# COMP809 – Data Mining and Machine Learning

## 4 –Classifier Models

- Two major objectives of this lab are to
  - configure Python's implementation of some of the most widely used classifiers
  - to evaluate these classifiers using a variety of different metrics.

- Configuring classifiers will be achieved using Python's sklearn library. Evaluation will also be done via sklearn but use a dedicated set of methods designed specifically for computingmterics.

Same Dataset with Lab 3.

We had made some experiments about Decision Tree classifier in Lab 3.
So the first task of Lab 4 is Random Forest Classifier.

*The **Random Forest Algorithm** combines the output of multiple (randomly created) **Decision Trees** to generate the final output.*

```
import pandas as pd import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import      train_test_split,
cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB, MultinomialNB from
sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier from
sklearn.neural_network import MLPClassifier
from sklearn.metrics import precision_score, recall_score,auc from
sklearn.metrics import roc_curve,roc_auc_score, plot_roc_curve
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics

path="P:\COMP809\Iris.xlsx"  #should  change  the  pathaccordingly
rawdata= pd.read_excel(path)          #pip install xlrd


pred_train, pred_test, tar_train, tar_test = train_test_split()

  #Create a Random Forest Classifier
  # https://www.datacamp.com/community/tutorials/random-forests-classifier-python

clf = RandomForestClassifier(n_estimators=100)
  #Train the model using the training sets
clf.fit(pred_train, tar_train)
  # prediction on test set
predictions =clf.predict(pred_test)
  # Model Accuracy
print("Accuracy score of our model with Random Forest:",
accuracy_score(tar_test, predictions))
```

1. **Configure the Naive Bayes classifier with the Gaussian option for numeric data**
   More details on: https://scikit-learn.org/stable/modules/naive_bayes.html#gaussian-naive-bayes

```
gnb = GaussianNB() #suitable for numeric features
gnb.fit(pred_train, np.ravel(tar_train,order='C'))
predictions = gnb.predict(pred_test)
print("Accuracy score of our model with Gaussian Naive
Bayes:", accuracy_score(tar_test, predictions))
```

2. **Configuring the Multinomial Naïve Bayes Classifier**

By referring suitable information online identify the version of Naive Bayes suitable for classifying discrete data and fill in the line below.

```
mnb = MultinomialNB() #optimized for nominal features but can
work for numeric ones as well
mnb.fit(pred_train, np.ravel(tar_train,order='C'))
predictions = mnb.predict(pred_test)
print("Accuracy score of our model with Multinomial Naive
Bayes:", accuracy_score(tar_test, predictions))
```

3. **Configuring the K Nearest Neighbor (KNN) classifier**

By referring to https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html specify the K parameter and the search method to be KDTree (a type of indexing method) that searches for neighbors faster than brute force search.

Examine the effects of different values of K on accuracy.

```
nbrs = KNeighborsClassifier()
nbrs.fit(pred_train,np.ravel(tar_train,order='C'))
predictions = nbrs.predict(pred_test)
print("Accuracy score of our model with kNN :",
accuracy_score(tar_test, predictions))
```

## 4. Configuring the Neural Network (Multi-Layer Perceptron) MLP Classifier

By referring to https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html specify the activation function to be logistic, solver to be stochastic gradient descent, the learning rate to be 0.1, two hidden layers, containing 5 and 2 neurons respectively.

```
clf = MLPClassifier()
clf.fit(pred_train,np.ravel(tar_train,order='C'))
predictions = clf.predict(pred_test)
print("Accuracy score of our model with MLP :",
accuracy_score(tar_test, predictions))
scores = cross_val_score(clf, predictors, target, cv=10)
print("Accuracy score of our model with MLP under cross
validation :", scores.mean())
```

## 5. Present the results to a table in a word document, like the given examples.

**Normally, a table with the accuracy is enough, but it would be good to have more metrics like precision, recall, F Measure and confusion matrix in a report.**

**Don't have to submit it. Just want you to try.**

# Examples

**Table IV. Total by Technique**

| Technique | Average accuracy |
|---|---|
| Decision Trees | 83,10% |
| Artificial Neural Networks | 82,85% |
| Rules | 82,76% |
| Clustering | 82,07% |
| Bayesian Classifiers | 81,76% |
| SVM | 74,72% |

| *Method* | *Classification Accuracy* | *F Measure* | *Precision* | *Recall* |
|---|---|---|---|---|
| Decision Tree | 0.9738 | 0.9709 | 0.9713 | 0.9704 |
| Naïve Bayesian | 0.4395 | 0.3064 | 0.4228 | 0.2402 |

| Classifier | Time Taken | Correctly Classified Instances | Incorrectly Classified Instances | Kappa statistic | Mean absolute error | Root mean squared error | Confusion Matrix |
|---|---|---|---|---|---|---|---|
| Bayes Net | 0.02 Sec | 217 (75.9%) | 69 (24.1%) | 0.3958 | 0.3018 | 0.4284 | a b<br>173 28 \|<br>41 44 \| |
| Naive bayes | 0.03 Sec | 215 (75.2%) | 71 (24.8%) | 0.3693 | 0.3012 | 0.4278 | a b<br>174 27 \|<br>44 41 \| |
| **Multi layer Perceptron** | **11.7 Sec** | **276 (96.5%)** | **10 (3.5%)** | **0.9157** | **0.0482** | **0.1567** | **a b<br>197 4 \|<br>6 79\|** |
| Simple Logistics | 0.87 Sec | 218 (76.2%) | 68 (23.8%) | 0.32 | 0.3535 | 0.4183 | a b<br>191 10 \|<br>58 27 \| |
| SMO | 0.11 Sec | 218 (76.2%) | 68 (23.8%) | 0.3615 | 0.2378 | 0.4876 | a b<br>183 18 \|<br>50 35 \| |
| **IBk** | **0 Sec** | **280 (97.9%)** | **6 (2.1%)** | **0.9491** | **0.0253** | **0.1053** | **a b<br>200 1 \|<br>5 80 \|** |
| **KStar** | **0 Sec** | **280 (97.9%)** | **6 (2.1%)** | **0.9494** | **0.0747** | **0.1399** | **a b<br>199 2\|<br>4 81\|** |
| **NNge** | **0.27 Sec** | **278 (97.2%)** | **8 (2.8%)** | **0.933** | **0.028** | **0.1672** | **a b<br>197 4 \|<br>4 81 \|** |
| PART | 0.21 Sec | 229 (80.1%) | 57 (19.9%) | 0.4825 | 0.299 | 0.3866 | a b<br>184 17 \|<br>40 45 \| |
| ZeroR | 0 Sec | 201 (70.3%) | 85 (29.7%) | 0 | 0.4183 | 0.457 | a b<br>201 0 \|<br>85 0 \| |
| ADTree | 0.08 Sec | 223 (78.0%) | 63 (22.0%) | 0.4522 | 0.3659 | 0.4024 | a b<br>175 26 \|<br>37 48 \| |
| J48 | 0.02 Sec | 217 (75.9%) | 69 (24.1%) | 0.2899 | 0.3658 | 0.4269 | a b<br>194 7 \|<br>62 23 \| |
| **Random Forest** | **0.24 Sec** | **278 (97.2%)** | **8 (2.8%)** | **0.9326** | **0.1439** | **0.204** | **a b<br>193 8 \|<br>5 80 \|** |
| Simple Cart | 1.1 Sec | 201 (70.3%) | 85 (29.7%) | 0 | 0.4177 | 0.457 | a b<br>201 0 \|<br>85 0 \| |