PREDICTING HOUSE PRICES USING MACHINE LEARNING

**PHASE 2 SUBMISSION**

**INTRODUCTION:**

The prediction of house prices using machine learning has become a compelling and transformative application in the field of real estate. This project involves leveraging advanced algorithms and data analysis techniques to create a model capable of estimating the market value of houses based on various features. By harnessing the power of machine learning, this project aims to provide accurate and data-driven predictions, offering valuable insights to both buyers and sellers in the real estate market.

In this project, I will use Linear Regression model to predict housing prices for regions in the USA, compare the accuracies of the models to find out the best suitable model.

**GRADIENT BOOSTING ALGORITHMS:**

Gradient boosting algorithms are a class of machine learning techniques used for both regression and classification tasks. They work by combining multiple weak learners, typically decision trees, into a strong predictive model. Some popular gradient boosting algorithms are included. Using the Dataset we can predict the house price using machine learning.

**Dataset Link:**[**https://www.kaggle.com/datasets/vedavyasv/usa-housing**](https://www.kaggle.com/datasets/vedavyasv/usa-housing)

**XGBoost:**

XGBoost (Extreme Gradient Boosting) is an optimized and efficient implementation of gradient boosting.

**1. Installation:**

|  |
| --- |
| ***bash*** |
| pip install xgboost |

**2. Import Libraries:**

|  |
| --- |
| ***python*** |
| import xgboost as xgb  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import mean\_absolute\_error |

**3. Data Preparation:**

- Load and preprocess your dataset, including handling missing values, encoding categorical variables, and splitting into training and testing sets.

**4. XGBoost Model Training:**

|  |
| --- |
| ***python*** |
| # Assuming X\_train, X\_test, y\_train, y\_test are your training and testing data  model = xgb.XGBRegressor()  model.fit(X\_train, y\_train) |

**5. Prediction:**

|  |
| --- |
| ***python*** |
| y\_pred = model.predict(X\_test) |

**6. Evaluation:**

|  |
| --- |
| python |
| mae = mean\_absolute\_error(y\_test, y\_pred)  print(f'Mean Absolute Error: {mae}') |

**7. Hyperparameter Tuning:**

- XGBoost has numerous hyperparameters to tune. You can use techniques like grid search or random search to find optimal hyperparameter values.

**LightGBM:**

LightGBM is a gradient boosting framework developed by Microsoft that is designed for distributed and efficient training.

**1. Installation:**

|  |
| --- |
| ***bash*** |
| pip install lightgbm |

**2. Import Libraries:**

|  |
| --- |
| ***python*** |
| import lightgbm as lgb |

**3. Data Preparation:**

- Similar to XGBoost, prepare your dataset.

**4. LightGBM Model Training:**

|  |
| --- |
| ***python*** |
| # Assuming X\_train, X\_test, y\_train, y\_test are your training and testing data  model = lgb.LGBMRegressor()  model.fit(X\_train, y\_train) |

**5. Prediction:**

|  |
| --- |
| ***python*** |
| y\_pred = model.predict(X\_test) |

**6. Evaluation:**

|  |
| --- |
| ***python*** |
| mae = mean\_absolute\_error(y\_test, y\_pred)  print(f'Mean Absolute Error: {mae}') |

**7. Hyperparameter Tuning:**

- LightGBM also has a range of hyperparameters to optimize. Experiment with different settings to improve performance.

**CATBOOST**:

This is designed to handle categorical features naturally, without the need for extensive preprocessing. It automatically encodes categorical data and handles missing values.

**ADABOOST**:

While not a gradient boosting method, AdaBoost is an ensemble technique that can be used for boosting. It combines weak learners with different weights, giving more importance to the data points that were misclassified in previous iterations.

**CONCLUSION:**

Both XGBoost and LightGBM are highly efficient and can handle large datasets with many features. Keep in mind that hyperparameters tuning is crucial for achieving the best performance. It's also recommended to understand the specific characteristics of your dataset to make informed choices regarding feature engineering and model selection.