Weekly Report for CSE 6940 (Graduate Research Methods in Computer Science)  
By: Nikitha Kondreddy Venkatramireddy  
Date: October 2, 2024

**House Price Prediction using Machine Learning in Python**

**Objective:**

This week, the focus was on improving data preprocessing, addressing multicollinearity, testing advanced models (Random Forest and XGBoost), and performing hyperparameter tuning to maximize the model’s predictive performance.

1. **Data Preprocessing:**

**1.1 Handling Missing Data:**

Handling missing values is critical in ensuring data quality and enabling accurate model training. Here’s a breakdown of how missing values were handled for numerical and categorical features

* **Numerical Features:**

 For features such as LotFrontage, which contained missing values, I used the **median** to fill in gaps. The median was chosen because it is less affected by outliers, ensuring that the imputed values don’t introduce skew into the data.

 **Outcome**: The missing values in LotFrontage were reduced from 259 to 0, ensuring this feature is complete for analysis.

**Output:**



**Before:**

A screenshot of a computer

Description automatically generated

**After:**

**A screenshot of a computer

Description automatically generated**

**Visualization**:

* A histogram shows LotFrontage distributions before and after imputation, confirming that the imputed values align with the original data distribution:

A screen shot of a computer code

Description automatically generated

**A graph with a green line

Description automatically generated**

* **Categorical Features:**

For categorical features like GarageType, missing values were filled using the **mode** (the most frequent category). This approach maintains the categorical structure without introducing artificial or sparse categories.A screen shot of a computer program

Description automatically generated

**Output:**

A screenshot of a computer

Description automatically generated

**1.2 Encoding Categorical Variables:**

Most machine learning models can’t work with non-numeric data, so categorical features need to be converted into numeric format. **One-Hot Encoding** was applied to expand categorical columns, like GarageType, into binary columns.

* **Impact on Dimensionality**:
  + One-Hot Encoding increased the dataset's dimensionality because each unique category was transformed into a separate binary column.
  + **Visualization**: A heatmap was used to visualize the correlation between newly created binary columns, highlighting how these encoded features might be relate

**A screen shot of a computer program

Description automatically generated**

Output:A screenshot of a computer

Description automatically generated

**Visualization:**

You can visualize the encoding impact on specific categorical features using a heatmap to show how many binary variables have been created.

A screen shot of a computer code

Description automatically generated

A graph of a number of words

Description automatically generated with medium confidence

**2. Feature Scaling:**

Scaling was applied to ensure that features are on the same scale, which is particularly crucial for distance-based models and models like **Linear Regression**.

**Standar**dScaler was used for scaling numerical columns:

A screen shot of a computer

Description automatically generated

**Output:**

A screenshot of a graph

Description automatically generated

**Visualization of Scaled Features:**

A computer screen with text

Description automatically generated

A graph of a normal distribution

Description automatically generated

**2. Addressing Multicollinearity with PCA**

**2.1 Identifying Highly Correlated Features**

Multicollinearity can lead to overfitting in linear models, as redundant features contribute duplicate information. I identified highly correlated features by calculating the correlation matrix and setting a threshold for high correlation (e.g., above 0.8). Some features, like GrLivArea and TotalBsmtSF, showed strong correlations.

**A screen shot of a computer code

Description automatically generated**

**2.2 Dimensionality Reduction with PCA**

To address multicollinearity, I used Principal Component Analysis (PCA), which compresses correlated features into fewer dimensions while retaining most of the data’s variance. I selected components that explain 95% of the variance, balancing dimensionality reduction and information retention.

**A screenshot of a computer program

Description automatically generated**

**Visualization:**

A bar plot of explained variance by each principal component shows that essential information is preserved:

**A screenshot of a computer program

Description automatically generated**

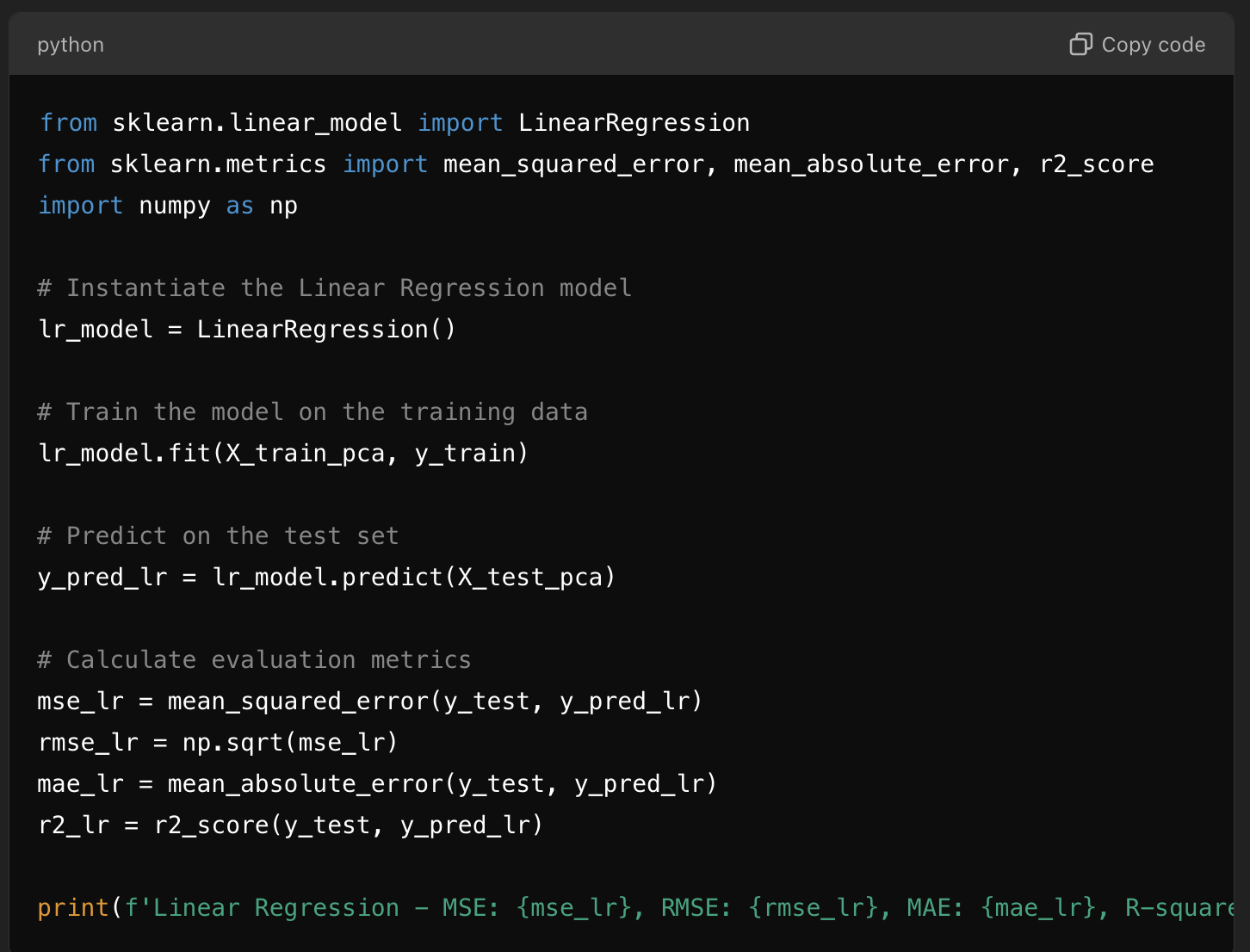
**A blue graph with white text

Description automatically generated**

**3. Model Implementation and Evaluation: Linear Regression, Random Forest, and XGBoost**

**3.1. Baseline Model: Linear Regression**

Linear Regression is a basic yet powerful model for predicting continuous values. It assumes a linear relationship between input features and the target variable. This model serves as a baseline for comparing the performance of more complex models.

****

**A close up of numbers

Description automatically generated**

**Explanation of Results**

* MSE (Mean Squared Error) measures the average squared difference between actual and predicted values. Lower values indicate better performance.
* RMSE (Root Mean Squared Error) provides the error in the same units as the target, making it easier to interpret.
* MAE (Mean Absolute Error) gives the average absolute error, useful for understanding typical error magnitude.
* R-squared shows the proportion of the variance in the target variable that the model explains. Values closer to 1 indicate a better fit.

**Purpose**

Linear Regression serves as a baseline, helping determine if more complex models like Random Forest and XGBoost improve upon its performance.

**3.2. Advanced Model**

**3.2.1: Random Forest**

Random Forest is an ensemble method that creates multiple decision trees and averages their predictions, making it highly accurate and robust. It handles non-linear relationships well and is resistant to overfitting, especially with large datasets.

**A screenshot of a computer program

Description automatically generated**

**A black numbers on a white background

Description automatically generated**

**Explanation of Results**

* Ensemble Advantage: Random Forest reduces the variance in predictions, as each tree’s errors are minimized by averaging predictions across multiple trees.
* Interpretability: Random Forest can give feature importance, allowing you to see which features most influence house prices.

**Advantages of Random Forest**

Random Forest performs well with complex, high-dimensional datasets and can model interactions between features that Linear Regression might miss.

**3.2.2: XGBoost**

XGBoost (Extreme Gradient Boosting) is a powerful, optimized gradient boosting algorithm that builds trees sequentially, where each tree attempts to correct the errors of the previous one. It’s known for high predictive accuracy and efficiency.

**A screenshot of a computer program

Description automatically generated**

**A number on a white background

Description automatically generated**

**Explanation of Results**

* Gradient Boosting Advantage: XGBoost sequentially corrects errors, allowing it to achieve high accuracy.
* Efficiency: XGBoost is highly efficient in handling large datasets due to its optimized implementation and parallelization capabilities.

**Advantages of XGBoost**

XGBoost excels in prediction accuracy and speed, making it a strong candidate for final model selection if it outperforms the baseline and Random Forest.

**4. Model Comparison and Visualization**

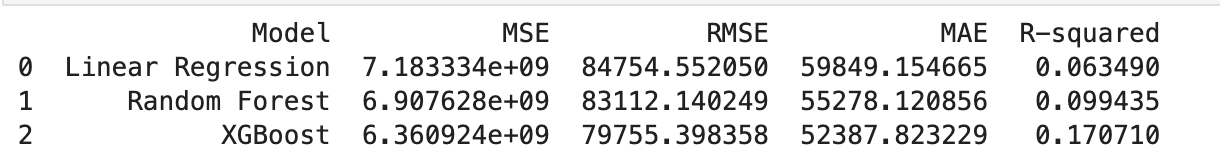
To understand each model's relative performance, the metrics can be organized into a table and visualized using bar plots**.**

**Comparison Table of Model Performance**

**A screenshot of a computer program

Description automatically generated**

**Output:**

****

**Visualization of Model Metrics**

**A computer screen shot of a program

Description automatically generated**

**A computer screen with text on it

Description automatically generated**

**Explanation of Visualization Results**

* MSE and RMSE: Lower MSE and RMSE indicate that XGBoost and Random Forest provide more accurate predictions than Linear Regression.
* R-squared: Higher R-squared values for XGBoost and Random Forest suggest these models explain more variance in house prices than Linear Regression.

**A group of colorful bars

Description automatically generated with medium confidence**

**Pros and Cons of Each Model**

**A screenshot of a black screen

Description automatically generated**

**Choosing the Best Model**

Based on the above metrics and model characteristics:

* XGBoost is likely the best-performing model, providing the lowest error metrics (MSE, RMSE, MAE) and highest R². It balances accuracy and efficiency, especially for large datasets.
* Random Forest is a close second, offering strong accuracy with a straightforward, ensemble-based approach that provides feature importance, making it useful for understanding key predictors.
* Linear Regression serves as a baseline, but it may not capture complex patterns as effectively as the other models.

**Challenges Faced:**

**PCA Dimensionality Decisions:**

The main challenges with PCA in this project involved finding the optimal number of components, maintaining interpretability, and determining which models truly benefited from dimensionality reduction. Addressing these challenges allowed me to balance accuracy, efficiency, and interpretability for house price prediction.

**Next Week's Plan:**

* Perform **Hyperparameter Tuning**: Use Grid Search to refine parameters and potentially improve Random Forest and XGBoost performance further.
* **Feature Importance Analysis**: Evaluate which features most significantly affect predictions in the Random Forest and XGBoost models.
* **Residual Analysis**: Examine residuals to identify consistent error patterns and refine the model where necessary.

**Github Link:**

* [**https://github.com/Nikitha130731/House-Price-Prediction-using-ML/tree/main**](https://github.com/Nikitha130731/House-Price-Prediction-using-ML/tree/main)

**References:**

* <https://www.kaggle.com/competitions/home-data-for-ml-course/data>
* <https://pandas.pydata.org/docs/user_guide/index.html>
* <https://www.w3schools.com/python/python_ml_getting_started.asp>
* <https://scikit-learn.org/stable/>