Tanmay Bhatkar Roll No.9 TEITA Batch 1

DSL-Experiment 5

To implement Regression.

Experiment Exercise

```
[ ]  import numpy as np
     import matplotlib.pyplot as plt

     import pandas as pd
     import seaborn as sns

     %matplotlib inline
```

Data Collection

```
[ ]  from sklearn.datasets import fetch_california_housing
     california_dataset = fetch_california_housing()
     print(california_dataset.keys())
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'feature_names', 'DESCR'])
```

```
[ ]  #Load the data into pandas dataframe

     # Convert to a DataFrame
     california_df = pd.DataFrame(california_dataset.data, columns=california_dataset.feature_names)

     # Display the first few rows
     california_df.head()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25

```
#The target values is missing from the data. Create a new column of target values and add it to dataframe
# Add the target variable (median house value) to the DataFrame
california_df['MedHouseVal'] = california_dataset.target

# Display the first few rows
california_df.head()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

Data Preprocessing

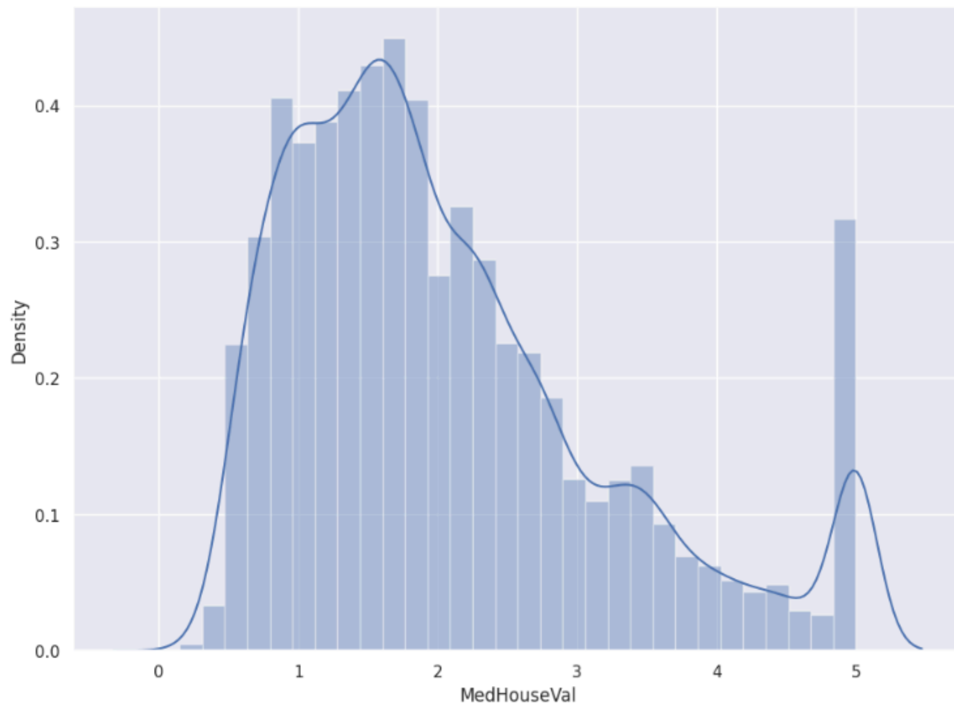
```
# check for missing values in all the columns
california_df.isnull().sum()
```

	0
MedInc	0
HouseAge	0
AveRooms	0
AveBedrms	0
Population	0
AveOccup	0
Latitude	0
Longitude	0
MedHouseVal	0

dtype: int64

Data Visualization

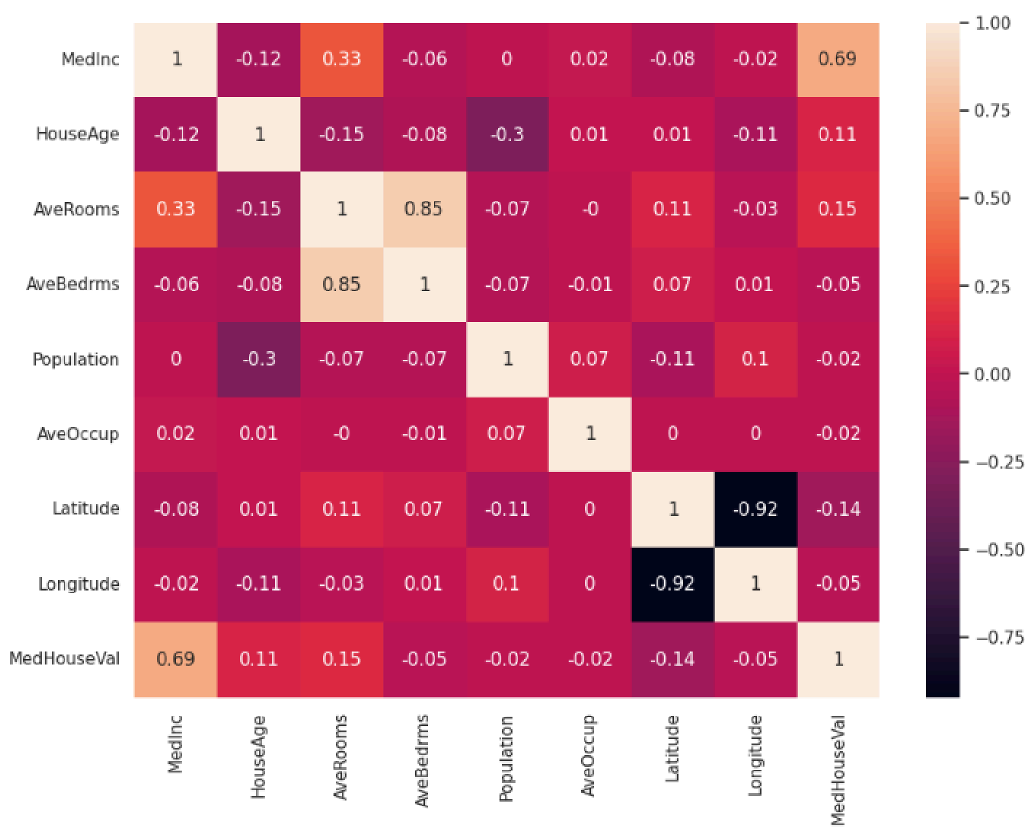
```
# set the size of the figure
sns.set(rc={'figure.figsize':(11,8)})

# plot a histogram showing the distribution of the target values
sns.distplot(california_df['MedHouseVal'], bins=30)
plt.show()
```



Corelation Matrix

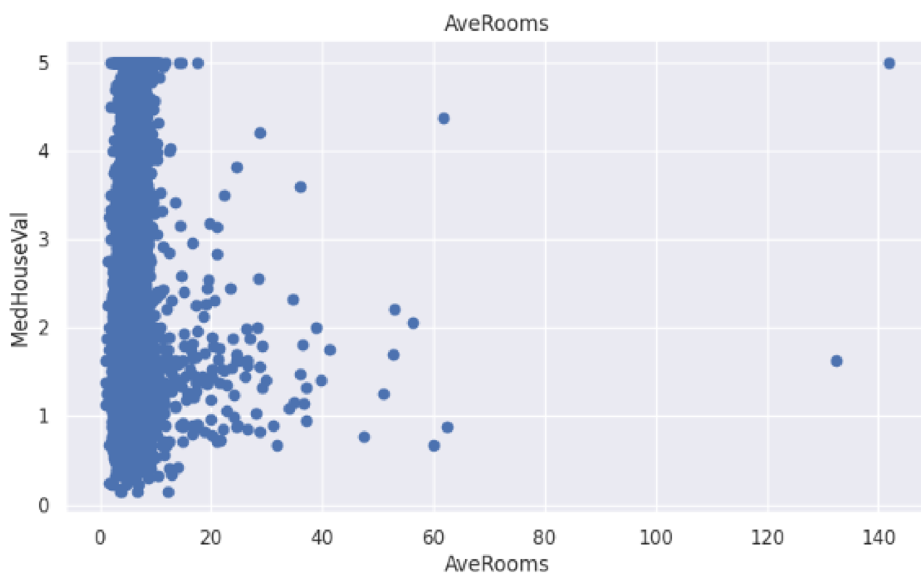
```
[ ] # compute the pair wise correlation for all columns
    correlation_matrix = california_df.corr().round(2)
```

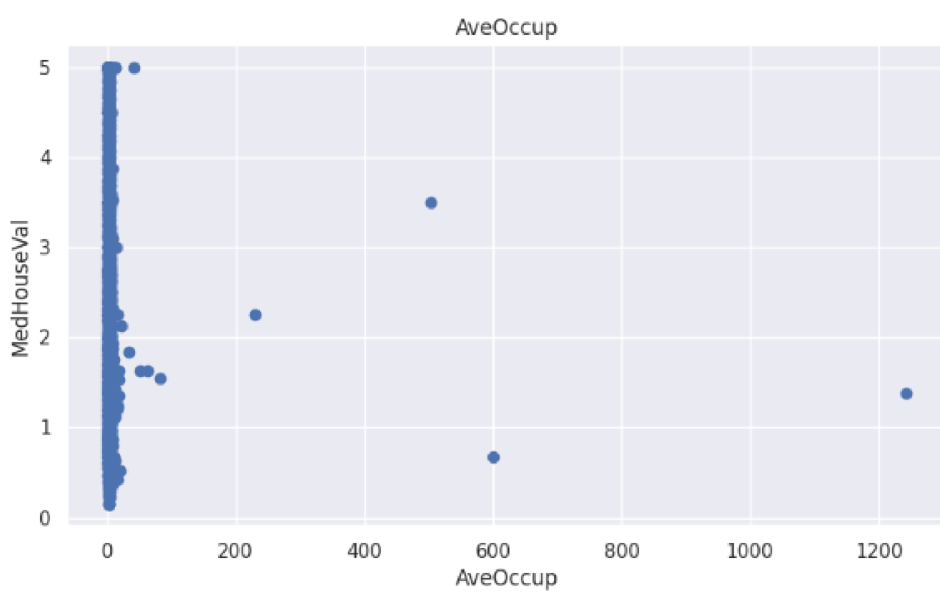
```
[ ] # use the heatmap function from seaborn to plot the correlation matrix
    # annot = True to print the values inside the square
    sns.heatmap(data=correlation_matrix, annot=True)
```



	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
MedInc	1	-0.12	0.33	-0.06	0	0.02	-0.08	-0.02	0.69
HouseAge	-0.12	1	-0.15	-0.08	-0.3	0.01	0.01	-0.11	0.11
AveRooms	0.33	-0.15	1	0.85	-0.07	-0	0.11	-0.03	0.15
AveBedrms	-0.06	-0.08	0.85	1	-0.07	-0.01	0.07	0.01	-0.05
Population	0	-0.3	-0.07	-0.07	1	0.07	-0.11	0.1	-0.02
AveOccup	0.02	0.01	-0	-0.01	0.07	1	0	0	-0.02
Latitude	-0.08	0.01	0.11	0.07	-0.11	0	1	-0.92	-0.14
Longitude	-0.02	-0.11	-0.03	0.01	0.1	0	-0.92	1	-0.05
MedHouseVal	0.69	0.11	0.15	-0.05	-0.02	-0.02	-0.14	-0.05	1

```
[ ] import matplotlib.pyplot as plt

    plt.figure(figsize=(20, 5))  # width and height in inches

    # Features to plot
    features = ['AveRooms', 'AveOccup']
    target = california_df['MedHouseVal']

    # Loop through each feature
    for i, col in enumerate(features):
        plt.subplot(1, len(features), i+1)
        x = california_df[col]
        y = target
        plt.scatter(x, y, marker='o')
        plt.title(col)
        plt.xlabel(col)
        plt.ylabel('MedHouseVal')

    plt.show()
```





```python
#Prepare data for training
X = pd.DataFrame(np.c_[california_df['AveRooms'], california_df['AveOccup']], columns=['AveRooms', 'AveOccup'])
Y = california_df['MedHouseVal']
```

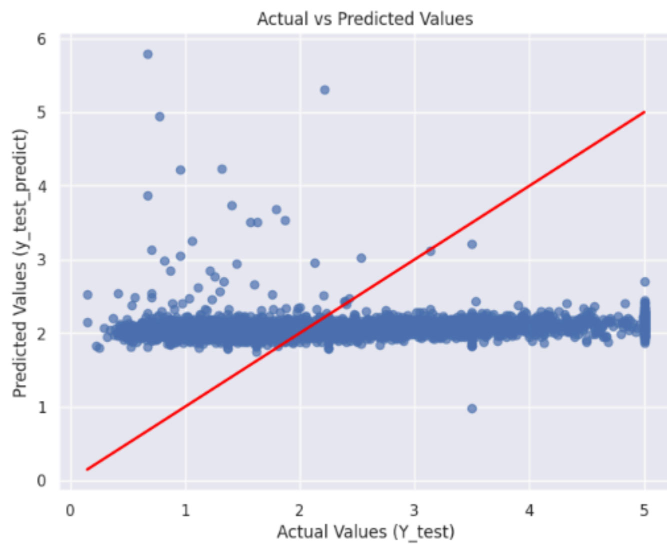
```python
from sklearn.model_selection import train_test_split

# Split the data into training and testing data (80% train, 20% test)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=5)

# Print the shapes of the resulting datasets
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(16512, 2)
(4128, 2)
(16512,)
(4128,)
```

```python
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

# Model evaluation for the training set
y_train_predict = lin_model.predict(X_train)
rmse_train = np.sqrt(mean_squared_error(Y_train, y_train_predict))
r2_train = r2_score(Y_train, y_train_predict)

print("The model performance for the training set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse_train))
print('R2 score is {}'.format(r2_train))
print("\n")

# Model evaluation for the testing set
y_test_predict = lin_model.predict(X_test)
rmse_test = np.sqrt(mean_squared_error(Y_test, y_test_predict))
r2_test = r2_score(Y_test, y_test_predict)

print("The model performance for the testing set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse_test))
print('R2 score is {}'.format(r2_test))
```

```
The model performance for the training set
--------------------------------------
RMSE is 1.1352790469141616
R2 score is 0.023154083200405018


The model performance for the testing set
--------------------------------------
RMSE is 1.159890968485926
R2 score is 0.024887570929574054
```

```python
import matplotlib.pyplot as plt

# Plotting y_test vs y_pred
plt.figure(figsize=(8, 6))
plt.scatter(Y_test, y_test_predict, alpha=0.7)
plt.plot([min(Y_test), max(Y_test)], [min(Y_test), max(Y_test)], color='red', lw=2)  # Ideal line (y = x)
plt.xlabel('Actual Values (Y_test)')
plt.ylabel('Predicted Values (y_test_predict)')
plt.title('Actual vs Predicted Values')
plt.show()
```



A. Extended Theory: (Soft Copy)

● Logistic regression

Logistic Regression is a supervised learning algorithm primarily used for classification tasks. It is particularly effective for binary classification problems, such as predicting whether a customer will buy a product (Yes/No) or whether an email is spam. Unlike linear regression, which predicts continuous values, logistic regression predicts the probability of an event occurring. The model outputs probabilities between 0 and 1, which are then converted into class labels using a threshold (commonly 0.5). The sigmoid (logistic) function is central to this approach, transforming the linear combination of input features into a probabilistic output.

Types of Logistic Regression

1. **Binary Logistic Regression** – Two possible outcomes (e.g., Fraud/Not Fraud).
   This is the most common type of logistic regression. It uses the sigmoid function to classify inputs into two distinct categories. The decision boundary is based on whether the predicted probability exceeds a defined threshold.
2. **Multinomial Logistic Regression** – More than two categories without order (e.g., Dog, Cat, Bird).
   Used when the dependent variable has more than two categories that do not have any inherent ordering. It extends binary logistic regression by using a softmax function to assign probabilities across multiple classes and selects the class with the highest probability.
3. **Ordinal Logistic Regression** – More than two categories with a meaningful order (e.g., Low, Medium, High).
   In this variant, the response variable has categories with a natural order. The model assumes proportional odds between the categories, meaning the relationship between each pair of outcome groups is the same. This is often used in survey data or ratings

Key Assumptions

- The dependent variable is **categorical**.
  Logistic regression requires a categorical target variable, typically binary, though extensions allow for multiple classes.
- Independent variables should not be **highly correlated**.
  High multicollinearity among predictors can distort the estimated coefficients and reduce the interpretability and predictive power of the model. Techniques like Variance Inflation Factor (VIF) are used to detect multicollinearity.
- The dataset should be **large enough** for accurate predictions.
  A sufficiently large dataset ensures stable and reliable estimation of parameters. Small sample sizes may result in overfitting or underfitting.
- There exists a **linear relationship** between independent variables and log-odds.
  Though logistic regression does not assume a linear relationship between the predictors and the outcome, it assumes a linear relationship between the predictors and the log-odds (logit) of the outcome.

Model Training & Performance

- **Trained using Maximum Likelihood Estimation (MLE) to find the best-fitting coefficients.MLE determines the values of the model parameters that maximize the likelihood of observing the data. It is an iterative optimization technique often solved using methods like gradient descent or Newton-Raphson.**
- **Evaluated using accuracy, precision, recall, F1-score, and ROC-AUC curve.Model evaluation goes beyond accuracy to include metrics that reflect class imbalance and misclassification costs:**
  - **Precision: Measures the accuracy of positive predictions.**
  - **Recall: Measures the ability to detect actual positives.**

- **F1-score: Harmonic mean of precision and recall.**
- **ROC-AUC: Reflects the model's ability to distinguish between classes across thresholds.**

Applications

- Medical Diagnosis (e.g., Cancer Detection)
  - Used to predict the likelihood of a disease based on patient features like age, symptoms, or lab results.
- Email Spam Filtering
  - Helps in automatically classifying emails as spam or not spam based on keywords, sender reputation, and other metadata.
- Customer Churn Prediction
  - Identifies customers likely to stop using a service by analyzing behavior patterns, demographics, or transaction history.
- Loan Default Prediction
  - Used in the financial sector to assess the risk of a customer defaulting on a loan using credit history, income, and employment data.
- Fraud Detection
  - Widely used in banking and e-commerce to flag suspicious transactions or activities by modeling patterns that typically indicate fraud.

Use MNIST Dataset and apply logistic regression.

```python
import numpy as np
import tensorflow as tf
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Load MNIST dataset
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

# Flatten images (28x28 → 784) and normalize pixel values (0-255 → 0-1)
x_train = x_train.reshape(-1, 784) / 255.0
x_test = x_test.reshape(-1, 784) / 255.0

# Train logistic regression model
log_reg = LogisticRegression(max_iter=1000, solver='lbfgs', multi_class='multinomial')
log_reg.fit(x_train, y_train)

# Predict and evaluate
y_pred = log_reg.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)

print(f"Test Accuracy: {accuracy:.4f}")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureW
  warnings.warn(
Test Accuracy: 0.9258
```