassoCV, Ridge, LassoLarsCV,ElasticNetCV
from sklearn.model_selection import GridSearchCV, cross_val_score, learning_curve
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor, ExtraTreesRegressor,
GradientBoostingRegressor
from sklearn.preprocessing import StandardScaler, Normalizer, RobustScaler
warnings.filterwarnings('ignore')
sns.set(style='white', context='notebook', palette='deep')
%config InlineBackend.figure_format = 'retina' #set 'png' here when working on notebook
%matplotlib inline
```

In [2]:

```
# Load train and Test set
train = pd.read_csv("../input/train.csv")
test = pd.read_csv("../input/test.csv")
```

In [3]:

```
# Check the numbers of samples and features
print("The train data size before dropping Id feature is : {} ".format(train.shape))
print("The test data size before dropping Id feature is : {} ".format(test.shape))

# Save the 'Id' column
train_ID = train['Id']
test_ID = test['Id']
```

```
# Now drop the 'Id' column since it's unnecessary for the prediction process.
train.drop("Id", axis = 1, inplace = True)
test.drop("Id", axis = 1, inplace = True)

# Check data size after dropping the 'Id' variable
print("\nThe train data size after dropping Id feature is : {} ".format(train.shape))
print("The test data size after dropping Id feature is : {} ".format(test.shape))
```

```
The train data size before dropping Id feature is : (1460, 81)
The test data size before dropping Id feature is : (1459, 80)

The train data size after dropping Id feature is : (1460, 80)
The test data size after dropping Id feature is : (1459, 79)
```

So the training set has 1460 rows and 81 features. The test set has 1459 rows and 80 features.

Objective & Data

The competition goal is to predict sale prices for homes in USA. You're given a training and testing data set in csv format as well as a data dictionary.
Dataset:
file:///C:\Users\Thahira\Desktop\MyProject\USA_housing.csv

Testing: The test data set consists of 1,459 examples with the same number of features as the training data. Our test data set excludes the sale price because this is what we are trying to predict. Once our models have been built we will run the best one the test data and submit it to the Kaggle leaderboard.

Task: Machine learning tasks are usually split into three categories; supervised, unsupervised and reinforcement. For this competition, our task is supervised learning.
Supervised learning uses examples and labels to find patterns in data.
It's easy to recognise the type of machine learning task in front of you from the data you have and your objective. We've been given housing data consisting of features and labels, and we're tasked with predicting the labels for houses outside of our training data.

Tools

I used Python and Jupyter notebooks for the competition. Jupyter notebooks are popular among data scientist because they are easy to follow and show your working steps.
Libraries: These are frameworks in python to handle commonly required tasks. I Implore any budding data scientists to familiarise themselves with these libraries:
Pandas — For handling structured data
Scikit Learn — For machine learning
NumPy — For linear algebra and mathematics
Seaborn — For data visualization

Project Pipeline

Generally speaking, machine learning projects follow the same process. Data ingestion, data cleaning, exploratory data analysis, feature engineering and finally machine learning.

The pipeline is not linear and you might find you have to jump back and forth between different stages. It's important I mention this because tutorials often make you believe the process is much cleaner than in reality. So please keep this in mind, your first machine learning project might be a mess.

SUBMISSION:-

Predicting house prices is a common machine learning task. Here's a Python code example using the popular scikit-learn library to predict house prices, including data preprocessing, model training, and evaluation steps. In this example, we'll use a simple linear regression model, but you can explore more complex models like Random Forest, Gradient Boosting, or neural networks for better performance.

```
# Import necessary libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler

# Load your dataset
data = pd.read_csv(USA_housing.csv)

# Data preprocessing
# Select features (X) and target variable (y)
X = data.drop('Price', axis=1)
y = data['Price']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the feature data (optional but can improve model performance)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Model training
model = LinearRegression()
model.fit(X_train, y_train)

# Model evaluation
y_pred = model.predict(X_test)

# Calculate evaluation metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
print(f"Mean Squared Error: {mse}")
print(f"R-squared (R2) Score: {r2}")

# Now you can use the trained model to make predictions on new data
# For example, to predict the price of a new house:
new_house_features = np.array([...])  # Replace ... with feature values of the new house
new_house_features = scaler.transform(new_house_features.reshape(1, -1))
predicted_price = model.predict(new_house_features)
print(f"Predicted Price: {predicted_price[0]}")
```

COMPILE THE CODE:

```
# Import necessary libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler

# Load your dataset
data = pd.read_csv(USA_housing.csv)
# Data preprocessing
# Select features (X) and target variable (y)
X = data.drop('Price', axis=1)
y = data['Price']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the feature data (optional but can improve model performance)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Model training
model = LinearRegression()
model.fit(X_train, y_train)

# Model evaluation
y_pred = model.predict(X_test)

# Calculate evaluation metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
```

```
print(f"R-squared (R2) Score: {r2}")

# Now you can use the trained model to make predictions on new data
# For example, to predict the price of a new house:
new_house_features = np.array([Avg. Area Income, House Age,Number of Rooms,Avg. Area Number of Bedrooms,Area Population,PriceAddress,24*7 security
])
new_house_features = scaler.transform(new_house_features.reshape(1, -1))
predicted_price = model.predict(new_house_features)
print(f"Predicted Price: {predicted_price[0]}")
```

CONCLUSION:

Thus, we studied and applied the concept of Linear Regression in real time implementation so as to ease the life of human.
Determining the price of property without complete knowledge about the surrounding is quite riskier for both customer and the seller.
In order to overcome this problem we have tried to develop application which determines the price of the property based on various parameters of the surrounding.

- Data provide us with the complete data about the surrounding in the form of dataset.
- Dataset helps to get the insight of the surrounding and machine learning model helps to predict the price of the property based on the training provided by the dataset.
- We successfully implemented linear regression model to predict the price of the houses.

A model for house price prediction that assists both buyer and seller had been proposed. The proposed house prediction system helps seller to sell house at best price and it also helps buyer to buy house at best price. Price of said land costs, material varies from place to place. Prediction of price of house is difficult as it varies from place to place as all attributes doesn’t have same proportion in all places. Deep learning algorithms may enhance the prediction of price of house and decrease test error rate percentage. The XGBoost regression algorithm helps to satisfy the needs of customers by increasing accuracy of the choice of estates and decreasing the risk for Prediction of House Price Using XGBoost Regression Algorithm 2155 customers to invest in real estate. The system can be made widely acceptable by including more number of features. Future scope is to create estate database including more cities in order to help customers explore more number of estates and get accurate decision.