HOUSE PRICE PREDICTION ANALYSIS

USING MACHINE LEARNING

PHASE 2:INNOVATION

INTRODUCTION:

Gradient Boosting and XGBoost are fantastic choices for improving prediction accuracy, especially in regression tasks like house price prediction. These techniques belong to the ensemble learning family, where multiple models are combined to create a stronger, more accurate predictor. Gradient Boosting builds trees sequentially, with each tree correcting the errors of the previous ones. XGBoost, or Extreme Gradient Boosting, is an optimized implementation of gradient boosting with additional features like regularization and parallel processing, making it even more powerful and efficient.

These algorithms have proven to be highly effective in various machine learning competitions and real-world applications. They handle complex relationships between features and the target variable, capture non-linear patterns, and often outperform traditional linear models.When implementing these techniques, it's crucial to tune hyperparameters carefully to achieve the best performance. Cross-validation and grid search can help identify the optimal combination of parameters for your specific dataset.

DATASET:

Our dataset have the following features:

- ➢ Avg. Area Income
- ➢ Avg. Area House Age
- ➢ Avg. Area Number of Rooms
- ➢ Avg. Area Number of Bedrooms
- ➢ Area Population
- ➢ Price
- ➢ Address

RANDOM FOREST:

The Random Forest algorithm functions by creating an ensemble of decision trees, each trained independently on random subsets of both the dataset and features. This diversity is introduced through feature selection and bootstrapping, which mitigates overfitting and enhances the model's resilience. During prediction, each decision tree contributes its output, and in regression tasks, the final prediction is typically the average of these outputs. This ensemble strategy maintains a balance between low bias and low variance, making it adept at capturing complex relationships in the data without overfitting. Additionally, the model can perform out-of-bag (OOB) evaluation to estimate its performance without requiring a separate validation set, ensuring its effectiveness in predicting housing prices within our dataset.

GRADIENT BOOSTING:

Gradient Boosting is an ensemble machine learning technique that iteratively combines the predictions of weak models, often decision trees, to create a strong predictive model. It improves accuracy by learning from the errors of the previous models, making it a popular choice for various supervised learning tasks.

Here's a high-level overview of how Gradient Boosting works:

Initialization:

In this house price prediction dataset, the initial model is usually a simple one, such as a decision tree or a regression model. This serves as the starting point for the boosting process.

Iteration:

Certainly, in your house price prediction model utilizing the dataset with features like "Avg. Area Number of Rooms," "Avg. Area Number of Bedrooms," and "Price," the concept of iterations is central. The iterative process begins with model initialization, typically using a simple decision tree. In each iteration, a new model is trained to correct the errors made by the ensemble of previous models, progressively capturing nuances and patterns in the data that enhance the model's accuracy. The predictions from each round are

aggregated, resulting in a refined predictive model, crucial for accurate house price predictions

Learning Rate:

In this house price prediction model using the provided dataset, the learning rate parameter controls the speed of model convergence and the balance between accuracy and overfitting. A lower learning rate, such as 0.01, makes the model's adjustments more conservative, leading to smoother convergence and higher accuracy but longer training times. In contrast, a higher learning rate can speed up convergence but requires careful monitoring to prevent overfitting. The choice of learning rate impacts the model's effectiveness in predicting house prices based on features like "Avg. Area Income" and "Avg. Area House Age" in your dataset.

Handles Non-Linearity:

In our house price prediction dataset, effective handling of non-linearity is pivotal for capturing the intricate relationships between features and property prices. The dataset, featuring variables such as "Avg. Area Income," "Avg. Area Number of Rooms," and "Price," presents a range of complex interactions that require a model capable of addressing non-linear patterns. Utilizing Gradient Boosting, our model excels in this aspect. By employing ensemble learning, each boosting round corrects the errors of the previous model, creating an ensemble that inherently accounts for non-linearity. This allows our model to discern intricate connections between factors like property size, room numbers, and location, enhancing its ability to accurately predict house prices. The model's adaptability to non-linearity is particularly advantageous in real estate, where factors influencing property values are often far from linear, and the dataset's diverse attributes underscore the effectiveness of this approach.

Stopping Criteria:

In our house price prediction model, it's essential to establish stopping criteria to prevent overfitting and guide the model's training process. Utilizing the dataset featuring attributes like "Avg. Area Number of Rooms," "Avg. Area Number of Bedrooms," and "Price," stopping criteria play a crucial role. We determine these criteria by specifying conditions that, when met, halt the iterative boosting process. Common stopping criteria include setting a maximum number of boosting rounds or monitoring the improvement in

performance metrics such as Root Mean Squared Error (RMSE) or Mean Absolute Error (MAE) on a validation dataset. Stopping the training process when a predefined threshold is reached ensures that our model generalizes well to unseen data and prevents it from fitting the training data too closely, thus enhancing its reliability in predicting house prices. Carefully chosen stopping criteria contribute to model stability and the avoidance of potential overfitting pitfalls.

XGBOOST:

.XGBoost, short for Extreme Gradient Boosting, is an optimized and highly efficient implementation of the Gradient Boosting algorithm. It's renowned for its speed and performance, thanks to techniques like parallel processing, tree pruning, and regularization. XGBoost is a preferred choice in many data science competitions and real-world applications due to its impressive accuracy and efficiency.

Key features of XGBoost include:

Regularization:

In our model, XGBoost uses L1 and L2 regularization to control feature importance and complexity, ensuring our model is accurate and reliable.

Parallelization:

We use parallel processing capabilities to speed up model training, which is crucial when working with large datasets and numerous features

Missing Value Handling:

We systematically address missing data, ensuring our model operates with complete and reliable information, leading to more accurate house price predictions.

Tree Pruning:

We optimize our model's decision trees by trimming them after construction, preventing overfitting and ensuring that our model is both accurate and interpretable.

Cross-Validation:

Cross-validation assesses the model's reliability and how well it performs on unseen data, providing insights into its real-world accuracy.

Optimized Implementation:

We focus on streamlining our model's performance with efficient data processing, parallelization, and hardware acceleration, striking a balance between computational speed and predictive accuracy. This ensures our house price predictions are both efficient and reliable for real estate applications

PURPOSE OF USING THESE TECHNIQUES:

What sets Gradient Boosting and XGBoost apart from traditional regression techniques is their ability to handle complex non-linear relationships in the data. Unlike simple linear regression, these ensemble learning methods iteratively build multiple models to correct errors, resulting in highly accurate

predictions, particularly suitable for datasets with intricate patterns and interactions. Additionally, they offer robust regularization options, efficient handling of missing data, and advanced optimization techniques, making them a superior choice for complex predictive modeling compared to traditional regression methods.

CODE:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.ensemble import IsolationForest
dataset = pd.read_csv('/content/USA_Housing.csv')
dataset.fillna(dataset.mean(), inplace=True)
numerical_cols = dataset.select_dtypes(include=['float64', 'int64']).columns
categorical_cols = dataset.select_dtypes(include=['object']).columns
if categorical_cols.any():
```

```python
    dataset = pd.get_dummies(dataset, columns=categorical_cols)
iso_forest = IsolationForest(contamination=0.05, random_state=42)
outliers = iso_forest.fit_predict(dataset[numerical_cols])
dataset = dataset[~(outliers == -1)]
dataset
dataset.info()
dataset.columns
dataset.describe()


from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
# Random Forest Regressor
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)
rf_regressor.fit(X_train, y_train)
# Make predictions
y_pred = rf_regressor.predict(X_test)
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)


import xgboost as xgb
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error
# XGBoost Regressor
xgb_regressor = XGBRegressor(n_estimators=100, random_state=42)
xgb_regressor.fit(X_train, y_train)
# Make predictions
y_pred = xgb_regressor.predict(X_test)
```

```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

CONCLUSION:

Gradient Boosting and XGBoost in our house price prediction model. These techniques excelled in handling complex, non-linear relationships within our dataset, comprising attributes like "Avg. Area Income" and "Avg. Area Number of Rooms." By iteratively correcting model errors and fine-tuning parameters like the learning rate and stopping criteria, Gradient Boosting proved crucial in capturing intricate patterns. Meanwhile, XGBoost's optimized implementation added efficiency, with features like regularization, parallelization, and missing data handling. The combination of tree pruning, cross-validation, and optimized implementation further refined our model's predictive prowess. Ultimately, these methods have the potential to make a substantial real-world impact on property valuation, offering valuable insights to both buyers and sellers. Future work aims to build upon this foundation, continuously improving predictive accuracy in the dynamic real estate market.