Name: Vikas Chaurasiya

Rollno.: 21102B0024

Class : BE CMPN B

Experiment 1 : Predict Housing Prices using Linear Regression

Github : https://github.com/vikas-a-chaurasiya/ML\_exp

Dataset: <https://www.kaggle.com/datasets/camnugent/california-housing-prices>

|  |
| --- |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  import seaborn as sns  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.preprocessing import StandardScaler  # Load the dataset  df = pd.read\_csv('housing.csv')  print(df.head()) |

longitude latitude housing\_median\_age total\_rooms total\_bedrooms \

0 -122.23 37.88 41.0 880.0 129.0

1 -122.22 37.86 21.0 7099.0 1106.0

2 -122.24 37.85 52.0 1467.0 190.0

3 -122.25 37.85 52.0 1274.0 235.0

4 -122.25 37.85 52.0 1627.0 280.0

population households median\_income median\_house\_value ocean\_proximity

0 322.0 126.0 8.3252 452600.0 NEAR BAY

1 2401.0 1138.0 8.3014 358500.0 NEAR BAY

2 496.0 177.0 7.2574 352100.0 NEAR BAY

3 558.0 219.0 5.6431 341300.0 NEAR BAY

print(df.isnull().sum())

df['total\_bedrooms'].fillna(df['total\_bedrooms'].median(), inplace=True)

print(df.isnull().sum())

longitude 0

latitude 0

housing\_median\_age 0

total\_rooms 0

total\_bedrooms 207

population 0

households 0

median\_income 0

median\_house\_value 0

ocean\_proximity 0

dtype: int64

longitude 0

latitude 0

housing\_median\_age 0

total\_rooms 0

total\_bedrooms 0

population 0

households 0

median\_income 0

median\_house\_value 0

ocean\_proximity 0

dtype: int64

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

scaler = StandardScaler()

df[features] = scaler.fit\_transform(df[features])

# Display the first few rows of the scaled dataset

print(df.head())

longitude latitude housing\_median\_age total\_rooms total\_bedrooms \

0 -1.327835 1.052548 0.982143 -0.804819 -0.972476

1 -1.322844 1.043185 -0.607019 2.045890 1.357143

2 -1.332827 1.038503 1.856182 -0.535746 -0.827024

3 -1.337818 1.038503 1.856182 -0.624215 -0.719723

4 -1.337818 1.038503 1.856182 -0.462404 -0.612423

population households median\_income median\_house\_value ocean\_proximity

0 -0.974429 -0.977033 2.344766 452600.0 NEAR BAY

1 0.861439 1.669961 2.332238 358500.0 NEAR BAY

2 -0.820777 -0.843637 1.782699 352100.0 NEAR BAY

3 -0.766028 -0.733781 0.932968 341300.0 NEAR BAY

4 -0.759847 -0.629157 -0.012881 342200.0 NEAR BAY

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

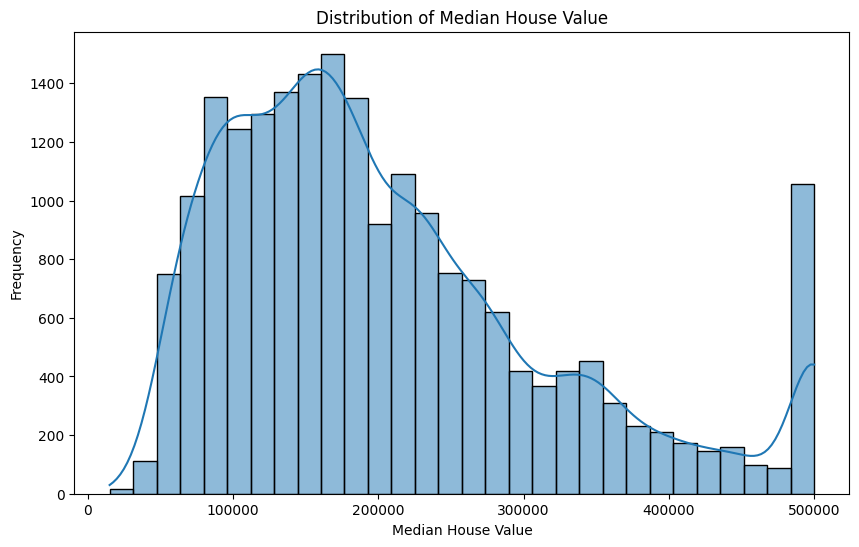
sns.histplot(df['median\_house\_value'], bins=30, kde=True)

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

plt.xlabel('Median House Value')

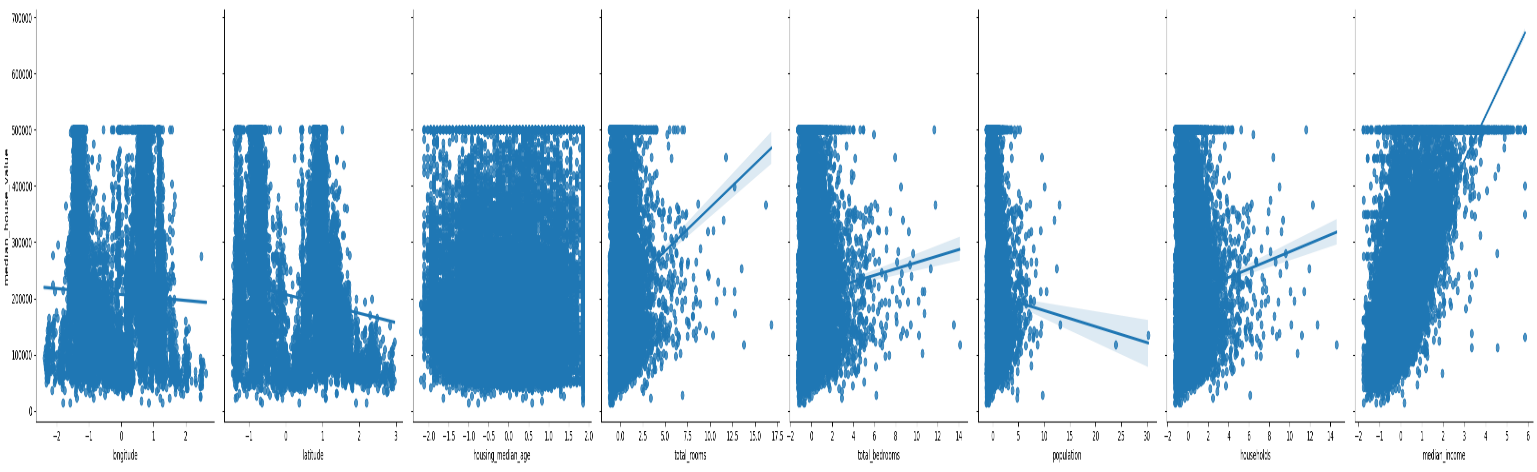
plt.ylabel('Frequency')

plt.show()



sns.pairplot(df, x\_vars=features, y\_vars='median\_house\_value', kind='reg', height=5)

plt.show()



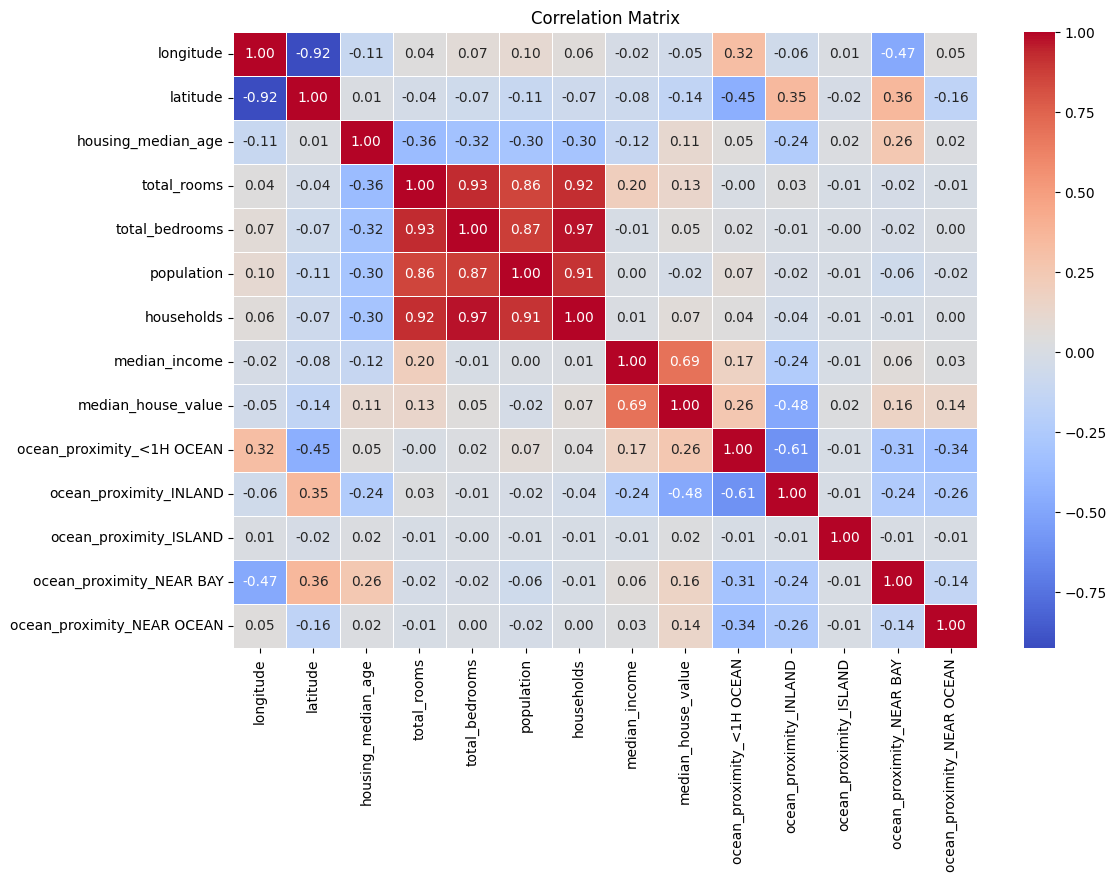
corr\_matrix = df.corr()

plt.figure(figsize=(12, 8))

sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)

plt.title('Correlation Matrix')

plt.show()



# Boxplot to identify outliers

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

sns.boxplot(data=df, x='median\_house\_value')

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

plt.show()

# Option to handle outliers: Remove outliers using the IQR method

Q1 = df['median\_house\_value'].quantile(0.25)

Q3 = df['median\_house\_value'].quantile(0.75)

IQR = Q3 - Q1

df = df[~((df['median\_house\_value'] < (Q1 - 1.5 \* IQR)) | (df['median\_house\_value'] > (Q3 + 1.5 \* IQR)))]

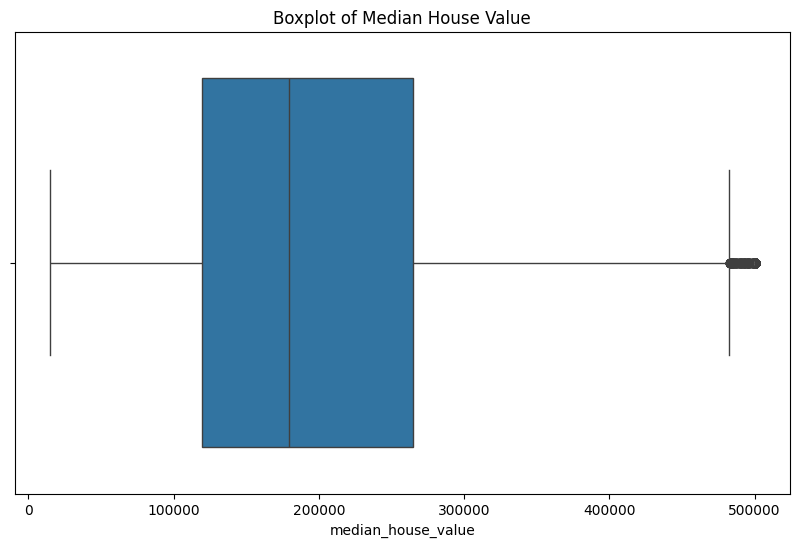
# Verify the removal of outliers

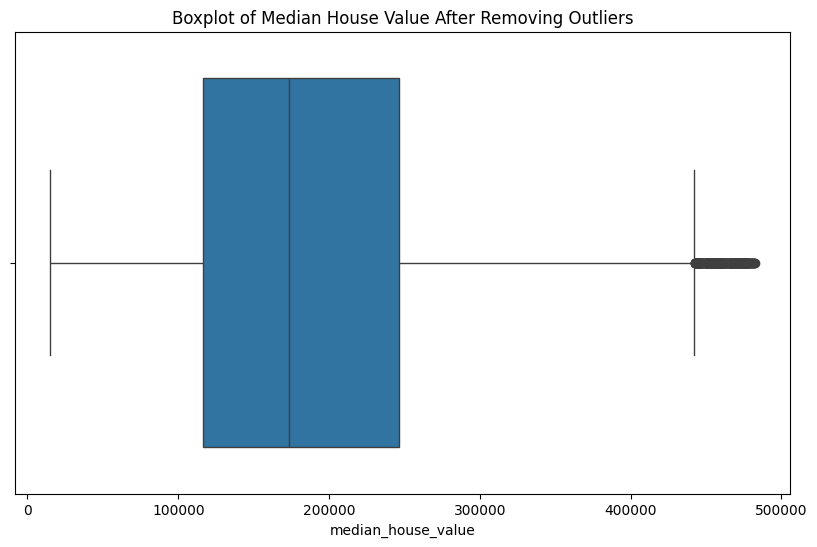
plt.figure(figsize=(10, 6))

sns.boxplot(data=df, x='median\_house\_value')

plt.title('Boxplot of Median House Value After Removing Outliers')

plt.show()





# Initialize the Linear Regression model

lr = LinearRegression()

# Fit the model to the training data

lr.fit(X\_train, y\_train)

# Predict on the test data

y\_pred = lr.predict(X\_test)

# Calculate the Mean Squared Error and R-squared value

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f'Mean Squared Error: {mse}')

print(f'R-squared Value: {r2}')

# Visualize the predicted vs actual values

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

plt.scatter(y\_test, y\_pred, alpha=0.5)

plt.xlabel('Actual Median House Value')

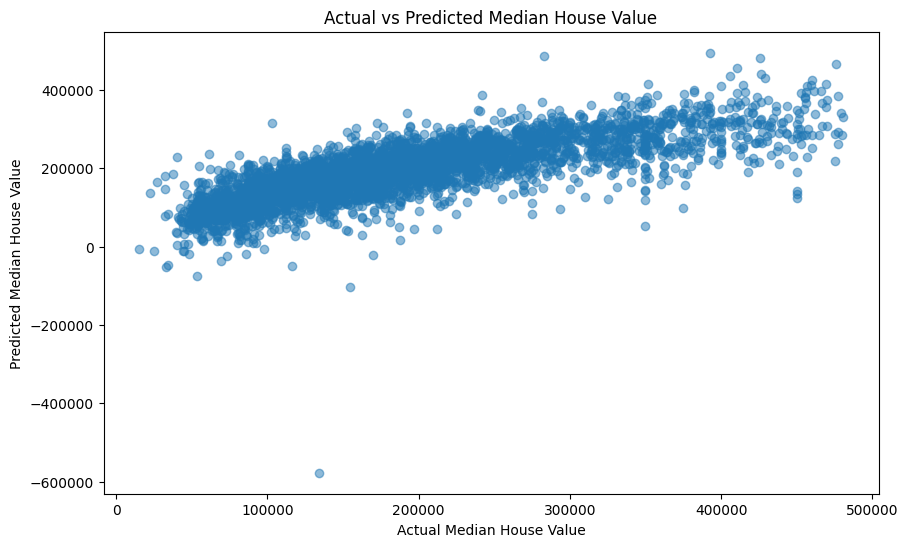
plt.ylabel('Predicted Median House Value')

plt.title('Actual vs Predicted Median House Value')

plt.show()

Mean Squared Error: 3587617293.3291674

R-squared Value: 0.6120740998708538



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

# Perform 5-fold cross-validation

cv\_scores = cross\_val\_score(lr, X, y, cv=5, scoring='neg\_mean\_squared\_error')

cv\_mse = -cv\_scores

cv\_mse\_mean = cv\_mse.mean()

print(f'Cross-Validated Mean Squared Error: {cv\_mse\_mean}')

Cross-Validated Mean Squared Error: 3904461854.3761964

ridge = Ridge()

# Define the hyperparameter grid

param\_grid = {'alpha': [0.1, 1, 10, 100, 1000]}

# Initialize the GridSearchCV object

grid\_search = GridSearchCV(ridge, param\_grid, cv=5, scoring='neg\_mean\_squared\_error')

grid\_search.fit(X\_train, y\_train)

# Get the best hyperparameters

best\_params = grid\_search.best\_params\_

ridge\_best = Ridge(alpha=best\_params['alpha'])

ridge\_best.fit(X\_train, y\_train)

y\_pred\_ridge = ridge\_best.predict(X\_test)

mae\_ridge = mean\_absolute\_error(y\_test, y\_pred\_ridge)

mse\_ridge = mean\_squared\_error(y\_test, y\_pred\_ridge)

r2\_ridge = r2\_score(y\_test, y\_pred\_ridge)

print(f'Best Ridge Alpha: {best\_params["alpha"]}')

print(f'Mean Absolute Error (Ridge): {mae\_ridge}')

print(f'Mean Squared Error (Ridge): {mse\_ridge}')

print(f'R-squared Value (Ridge): {r2\_ridge}')

Mean Absolute Error: 44610.13010189545

Mean Squared Error: 3587617293.3291674

R-squared Value: 0.6120740998708538

Cross-Validated Mean Squared Error: 3904461854.3761964

Best Ridge Alpha: 1

Mean Absolute Error (Ridge): 44611.1815161389

Mean Squared Error (Ridge): 3587476327.8117924

R-squared Value (Ridge): 0.6120893423481706