CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY
DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH
Department of Computer Engineering

Subject Name: Machine Learning
Semester: V
Subject Code: CE362
Academic year: 2025-26 [ODD]

Internal Practical Examination

No.	Aim of the Practical
1.	The aim of this practical is to develop a predictive model for estimating Boston housing prices using historical neighborhood data. By applying preprocessing, feature scaling, Random Forest regression, and visual analysis, the project seeks to identify key factors influencing prices and provide accurate predictions for policy-making.

CODE:

```
import kagglehub
from kagglehub import KaggleDatasetAdapter
# Set the path to the file you'd like to load
file_path = "housing.csv"
# Load the latest version
df = kagglehub.load_dataset(
  KaggleDatasetAdapter.PANDAS,
  "schirmerchad/bostonhoustingmlnd",
  file_path,
  # Provide any additional arguments like
  # sql_query or pandas_kwargs. See the
  # documenation for more information:
  #
https://github.com/Kaggle/kagglehub/blob/main/README.md#kaggledatasetadapterpan
das
)
```

```
print("First 5 records:", df.head())
```

OUTPUT:

```
/tmp/ipython-input-547231928.py:9: DeprecationWarning: Use dataset_load() instead of load_dataset(). load_dataset() will be removed in a future version.
  df = kagglehub.load_dataset(
Using Colab cache for faster access to the 'bostonhoustingmlnd' dataset.
First 5 records:       RM  LSTAT  PTRATIO      MEDV
0  6.575   4.98     15.3  504000.0
1  6.421   9.14     17.8  453600.0
2  7.185   4.03     17.8  728700.0
3  6.998   2.94     18.7  701400.0
4  7.147   5.33     18.7  760200.0
```

CODE:

```
print("Missing values before handling:")
print(df.isnull().sum())
```

OUTPUT:

```
Missing values before handling:
RM         0
LSTAT      0
PTRATIO    0
MEDV       0
dtype: int64
```

CODE:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
X = df.drop('MEDV', axis=1)
y = df['MEDV']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print("\nShape of X_train:", X_train.shape)
print("Shape of X_test:", X_test.shape)
print("Shape of y_train:", y_train.shape)
print("Shape of y_test:", y_test.shape)
```

OUTPUT:

```

Shape of X_train: (391, 3)
Shape of X_test: (98, 3)
Shape of y_train: (391,)
Shape of y_test: (98,)
```

CODE:

```
scaler = StandardScaler()
```

```
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
print("\nFirst 5 rows of scaled X_train:")
display(X_train_scaled[:5])
print("\nFirst 5 rows of scaled X_test:")
display(X_test_scaled[:5])
```

OUTPUT:

```
First 5 rows of scaled X_train:
array([[-0.57729555, -0.42657057,  0.80699513],
       [-0.12427302,  1.59067125,  1.26588852],
       [ 0.72978585,  0.64666951,  0.80699513],
       [ 0.26488076,  0.47528824,  0.80699513],
       [ 0.79662524, -0.89295239,  1.17410984]])

First 5 rows of scaled X_test:
array([[-0.49263232,  0.74359826,  0.80699513],
       [ 0.19507073, -0.45185633,  0.02687636],
       [ 0.59016579,  0.68740768,  0.80699513],
       [-1.25311605,  1.56398073,  0.76110579],
       [ 0.29904311,  1.56538549,  0.80699513]])
```

CODE:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train_scaled, y_train)
y_pred = rf_model.predict(X_test_scaled)
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error (MSE): {mse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"R-squared (R2): {r2}")
```

OUTPUT:

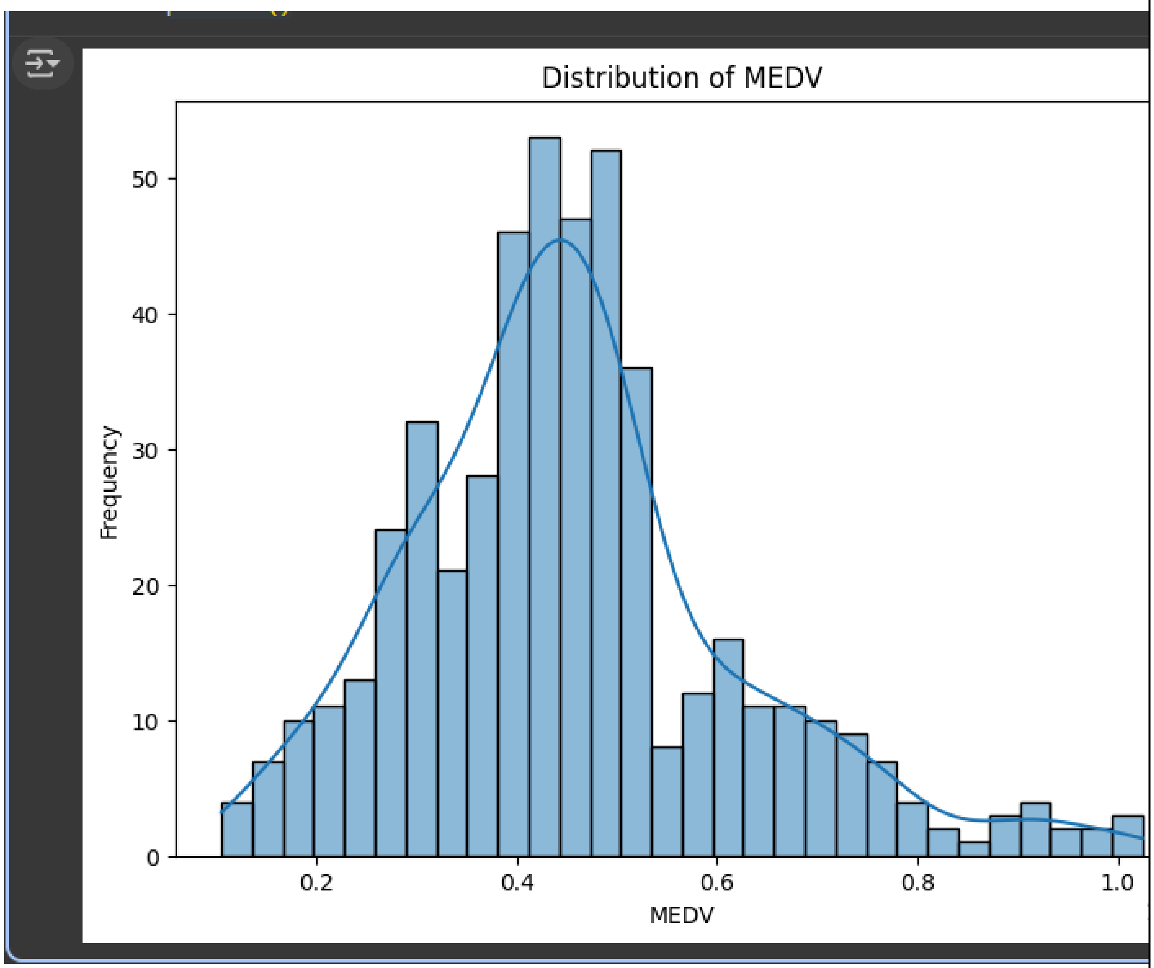
```
Mean Squared Error (MSE): 3241181641.5
Mean Absolute Error (MAE): 44046.642857142855
R-squared (R2): 0.8525234010221985
```

CODE:

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8, 6))
sns.histplot(df['MEDV'], bins=30, kde=True)
plt.title('Distribution of MEDV')
plt.xlabel('MEDV')
plt.ylabel('Frequency')
plt.show()
```

OUTPUT:

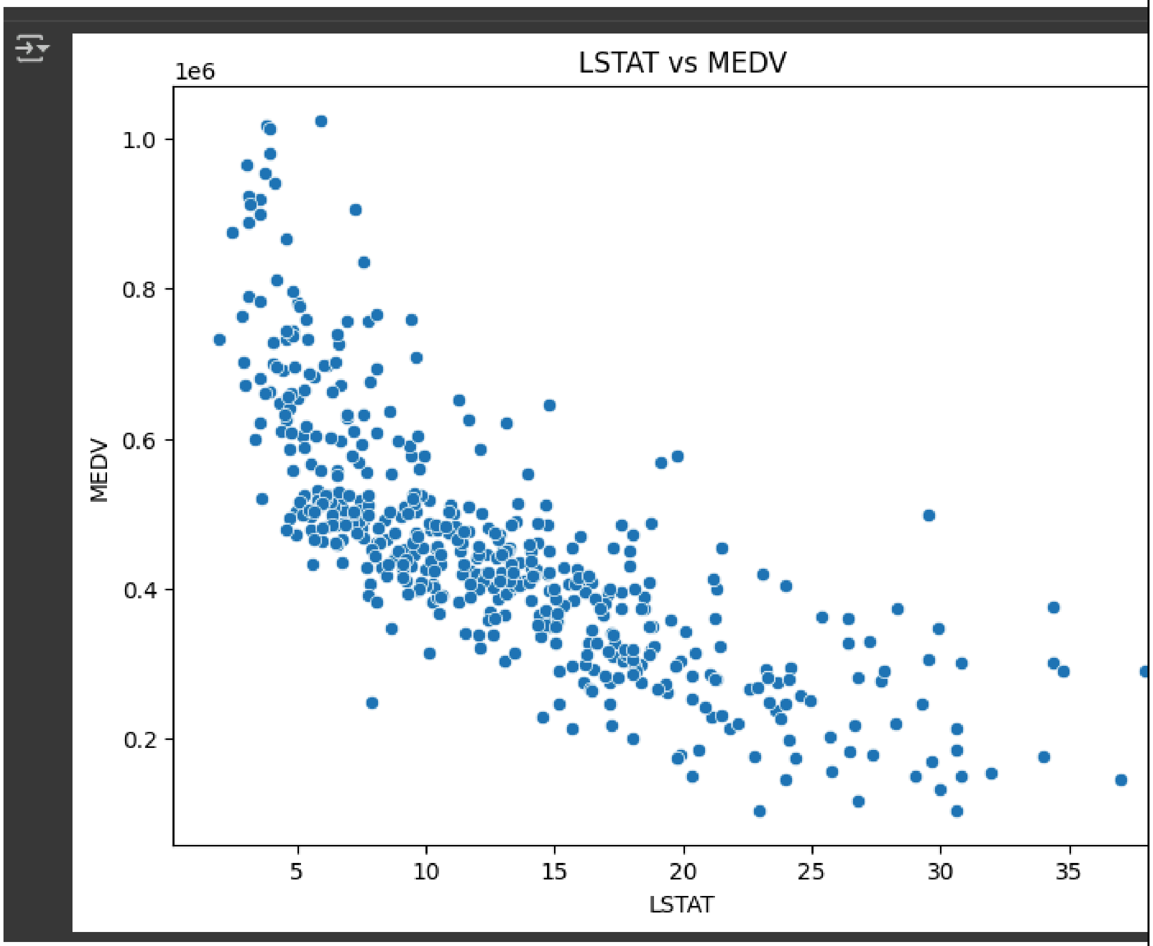


CODE:

```
plt.figure(figsize=(8, 6))
sns.scatterplot(x='LSTAT', y='MEDV', data=df)
plt.title('LSTAT vs MEDV')
plt.xlabel('LSTAT')
```

```
plt.ylabel('MEDV')
plt.show()
```

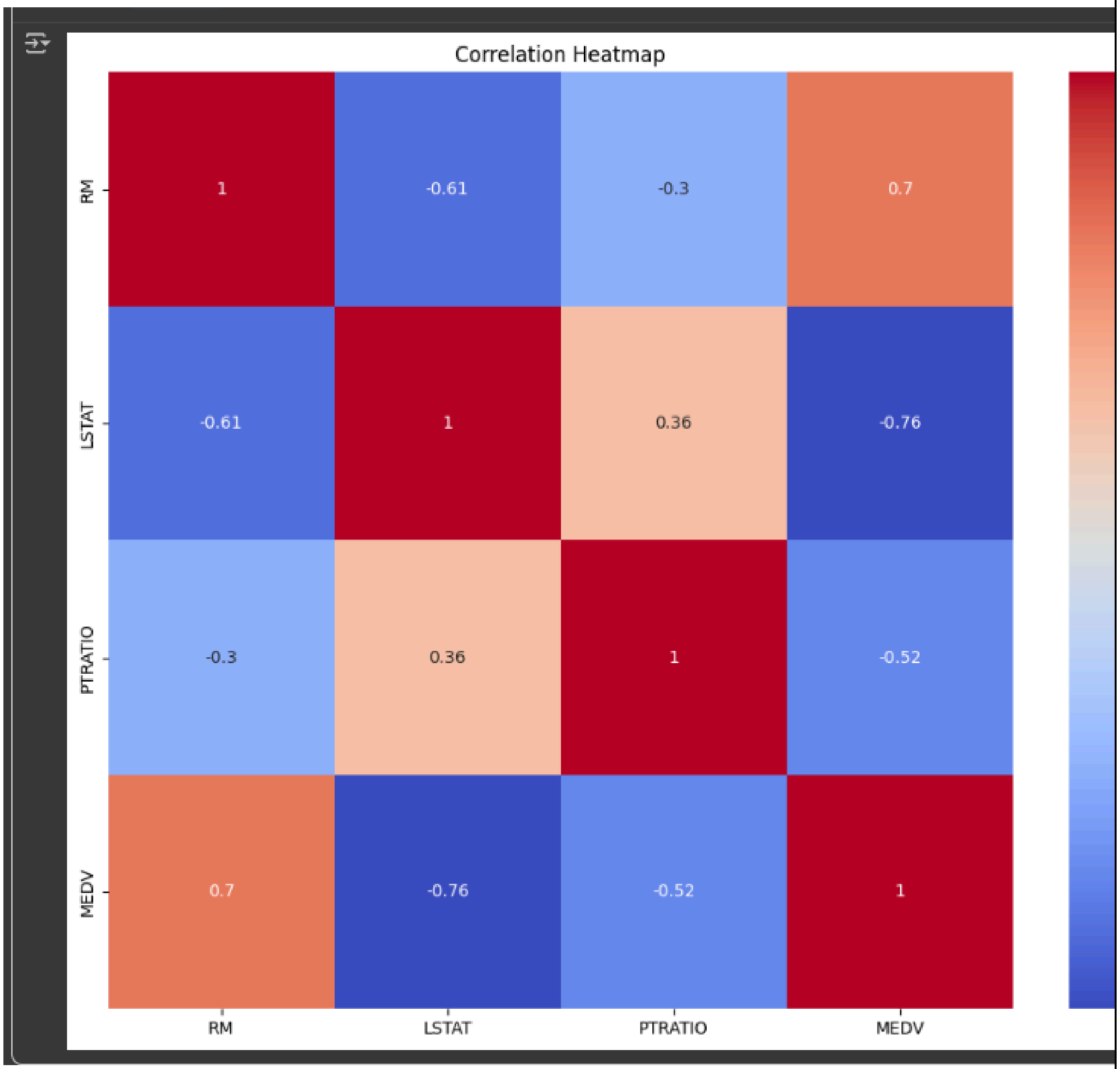
OUTPUT:



CODE:

```
plt.figure(figsize=(12, 10))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

OUTPUT:



Here are two factors that influence housing prices(MEDV):
RM (Average number of rooms): There is a strong positive correlation between the average number of rooms (RM) and the median value of homes (MEDV). This suggests that houses with more rooms tend to have higher prices.

LSTAT (Percentage of lower status population): There is a strong negative correlation between the percentage of the lower status population (LSTAT) and the median value of homes (MEDV). This indicates that neighborhoods with a higher percentage of lower status population tend to have lower housing prices.

CODE:

```
import numpy as np

# 'high rooms' is 7.5, 'low LSTAT' is 5, and 'low PTRATIO' is 15
test_case_1 = np.array([[7.5, 5.0, 15.0]])
#'low rooms' is 5.5, 'high LSTAT' is 25, and 'high PTRATIO' is 20
test_case_2 = np.array([[5.5, 25.0, 20.0]])

test_case_1_scaled = scaler.transform(test_case_1)
test_case_2_scaled = scaler.transform(test_case_2)

predicted_price_1 = rf_model.predict(test_case_1_scaled)
predicted_price_2 = rf_model.predict(test_case_2_scaled)

print(f"Predicted house price for Case 1: ${predicted_price_1[0]:,.2f}")
print(f"Predicted house price for Case 2: ${predicted_price_2[0]:,.2f}")
```

OUTPUT:

```
Predicted house price for Case 1: $913,605.00
Predicted house price for Case 2: $247,674.00
```