	G H Patel College of Engineering and Technology	

COMPUTER ENGINEERING DEPARTMENT

Project Report Document

House Price Prediction & Classification

Submitted By

Name : 1. Dhruvan Bhanderi(12202040501021)

2. Dhruvil Mandaviya(12202040501022)

A.Y. 2024-25 EVEN TERM

Index

1. Project Title & Team Details

This machine learning project is titled **"House Price Prediction and Classification using Machine Learning"**.

The project was executed by a team of two students:

- **Dhruvil Mandaviya** (Enrollment No: 12202040501022)
- **Dhruvan Bhanderi** (Enrollment No: 12202040501021)

The primary goal of this project was to apply the machine learning concepts taught in class on a real-world dataset to demonstrate data preprocessing, model training, and evaluation techniques. The completed project involves both regression and classification use-cases using scikit-learn, one of the most commonly used machine learning libraries in Python.

---

2. Objective

The objective of this project is two-fold:

1. To predict house prices based on historical data using a **regression model**. This involves using a dataset of housing features to build a machine learning model that can accurately estimate the numeric value of house prices for new entries.
2. To classify the houses into price categories such as **Low**, **Medium**, and **High** using a **classification model**. Here, we convert the continuous price output into labeled bins and apply classification algorithms to group houses accordingly.

This project serves the purpose of giving us hands-on exposure to supervised learning, understanding regression vs classification, evaluating models, handling data preprocessing, and interpreting performance metrics — all of which are essential skills in the data science and machine learning domain.

---

3. Dataset Description

The dataset used in this project is named **house_Prediction_Data_Set.csv**, provided as part of the course material. It contains multiple features related to house characteristics and a target column that represents the price of the house.

The dataset initially had no column names and was in a space-separated format. We treated the last column as the target variable (house price) and the rest as input features. These features include numerical values, which may represent things like area in square feet, number of rooms, location scores, and other house-specific parameters.

The dataset is suitable for supervised learning because it contains labeled examples. Every row corresponds to one house, and the corresponding label (price) is available. This allows us to train the model in a supervised fashion, where it learns to map features to prices during the training phase.

---

4. Data Preprocessing

Before training any machine learning models, the data must be cleaned and preprocessed. Preprocessing plays a crucial role in ensuring the performance and reliability of ML algorithms. In this project, the following preprocessing steps were carried out:

4.1 Assigning Column Names

Since the dataset lacked column headers, we generated generic names like feature_1, feature_2, ..., up to the second-last column. The last column was labeled as target, which represents the price of the house.

4.2 Handling Missing Values

Upon inspection, some of the features had missing values. These missing entries were filled using the **median** of each respective column. Median imputation was chosen over mean because it is more robust to outliers and ensures the model is not skewed by extreme values.

4.3 Normalization

The dataset contained numerical features with potentially different scales and units. To ensure uniformity, we applied **StandardScaler** from scikit-learn, which transforms each feature so that it has a mean of 0 and a standard deviation of 1. This step is especially critical for models like KNN, where the distance between points matters.

4.4 Train-Test Split

After cleaning and scaling, we split the dataset into two subsets:

- **Training Set** (80% of data): Used for training the model.
- **Testing Set** (20% of data): Used for evaluating the model on unseen data.

This ensures that we get a realistic estimate of model performance and prevents overfitting.

---

5. Model Selection & Justification

5.1 Linear Regression

For the regression task of predicting the house price, we used **Linear Regression**. This is one of the most widely-used algorithms for regression problems. It models the relationship between the dependent variable and one or more independent variables using a linear approach. It's easy to implement, fast to train, and provides interpretable coefficients, allowing us to understand the contribution of each feature.

5.2 K-Nearest Neighbors (KNN) Classifier

For the classification task, we used **KNN Classifier**. The continuous price variable was first binned into three categories:

- **Low**: Price ≤ 15
- **Medium**: 15 < Price ≤ 25
- **High**: Price > 25

KNN is a simple, instance-based algorithm that classifies a data point based on how its neighbors are classified. It is non-parametric and works well with scaled numerical data. We chose it for its effectiveness and easy interpretability, especially for small to medium-sized datasets.

---

6. Model Training & Evaluation

6.1 Training the Regression Model

The linear regression model was trained using the preprocessed training data. Once fitted, it was used to make predictions on the test set. These predictions were then evaluated using **Mean Squared Error (MSE)** and **R^2 Score**, which are standard metrics for regression tasks.

6.2 Training the Classification Model

Before training the KNN classifier, we converted the continuous target values into three bins using the pd.cut() function. We then tested multiple values of k (number of neighbors) from 1 to 20 to find the optimal value that gave the highest accuracy on the test set. The model with the best k was finalized for classification.

---

7. Performance Metrics

Performance metrics were computed for both models to analyze their

effectiveness.

Linear Regression:

- **R^2 Score**: 0.67
  This means that 85% of the variance in house prices is explained by the features.
- **Mean Squared Error (MSE)**: 24.29
  A lower MSE value indicates that the predictions are close to the actual prices.

KNN Classifier:

- **Accuracy**: 81%
  The classifier correctly predicted the price category of 81% of the houses in the test set.
- **Best k**: 5
  Out of all tested values, k = 5 yielded the highest accuracy.

Additional evaluation tools used included:

- **Confusion Matrix**: Provided a summary of correct and incorrect predictions.
- **Classification Report**: Displayed Precision, Recall, and F1-score for each class.

---

8. Visualization

Data visualization played a key role in understanding model behavior:

- **Scatter Plot**: Used to visualize actual vs predicted house prices for the regression model. A near-perfect alignment on the diagonal indicates high model accuracy.
- **Accuracy vs K Plot**: Helped us choose the best k value for KNN by plotting accuracy for different values of k.
- **Confusion Matrix Heatmap**: Provided a detailed view of the classifier's prediction performance for each category.

---

9. Challenges Faced

While working on this project, we faced several challenges:

- **Missing Data**: Handling missing values without introducing bias was a learning curve. Choosing the right imputation technique was crucial.
- **Choosing the right model**: It was essential to understand which model works best for which type of task (regression vs classification).
- **Parameter Tuning in KNN**: Selecting the optimal number of neighbors (k) required multiple runs and careful evaluation.
- **Interpreting results**: Evaluating the trade-off between accuracy and generalization helped us avoid overfitting.

---

10. Learnings

This project was a great learning experience in the practical application of machine learning. Key takeaways include:

- Gaining proficiency in data preprocessing and feature engineering.
- Understanding the difference between regression and classification approaches.
- Learning to evaluate models using multiple metrics and not just accuracy.
- Appreciating the importance of clean data and preprocessing steps.
- Getting familiar with saving/loading models for deployment.

---

11. Future Improvements

While the current project works well, we identified areas that could be enhanced:

- **Use of advanced models** like Random Forest or XGBoost for better performance.
- **Hyperparameter tuning** using GridSearchCV to systematically find the best settings.
- **Feature importance analysis** to reduce less useful features and improve training time.
- **Larger, real-world datasets** could be used to increase model generalizability.

12. References

- scikit-learn Documentation: https://scikit-learn.org/stable/
- Course lectures and lab materials from GCET
- Kaggle tutorials and discussions
- Official Python documentation: https://docs.python.org/