2025-2026

Welcome to the 3rd Lab session of CO3519 module.

For Two Hours Lab Session:

#	Task	Time (m)	Guidance
1	Please implement/run the code to solve house price prediction using Linear Regression as discussed in the lecture this morning.	50	Please follow the instruction on Page 2 to Page 4, for **Step 1** given there.
	Break	10	
2	Please follow the second step given on Page 5.	50	Download the dataset and update the code for it. Make necessary changes.

For One Hour Lab Session:

#	Task	Time (m)	Guidance
1	Please perform the first task from the two hours sessions.	50	Please follow the instruction on Page 2 to Page 5, for Step 1 given there.

House Price Prediction

As solved in the lecture, LR provides a simple yet effective baseline model for house price prediction. Model performance is evaluated using metrics like MSE and RMSE, complemented by visualizations to analyze the fit. Though straightforward, the model's effectiveness depends on the feature set and data quality.

Description before running into the code.

The California Housing dataset is loaded and converted into a Pandas DataFrame for manipulation. Features (independent variables) include housing characteristics like average income, housing age, and more. The target variable (dependent variable) is PRICE, representing median house prices. The dataset is divided into training (80%) and testing (20%) sets using train_test_split, ensuring a balanced evaluation. A Linear Regression model is initialized and trained on the training data (X_train and y_train). The model calculates coefficients for each feature, indicating their influence on house prices, and an intercept value for the baseline prediction. The trained model predicts house prices for the test dataset (X_test), generating y_pred.

Mean Squared Error (MSE) and **Root Mean Squared Error (RMSE)** are used to assess model performance. Lower values indicate better accuracy.

Scatter Plots: Show the relationship between actual (y_test) and predicted (y_pred) prices.

Line Plots: Highlight trends in actual and predicted prices across sorted indices.

Result Analysis: A summary DataFrame compares actual and predicted prices for deeper insights. Sorting and plotting help identify patterns and discrepancies.

1. Please run the following code for house price prediction:

First, import necessary libraries where numpy for numerical operations. pandas for data manipulation and analysis. train_test_split to split data into training and testing sets. LinearRegression for creating and training a linear regression model. mean_squared_error for model evaluation. matplotlib.pyplot for data visualization. fetch_california_housing to load a sample dataset of California housing prices.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
```

Next, import and load the dataset and show the first few samples of the data. Fetch the California housing dataset (fetch_california_housing). Convert it to a Pandas DataFrame for easier handling. Add

the target variable (PRICE) to the DataFrame. data.head() displays the first few rows for a quick preview of the data.

```
from sklearn.datasets import fetch_california_housing
```

Loads the California housing dataset into housing.
housing.data shows feature values (like number of rooms, population, income, etc.).
housing.feature_names shows names of these features.
housing.target shows target values (house prices).
data shows a pandas DataFrame with features and house price.
Adds a new column 'PRICE' which contains the target variable (housing.target).
data.head() shows the first 5 rows.
Example columns:
MedInc → median income in the area
HouseAge → average age of houses
AveRooms → average number of rooms per household
AveOccup → average occupants per household
PRICE → target variable (house price in $100,000s)

```
housing = fetch_california_housing()

data = pd.DataFrame(housing.data, columns=housing.feature_names)
data['PRICE'] = housing.target  # Target variable is the house price


data.head()
```

✓ 0.0s

Similarly, Splitting Data into Features (X) and Target (y) where X → all independent variables (features) = everything except 'PRICE'.
y → dependent variable (target) = house price.

```
X = data.drop('PRICE', axis=1)
y = data['PRICE']
```

Now we make Train-Test Split. Split the dataset into training (80%) and testing (20%) sets. random_state ensures reproducibility of the split. We will also display data shapes, Output the number of rows and columns in the training and testing datasets for verification. random_state=42 → ensures reproducibility. Prints dataset shapes.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print("Training data shape:", X_train.shape)
print("Testing data shape:", X_test.shape)
```

Now, initialize the Linear Regression model and train the model). Train the linear regression model using the training data (`X_train` and `y_train`). model.fit() → trains the model using training features (X_train) and labels (y_train).
(Initialize the Linear Regression model and train the model)

```
model = LinearRegression()
model.fit(X_train, y_train)
```

y_pred → predicted house prices on test data (X_test).
Make predictions on the test data using the mse and rmse. Evaluate the model's performance using Mean Squared Error (MSE). Calculate Root Mean Squared Error (RMSE) for interpretability in the same unit as the target variable.

```
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)

rmse = np.sqrt(mse)
print("Root Mean Squared Error:", rmse)
```

Plot a scatter plot comparing actual house prices (`y_test`) and predicted prices (`y_pred`).

Ideal case: all points should lie on a diagonal line. (Plot the predicted vs actual prices)

```
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted House Prices")
plt.show()

✓ 0.1s
```

Create a more detailed scatter plot with labels, legends, grid, and improved aesthetics.
(With different and improved view)
Larger figure (10x10).
Blue points → predicted prices.

Red diagonal line → actual prices (perfect predictions)

```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', label='Predicted Prices', alpha=0.6)  # Predicted prices
plt.scatter(y_test, y_test, color='red', label='Actual Prices', alpha=0.6)  # Actual prices
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted House Prices")
plt.legend()
plt.grid()
plt.show()
```

(With different view)

```
# Create a DataFrame for better visualization
results = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})

# Sort by actual values for better line plotting
results.sort_index(inplace=True)

# Plot the actual vs predicted prices
plt.figure(figsize=(10, 6))
plt.plot(results.index, results['Actual'], color='red', label='Actual Prices', alpha=0.6)
plt.plot(results.index, results['Predicted'], color='blue', label='Predicted Prices', alpha=0.6)
plt.xlabel("Index")
plt.ylabel("House Prices")
plt.title("Actual vs Predicted House Prices")
plt.legend()
plt.grid()
plt.show()

✓ 0.1s
```

Sort the results DataFrame by actual prices to make it easier to interpret and display the sorted DataFrame.

```
# Create a DataFrame for actual and predicted prices
results = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})

# Optionally, sort the DataFrame by actual values for better visibility
results_sorted = results.sort_values(by='Actual').reset_index(drop=True)

# Display the results
print(results_sorted)

✓ 0.0s
```

This snippet should give actual and predicted values. The previous 3 snippets will output some graphs.

2. After running the code above, please change the dataset and see the predictions:

Please visit this (https://www.kaggle.com/datasets/yasserh/housing-prices-dataset/data) where you will find UK house data. Download and use “Housing Prices Dataset”. Please make sure to update the code for the path and column where prices are given. After this activity, search another (ANY) dataset with the same nature and display the prediction graph and then its values.