COMP809 – Data Mining and Machine Learning

Week 8– Naive Bayes Classifier

In machine learning, Naïve Bayes classification is a powerful algorithm for the classification task. Naïve Bayes classification is based on applying Bayes' theorem with strong independence assumption between the features. Naïve Bayes models are also known as simple Bayes or independent Bayes.

Naïve Bayes Classifier uses the Bayes' theorem to predict membership probabilities for each class such as the probability that given record or data point belongs to a particular class. The class with the highest probability is considered as the most likely class. This is also known as the **Maximum A Posteriori (MAP)**.

- Two major objectives of this lab are to
  - configure Python's implementation of Naive Bayes classifier.
  - 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 computing metrics.

In this lab you will build Naive Bayes Classifiers for two different datasets. First use the same Iris dataset experiments used in Lab 7 for building a Decision Tree classifier.

Part I: Naïve Bayes Classifier

Implementation of the Naïve Bayes Classifier to classify the Iris dataset used in Lab 7 for building a Decision Tree classifier.

To start with, download lab 8 files: Iris dataset ('Iris.xlsx) and the python code file (Comp615_Lab8_NaiveBayesClassifier.ipynb).

Task 1. Load library that is needed for building this model. Load dataset and declare the predictor and target variable. GaussianNB is a library for building Naïve Bayes model.

```
from sklearn.naive_bayes import GaussianNB
```

Task 2: Visualization

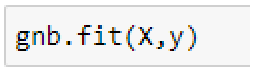
To check if the features in your dataset resemble a bell curve (indicating normal distribution) for suitability with Gaussian Naive Bayes, follow these steps:

- Plotting Features: Use histograms or density plots to visualize the distribution of each feature in your dataset.
- Assessing Normality: Look for bell-shaped curves in these plots. Features that are normally distributed (bell-shaped) are more suitable for Gaussian Naive Bayes.

Explain your findings.

Data Preprocessing Implementing k-Fold Cross-Validation

Task 3: Fit a Gaussian Naive Bayes model using your training sets, specifically the X (features) and y (labels) data.



```
gnb.fit(X,y)
```

Implementing k-fold cross-validation with a Naive Bayes model is an effective approach to evaluate its performance more reliably by using different subsets of dataset for training and testing.

Task 3.1: By using k-fold cross-validation (e.g. k= 10), split your dataset into training and testing subset with proportion 33% of the dataset is test sets.

Task 3.2: Try to change your proportion to 20% for test sets. Explain your findings.

Model Evaluation

Task 4: Generate the confusion matrix for your model and provide matrix such as True Positives, False Negatives, and False Positives based on your predictions. Explain your findings!

Task 4.1: Using the classification report, provide the performance metrics of the model. How do you think about the performance of your model?

Part II: Predict whether income exceeds $50K/year based on census data. Also known as "Census Income" dataset.

Download Files: Download (adult.csv) and (COMP809_Lab8_NB_Part_Two.ipynb)

By following steps outlined in Tasks 1 through 4 in part I, build a Naïve Bayes Classifier for the the “adult’ dataset.

Perform Data Exploration and build a Naïve Bayes model and answers questions in the code provided. The code is not complete, and you need to complete it in order to answer the questions.