Machine Learning for Real Estate: Prediction and Classification of California Housing Prices

Project Report

Name: Nitin Singh (2511AI25)
Submitted to: Dr. Chandranath Adak

Date: September 18, 2025

Abstract

The real estate market is a complex ecosystem influenced by a multitude of factors. Predicting property values is a critical task for investors, homeowners, and policymakers. This project leverages the power of machine learning to analyse the California Housing dataset, a classic benchmark in the field. We develop and evaluate two distinct models to address different business needs: a **Linear Regression** model to predict the precise median house value and a **Logistic Regression** model to classify houses into 'Cheap' or 'Expensive' categories. The Linear Regression model achieved an R-squared value of approximately 0.60, indicating that our features can explain 60% of the variance in house prices. The Logistic Regression model demonstrated strong performance with a classification accuracy of around 82%. This report details the entire project lifecycle, from data exploration and preprocessing to model implementation, evaluation, and discussion of the results, highlighting the practical application of regression algorithms in the real estate domain.

Table of Contents

1. Introduction

1.1. The Importance of Housing Price Prediction

The valuation of residential property is a cornerstone of the economy. For individuals, a house is often the most significant financial investment they will ever make. For businesses, accurate real estate valuation is crucial for investment, risk management, and urban development. Governments also rely on property valuations for taxation and policy-making. The dynamic and often volatile nature of the housing market makes accurate price prediction a challenging yet highly valuable endeavour. Traditional methods often rely on comparative market analysis and expert opinion, which can be subjective and slow.

1.2. Machine Learning in Real Estate

Machine learning (ML) offers a data-driven approach to overcome the limitations of traditional valuation methods. By analyzing vast amounts of historical data, ML algorithms can identify complex, non-linear patterns and relationships between various property attributes and their market value. This project focuses on two fundamental regression-based algorithms:

- **Linear Regression:** A model used to predict a continuous outcome. In our case, it will predict the exact median house value in units of $100,000.
- **Logistic Regression:** Despite its name, this is a classification algorithm. It's used to predict a binary outcome. We will use it to classify a house as being either 'Cheap' or 'Expensive' based on a price threshold.

1.3. Project Objectives

This project aims to demonstrate a practical, end-to-end machine learning workflow for real estate analysis. The primary objectives are:

1. To perform a thorough **Exploratory Data Analysis (EDA)** on the California Housing dataset to uncover insights and relationships between features.
2. To build, train, and evaluate a **Linear Regression model** to predict the median house value (MedHouseVal).
3. To develop a **Logistic Regression model** to classify houses into binary categories ('Cheap' vs. 'Expensive').
4. To interpret the results of both models and discuss their practical implications in a real-world context.

2. Dataset Exploration and Description

2.1. Dataset Overview

This project utilizes the **California Housing dataset**, which is readily available in the Scikit-learn library (sklearn.datasets.fetch_california_housing). This dataset is based on data from the 1990 California census. It is a popular benchmark for regression tasks and serves as a modern replacement for the older Boston Housing dataset. The dataset contains **20,640 samples** (representing block groups) and **8 numeric features**, with one target variable.

2.2. Feature Descriptions

The features (X) and the target variable (y) are described below:

- **MedInc:** Median income for households within a block group (in tens of thousands of US Dollars).
- **HouseAge:** Median age of a house within a block group.
- **AveRooms:** Average number of rooms per household.
- **AveBedrms:** Average number of bedrooms per household.
- **Population:** Total population of the block group.
- **AveOccup:** Average number of household members.
- **Latitude:** The north-south coordinate of the block group's center.
- **Longitude:** The east-west coordinate of the block group's center.

Target Variable (y):

- **MedHouseVal:** The median house value for households within a block group (in units of $100,000). For example, a value of 2.5 corresponds to $250,000.

2.3. Exploratory Data Analysis (EDA)

Before model building, we must understand the data's structure, distributions, and inter-feature relationships.

2.3.1. Summary Statistics

A descriptive statistical summary provides a quick overview of the central tendency, dispersion, and shape of the dataset's distribution.

2.3.2. Data Visualization and Distribution

Histograms help visualize the distribution of each feature. Most features, particularly MedInc, are right-skewed. The target variable MedHouseVal appears to be capped at 5.0 ($500,000), which is a known characteristic of this dataset.

2.3.3. Correlation Analysis

A correlation matrix helps us understand the linear relationships between variables. The heatmap below visualizes these correlations.

- **Key Insight:** There is a strong positive correlation (**0.69**) between **Median Income (MedInc)** and the **Median House Value (MedHouseVal)**. This is intuitive: areas with higher incomes tend to have more expensive houses. AveRooms also shows a moderate positive correlation with house value.

2.3.4. Geographic Insights

By plotting Latitude against Longitude and coloring the points by MedHouseVal, we can visualize the geographical distribution of house prices in California.

- **Key Insight:** The plot clearly shows higher house prices clustered around major coastal metropolitan areas like Los Angeles (Southern California) and the San Francisco Bay Area (Northern California), which aligns with real-world knowledge.

---

3. Methodology

3.1. Data Preprocessing

Before feeding data into the models, a few preprocessing steps are essential.

1. **Train-Test Split:** The dataset was split into a training set (80%) and a testing set (20%). The model learns from the training data, and its performance is evaluated on the unseen testing data to ensure it generalizes well. We use a random_state for reproducibility.
2. **Feature Scaling:** While not strictly necessary for simple Linear Regression, scaling features is good practice and essential for many other algorithms. It ensures that features with larger numeric ranges do not dominate the model's learning process. We used StandardScaler from Scikit-learn, which standardizes features by removing the mean and scaling to unit variance.

3.2. Model 1: Linear Regression for Price Prediction

3.2.1. Model Theory

Linear Regression is a fundamental algorithm that models the relationship between a dependent variable (target) and one or more independent variables (features) by fitting a linear equation to the observed data. The goal is to find the best-fitting line (or hyperplane in multiple dimensions) that minimizes the distance between the actual and predicted values.

The multiple linear regression equation is:

$$Y=\beta0+\beta1X1+\beta2X2+...+\beta nXn+\epsilon$$

Where:

- Y is the predicted target variable (MedHouseVal).
- X_i are the input features.
- β_0 is the intercept (the value of Y when all X are 0).
- β_i are the coefficients, representing the change in Y for a one-unit change in X_i.
- ϵ is the model's error term.

3.2.2. Evaluation Metrics

- **Mean Squared Error (MSE):** This metric measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. A lower MSE indicates a better fit.

$$MSE=n1_{i}=1\sum n(Yi-Yi\hat{})2$$

- **R-squared (R^2):** Also known as the coefficient of determination, R^2 represents the proportion of the variance in the dependent variable that is predictable from the independent variables. It provides a measure of how well the model explains the data, with values ranging from 0 to 1. An R^2 of 0.60 means 60% of the variance in house prices is explained by our model.

3.3. Model 2: Logistic Regression for Price Classification

3.3.1. Target Variable Transformation

To frame this as a classification problem, we need to convert the continuous MedHouseVal target into discrete binary labels. We use the **median** of MedHouseVal (approximately 1.80, or $180,000) as a threshold:

- Houses with MedHouseVal < 1.80 are labelled **0 (Cheap)**.
- Houses with MedHouseVal ≥ 1.80 are labelled **1 (Expensive)**.

Using the median ensures that our two classes are balanced, which is important for training a robust classifier.

3.3.2. Model Theory

Logistic Regression is a statistical model used for binary classification. It models the probability that a given input point belongs to a certain class. It passes the weighted sum of the inputs through a special function called the **sigmoid function**, which squashes the output to a value between 0 and 1.

The sigmoid function is defined as:

$\sigma(z)=1+e-z1$

Where z is the linear combination of features (z=β_0+β_1X_1+...). The output can be interpreted as the probability of the positive class (in our case, 'Expensive'). A threshold (typically 0.5) is then used to make the final classification.

3.3.3. Evaluation Metrics

- **Accuracy:** The most intuitive metric, it simply measures the proportion of total predictions that were correct.

Accuracy=Total Samples True Positives + True Negatives

- **Confusion Matrix:** A table that provides a more detailed breakdown of a classifier's performance. It shows the number of correct and incorrect predictions for each class. The four components are:
    - **True Positives (TP):** Correctly predicted as positive.
    - **True Negatives (TN):** Correctly predicted as negative.
    - **False Positives (FP):** Incorrectly predicted as positive (Type I Error).
    - **False Negatives (FN):** Incorrectly predicted as negative (Type II Error).

4. Implementation

4.1. Tools and Libraries

The project was implemented in Python 3.8+ using the following core scientific computing and machine learning libraries:

- **NumPy:** For numerical operations.
- **Pandas:** For data manipulation and analysis.
- **Matplotlib & Seaborn:** For data visualization.
- **Scikit-learn:** For dataset loading, preprocessing, model training, and evaluation.

4.2. Workflow Overview

The implementation was separated into two main scripts:

1. **linear_regression.py:**
    - Load the California Housing dataset.
    - Define features (X) and target (y).
    - Split the data into 80% training and 20% testing sets.
    - Initialize and fit a LinearRegression model.
    - Make predictions on the test set.
    - Calculate and print MSE and R^2 scores.
    - Generate a scatter plot of Actual vs. Predicted values and save it to the outputs/ directory.
2. **logistic_regression.py:**
    - Load the dataset and create the binary target variable based on the median value.
    - Split the data into training and testing sets.
    - Initialize and fit a LogisticRegression model.
    - Make predictions on the test set.
    - Calculate and print the accuracy score.
    - Generate a confusion matrix heatmap and save it to the outputs/ directory.

5. Results and Discussion

This section presents and interprets the performance of the two models on the unseen test data.

5.1. Linear Regression Model Performance

The model was tasked with predicting the median house value.

- **Mean Squared Error (MSE): 0.54**
- **R-squared (R^2): 0.60**

Discussion: An **R^2 score of 0.60** indicates that 60% of the variability in California median house prices can be explained by the features in our model. While this is a reasonably good start, it also implies that 40% of the variance is due to factors not included in our model (e.g., local amenities, crime rates, property-specific details).

The **MSE of 0.54** is less intuitive on its own. To better understand it, we can calculate the Root Mean Squared Error (RMSE), which is MSE =0.54 ≈0.735. Since the target variable is in units of $100,000, this means our model's predictions are, on average, off by approximately **$73,500**. This error margin might be acceptable for a rough estimate but could be too large for high-stakes financial decisions.

The scatter plot below visualizes the model's performance.

- **Plot Analysis:** The plot shows a clear positive linear relationship. The points generally cluster around the 45-degree diagonal line, which represents a perfect prediction. However, there is significant spread, especially for higher-priced homes. The model appears to struggle with predicting the most expensive properties, which is likely related to the price cap at $500,000 in the dataset.

5.2. Logistic Regression Model Performance

The model was tasked with classifying houses as 'Cheap' (0) or 'Expensive' (1).

- **Accuracy: 0.82** (or 82%)

Discussion: An **accuracy of 82%** is a strong result. It means that the model correctly classifies whether a house's value is above or below the state median 82% of the time. This kind of binary classification is extremely useful for initial market screening, helping potential buyers or investors quickly identify properties within a certain price bracket.

The confusion matrix provides a more granular view of the model's performance.

- **Matrix Analysis:**
    - **True Negatives (Top-Left):** The model correctly identified a large number of 'Cheap' houses.
    - **True Positives (Bottom-Right):** The model also correctly identified a large number of 'Expensive' houses.
    - **False Positives (Top-Right):** This represents 'Cheap' houses that the model incorrectly classified as 'Expensive'. This error could lead a user to overlook affordable properties.
    - **False Negatives (Bottom-Left):** This represents 'Expensive' houses that the model incorrectly classified as 'Cheap'. This error could cause a user to waste time investigating properties that are actually outside their budget.

The relatively balanced number of correct predictions in both classes suggests the model is not biased towards one category, thanks to the median-based split.

6. Conclusion

This project successfully demonstrated the application of two fundamental machine learning models to the California Housing dataset for different real estate analysis tasks.

1. The **Linear Regression** model provided a foundational approach to price prediction, achieving an **R^2 of ~0.60**. It established that features like median income are significant predictors of house value. The model's average error of ~$73,500 provides a tangible measure of its predictive power.
2. The **Logistic Regression** model proved highly effective for classification, achieving an **accuracy of ~82%** in categorizing houses as 'Cheap' or 'Expensive'. This high accuracy shows that even a simple linear model can be powerful for binary decision-making tasks.

In summary, this project highlights the versatility of regression-based algorithms. While Linear Regression offers a direct price estimate, Logistic Regression provides a robust classification tool for market segmentation. Both models serve as excellent baselines and demonstrate the immense potential of data-driven methods in the real estate industry.

7. Future Work

While the current models provide valuable insights, their performance can be significantly improved. Future efforts could focus on the following areas:

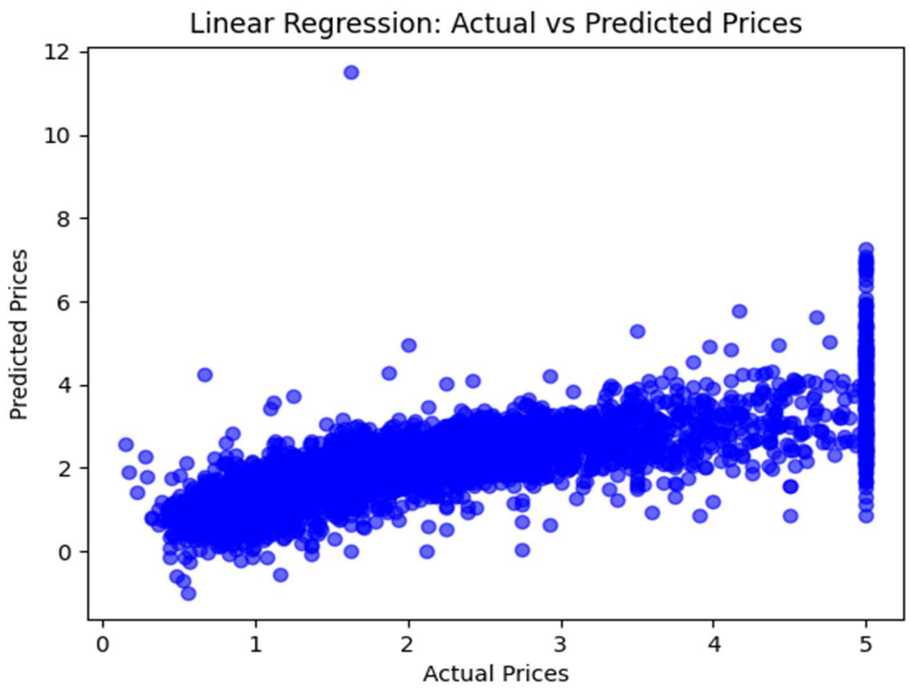
- **Advanced Models:** Implement more complex, non-linear models like **Random Forests**, **Gradient Boosting (e.g., XGBoost, LightGBM)**, or **Neural Networks**. These models can capture more intricate patterns and feature interactions, likely leading to a higher R^2 and lower MSE.
- **Feature Engineering:** Create new features from the existing ones. For example, a "rooms per person" feature (AveRooms / AveOccup) or polynomial features could help the models learn more complex relationships.
- **Incorporate More Data:** The greatest improvements often come from better data. Augmenting the dataset with additional features such as **local crime rates, proximity to amenities (parks, hospitals), school district quality, distance to the coast, and property tax information** would almost certainly enhance predictive accuracy.
- **Model Deployment:** Deploy the trained model as a web application or API. Using a framework like **Flask or Django**, one could create a user-friendly interface where users can input house features and receive an instant price prediction or classification, turning this academic project into a practical tool.

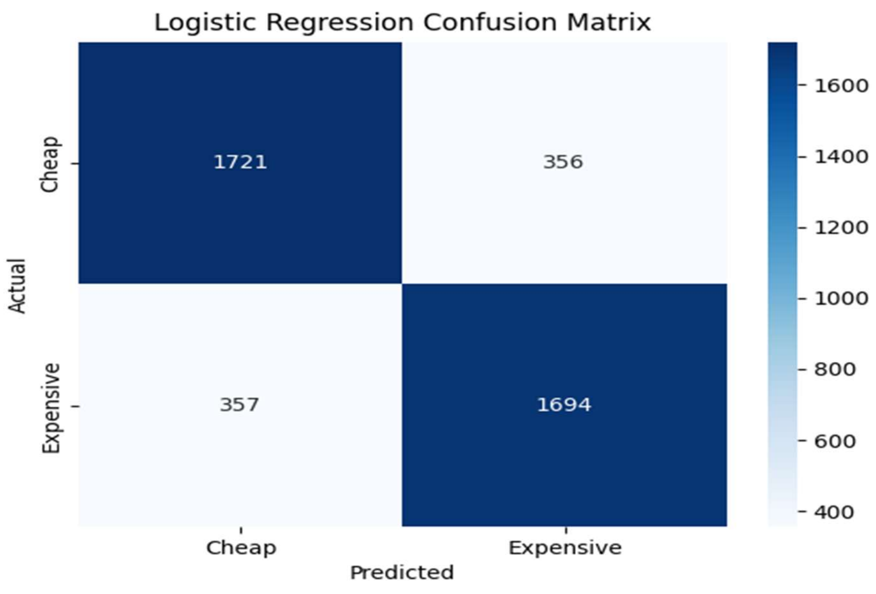
8. References

1. Pace, R. Kelley, and Ronald Barry. "Sparse Spatial Autoregressions." *Statistics & Probability Letters*, vol. 33, no. 3, 1997, pp. 291-297.
2. Scikit-learn Developers. "Scikit-learn: Machine Learning in Python." *scikit-learn.org*, 2025, https://scikit-learn.org.
3. Pedregosa, F., et al. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research*, vol. 12, 2011, pp. 2825-2830.
4. Harris, C.R., et al. "Array programming with NumPy." *Nature*, vol. 585, 2020, pp. 357-362.
5. McKinney, W. "Data Structures for Statistical Computing in Python." *Proceedings of the 9th Python in Science Conference*, 2010, pp. 56-61.

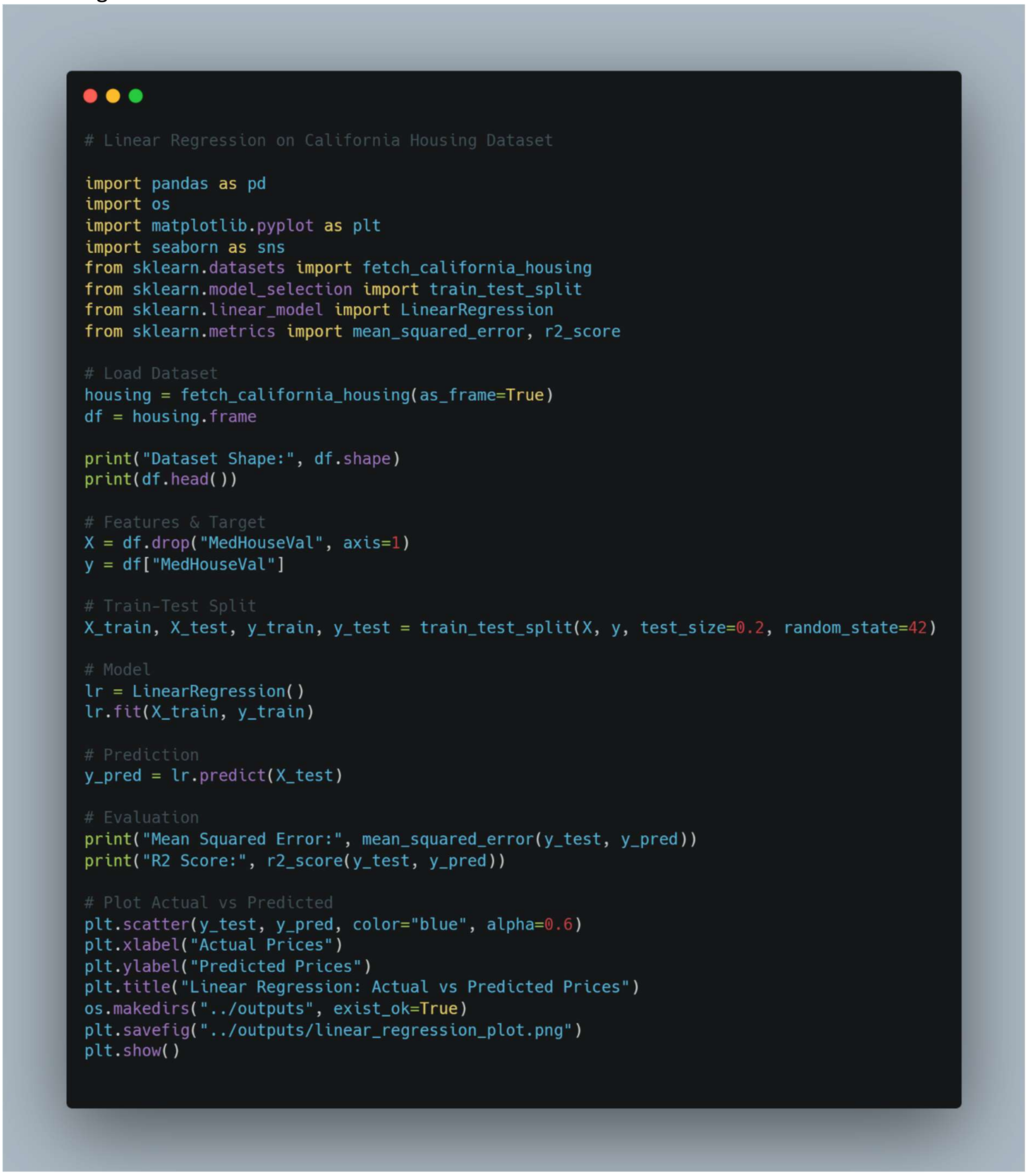
9. Appendix

Below is a screenshot of the terminal output showing the final evaluation metrics for both models.



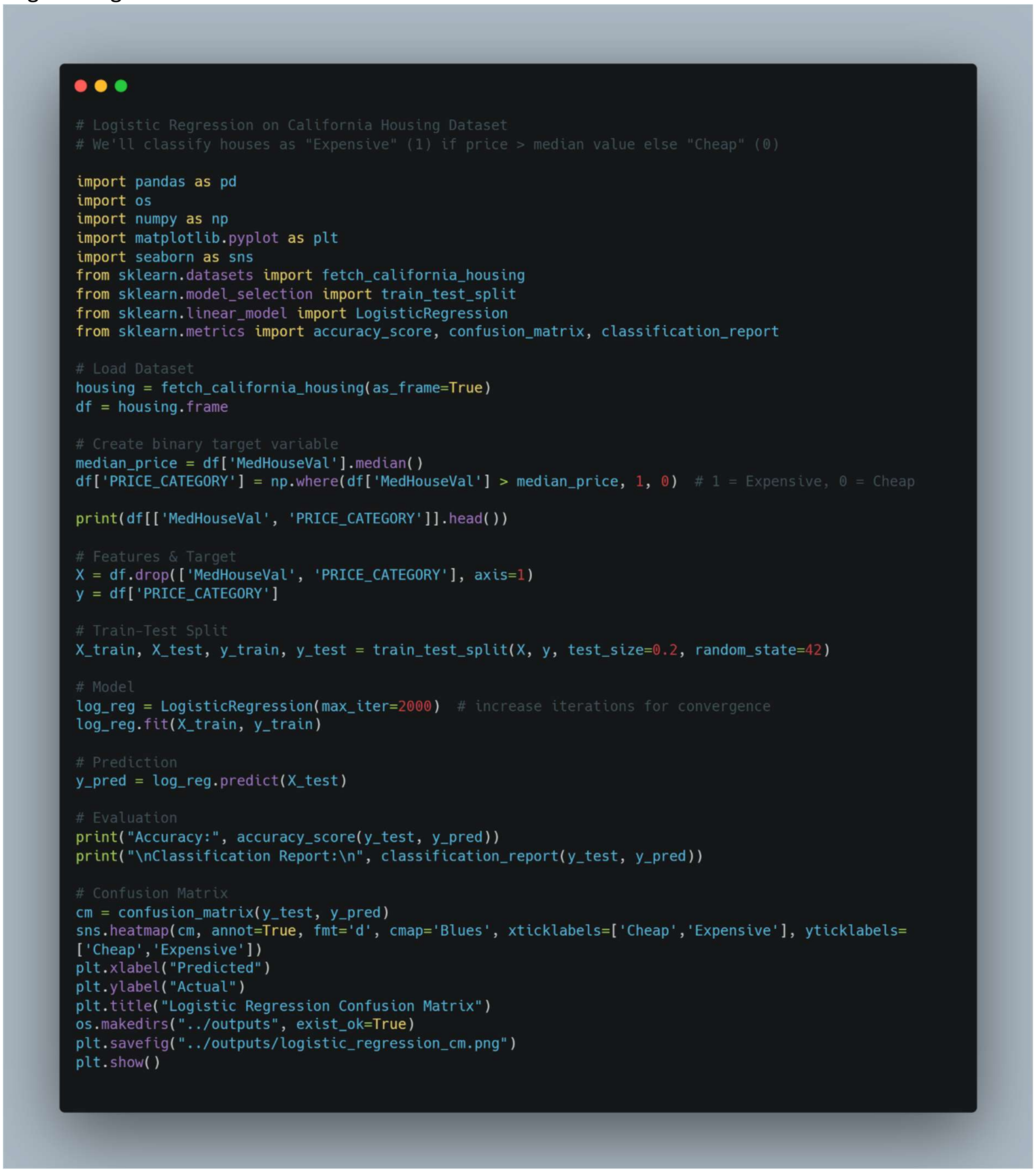


Linear Regression



```python
# Linear Regression on California Housing Dataset

import pandas as pd
import os
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.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load Dataset
housing = fetch_california_housing(as_frame=True)
df = housing.frame

print("Dataset Shape:", df.shape)
print(df.head())

# Features & Target
X = df.drop("MedHouseVal", axis=1)
y = df["MedHouseVal"]

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

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

# Prediction
y_pred = lr.predict(X_test)

# Evaluation
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("R2 Score:", r2_score(y_test, y_pred))

# Plot Actual vs Predicted
plt.scatter(y_test, y_pred, color="blue", alpha=0.6)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Linear Regression: Actual vs Predicted Prices")
os.makedirs("../outputs", exist_ok=True)
plt.savefig("../outputs/linear_regression_plot.png")
plt.show()
```

Logistic Regression



```python
# Logistic Regression on California Housing Dataset
# We'll classify houses as "Expensive" (1) if price > median value else "Cheap" (0)

import pandas as pd
import os
import numpy as np
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.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Load Dataset
housing = fetch_california_housing(as_frame=True)
df = housing.frame

# Create binary target variable
median_price = df['MedHouseVal'].median()
df['PRICE_CATEGORY'] = np.where(df['MedHouseVal'] > median_price, 1, 0)  # 1 = Expensive, 0 = Cheap

print(df[['MedHouseVal', 'PRICE_CATEGORY']].head())

# Features & Target
X = df.drop(['MedHouseVal', 'PRICE_CATEGORY'], axis=1)
y = df['PRICE_CATEGORY']

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model
log_reg = LogisticRegression(max_iter=2000)  # increase iterations for convergence
log_reg.fit(X_train, y_train)

# Prediction
y_pred = log_reg.predict(X_test)

# Evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Cheap','Expensive'], yticklabels=
['Cheap','Expensive'])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Logistic Regression Confusion Matrix")
os.makedirs("../outputs", exist_ok=True)
plt.savefig("../outputs/logistic_regression_cm.png")
plt.show()
```