California Housing Price Prediction and Analysis

Project Report

Course Code: CS33002 – Applications Development Laboratory

Student Name: Tanishq Kumar Sinha

Roll Number: 2305261

Semester & Year: 6th & 2026

Instructor Name: Murari Mandal Sir

Abstract

This project develops an end-to-end machine learning pipeline using the California Housing Prices dataset from the sklearn library. The objective is to apply multiple machine learning techniques within a single workflow, including data preprocessing, exploratory data analysis, regression, classification, support vector machine modeling, clustering with PCA, neural networks, and web deployment.

The dataset was divided into training, validation, and testing sets to ensure fair evaluation and avoid data leakage. Regression models were used to predict median house values, while a classification task was created by converting house prices into low, medium, and high categories. Several models such as logistic regression, decision trees, random forests, and support vector machines were trained and compared using validation and test performance. Unsupervised learning techniques like K-means clustering and Principal Component Analysis were also applied to identify patterns and visualize data structure.

A neural network was trained using validation-based monitoring and early stopping to improve generalization. Finally, the trained models were integrated into a Flask web application that allows users to input housing features and receive real-time predictions. The project demonstrates a complete machine learning workflow from data analysis to deployment and highlights the comparative performance of different algorithms.

📘 Introduction

• Background of housing price prediction:

Housing price prediction is a common real-world machine learning problem where numerical and geographical data are used to estimate house values. Accurate prediction models help analysts, investors, and planners understand market trends.

• Motivation:

Machine learning can capture complex relationships between housing features such as income, location, and population. This project explores multiple ML techniques to compare their effectiveness on the same dataset.

• Objectives of the project:

- Build an end-to-end machine learning pipeline.
- Perform data preprocessing and exploratory analysis.
- Train and evaluate regression and classification models.
- Apply SVM, neural networks, and clustering techniques.
- Deploy trained models using a web application for real-time prediction.

📊 Dataset Description

• Dataset source:

California Housing Prices dataset obtained from sklearn.datasets.fetch_california_housing(). The dataset contains housing-related information collected from California census data.

• Feature descriptions:

- **MedInc:** Median income of households in a block group
- **HouseAge:** Median age of houses
- **AveRooms:** Average number of rooms per household
- **AveBedrms:** Average number of bedrooms per household
- **Population:** Total population in a block

- **AveOccup:** Average number of occupants per household
- **Latitude:** Geographic latitude coordinate
- **Longitude:** Geographic longitude coordinate

• **Target variable:**
MedHouseVal – Median house value in each block, represented as a continuous numerical variable.

• **Problem reformulation (regression & classification):**

- **Regression Task:** Predict the continuous median house value using input features.
- **Classification Task:** Convert house values into three categories (Low, Medium, High) using quantile-based thresholds derived from training data.

⚙️ Data Preprocessing

• **Missing value handling:**
The California Housing dataset did not contain missing values; therefore, no imputation or removal of samples was required.

• **Feature scaling:**
Standardization was applied using StandardScaler to normalize feature distributions. Scaling parameters were learned only from the training set and then applied to validation and test sets.

• **Train-test split:**
The dataset was divided into training, validation, and test sets using a 70% / 15% / 15% ratio with random_state=42 to ensure reproducibility. The validation set was used for model selection and tuning, while the test set was reserved for final evaluation.

• **Justification of preprocessing steps:**
Preprocessing ensures consistent feature scales for algorithms sensitive to feature magnitude (e.g., SVM and neural networks). Using training data statistics for scaling prevents data leakage and ensures fair model evaluation.

📈 Exploratory Data Analysis

• Feature distributions:

Histograms were plotted to examine the distribution of individual features such as median income, house age, and population. Some features showed skewed distributions, indicating varying data density across regions.

• Scatter plots:

Scatter plots were used to visualize relationships between important variables, particularly median income versus house value. These plots helped identify trends and potential nonlinear relationships.

• Correlation heatmap:

A correlation heatmap was generated to analyze relationships among features and the target variable. This helped identify highly correlated attributes and understand feature importance.

• Key insights:

- Median income showed strong positive correlation with house value.
- Geographical features (latitude and longitude) influenced housing prices significantly.
- Certain features displayed moderate correlation, suggesting that multi-feature models would perform better than single-feature models.

📉 Regression Models

5.1 Simple Linear Regression

• Model description:

A simple linear regression model was trained using a single feature (median income) to predict median house value. This model establishes a baseline and helps understand the relationship between income and housing prices.

• **Evaluation metrics:**
Model performance was evaluated using Mean Squared Error (MSE) and R^2 score on validation and test sets.

• **Plots:**
An actual vs predicted scatter plot was generated to visualize model performance and prediction errors.

5.2 Multiple Linear Regression

• **Model description:**
A multiple linear regression model was trained using all available features to capture combined effects of socioeconomic and geographic factors on house value.

• **Performance comparison:**
Multiple linear regression showed improved accuracy compared to simple linear regression due to the inclusion of additional explanatory features, resulting in lower error and higher R^2 score.

🧠 Classification Models

• **Logistic Regression:**
A logistic regression classifier was trained on scaled features to establish a baseline classification performance for predicting low, medium, and high house value categories.

• **Decision Tree:**
A decision tree classifier was trained to capture nonlinear relationships and hierarchical decision boundaries within the dataset.

• **Random Forest:**
A random forest classifier was implemented as an ensemble of decision trees to improve robustness, reduce overfitting, and achieve higher classification accuracy.

• **Confusion matrices and classification reports:**
Confusion matrices were generated to visualize prediction errors across classes.

Classification reports including precision, recall, and F1-score were used to evaluate overall model quality.

• **Comparative discussion:**
Random Forest achieved the best performance among the classification models due to ensemble learning and its ability to model complex feature interactions. Logistic Regression provided stable baseline results, while Decision Tree showed higher variance and potential overfitting.

🎯 Support Vector Machine

• **Kernel used:**
Support Vector Machine (SVM) classifiers were trained using linear and RBF kernels. Feature scaling was applied before training, and the kernel with better validation performance was selected.

• **Results:**
The SVM model achieved strong classification accuracy, showing effective separation between house value categories. Performance metrics were evaluated using validation and test datasets.

• **Comparison with other classifiers:**
SVM provided competitive performance compared to Random Forest and Logistic Regression. While it offered strong decision boundaries, training time was higher compared to tree-based models. Random Forest remained the best overall performer due to higher robustness and easier interpretability.

🔍 Principal Component Analysis (PCA)

• **Purpose:**

PCA is used to reduce the dimensionality of the dataset while preserving the most important information.

• **Working principle:**

It transforms original features into a new set of uncorrelated variables called principal components.

• **Variance preservation:**

The first few principal components capture the maximum variance present in the data.

• **Dimensionality reduction:**

In this project, PCA was used to reduce the dataset to two components for visualization.

• **Visualization:**

The reduced data was plotted to observe clustering patterns and data distribution.

🧠 Neural Network

• **Model type:**

A feedforward neural network (deep learning model) designed to learn complex patterns in data.

• **Architecture:**

The model consists of an input layer, two hidden layers with ReLU activation, and an output layer with softmax activation for multi-class classification.

• **Activation functions:**

ReLU (Rectified Linear Unit) was used in hidden layers to introduce nonlinearity, while softmax was used in the output layer to produce class probabilities.

• **Training process:**

The model was trained using the Adam optimizer and sparse categorical crossentropy loss function.

• **Regularization technique:**

Early stopping was applied to monitor validation loss and prevent overfitting by stopping training when performance stopped improving.

🌐 Web Deployment

• System architecture:

The deployed system follows a client–server architecture where users enter housing features through a web interface. The input data is sent to the Flask backend, processed using trained machine learning models, and prediction results are returned and displayed on the webpage.

Workflow:

User Input → Flask Backend → Data Preprocessing → ML Models → Prediction Output

---

• Backend implementation:

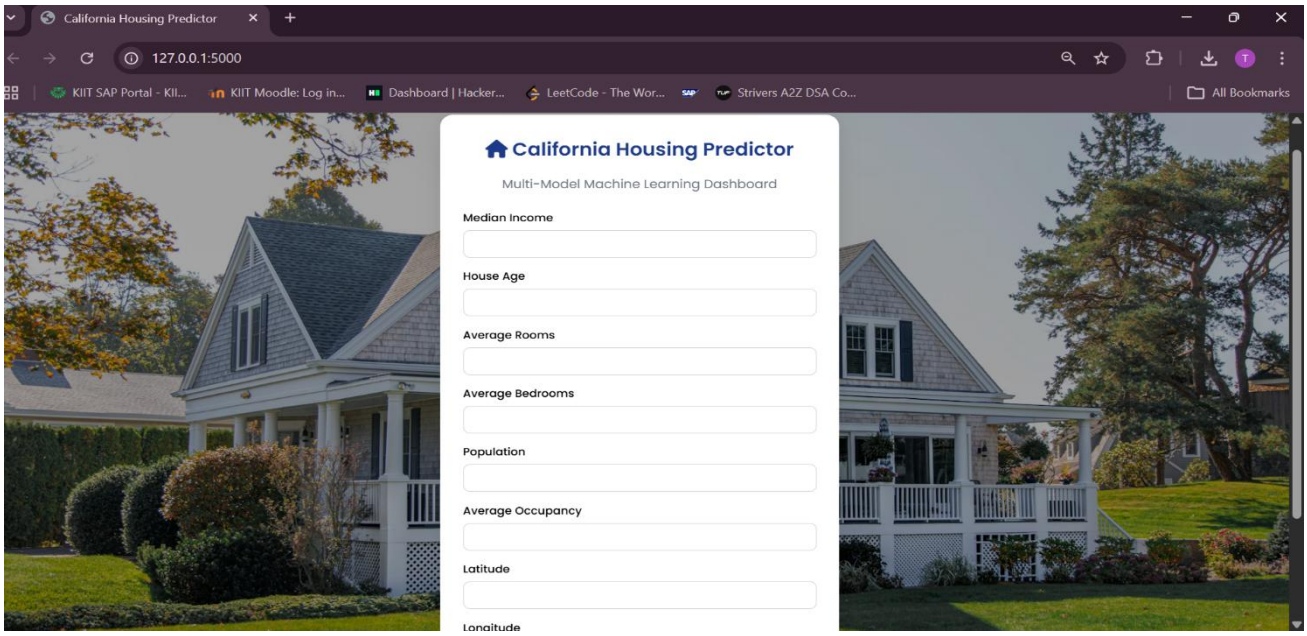
- Flask framework was used to build the backend web application.
- Trained models (Regression, Random Forest, SVM, and Neural Network) were loaded using Joblib and TensorFlow.
- Input validation and error handling were implemented using Flask request handling and flash messages.
- StandardScaler was applied before prediction to maintain consistency with training preprocessing steps.

• Frontend interface:

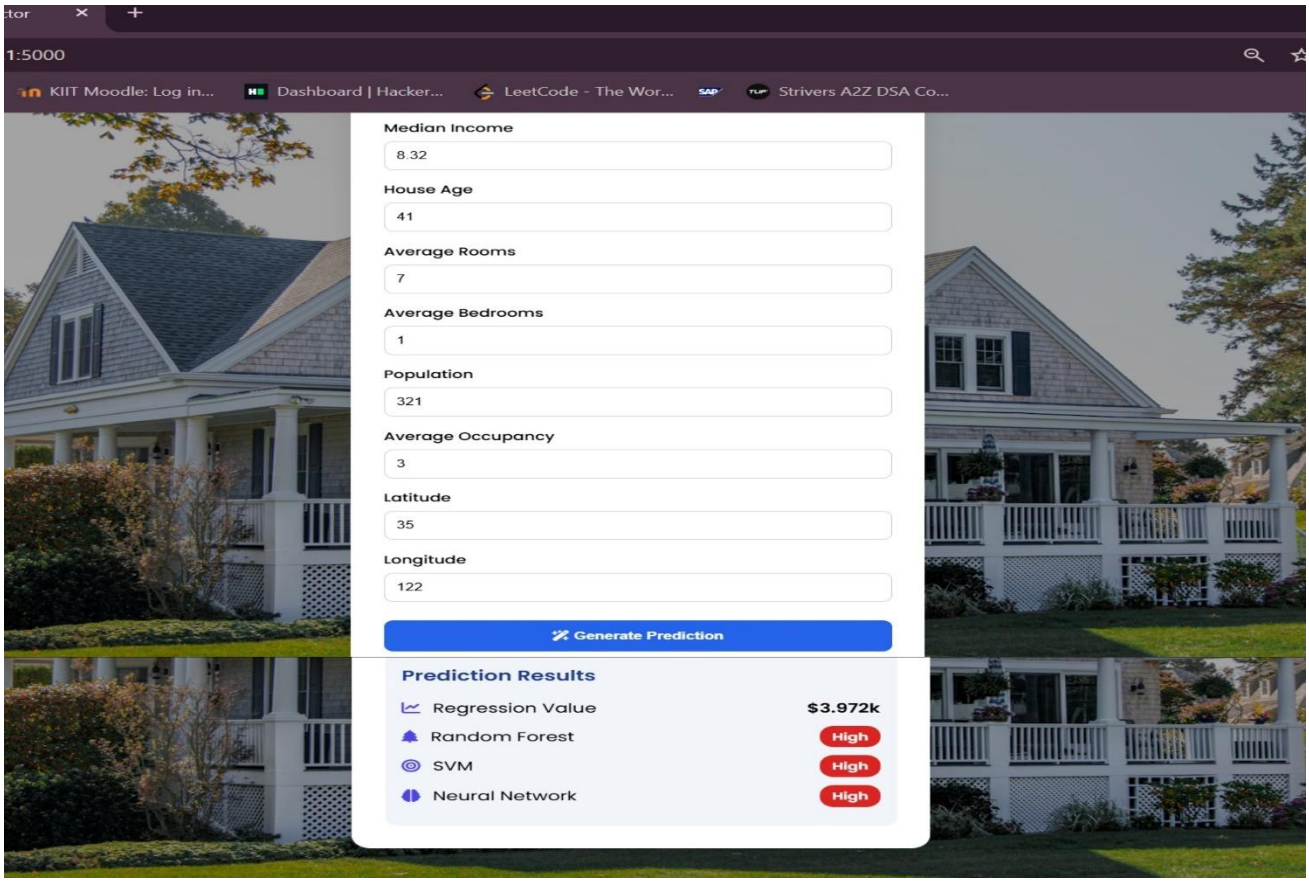
- HTML and CSS were used to create an interactive user interface.
- The interface includes input fields for all housing features.
- Model predictions are displayed as styled result cards with color-coded badges (Low, Medium, High).
- Icons, modern fonts, and background styling were added to improve visual appearance and usability.

Screenshots

1 Home Page Screenshot



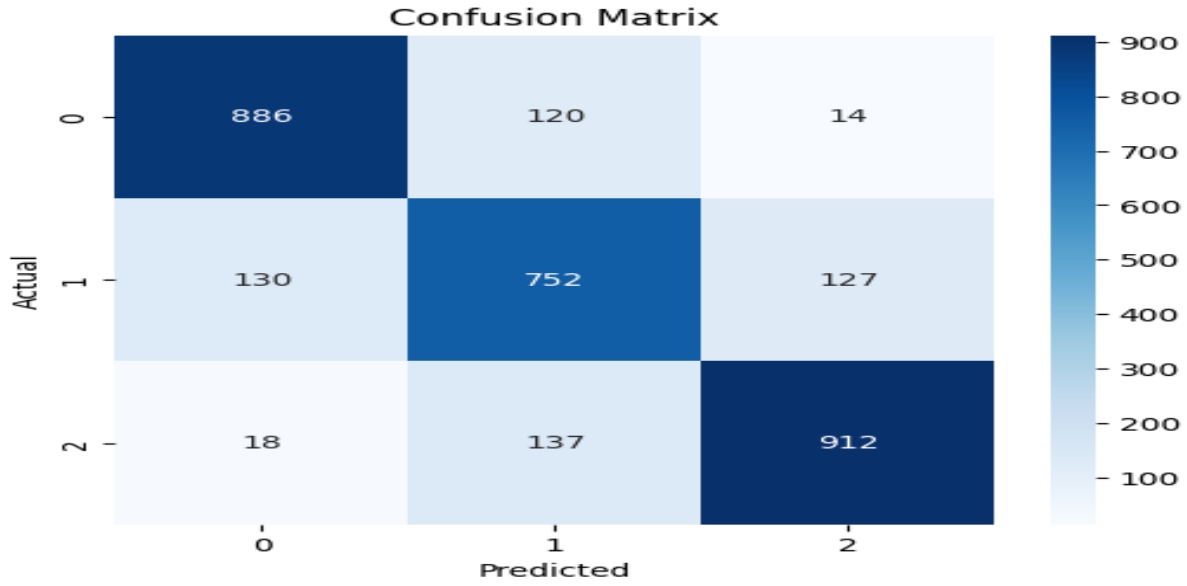
2 Prediction Output Screenshot



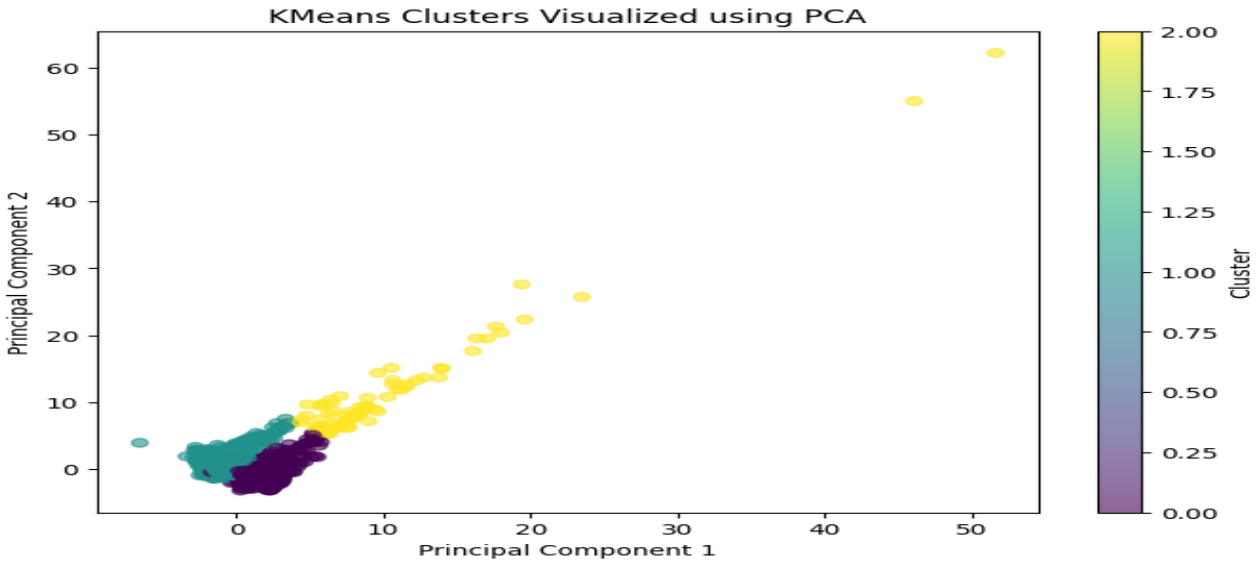
3 Model Evaluation (Metrics)

```
Test Accuracy: 0.8236434108527132

Classification Report:

              precision    recall  f1-score   support

           0       0.86      0.87      0.86      1020
           1       0.75      0.75      0.75      1009
           2       0.87      0.85      0.86      1067

    accuracy                           0.82      3096
   macro avg       0.82      0.82      0.82      3096
weighted avg       0.82      0.82      0.82      3096
```

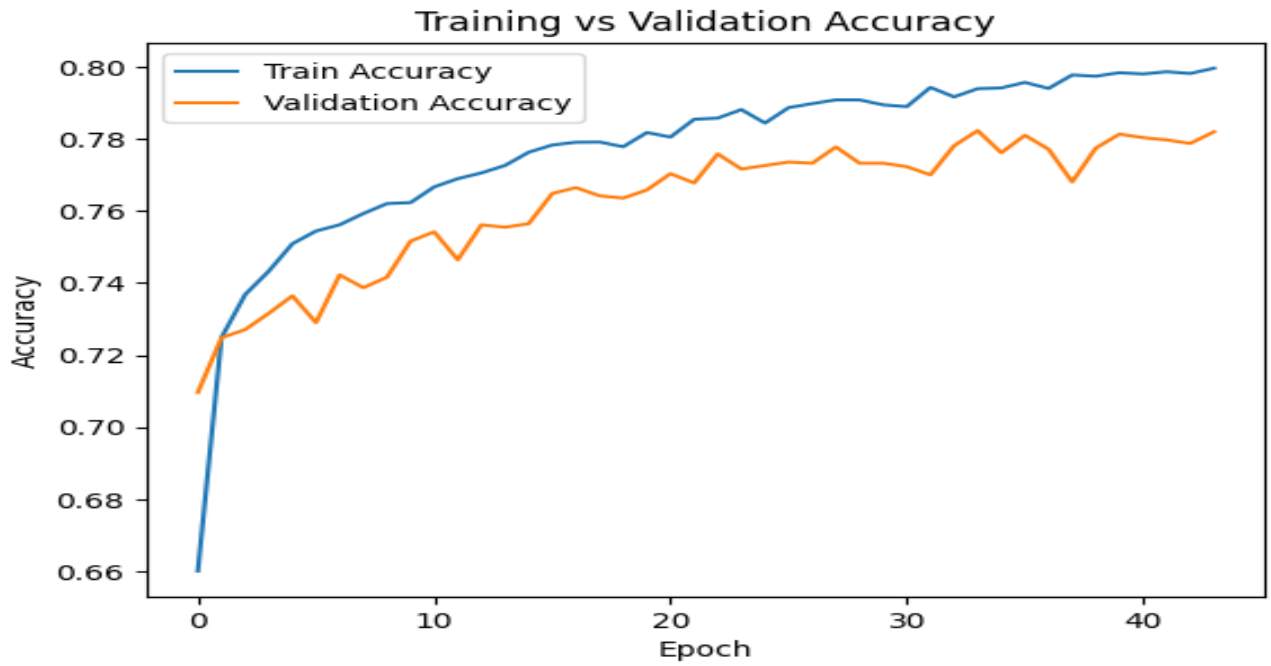
4 Confusion Matrix



5 PCA / Clustering Plot



6️⃣ Neural Network Training Plot



📊 Results & Discussion

• Overall comparison of models:

Multiple models were evaluated across regression and classification tasks. Multiple Linear Regression outperformed Simple Linear Regression due to the use of all features. Among classification models, Random Forest achieved the highest accuracy, followed by SVM and Logistic Regression. The neural network also provided competitive performance with stable generalization.

• Observations:

- Median income was the most influential feature in predicting housing prices.
- Ensemble methods like Random Forest handled nonlinear relationships effectively.
- SVM performed well after feature scaling but required higher computation.
- Neural networks showed consistent performance with proper tuning and early stopping.

- Different models sometimes produced varying class predictions, indicating diverse learning patterns.

• **Limitations:**

- Limited hyperparameter tuning was performed, which may affect optimal performance.
- Neural network performance depends on architecture and training configuration.
- Dataset size and features are restricted to the provided California dataset.
- The web application is deployed locally and not on a cloud platform.

✅ Conclusion & Future Work

• **Summary of outcomes:**
An end-to-end machine learning pipeline was successfully developed for California housing price prediction. The project included data preprocessing, exploratory analysis, regression, classification, SVM, clustering, PCA, and neural network modeling. Multiple models were compared, and a web application was built to provide real-time predictions, demonstrating practical deployment of machine learning techniques.

• **Possible improvements:**

- Perform advanced hyperparameter tuning using GridSearchCV or RandomizedSearchCV.
- Deploy the application on cloud platforms such as AWS or Heroku.
- Incorporate additional features or external datasets to improve prediction accuracy.
- Enhance the neural network architecture for better performance.
- Add user authentication and database integration for a complete production-level system.

📚 References

• **Dataset source:**

- California Housing Dataset – Available via Scikit-learn (sklearn.datasets.fetch_california_housing).

• **Libraries and tools used:**

- Scikit-learn – For machine learning models and preprocessing
- TensorFlow / Keras – For neural network implementation
- Pandas – For data manipulation and analysis
- NumPy – For numerical computations
- Matplotlib & Seaborn – For data visualization
- Flask – For web application deployment
- Joblib – For saving and loading trained models
- Python – Programming language used for implementation