|  |  |
| --- | --- |
|  | **DEPARTMENT OF COMPUTER ENGINEERING** |

**Experiment No. 01**

|  |  |
| --- | --- |
| Semester | B.E. Semester VII – Computer Engineering |
| Subject | Machine Learning |
| Subject Professor In-charge | Sanjeev Drivedi |
| Academic Year | 2024-25 |

|  |  |
| --- | --- |
| Student Name | Priyanka Kamble |
| Roll Number | 21102B0068 |

**Description:** Predicting Hosing Price Using Linear Regression

**Code:**

Data Preprocessing

1. Handle Missing Values

2. Features scaling and Normalisation

3. Encoding Categorical variables if necessary

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.metrics import accuracy\_score

import joblib

from sklearn.model\_selection import cross\_val\_score

df = pd.read\_csv('housing.csv')

df.fillna(df.median(numeric\_only=True), inplace=True)

data.head()

|  | **longitude** | **latitude** | **housing\_median\_age** | **total\_rooms** | **total\_bedrooms** | **population** | **households** | **median\_income** | **median\_house\_value** | **ocean\_proximity** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | -122.23 | 37.88 | 0.982163 | -0.803813 | -0.970325 | -0.973320 | -0.976833 | 2.345163 | 452600.0 | NEAR BAY |
| 1 | -122.22 | 37.86 | -0.606210 | 2.042130 | 1.348276 | 0.861339 | 1.670373 | 2.332632 | 358500.0 | NEAR BAY |
| 2 | -122.24 | 37.85 | 1.855769 | -0.535189 | -0.825561 | -0.819769 | -0.843427 | 1.782939 | 352100.0 | NEAR BAY |
| 3 | -122.25 | 37.85 | 1.855769 | -0.623510 | -0.718768 | -0.765056 | -0.733562 | 0.932970 | 341300.0 | NEAR BAY |
| 4 | -122.25 | 37.85 | 1.855769 | -0.461970 | -0.611974 | -0.758879 | -0.628930 | -0.013143 | 342200.0 | NEAR BAY |

data.shape

(20433, 10)

# Feature scaling and normalization

scaler = StandardScaler()

df\_scaled = pd.DataFrame(scaler.fit\_transform(df), columns=df.columns)

# Split the data into training and testing sets

X = df\_scaled.drop('median\_house\_value', axis=1)

y = df\_scaled['median\_house\_value']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

X.isnull().sum()

longitude 0

latitude 0

housing\_median\_age 0

total\_rooms 0

total\_bedrooms 0

population 0

households 0

median\_income 0

dtype: int64

X.fillna(X.mean(),inplace=True)

X.isnull().sum()

longitude 0

latitude 0

housing\_median\_age 0

total\_rooms 0

total\_bedrooms 0

population 0

households 0

median\_income 0

dtype: int64

# Feature scaling and normalization

scaler = StandardScaler()

df\_scaled = pd.DataFrame(scaler.fit\_transform(df), columns=df.columns)

scaler.fit(X)

StandardScaler

StandardScaler()

standardized\_data = scaler.transform(X)

print(standardized\_data)

[[-1.32783522 1.05254828 0.98214266 ... -0.9744286 -0.97703285

2.34476576]

[-1.32284391 1.04318455 -0.60701891 ... 0.86143887 1.66996103

2.33223796]

[-1.33282653 1.03850269 1.85618152 ... -0.82077735 -0.84363692

1.7826994 ]

...

[-0.8237132 1.77823747 -0.92485123 ... -0.3695372 -0.17404163

-1.14259331]

[-0.87362627 1.77823747 -0.84539315 ... -0.60442933 -0.39375258

-1.05458292]

[-0.83369581 1.75014627 -1.00430931 ... -0.03397701 0.07967221

-0.78012947]]

Step 2: Exploratory Data Analysis (EDA)

1. Visualize the distribution of the target variable.

2. Analyze the relationship between features and the target variables

3. identify the outliers

# Visualize the distribution of the target variable

plt.figure(figsize=(10, 6))

sns.histplot(df['median\_house\_value'], bins=50, kde=True)

plt.xlabel('Median House Value')

plt.ylabel('Frequency')

plt.title('Distribution of Median House Value')

plt.show()

A diagram of a house value

Description automatically generated

for feature in X.columns:

plt.figure(figsize=(8, 6)) # Adjust figure size as needed

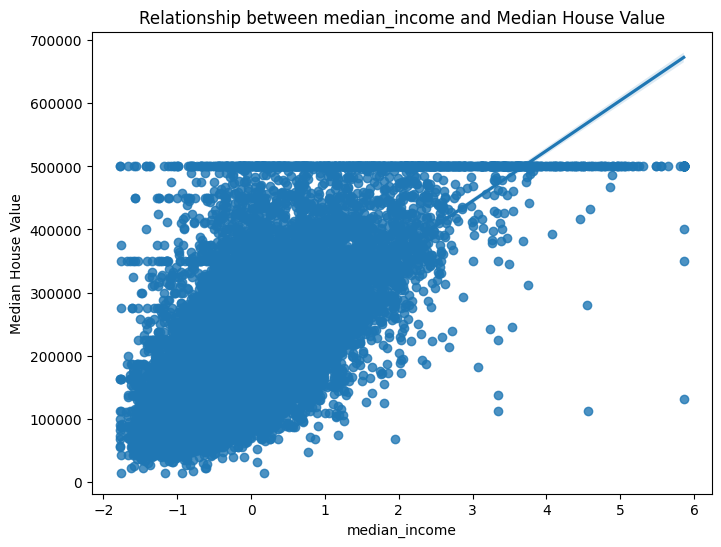
sns.regplot(x=feature, y='median\_house\_value', data=data)

plt.title(f"Relationship between {feature} and Median House Value")

plt.xlabel(feature)

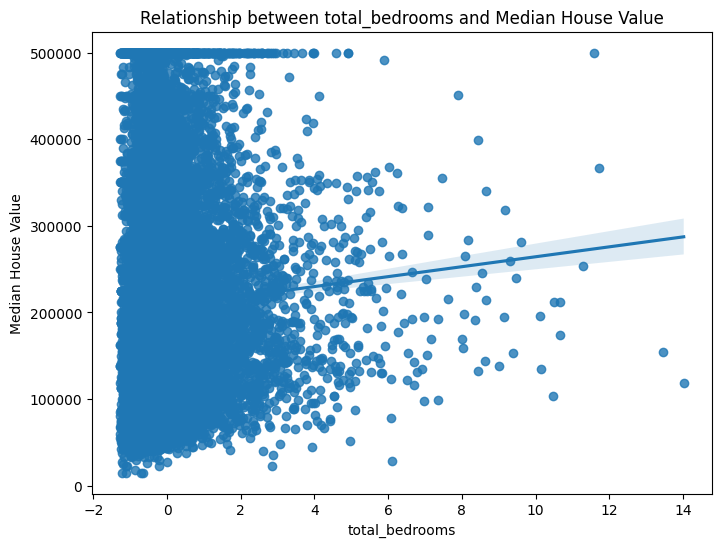
plt.ylabel("Median House Value")

plt.show()

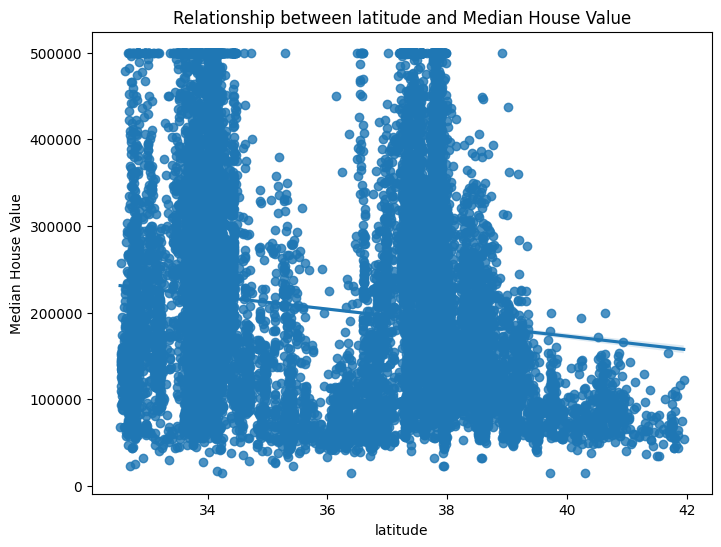
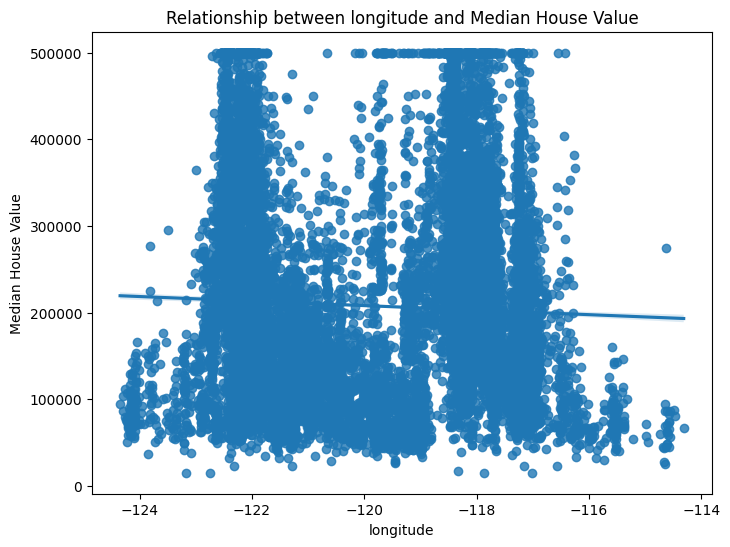


A graph of blue dots

Description automatically generated A graph with blue dots

Description automatically generated  A graph of blue dots

Description automatically generated A graph of blue dots

Description automatically generated   

# Analyze the relationship between features and the target variable

# Scatterplot for numeric features

features = ['median\_income', 'housing\_median\_age', 'total\_rooms', 'total\_bedrooms', 'population', 'households']

plt.figure(figsize=(20, 15))

for i, feature in enumerate(features, 1):

plt.subplot(3, 2, i)

plt.scatter(df[feature], df['median\_house\_value'])

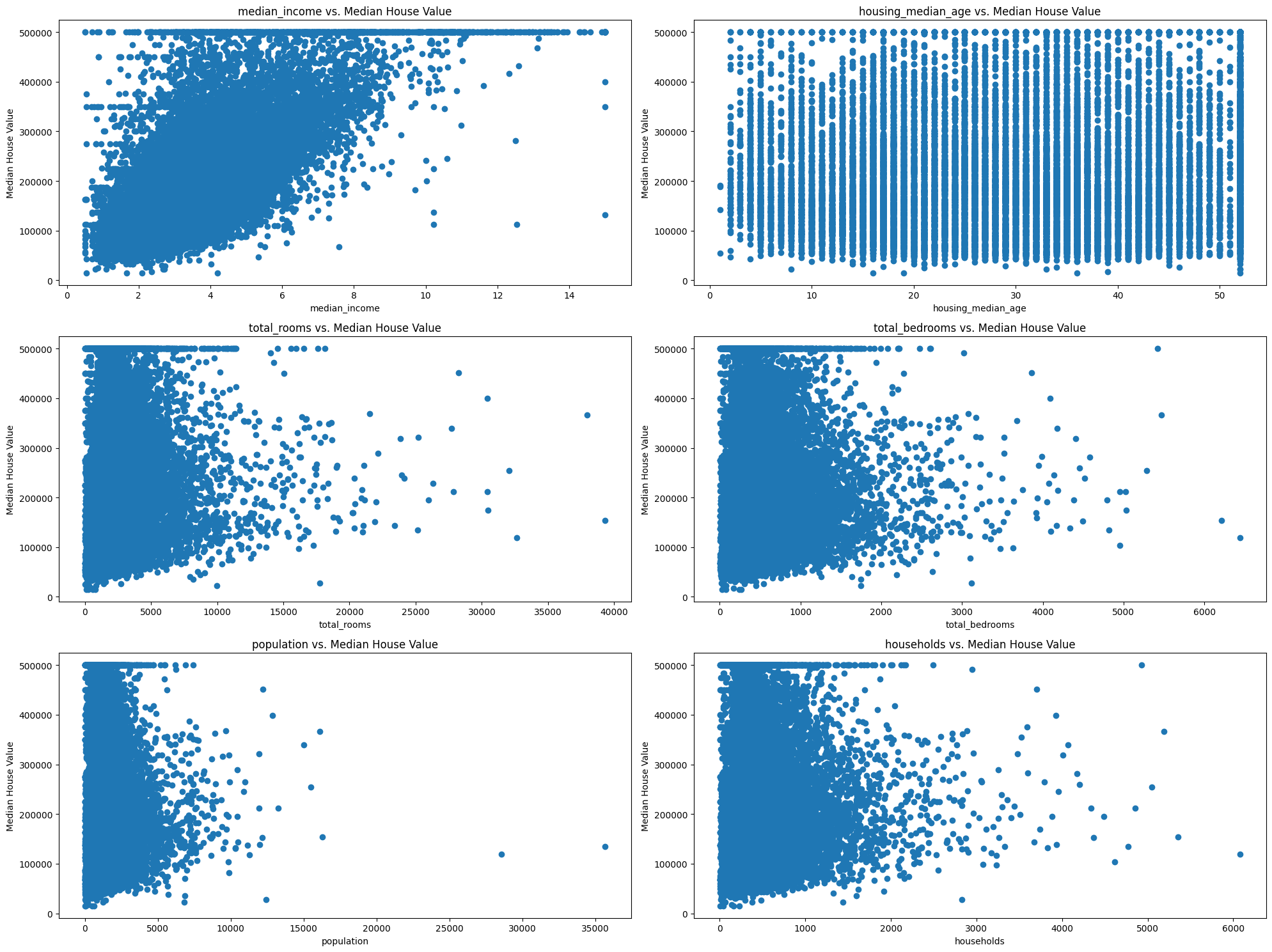
plt.xlabel(feature)

plt.ylabel('Median House Value')

plt.title(f'{feature} vs. Median House Value')

plt.tight\_layout()

plt.show()



# Identify the outliers using boxplots

plt.figure(figsize=(20, 15))

for i, feature in enumerate(features, 1):

plt.subplot(3, 2, i)

sns.boxplot(data=df, x=feature)

plt.title(f'Boxplot of {feature}')

plt.tight\_layout()

plt.show()

A group of blue and black squares

Description automatically generated with medium confidence

# Using IQR to identify outliers

def identify\_outliers(df, feature):

Q1 = df[feature].quantile(0.25)

Q3 = df[feature].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

outliers = df[(df[feature] < lower\_bound) | (df[feature] > upper\_bound)]

return outliers

for feature in features:

outliers = identify\_outliers(df, feature)

print(f'{feature} has {len(outliers)} outliers')

median\_income has 681 outliers

housing\_median\_age has 0 outliers

total\_rooms has 1287 outliers

total\_bedrooms has 1306 outliers

population has 1196 outliers

households has 1220 outliers

plt.figure(figsize=(10, 6))

sns.boxplot(x=df['median\_house\_value'])

plt.xlabel('Median House Value')

plt.title('Boxplot of Median House Value')

plt.show()

A blue rectangular bar graph

Description automatically generated

# Bar chart

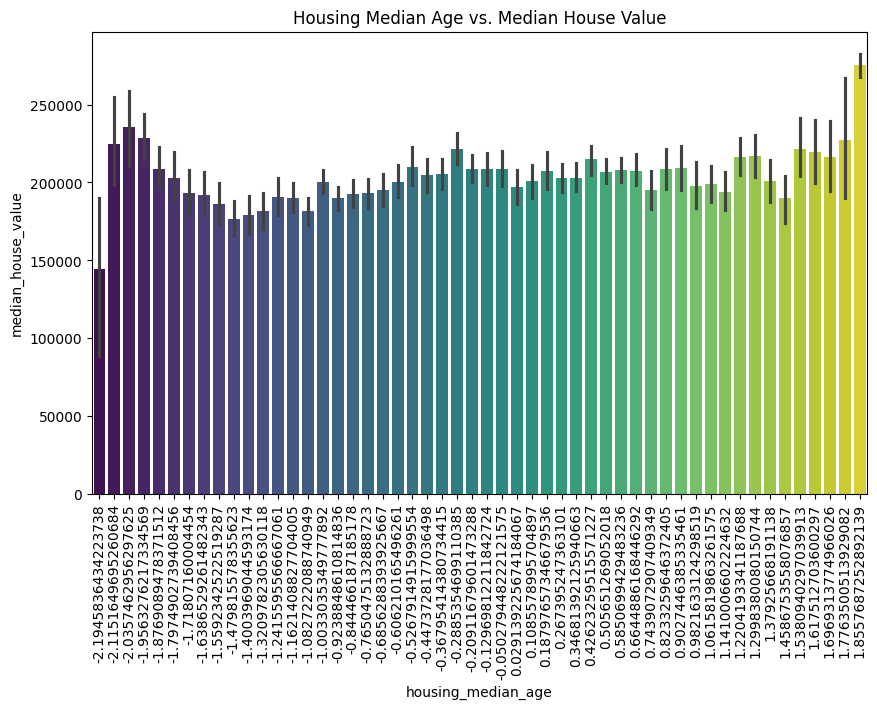
plt.figure(figsize=(10, 6))

sns.barplot(x='housing\_median\_age', y='median\_house\_value', data=data, palette='viridis')

plt.title('Housing Median Age vs. Median House Value')

plt.xticks(rotation=90)

plt.show()



Model Development:

1. Split the data

2. Train a linear regression model on the training data.

3. Evaluate the model using metrics such as MAE,MSE,R-squared

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 2)

print( X\_train.shape, X\_test.shape,Y\_train.shape, Y\_test.shape)

(16512, 8) (4128, 8) (16512,) (4128,)

from sklearn import linear\_model

from sklearn.metrics import mean\_squared\_error, r2\_score, mean\_absolute\_error

model = linear\_model.LinearRegression()

LinearRegression

LinearRegression()

Y\_pred = model.predict(X\_test)

print('Coeffecients : ',model.coef\_)

print('Intercept : ',model.intercept\_)

print('Mean Squared error: %.2f' % mean\_squared\_error(Y\_test, Y\_pred))

print('R2 score: %.2f' % r2\_score(Y\_test, Y\_pred))

print('Mean Absolute error: %.2f' % mean\_absolute\_error(Y\_test, Y\_pred))

Coeffecients : [-0.73332474 -0.7812548 0.12463204 -0.12895359 0.28342985 -0.38461431

0.27535944 0.65608161]

Intercept : 0.0011979959586306248

Mean Squared error: 0.37

R2 score: 0.63

Mean Absolute error: 0.45

Model Evaluation and Tuning:

1. Perform cross-validation to ensure the model's robustness.

2. Tune the hyperparameters of the model to improve its performance.

from sklearn.model\_selection import cross\_val\_score

# Perform 5-fold cross-validation

scores = cross\_val\_score(model, X, y, cv=5, scoring='r2')

# Print the scores for each fold

print("Cross-validation scores:", scores)

# Print the average score

print("Average R2 score:", scores.mean())

Cross-validation scores: [0.58426664 0.52339781 0.58372153 0.54302958 0.69033068]

Average R2 score: 0.5849492476538065

input\_data =[-119.84,36.77,6,1853,473,1397,417,1.4817]

input\_data\_as\_numpy\_array = np.asarray(input\_data)

input\_data\_reshaped = input\_data\_as\_numpy\_array.reshape(1,-1)

prediction = model.predict(input\_data\_reshaped)

print(prediction[0])

-466.49398062766835

input\_data =[-122.23, 37.88,41,880,129,322,126,8.3252]

input\_data\_as\_numpy\_array = np.asarray(input\_data)

input\_data\_reshaped = input\_data\_as\_numpy\_array.reshape(1,-1)

prediction = model.predict(input\_data\_reshaped)

print(prediction[0])

-95.45375396661967

input\_data =[-121.83, 38,8,2572,738,1384,684,1.7161]

input\_data\_as\_numpy\_array = np.asarray(input\_data)

input\_data\_reshaped = input\_data\_as\_numpy\_array.reshape(1,-1)

prediction = model.predict(input\_data\_reshaped)

print(prediction[0])

-404.6803240993742