**Practical 4 – Naïve Bayes Classifier and KNN (Solutions)**

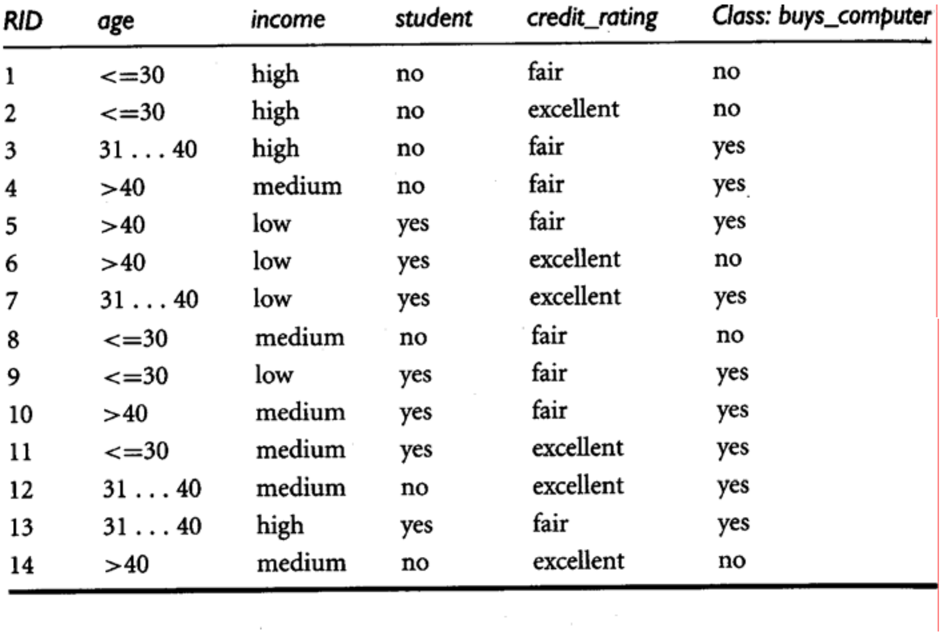
**CS5062 – Machine Learning**

## Overview

In this exercise, you will work on Naïve Bayes Classifier and use Python to implement the KNN algorithm introduced in the Lecture.

## Naïve Bayes Classifier

Given the training data in the table below (Buy Computer data), we need to predict the classes of a new example using Naïve Bayes classification. $E$ = age<=30, income=medium, student=yes, credit-rating=fair. $E_1$ is age<=30, $E_2$ is income=medium, $E_3$ is student=yes, $E_4$ is credit



| *RID* | *age* | *income* | *student* | *credit_rating* | *Class: buys_computer* |
|---|---|---|---|---|---|
| 1 | <=30 | high | no | fair | no |
| 2 | <=30 | high | no | excellent | no |
| 3 | 31 . . . 40 | high | no | fair | yes |
| 4 | >40 | medium | no | fair | yes |
| 5 | >40 | low | yes | fair | yes |
| 6 | >40 | low | yes | excellent | no |
| 7 | 31 . . . 40 | low | yes | excellent | yes |
| 8 | <=30 | medium | no | fair | no |
| 9 | <=30 | low | yes | fair | yes |
| 10 | >40 | medium | yes | fair | yes |
| 11 | <=30 | medium | yes | excellent | yes |
| 12 | 31 . . . 40 | medium | no | excellent | yes |
| 13 | 31 . . . 40 | high | yes | fair | yes |
| 14 | >40 | medium | no | excellent | no |

According to given data, please work on the following questions:

1) Please compute $P(yes)$ and $P(no)$.
   Solution:

$$P(yes) = \frac{9}{14} = 0.643$$

$$P(no) = \frac{5}{14} = 0.357$$

2) After obtaining $P(yes)$ and $P(no)$ from (1), please compute $P(E_i|yes)$ and $P(E_i|no)$, where $i = 1, 2, 3, 4$.
   Solution:
   $P(E_1|yes) = \frac{2}{9} = 0.222$, $P(E_1|no) = \frac{3}{5} = 0.6$
   $P(E_2|yes) = \frac{4}{9} = 0.444$, $P(E_2|no) = \frac{2}{5} = 0.4$

$P(E_3|yes) = \frac{6}{9} = 0.667, P(E_3|no) = \frac{1}{5} = 0.2$
$P(E_4|yes) = \frac{6}{9} = 0.667, P(E_4|no) = \frac{2}{5} = 0.4$

3) Please compute $P(yes|E)$ and $P(no|E)$.
Solution:

$$P(yes|E) = \frac{0.222 \times 0.444 \times 0.667 \times 0.668 \times 0.643}{P(E)} = \frac{0.028}{P(E)}$$

$$P(no|E) = \frac{0.6 \times 0.4 \times 0.2 \times 0.4 \times 0.357}{P(E)} = \frac{0.007}{P(E)}$$

Hence, the Naïve Bayes classifier predicts buys_computer=yes for the new example.

## K-Nearest Neighbours

The task of this exercise is to write Python code to implement KNN algorithm on a classification problem:

1) We use IRIS data for this workshop. The data can be loaded from `sklearn' using Python. (from sklearn import datasets; iris=datasets.load_iris()) This data has 150 observations of iris flowers from three different species, hence three classes: setosa, versicolor and virginica. There are four attributes in the data: sepal length, sepal width, petal length and petal width.
2) Your code to implement KNN algorithm should include the following functions:
    a. Load the data set; return 90% as train data and 10% as test data
    b. Normalize the features
    c. Calculate the Euclidean distance between any two points
    d. Find the neighbours given a test data point and number k=3
    e. Get the class labels; you should also handle ties
    f. Calculate prediction accuracy
3) Apply your KNN algorithm to the iris data.

Source codes:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import neighbors, datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score


n_neighbors = 3

# import some data to play with
iris = datasets.load_iris()

X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split\
    (X, y, test_size=0.1, random_state=42)
```

```
# Function of computing euclidean distance between two points
def dist(x, y):
    return np.sqrt(np.dot(x,x) - 2*np.dot(x, y) + np.dot(y, y))

print('Euclidean distance: ', dist(X[0,:],X[1,:]))

# we create an instance of Neighbours Classifier and fit the data.
KNN_clf = neighbors.KNeighborsClassifier(n_neighbors,
weights='uniform')
KNN_clf.fit(X_train, y_train)

# Predictions of test dataset
preds = KNN_clf.predict(X_test)

# Show the accuracy
print(accuracy_score(y_test,preds))
```