PHASE 1

Title: **Predicting House Prices using Machine Learning**

Introduction:

House Price prediction are very stressful work as we have to consider different things while buying a house like the structure and the rooms. 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. Which influence the house price. But by using the Machine learning we can easily find the house which is to be prefect for us and helps to predict the price accurately

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

Problem Definition: The problem is to predict house prices using machine learning techniques. The objective is to develop a model that Accurately predicts the prices of houses based on a set of features such as location, square footage number of bedrooms and Bathrooms, and other relevant factors. This project involves data pre-processing, feature engineering, model selection, training, and Evaluation

Design Thinking:

Data source: 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 House buyer: 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 House seller: 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.

Dataset Link: https://www.kaggle.com/datasets/vedavyasv/usa.housing

Importing required libraries.

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import r2_score

%matplotlibinline
```

Data processing: typically involves several steps of data processing. Here's an overview of the process:

1.Data collection: Gather a dataset that includes information about houses, such as square footage, number of bedrooms, location, and sale prices. You can obtain data from sources like real estate websites, government databases, or through web scraping.

2.Data Cleaning:

- Handle missing values: Determine how to handle missing data, either by imputing values or removing rows with missing values.

- Outlier detection: Identify and deal with outliers that could skew the model's predictions.

3. Feature Selection :

- Select relevant features: Choose which features (attributes) are most important for predicting house prices. Common features include square footage, number of bedrooms and bathrooms, location, and more.

- Engineer new features: Create new features that might improve prediction accuracy, such as the price per square foot or a distance metric to important amenities.

4.Data Transformation:

- Encode categorical variables: Convert categorical variables (like location or type of house) into numerical representations, such as one-hot encoding.

- Scale numerical features: Normalize or standardize numerical features to bring them to a similar scale.

5.Split Data:

- Divide the dataset into a training set and a testing/validation set. This allows you to train the model on one portion of the data and evaluate its performance on unseen data.

6. Model Selection:

- Choose a machine learning algorithm suitable for regression tasks, such as linear regression, decision trees, random forests, or neural networks.

- Train multiple models and compare their performance to select the best one.

7. Model Training:

- Train the selected model on the training data using appropriate hyperparameters.

- Regularize the model to prevent overfitting if needed.

8. Model Evaluation:

- Evaluate the model's performance on the testing/validation dataset using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE).

9. Hyperparameter Tuning:

- Fine-tune the model's Hyperparameter to optimize its performance.

10. Deployment:

- Once satisfied with the model's performance, deploy it for making predictions on new, unseen data.

11. Monitoring and Maintenance:

- Continuously monitor the model's performance in a production environment and retrain or update it as needed to account for changing real estate market trends.

Remember that predicting house prices is a complex task, and the success of your model depends on the quality of your data, feature engineering, and the choice of the appropriate machine learning algorithm. Additionally, ethical considerations, such as bias in data and model, should be taken into account when using such models for real-world applications.

Feature Selection

-This notebook tries various feature selection techniques.

- Based in these techniques, select features that contribute most to the output variable.

Techniques used are

-Correlation and Mutual Information for Numerical Features

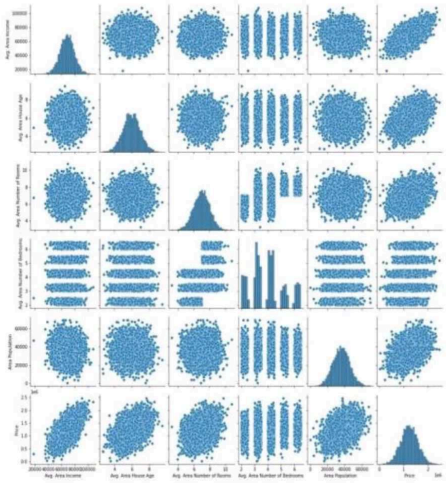
-SelectFromModel

-SelectKBest

Numerical Features

There are two methods by which one can choose numerical features. One is correlation and other is mutual information.

Correlation:



All numeric columns plots

- It is a way to understand the relationship between multiple variables and attributes in your dataset.

- It helps in predicting one attribute from another Can indicate presence of causal relationship

- Can help us identify redundant information/features Positive correlation means if A increases, B also increases. Usually indicated by positive value

- Negative correlation means if A increases, B decreases

- Multicolinearity happens when one predictor variable in a multiple regression model can linearly

- predict others with a high degree of accuracy. Decision trees and boosted tree algorithms are immune to this

- To deal with it use PCA

- Note that some models like Linear Regression must have correlation between dependent and

independent features.

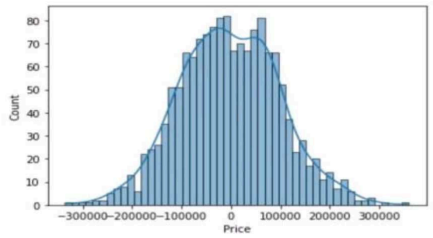
Mutual Information

-There is no redundancy when two variables are independent. As the variables get more and more dependent, redundancy increases and to quantity this, we use mutual information

-Mutual information is the amount of extra information needed if we are representing the tru joint distribution with the independent factorization

Model selection:

-If the mutual infcinitiation is high, the feature is a strong indicator of the class Measure of mutual



dependence between two variables.

Linear regression

Linear regression models assume that the relationship between a dependent continuous variable Y and one or more explanatory (independent) variables X is linear (that is, a straight line). It's used to predict values within a continuous range (e.g. sales, price)

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

Model training:

Now that we've pre-processed and explored our data, we have a much better understanding of the type of data that we're dealing with. Now, we can began to build and test different models for regression to predict the Sale Price of each house. We will import these models, train them, and evaluate them. In classification, we used accuracy as a evaluation metric; in regression, we will use the R^2 score as well as the RMSE to evaluate our model performance. We will also use cross validation to optimize our model hyperparameters.

Defining Training/Test Sets

We drop the Id and SalePrice columns for the training set since those are not involved in predicting the Sale Price of a house. The SalePrice column will become our training target. Remember how we transformed SalePrice to make the distribution more normal? Well we can apply that here and make TransformedPrice the target instead of SalePrice. This will improve model performance and yield a much smaller RMSE because of the scale.

Splitting into Validation

It is always good to split our training data again into validation sets. This will help us evaluate our model performance as well as avoid overfitting our model.

Regression Evaluation Metrics

All of these are **loss functions**, because we want to minimize them.

Mean Absolute Error (MAE)

is the mean of the absolute value of the errors:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} | y_i - \hat{y}_i |$$

The Mean absolute error represents the average of the absolute difference between the actual and predicted values in the dataset. It measures the average of the residuals in the dataset

Mean Squared Error (MSE)

$$\mathbf{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

is the mean of the squared errors:

Mean Squared Error represents the average of the squared difference between the original and predicted values in the data set. It measures the variance of the residual

Root Mean Squared Error (RMSE)

is the square root of the mean of the squared errors:

$$RMSE = \sqrt{\sum_{i=1}^{n}\frac{(\hat{y}_i - y_i)^2}{n}}$$

RMSE measures the standard deviation of residuals

Innovation:

a. Easy to use

b. open source

c. Best accuracy

d. GUI Based Application

XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Kubernetes, Hadoop, SGE, disk, Spark, PySpark) and can solve problems beyond billions of examples.

Conclusion:

The research presented in this paper demonstrates the potential of machine learning algorithms for accurately predicting house prices. With the proper data and features, a well-trained model based on Linear Regression can be used to accurately predict the price of a house. However the accuracy levels can vary based on the datasets used. While the results of this study are promising, there are many opportunities for future research. For instance, exploring different model architectures, such as deep learning and transfer learning, can improve model performance. Additionally, further research could be done to identify the most important features for house price prediction, as well as to explore the impact of different types of data, such as location and neighbourhood characteristics, on model performance. Finally, developing more efficient methods for training and deploying models could enable the use of machine learning algorithms in a wide range of applications.