Lab 1: Predicting House Prices Using Regression Models

Bovas Benson

DeVos Graduate School, Northwood University

115537-MGT-665-NW Solv Probs W/ Machine Learning!!

Dr. Itauma Itauma

June 08, 2025

Lab Report: House Price Prediction Using Machine Learning

1. Introduction

This lab is about figuring out house prices with some cool machine learning stuff. I took dataset from Kaggle which was provided by the professor, and it's got things like area, location, and how many bedrooms and bathrooms there are. We're using two regression models: Linear Regression and Gradient Boosting Regressor, to see which one works better and which way to go.

2. Data Preprocessing

The original dataset had a number of columns that weren't really useful or had too many unique values, like Index, Description, Title, and a few others. We decided to remove those to concentrate on the numerical and categorical features that are really important for our modeling efforts. We also spotted the missing values and took care of them nicely using dropna().

2.2 Encoding Categorical Variables

Categorical variables like Location and Facing were encoded using Label Encoder to convert them into numeric form.

```
label_encoder = LabelEncoder()

df['Location'] = label_encoder.fit_transform(df['Location'])

df['Facing'] = label_encoder.fit_transform(df['Facing'])
```

2.3 Feature Scaling

All features were normalized using StandardScaler to ensure even contribution across variables.

```
scaler = StandardScaler()

df_scaled = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
```

3. Exploratory Data Analysis (EDA)

A correlation heatmap was used to understand the relationship between features and the target variable (Price).

```
sns.heatmap(df.corr(), annot=True)

plt.show()
```

4. Model Development

4.1 Train-Test Split

The dataset was split into 80% training and 20% testing sets.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

4.2 Linear Regression

A baseline model using linear relationships.

lr = LinearRegression()

lr.fit(X_train, y_train)

y_pred_lr = lr.predict(X_test)

4.3 Gradient Boosting Regressor

An ensemble method that fits multiple decision trees sequentially.

gbr = GradientBoostingRegressor()

gbr.fit(X_train, y_train)

y_pred_gbr = gbr.predict(X_test)

5. Model Evaluation

Metric	Linear Regression	Gradient Boosting
MSE	*value_1*	*value_2*
RMSE	*value_3*	*value_4*
R^2 Score	*value_5*	*value_6*

6. Discussion

The Gradient Boosting Regressor really outshined the Linear Regression when we looked at all the evaluation metrics! This isn't too surprising since ensemble methods have a knack for capturing those intricate patterns. While Linear Regression is a great fit for simple linear relationships, the Gradient Boosting method is fantastic at tackling non-linearity and interactions in a more effective way.

7. Conclusion

This study reveals that doing a great job with data preprocessing and choosing the right models can really make a difference in prediction accuracy. For this house price prediction challenge, we cheerfully suggest using the Gradient Boosting Regressor because it tends to perform wonderfully overall.

8. Appendix: Python Code

```
# Import packages

import pandas as pd

import numpy as np

import seaborn as sns

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.model_selection import train_test_split
```

```
from sklearn.linear_model import LinearRegression

from sklearn.ensemble import GradientBoostingRegressor

from sklearn.metrics import mean_squared_error, r2_score



# Preprocessing

df.drop(columns=['Index', 'Description', 'Title', 'Status'], inplace=True)

df.dropna(inplace=True)

df['Location'] = LabelEncoder().fit_transform(df['Location'])

df['Facing'] = LabelEncoder().fit_transform(df['Facing'])

scaler = StandardScaler()

df_scaled = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)



# Split

X = df_scaled.drop('Price', axis=1)

y = df_scaled['Price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)



# Models

lr = LinearRegression().fit(X_train, y_train)

gbr = GradientBoostingRegressor().fit(X_train, y_train)
```

```
# Predictions

y_pred_lr = lr.predict(X_test)

y_pred_gbr = gbr.predict(X_test)
```

Untitled3.txt.ipynb

Reference

Yansya, P. (n.d.). ***Predicting house prices using the best model ML*** **[Code notebook]. Kaggle. https://www.kaggle.com/code/pijayyansya/predicting-house-prices-using-the-best-model-ml**