Junaid Asif
BSAI-144

Lab Report –KNN1

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

Imports: This section imports necessary libraries for data manipulation (pandas and numpy), splitting data (train_test_split), and scaling data (StandardScaler) for feature scaling.

```python
data=pd.read_csv(r'E:\Sem3-GH\Sem3Lab\AI\knn\iris.csv')
print(data.head())
```

Loading Data: It reads a CSV file named 'iris.csv' using Pandas' read_csv function from the specified file path and displays the first few rows of the dataset using head().

```python
x=data.drop('species','columns')
y=data['species']
```

Defining Features and Target: It separates the features (x) and the target variable (y) from the dataset. x contains all columns except the 'species' column, and y contains only the 'species' column.

```python
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20)
```

Train-Test Split: Splits the dataset into training and testing sets using train_test_split. 80% of the data is used for training (x_train, y_train), and 20% for testing (x_test, y_test).

```python
scaler=StandardScaler()
scaler.fit(x_train)
x_train=scaler.transform(x_train)
x_test=scaler.transform(x_test)
```

Feature Scaling: Initializes a StandardScaler and fits it on the training data (x_train). Then it transforms both the training and testing sets to have zero mean and unit variance using the fitted scaler.

```python
from sklearn.neighbors import KNeighborsClassifier
classifier=KNeighborsClassifier(n_neighbors=3)
classifier.fit(x_train,y_train)
```

K-Nearest Neighbors Classifier: Imports the K-Nearest Neighbors classifier from sklearn, initializes it with n_neighbors=3, and fits it to the scaled training data and corresponding target labels.

```python
result=classifier.predict(x_test)
```

Prediction: Uses the trained classifier to predict the target labels (species) for the test dataset (x_test).

```python
from sklearn.metrics import confusion_matrix,classification_report
print(classification_report(y_test,result))
print(confusion_matrix(y_test,result))
```

Model Evaluation: Generates a classification report and a confusion matrix by comparing the predicted results (result) against the actual target labels (y_test).