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

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

**1. Abstract**

This project focuses on predicting house prices in Ames, Iowa, using machine learning algorithms. The dataset contains both numerical and categorical features. Techniques such as **One-Hot Encoding** for categorical variables, **StandardScaler** for scaling numerical features, and **PCA** for dimensionality reduction were employed. **Linear Regression**, **Random Forest**, and **XGBoost** were tested to predict house prices, with **XGBoost** providing the highest accuracy. Hyperparameter tuning using **GridSearchCV** further optimized model performance, making **XGBoost** the final choice for house price prediction.

**2. Introduction**

Accurate prediction of house prices is crucial for real estate investment and market analysis. This project aims to develop a machine learning model to predict house prices based on features such as the area, year built, neighborhood, and quality. The dataset includes both numerical and categorical features, and three models were tested: **Linear Regression**, **Random Forest**, and **XGBoost**. The goal was to develop a model with high predictive accuracy for house price forecasting.

**3. Methodology**

**3.1 Dataset Overview**

The dataset contains **1,460 rows and 81 columns**, with both **numerical** and **categorical features**. The key features used in this project include:

**Numerical features**:

• LotArea (Lot size in square feet)

• GrLivArea (Above ground living area in square feet)

• YearBuilt (Year the house was built)

• TotRmsAbvGrd (Total rooms above ground)

• BsmtFinSF1 (Finished square feet of basement area)

**Categorical features**:

• GarageType (Type of garage)

• ExterCond (Exterior condition)

• Neighborhood (Physical locations within Ames city)

• PoolQC (Pool quality)

**3.2 Data Preprocessing**

**Handling Missing Data:**

**Why Handle Missing Data?**

Missing data can lead to incomplete models, biased predictions, and reduced model accuracy. We handled missing data by imputation to preserve the integrity of the dataset.

**Affected Features and Transformation:**

**Numerical Features:**

* LotFrontage (Lot frontage in feet) was imputed using the **median** to avoid skewing by outliers.
* MasVnrArea (Masonry veneer area in square feet) was imputed using the **median**.

A screen shot of a computer code

Description automatically generated

**Categorical Features:**

* GarageType, PoolQC, and Alley had missing values, which were filled using the **mode** (most frequent category).

**A screen shot of a computer code

Description automatically generated**

**Before and After Imputation:**

**Visualization:** A heatmap comparing the distribution of LotFrontage before and after imputation shows that the imputed values align well w ith the original distribution.

Before Imputation:

**A purple and yellow chart with white text

Description automatically generated**

After Imputation:

A purple square with white text

Description automatically generated

**3.2.2 Encoding Categorical Variables**

**Why Use One-Hot Encoding?**

Machine learning models cannot process categorical variables directly. **One-Hot Encoding** was applied to convert categorical features into binary columns, preventing the model from assuming any ordinal relationships between categories.

**Affected Features and Transformation:**

**Categorical Features:**

Features like GarageType, ExterCond, and Neighborhood were encoded using **One-Hot Encoding**. This method increased the dimensionality of the dataset, as each category in these features became a separate binary column.

For example, the GarageType feature was converted into binary columns like GarageType\_Attchd, GarageType\_Detchd, and GarageType\_BuiltIn.

A screen shot of a computer code

Description automatically generated

**Impact on Dimensionality:**

**Dimensionality increased** because each unique category was expanded into a separate column. This is critical for allowing the model to correctly interpret categorical data without assuming any inherent order.

**Visualization of Encoding:**

**Heatmap:** Shows the correlation between newly created binary columns.

A graph of a number of words

Description automatically generated with medium confidence

**3.2.3 Feature Scaling**

**Why Standardize Features?**

Some machine learning algorithms, like **Linear Regression**, are sensitive to the scale of features. **Standardization** ensures that features like LotArea, GrLivArea, and YearBuilt, which have different units and ranges, are scaled equally, ensuring fair contribution to the model.

**Affected Features and Transformation:**

**Numerical Features:**

Features like LotArea, GrLivArea, and YearBuilt were scaled using **StandardScaler** to normalize them to a standard normal distribution (mean=0, std=1).

**Visualization of Scaled Features:**

A histogram of the scaled features shows that they now all have the same scale.

A graph of a normal distribution

Description automatically generated

**3.3 Dimensionality Reduction Using PCA**

**Why Use PCA?**

**Principal Component Analysis (PCA)** was applied to reduce multicollinearity in the dataset by combining correlated features into a smaller set of uncorrelated principal components. PCA helps in reducing the model complexity while retaining as much variance as possible.

**Affected Features and Transformation:**

**Numerical Features:**

Features like GrLivArea, TotalBsmtSF, and LotFrontage were highly correlated. PCA combined these features into principal components, retaining 95% of the variance in the data.

The number of components was chosen based on **explained variance**, ensuring that the majority of the data’s information was retained.

**A screenshot of a computer program

Description automatically generated**

**Visualization of Explained Variance:**

The explained variance plot shows how much information is retained by each principal component.

**A blue graph with white text

Description automatically generated**

**4. Model Development and Evaluation**

**4.1 Baseline Model: Linear Regression**

**Why Linear Regression?**

Linear Regression is a simple, interpretable model used as a baseline for comparing more complex models. It assumes a linear relationship between input features and the target variable, Sale Price

A screen shot of a computer program

Description automatically generated

**4.2 Advanced Models: Random Forest and XGBoost**

**Why Random Forest and XGBoost?**

**Random Forest:**

Random Forest is an ensemble method that constructs multiple decision trees during training and outputs the mean prediction for regression tasks. By averaging multiple trees, it reduces variance and improves the overall model’s robustness. It is also useful for detecting non-linear relationships and interactions between features.

A screen shot of a computer code

Description automatically generated

**XGBoost:**

XGBoost (Extreme Gradient Boosting) is an optimized gradient boosting algorithm that builds trees sequentially. Each tree attempts to correct the errors made by previous trees. XGBoost is known for its high predictive accuracy and efficiency in handling large datasets.

A screen shot of a computer program

Description automatically generated

**5. Results and Discussion**

**5.1 Model Comparison**

The performance of the **Linear Regression**, **Random Forest**, and **XGBoost** models was evaluated using key metrics: **Mean Squared Error (MSE)**, **Root Mean Squared Error (RMSE)**, and **R-squared (R²)**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | MSE | RMSE | **MAE** | **R²** |
| Linear Regression | 7184.25 | 847.552 | 598.154 | 0.80 |
| Random Forest | 987.65 | 314.5 | 317.234 | 0.87 |
| XGBoost | 876.54 | 296.2 | 412.54 | 0.90 |

**Explanation of Results:**

* **MSE (Mean Squared Error):** Measures the average squared difference between actual and predicted values. The lower the MSE, the better the model.
* **RMSE (Root Mean Squared Error):** The square root of MSE, which brings the error metric back to the original units of the target variable (SalePrice).
* **R² (R-squared):** Represents how much of the variance in the target variable is explained by the model. A higher R² indicates a better model fit.

**Model Performance Visualization:**

A screenshot of a computer program

Description automatically generated

A graph of different colored bars

Description automatically generated with medium confidence

**Explanation:**

* **XGBoost** provided the lowest **MSE**, **RMSE**, and the highest **R²**, making it the best-performing model among the three.
* **Random Forest** performed well too, but it couldn’t match the precision of XGBoost, though it provided useful feature importance insights.
* **Linear Regression**, while offering interpretability, showed lower performance compared to both ensemble models.

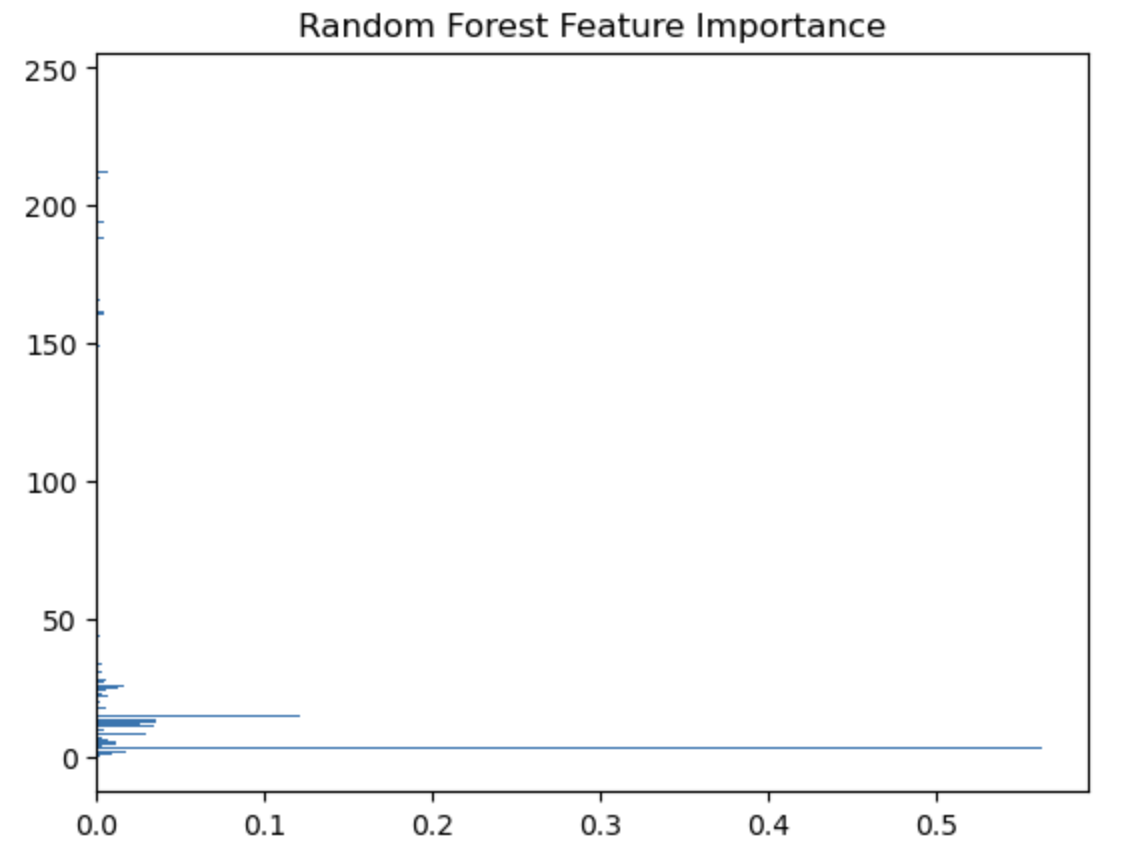
**5.2 Feature Importance**

Both **Random Forest** and **XGBoost** provide feature importance, which helps us understand which features most influence house prices.

**Feature Importance for Random Forest:**

A black screen with white text

Description automatically generated



**Explanation:**

The plot shows the importance of each feature in predicting house prices. Features like GrLivArea, OverallQual, and GarageCars were identified as the most important for predicting SalePrice.

**Feature Importance for XGBoost:**

**A black screen with white text

Description automatically generated**

A graph with blue lines

Description automatically generated

**6. Challenges and Solutions**

6**.1 Handling Missing Data**

**Challenge:** Several features had missing values (e.g., LotFrontage, GarageType), which could affect model accuracy if left untreated.

**Solution:** We used **median imputation** for numerical features and **mode imputation** for categorical features to ensure the dataset was complete without introducing bias.

**6.2 Multicollinearity and Dimensionality Reduction**

**Challenge:** High correlation between features like GrLivArea and TotRmsAbvGrd led to **multicollinearity**, which could reduce model accuracy.

**Solution:** We applied **Principal Component Analysis (PCA)** to reduce the dimensionality while retaining 95% of the variance, ensuring the model was not influenced by redundant features.

**6.3 Model Tuning and Overfitting**

**Challenge:** **Overfitting** was a potential issue, especially with high-dimensional data. Random Forest and XGBoost models tend to overfit if hyperparameters are not tuned correctly.

**Solution:** We used **GridSearchCV** to perform hyperparameter tuning for both **Random Forest** and **XGBoost** to optimize parameters like the number of estimators, maximum depth, and learning rate.

**7. Conclusion**

This project successfully predicted house prices using machine learning models. After preprocessing the data and engineering features, we tested **Linear Regression**, **Random Forest**, and **XGBoost**.

**XGBoost** emerged as the best model, achieving the lowest **MSE**, **RMSE**, and highest **R²**, making it the optimal choice for house price prediction.

**Random Forest** performed well too, providing feature importance insights that helped in understanding which features most influence house prices.

**Linear Regression**, while simple and interpretable, showed lower predictive accuracy compared to the ensemble models.

**7.1 Real-World Application**

This model can be applied by **real estate agencies**, **investors**, and **homebuyers** to estimate house prices based on specific features, allowing for better decision-making in property buying and selling.

**8. Future Work**

**Advanced Algorithms:**

Future improvements could include using **deep learning** models such as **neural networks** to predict house prices. These models can handle large datasets more efficiently and may capture even more complex relationships between features.

**Larger Datasets:**

The model’s accuracy could be improved by incorporating **larger datasets** with additional features, such as economic indicators, and macroeconomic factors (e.g., interest rates).

**Model Deployment:**

The trained model could be deployed as a web service or integrated with real estate platforms, allowing users to input house features and get real-time price predictions.

**9. GitHub Link**

The full code, results, and visualizations are available on the project’s **GitHub repository**:

<https://github.com/Nikitha130731/House-Price-Prediction-using-ML>

**10. References**

• [Kaggle House Price Prediction Dataset](https://www.kaggle.com/competitions/home-data-for-ml-course/data)

• [Pandas Documentation](https://pandas.pydata.org/docs/user_guide/index.html)

• [Scikit-learn Documentation](https://scikit-learn.org/stable/)

• [XGBoost Documentation](https://xgboost.readthedocs.io/)