LAB # 13: Machine Learning

Objective:

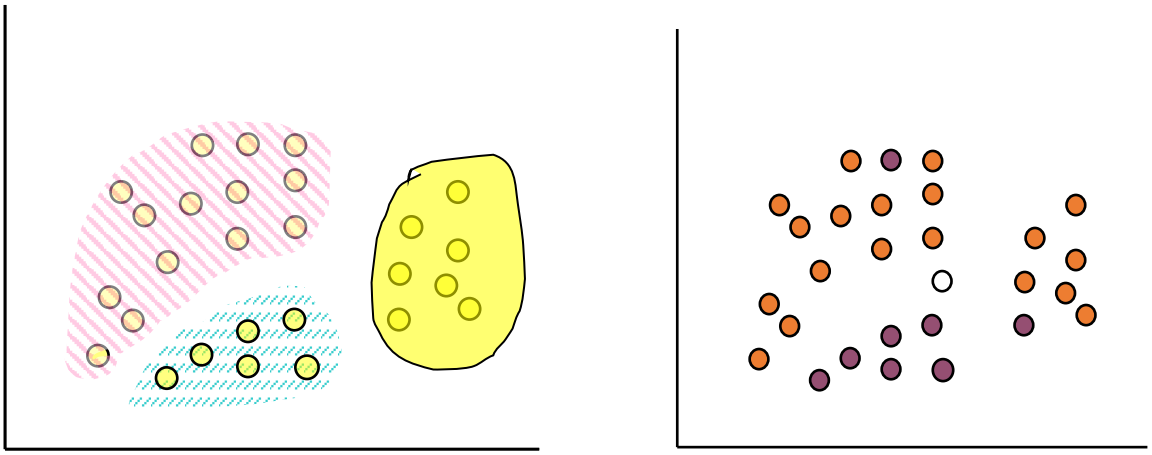
The objective of this lab is to understand and apply some well-known algorithms of machine learning.

Theory:

Machine Learning is the study of algorithms that improve their performance at some task with experience. It is to optimize a performance criterion using example data or past experience. Machine learning is programming computers to optimize a performance criterion using example data or past experience. Learning is used when:

- Human expertise does not exist (navigating on Mars),
- Humans are unable to explain their expertise (speech recognition)
- Solution changes in time (routing on a computer network)
- Solution needs to be adapted to particular cases (user biometrics)

Classification is one of the mostly used machine learning technique to enable any system to make proper decision. The purpose of classification is to decide proper class of unseen sample.



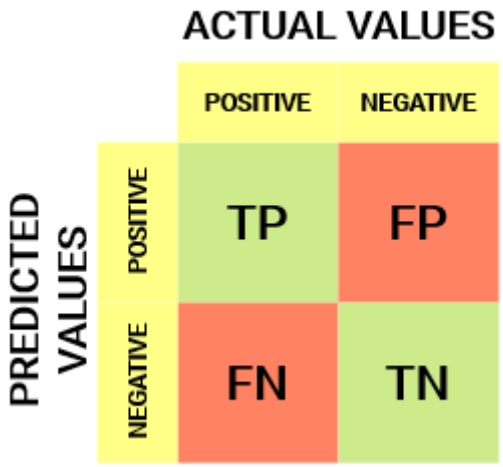
Minimum Distance Classification relies on calculating the distance of an unknown sample from all the classes and assigning the unknown sample to that class from which the distance is minimum. The mean of every feature of every class is calculated and this feature vector becomes a representation of that class. The distance of the unknown sample is calculated from this mean feature vector.

K-Nearest Neighbors classification checks k number of closest neighbors and then assigns the unknown sample to that class which has the most samples in the predefined number k.

KNN implementation:

i. **Handle data**
   Split the data set into train/test data sets.
ii. **Similarity**
   Calculate the (Euclidian) distances between the test sample and all points in the data.
iii. **Neighbors**
   Locate *k* nearest neighbors from the test sample.
iv. **Assign Output Class:**
   Assign that class to the test samples which is most prevalent in the *k* nearest neighbors.

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.



True Positive (TP)

- The actual value was positive, and the model predicted a positive value. (The predicted value matches the actual value)

True Negative (TN)

- The actual value was negative, and the model predicted a negative value. (The predicted value matches the actual value)

False Positive (FP) – Type 1 error

- The actual value was negative, but the model predicted a positive value (The predicted value was falsely predicted). Also known as the **Type 1 error**

False Negative (FN) – Type 2 error

- The actual value was positive, but the model predicted a negative value (The predicted value was falsely predicted). Also known as the **Type 2 error**

Precision:

Precision tells us how many of the correctly predicted cases actually turned out to be positive.

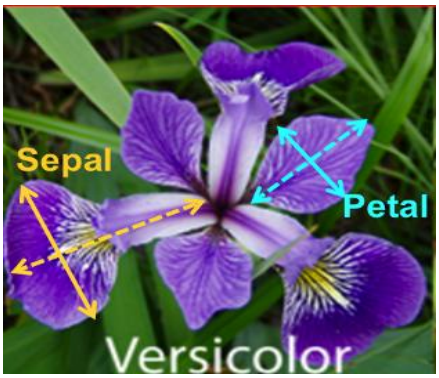
$$Precision = \frac{TP}{TP + FP}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Dataset:

The Iris dataset contains the three iris species with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

There are four features: sepal length, sepal width petal length and petal width.



Some Useful Commands:

The python package that will be needed for the following function is **pandas, sklearn and mlxtend.**

1. To read a csv file:

```
import pandas as pd
df = pd.read_csv('iris.csv')
print(df.head())
print(df.shape)
```

2. Selecting a single or multiple columns from he Pandas Dataframe:

```
Sepallength,sepalwidth=X['SepalLengthCm'],X['SepalWidthCm']
y = df['Species']
```

3. The dataset can be split into training and test sets using the following code:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

Where random state can be changed to alter the split. We can manually split the data as well. Using the manual split, we can also implement cross validation. We are going to use k = 5 for k-fold cross validation.

4. Confusion matrix can be plotted using the code:

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
cm = confusion_matrix(y_test, y_pred)
cm = confusion_matrix(y_test, predictions, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels_)
disp.plot()
plt.show()
```

5. You can use matplotlib to plot the value of accuracy as the value of k is changed. In addition, you can use the error bars to indicate the standard deviation obtained by cross validation for each value of k. The code to do so is:

```
plt.plot(accuracy_values, k_values)
plt.errorbar(accuracy_values, k_values, yerr = standard_deviation_at_k, fmt ='o')
```

6. Using sklearn to create a kNN classifier object:

```
from sklearn import neighbors
knn = neighbors.KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
```

7. Once you have created a built-in kNN object, you can use it to visualise the decision boundaries using mlxtend package. The code to do so is as follows:

```
from mlxtend.plotting import plot_decision_regions
plot_decision_regions(X, y, clf=knn, legend=2)
plt.xlabel('X')
plt.ylabel('Y')
```

```
plt.title('KNN with K=5')
plt.show()
```

Lab Tasks:

Lab Task 1:

Implement k nearest neighbor algorithm for the Iris dataset. You should create the following functions to break down your code:

- **Read_inputdata(file_path):** This function will read the csv file and save data into array.
- **Calculate_distance (instance1, instance2):** This function will find Euclidian distance between two samples.
- **Find_neighbours(k):** This function will return k nearest neighbours of unseen sample.
- **Get_response(nearest_neighbour_array):** This function will give the label for unseen sample it will belong to either class A or class B
- **Confusion_Matrix():** Calculates confusion matrix and its parameters accuracy and precision using all test samples.
- **Accuracy_with_k():** This function will plot a line graph of overall accuracy for different values of k. As you would be performing cross validation, your line graph should have the standard deviation lines as well.

Once you have implemented your own manual kNN, use the built-in k-NN function. Using this built-in function, plot the decision boundaries as well using the code above. Experiment with how it changes with changing the value of k. For k-fold cross validation, we are going to use k = 5 for our tasks.

Lab Task 2:

Using the Iris.csv file as input data, write the code for minimum distance classifier. Make a confusion matrix and calculate precision and accuracy parameters.

Conclusion:

This lab gives an introduction to widely used Machine Learning algorithms.