Subject: 22AIE213

Lab Session: 03

Notes:

1. Please read the assignment notes carefully and comply to the guidelines provided.
2. Code should be checked into the GitHub. These details shall be provided in the Lab.
3. If you have not completed the prerequisite assignments, please complete them before the next lab session.

Coding Instructions:

1. The code should be modularized; The asked functionality should be available as a function. Please create multiple functions if needed. However, all functions should be present within a single code block, if you are using Jupyter or Colab notebooks.
2. There should be no print statement within the function. All print statements should be in the main program.
3. Please use proper naming of variables.
4. For lists, strings and matrices, you may use your input values as appropriate.
5. Please make inline documentation / comments as needed within the code blocks.
6. **Please DO NOT use AI packages for automatic code generation.**

References:

Refer to lecture portions on k-NN. Also refer:

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

https://docs.scipy.org/doc//scipy-1.16.2/reference/generated/scipy.spatial.distance.minkowski.html

Please use the data associated with your own project. This assignment deals with classification models.

Main Section (Mandatory):

A1. Write your own functions to evaluate the following for vectors, A & B where A & B are vectors in N-dimensional space

- Dot product between A & B.
- Length of Vectors with Euclidean Norm

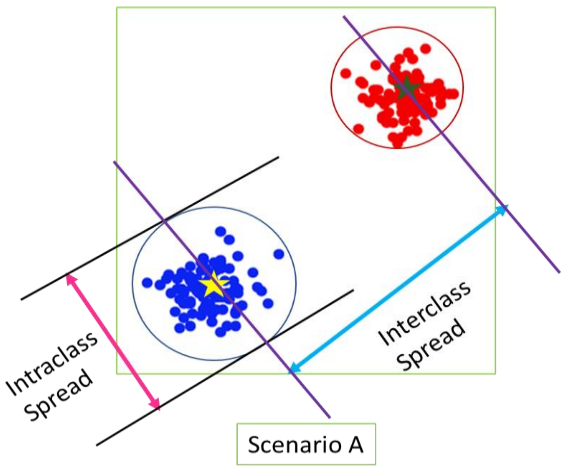
Compare the results obtained using ***`numpy.dot()`*** & ***`numpy.linalg.norm()`***

A2. Evaluate the intraclass spread and interclass distances between the classes in your dataset. If your data deals with multiple classes, you can take any two classes. Steps below (refer below diagram for understanding):

- Write your function for calculation of mean, variance and standard deviation for a set of data. Use this functions to develop a function for calculation of the values for a dataset represented as a matrix with columns as features.
- Calculate the mean for each class (also called as *class centroid*)

(*Suggestion: You may use* **`numpy.mean`**() *function for finding the average vector for all vectors in a given class. Please define the axis property appropriately to use this function. EX:* **`feat_vecs.mean(axis=0)`**)

- Calculate spread (standard deviation) for each class
  (*Suggestion: You may use* **`numpy.std`**() *function for finding the standard deviation vector for all vectors in a given class. Please define the axis property appropriately to use this function.*)
- Calculate the distance between mean vectors between classes.
  (*Suggestion:* **`numpy.linalg.norm(centroid1 – centroid2)`** *gives the Euclidean distance between two centroids.*)



A3. Take any feature from your dataset. Observe the density pattern for that feature by plotting the histogram. Use buckets (data in ranges) for histogram generation and study. Calculate the mean and variance from the available data.

(*Suggestion:* **`numpy.histogram()`** *gives the histogram data. Plot of histogram may be achieved with* **`matplotlib.pyplot.hist()`**)

A4. Take any two feature vectors from your dataset. Calculate the Minkwoski distance with p from 1 to 10. Make a plot of the distance and observe the nature of this graph. Develop your own function for calculation of the Minkowski distance metric.

A5. Compare the distance values for 2 vectors in your dataset using your own developed functions and the package function available as **`scipy.spatial.distance.minkowski().`**

A6. Divide dataset in your project into two parts – train & test set. To accomplish this, use the train-test_split() function available in SciKit. See below sample code for help:

```
>>> import numpy as np
>>> 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)
```

X is the feature vector set for your project and y is the class levels for vectors present in X.

<u>Note: Before set split, make sure you have only two classes. If your project deals with multi-class problem, take any two classes from them.</u>

A7. Train a kNN classifier (k =3) using the training set obtained from above exercise. Following code for help:

```
>>> import numpy as np
>>> from sklearn.neighbors import KNeighborsClassifier
>>> neigh = KNeighborsClassifier(n_neighbors=3)
>>> neigh.fit(X, y)
```

A8. Test the accuracy of the kNN using the test set obtained from above exercise. Following code for help.

```
>>> neigh.score(X_test, y_test)
```

This code shall generate an accuracy report for you. Please study the report and understand it.

A9. Use the predict() function to study the prediction behavior of the classifier for test vectors.

```
>>> neigh.predict(X_test)
```

Perform classification for a given vector using neigh.predict(<<test_vect>>). This shall produce the class of the test vector (test_vect is any feature vector from your test set).

A10. Repeat the kNN classification exercise with your own functions developed. Compare the results obtained from the developed kNN classifier (developed by you) against the package kNN classifier.

A11. Make k = 1 to implement NN classifier and compare the results with kNN (k = 3). Vary k from 1 to 11 and make an accuracy plot.

A12. Please evaluate confusion matrix for your classification problem. From confusion matrix, the other performance metrics such as precision, recall and F1-Score measures for both training and test data. Based on your observations, infer the models learning outcome (underfit / regularfit / overfit).

A13. Write your own code as functions to evaluate the confusion matrix and other performance metrics, namely, Accuracy, Precision. Recall and F_β-score.

A14. Compare the performance of kNN classifier with that of the matrix inversion technique.

Optional Section:

O1. Create a normal distribution data, plot the graph and compare the normal distribution plot against the histogram plot.

O2. Use different distance metric for kNN classifier by tuning the metric parameters of KNeighborsClassifier(). Observe the behaviour with change in the distance for classification.

O3. Make an AUROC plot for your project for kNN classifier. Compare the results with the area obtained and infer.

Report Assignment:

1. Update your understanding of your project in the introduction section of the report.
2. Study the downloaded papers & update the literature survey section of your report.
3. Expand the methodology and results sections with outcomes of this experiments & results obtained. Please discuss your observations, inferences in results & discussion section. Please

conclude the report appropriately with these experiments. Consider following points for observation analysis & inferences.

- Do you think the classes you have in your dataset are well separated? Justify your answer.
- Explain the behavior of the kNN classifier with increase in value of k. Explain the scenarios of over-fitting and under-fitting in kNN classifier.
- Do you think the kNN classifier is a good classifier based on the results obtained on various metrics?
- Do you think the model has regular fit situation? Use train and test set performances to arrive at this inference.
- When do you think a situation of overfit happens for kNN classifier?