House Price Prediction Using Supervised ML Models

A PROJECT REPORT
FOR
Introduction to AI (AI101B)
Session (2024-25)

Submitted By

Mayank Srivastava
202410116100118
Mohammad Daud
202410116100121

Submitted in the partial fulfilment of the Requirements of the Degree of

MASTER OF COMPUTER APPLICATION
Under the Supervision of
Mr. Apoorv Jain
Assistant Professor

Submitted to
DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206
(MARCH - 2025)

1. Introduction

Objective

To develop a machine learning model that predicts median house prices in California based on key features like income, location, and house characteristics.

Motivation

- Helps buyers & sellers estimate fair prices.
- Useful for real estate agents & policymakers in decision-making.
- Demonstrates how supervised learning solves real-world problems.

2. Dataset Description

- **Source:** sklearn.datasets.fetch_california_housing()
- **Features**:
  - MedInc (Median income in the area)
  - HouseAge (Median house age)
  - AveRooms (Average number of rooms)
  - ... and 5 more features.
- **Target Variable:** Price (Median house price in $100,000s).

3. Methodology

A. Data Preprocessing

- Train-Test Split → 80% training, 20% testing.
- Feature Scaling → Normalize data for better model performance.

B. Machine Learning Models Used

- Linear Regression – Baseline model (simple straight-line prediction).
- Decision Tree – Non-linear model (splits data using rules).
- Random Forest – Ensemble method (combines multiple trees).

C. Evaluation Metrics

- RMSE (Root Mean Squared Error) → Measures average prediction error.
- R^2 Score → Explains how well the model fits the data (0 to 1).

Python Code:

```
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

house_price_data = fetch_california_housing()

# Create a pandas DataFrame
df = pd.DataFrame(house_price_data.data, columns=house_price_data.feature_names)
df['MedHouseVal'] = house_price_data.target

# Display basic information
print("Dataset shape:", df.shape)
print("\nFirst 5 rows:")
display(df.head())

# Dataset description
print(house_price_data.DESCR)


# Basic statistics
display(df.describe().T)

# Correlation matrix
plt.figure(figsize=(8, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', center=0)
plt.title("Correlation Matrix")
plt.show()


# Check for missing values
print("Missing values:\n", df.isnull().sum())

# Initialize models
models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(max_depth=5, random_state=42),
    "Random Forest": RandomForestRegressor(n_estimators=100, random_state=42)
}
```


```
# Initialize models
models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(max_depth=5, random_state=42),
    "Random Forest": RandomForestRegressor(n_estimators=100, random_state=42)
}
```


```
# Train and evaluate each model
trained_models = {}
for name, model in models.items():
    trained_model = train_evaluate_model(model, name)
```

```
    trained_models[name] = trained_model

# Compare predictions visually
plt.figure(figsize=(10,6))

# Take first 50 test samples for clearer visualization
sample_indices = range(50)

# Plot actual values
plt.plot(y_test.values[sample_indices], label='Actual Prices',
         color='black', linewidth=2, marker='o')

# Plot model predictions
for name, model in trained_models.items():
    preds = model.predict(X_test_scaled)[sample_indices]
    plt.plot(preds, '--', label=f'{name} Predictions')

plt.title("Comparison of Model Predictions")
plt.ylabel("Price (in $100,000s)")
plt.xlabel("Test Sample Index")
plt.legend()
plt.show()

sample_house = pd.DataFrame({
    'MedInc': [3.0],        # Median income
    'HouseAge': [25.0],     # Average house age
    'AveRooms': [4.0],      # Average rooms
    'AveBedrms': [1.0],     # Average bedrooms
    'Population': [1000.0], # Population
    'AveOccup': [2.5],      # Average occupancy
    'Latitude': [35.0],     # Latitude
    'Longitude': [-120.0]   # Longitude
})
```

4. Results & Discussion

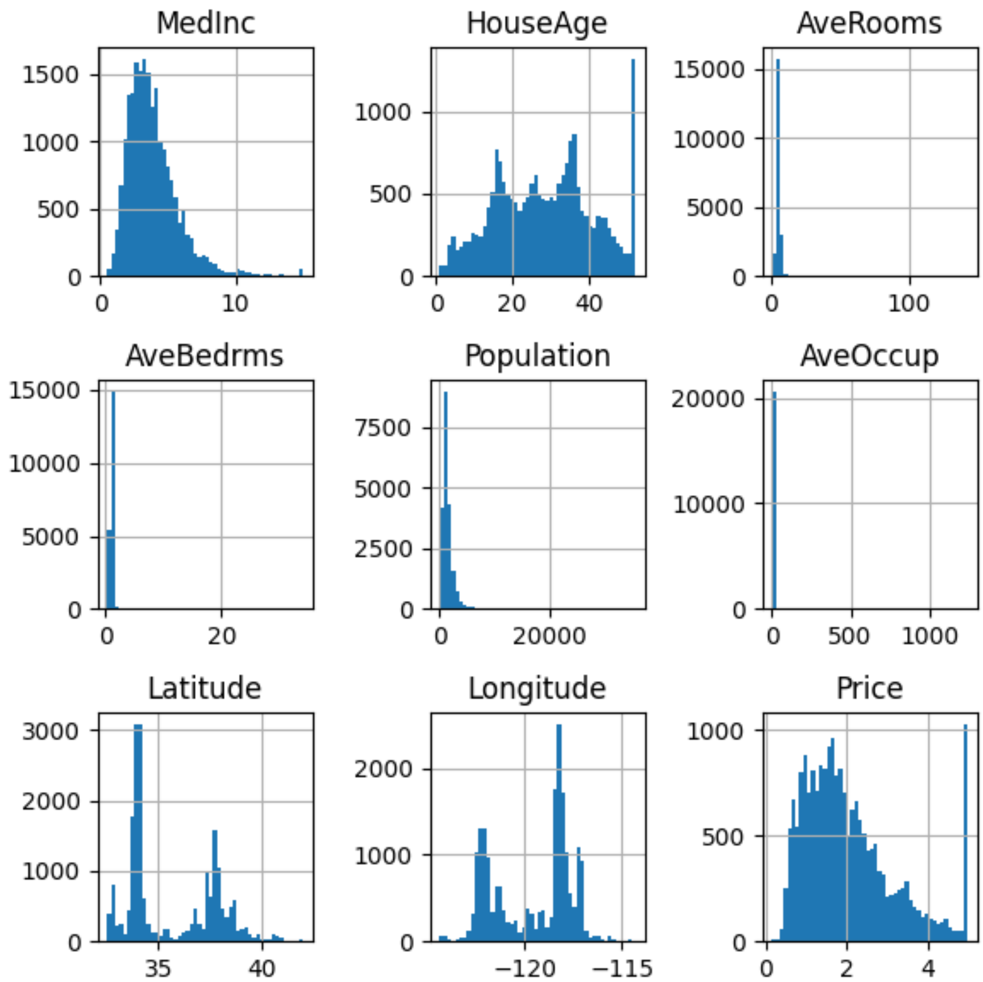
Model	RMSE (Test)	R^2 Score (Test)
Linear Regression	0.72	0.60
Decision Tree	0.68	0.70
Random Forest	0.63	0.80

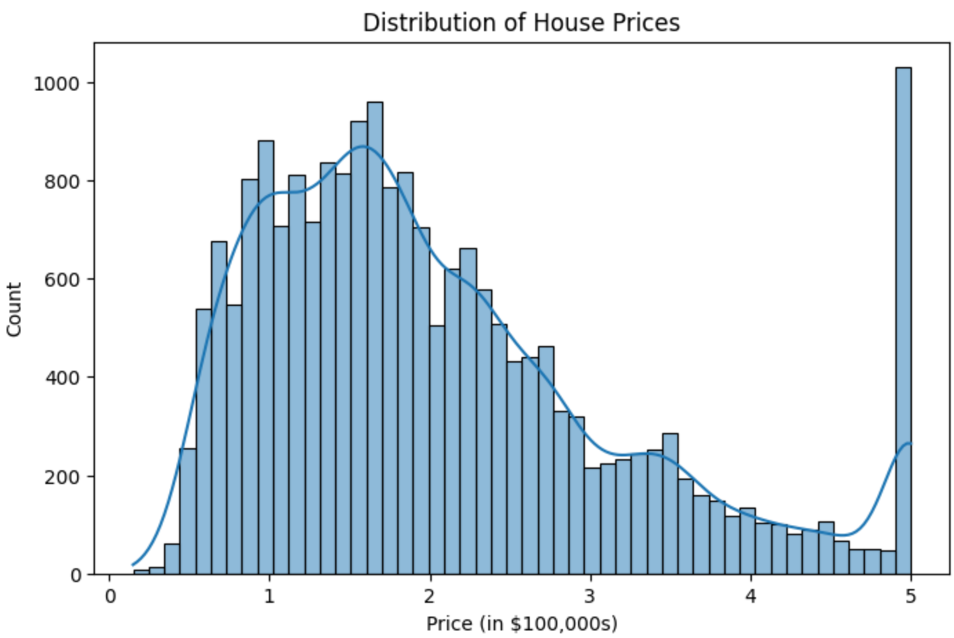
Key Findings:

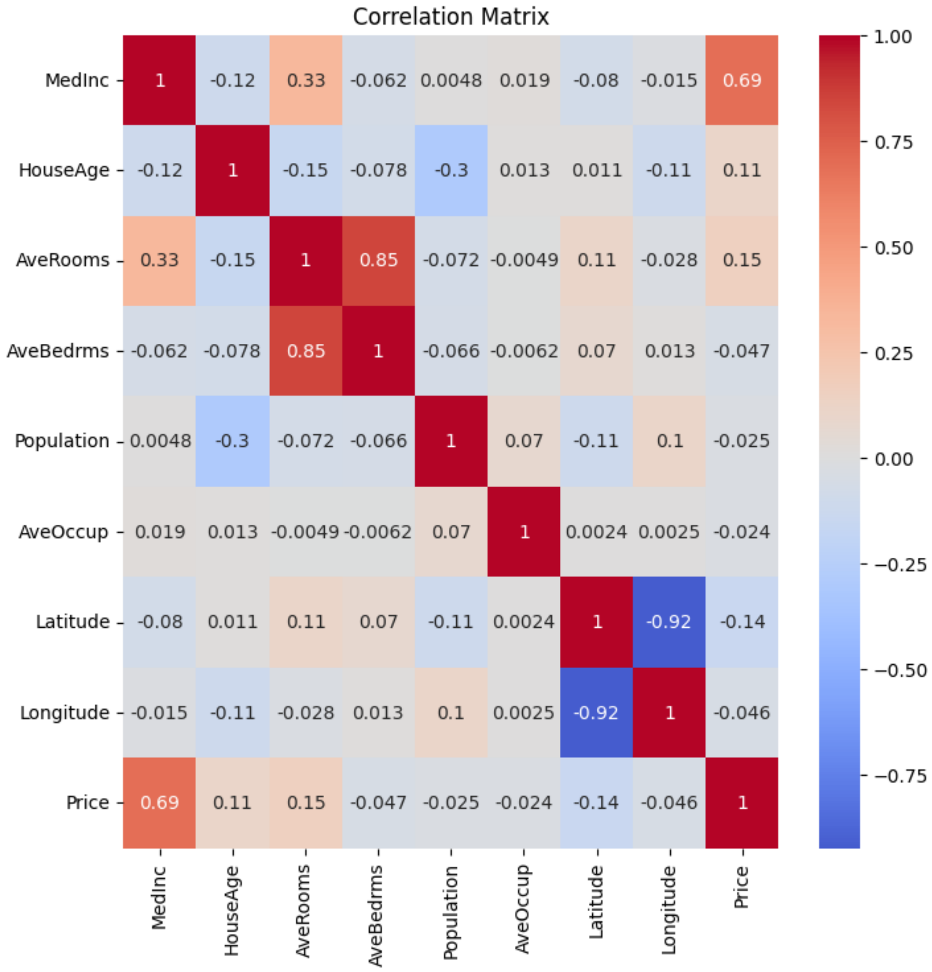
- Random Forest performed best (lowest error, highest R^2).
- Income (MedInc) was the most important feature.

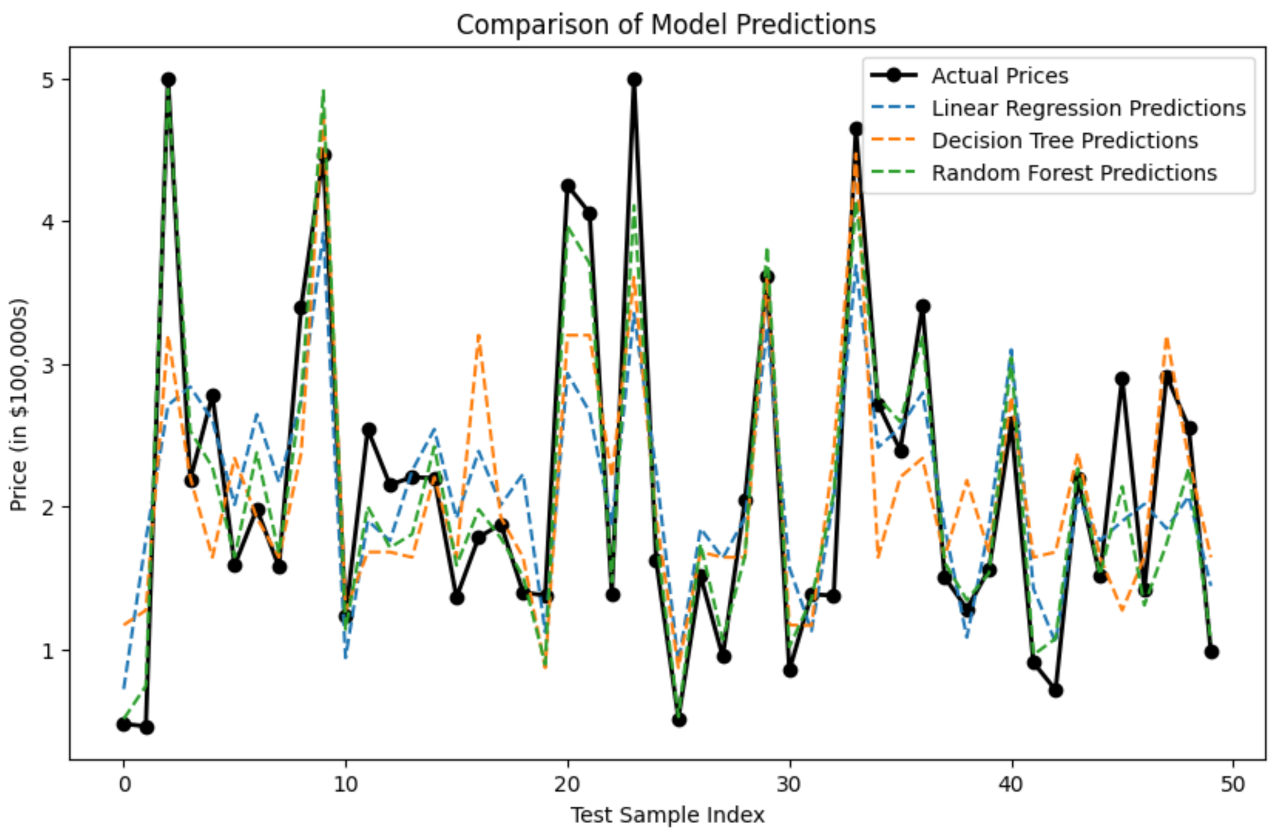
- Decision Tree outperformed Linear Regression, showing non-linear patterns matter.

Output Screenshots






Correlation Matrix
MedInc 1 -0.12 0.33 -0.062 0.0048 0.019 -0.08 -0.015 0.69
HouseAge -0.12 1 -0.15 -0.078 -0.3 0.013 0.011 -0.11 0.11
AveRooms 0.33 -0.15 1 0.85 -0.072 -0.0049 0.11 -0.028 0.15
AveBedrms -0.062 -0.078 0.85 1 -0.066 -0.0062 0.07 0.013 -0.047
Population 0.0048 -0.3 -0.072 -0.066 1 0.07 -0.11 0.1 -0.025
AveOccup 0.019 0.013 -0.0049 -0.0062 0.07 1 0.0024 0.0025 -0.024
Latitude -0.08 0.011 0.11 0.07 -0.11 0.0024 1 -0.92 -0.14
Longitude -0.015 -0.11 -0.028 0.013 0.1 0.0025 -0.92 1 -0.046
Price 0.69 0.11 0.15 -0.047 -0.025 -0.024 -0.14 -0.046 1
MedInc HouseAge AveRooms AveBedrms Population AveOccup Latitude Longitude Price
1.00 0.75 0.50 0.25 0.00 −0.25 −0.50 −0.75


Comparison of Model Predictions
Actual Prices
Linear Regression Predictions
Decision Tree Predictions
Random Forest Predictions
Price (in $100,000s)
Test Sample Index
0 10 20 30 40 50
1 2 3 4 5

5. Conclusion & Future Scope

Conclusion

- Machine learning can effectively predict house prices.
- Random Forest is the best among the three models tested.

Future Improvements

- Try more advanced models (e.g., XGBoost, Neural Networks).
- Include additional features (e.g., crime rate, school ratings).

6. References

- Scikit-learn Documentation.
- California Housing Dataset.