**CAC -1 Machine Learning**

**Mishan Regmi 1841030 5BCA -A (Fast Track)**

**Question Number 1**

Compare and Contrast – KNN, Naïve Bayes, Decision tree classification

ANS

# KNN

1. Comparing KNN with other, we choose KNN if having a conditional independence will bring negative effect on classification. And Naive Bayes can suffer from zero probability problem when conditional probability equals zero and fail to produce a valid prediction. So KNN become an easy choice in this.

2. The contrast of KNN over Naive Bayes is it can work on more boundary like linear, elliptic, or parabolic in which Naive Bayes work.

# Naive Bayes

1. Comparing Naive Bayes which is a linear classification tend to work much better and faster than other classification on big data whereas KNN work slower because it requires more classification. So Naive Bayes is more use of we need speed. But KNN have more accuracy when we are not considering the factor speed. and error rate are also less.

2. Naive Bayes offers two hyperparameters to tune for smoothing. A hyperparameter is a prior parameter that are tuned on the training set to optimize it but on other side KNN have only one option for tuning.

# Decision Tree

1. Of the three methods, decision trees are the easiest to explain and understand. Most people understand hierarchical trees, and the availability of a clear diagram can help you to communicate your results. Conversely, the underlying mathematics behind Bayes Theorem can be very challenging to understand for the layperson. K-NN meets somewhere in the middle; Theoretically, you could reduce the K-N

2. Decision trees have easy to use features to identify the most significant dimensions, handle missing values, and deal with outliers.

**Question Number 2 & 3**

import NumPy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.metrics import precision\_recall\_fscore\_support

from sklearn.metrics import confusion\_matrix , accuracy\_score, classification\_report

from sklearn import preprocessing

from sklearn import metrics

df=pd.read\_csv("mushroom.csv")

df.head()

x=df.drop('class',axis = 1)

y=df['class']

cols=['cap-shape','cap-surface','cap-color','bruises','odor','gill-attachment','gill-spacing','gill-size','gill-color','stalk-shape', 'stalk-root',

      'stalk-surface-above-ring','stalk-surface-below-ring', 'stalk-color-above-ring','stalk-color-below-ring', 'veil-type', 'veil-color',

      'ring-number','ring-type', 'spore-print-color', 'population', 'habitat']

# Convert the data into numeric form

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

for col in cols:

     x[col]=le.fit\_transform(x[col])

y=le.fit\_transform(y)

x.head()

# Question 2:  K=1 case for KNN classification leads to over fitting. Demonstrate it using a suitable dataset and sample program.

xk\_train,xk\_test,yk\_train,yk\_test= train\_test\_split(x,y,test\_size=0.3, random\_state=42)

k = KNeighborsClassifier(n\_neighbors=1)

k.fit(xk\_train, yk\_train)

yk\_pred = k.predict(xk\_test)

yk\_pred

print("Testing Accuracy:",k.score(xk\_test, yk\_test)\*100)

print("Training Accuracy:",k.score(xk\_train,yk\_train)\*100)

# As we can see that train accuracy is 100% so model is good at training dataset

# but on test data the accuracy drops to 68.8%. at it is leading towards overfitting.

# # Question 3:  Explain the suitability of F measure as accuracy metric for class imbalanced data with an example.

data=pd.read\_csv("diabetes.csv")

data.head()

data['Outcome'].value\_counts()

data.shape

x=data.drop('Outcome',axis = 1)

y=data['Outcome']

x\_train\_ib,x\_test\_ib,y\_train\_ib,y\_test\_ib=train\_test\_split(x,y,test\_size=0.3, random\_state=42)

knn\_ib=KNeighborsClassifier()

knn\_ib.fit(x\_train\_ib,y\_train\_ib)

y\_predict\_ib=knn\_ib.predict(x\_test\_ib)

y\_predict\_ib

knn\_ib.score(x\_test\_ib,y\_test\_ib)

print(classification\_report(y\_test\_ib,knn\_ib.predict(x\_test\_ib)))

# When we need to find FP and FN we use F measure over accuracy and above the F1 score is good.