Assignment 3 Advanced Data Mining

Dr. Minaei

Deadline: Friday 12th Khordad 1402

Faezeh Sadeghi
Farbod Davoodi

Bayesian Classification

- ### Part 1

Background

Bayesian classification is a probabilistic method for classifying objects into one of several classes based on the values of a set of input features. The method is based on Bayes' theorem, which states that the probability of a hypothesis (in this case, a class label) given some observed evidence (in this case, the input features) is proportional to the probability of the evidence given the hypothesis, multiplied by the prior probability of the hypothesis.

Bayesian classification is used in a wide range of applications, including medical diagnosis, fraud detection, and fault diagnosis.

Objective

The objective of this assignment is to implement a Bayesian classification algorithm to diagnose faults in a manufacturing process.

Dataset : Steel Plates Faults Data Set

- Description: This dataset contains measurements of steel plates with various faults, such as rolled-in scale, patches, and scratches. The goal is to classify the type of fault based on the measurements.
- https://archive.ics.uci.edu/ml/datasets/steel+plates+faults

Requirements

1. Preprocess the dataset.
    - Split the dataset into a training set (80%) and a test set (20%).
    - Preprocess the input features by normalizing them to have zero mean and unit variance.
2. Implement a Naive Bayes classifier for fault diagnosis.
    - Use the Laplace smoothing technique to handle zero-count feature values.
    - Train the classifier on the training set.
    - Evaluate the classifier on the test set by calculating precision, recall, F1-score, and accuracy.
3. Experiment with different values of the Laplace smoothing parameter and report the performance of the classifier for each value.
    - Plot the performance metrics (precision, recall, F1-score, and accuracy) as a function of the smoothing parameter value.

4. Implement a feature selection technique to improve the performance of the classifier.
    - Use the mutual information criterion to rank the features based on their relevance to the output variable.
    - Experiment with different numbers of top-ranked features to determine the optimal number of features for the classifier.
    - Train the classifier using the top-ranked features and evaluate its performance on the test set.

Deliverables

- Python code implementing the Naive Bayes classifier and the feature selection technique.
- A report describing the implementation details and the experimental results.

Bayesian Network Inference

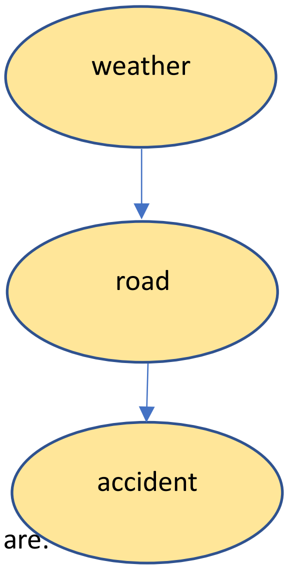
In this assignment, you will be working with a Bayesian network that represents the relationship between weather conditions, road conditions, and the probability of a car accident. Your task is to use the network to answer some questions about the probability of accidents under different conditions.

- **Part 2**

Bayesian Network Inference

In this assignment, you will be working with a Bayesian network that represents the relationship between weather conditions, road conditions, and the probability of a car accident. Your task is to use the network to answer some questions about the probability of accidents under different conditions.

The Bayesian network has the following structure:



The nodes in the network are:

- Weather: The weather condition, which can be either "sunny" or "rainy".
- Road: The road condition, which can be either "dry" or "wet".
- Accident: The probability of a car accident, which can be either "yes" or "no".

The following conditional probabilities are known:

- P(Weather=sunny) = 0.7
- P(Weather=rainy) = 0.3
- P(Road=dry|Weather=sunny) = 0.9
- P(Road=dry|Weather=rainy) = 0.6
- P(Road=wet|Weather=sunny) = 0.1
- P(Road=wet|Weather=rainy) = 0.4
- P(Accident=yes|Road=dry) = 0.05
- P(Accident=yes|Road=wet) = 0.25
- P(Accident=no|Road=dry) = 0.95
- P(Accident=no|Road=wet) = 0.75

Task 1: Calculate the probability of a car accident when the weather is sunny and the road is dry.

Task 2: Calculate the probability of a car accident when the weather is rainy and the road is wet.

Task 3: Prove the following conditional probabilities using the chain rule of probability:

P(Accident=yes|Weather=sunny, Road=dry) = 0.0315

P(Accident=yes|Weather=rainy, Road=wet) = 0.135

Task 4: Calculate the joint probability distribution of all three variables.

Task 5: Calculate the marginal probability distribution of each variable.

Task 6: Calculate the conditional probability distribution of Accident given Weather=rainy.

Task 7: Using the Markov Blanket of Accident, calculate the conditional probability distribution of Accident given Weather=sunny and Road=wet.

Task 8: Using the network structure, explain why knowing the value of Weather makes Road and Accident conditionally independent.

For this assignment, you will need to implement a Bayesian network in order to answer the questions in each task. You will need to define the network structure, set the conditional probability distributions for each node, and use the network to calculate probabilities and conditional probabilities.

You can use any programming language or library that you are comfortable with to implement the Bayesian network. There are several libraries available for implementing Bayesian networks, such as pgmpy, Pomegranate, and BayesPy in Python. These libraries provide pre-defined classes and functions for defining the network structure, learning the parameters, and making predictions.

In addition to implementing the network, you will also need to explain your methodology and reasoning behind your calculations and solutions for each task. Your explanations should be clear and concise, and should demonstrate your understanding of the Bayesian network theory and its application to real-world problems.

Submission Guidelines:

Submit a Jupyter Notebook that includes your code and explanations for each task. Your explanations should be clear and concise, and your code should be well-documented and organized. You can use any Python libraries or frameworks that you like, but please make sure to cite any external sources that you use.

SVM classification

In this section, we aim to implement Support Vector Machines (SVM) and Decision Trees using Python and scikit-learn library to help predict whether a patient has a heart disease or not. The following steps outline the process, and in each step, please provide a comprehensive report including the outputs, explanations, and necessary analyses.

1. Preprocessing the data

we will use a dataset on heart diseases. This dataset allows us to identify individuals who have been diagnosed with a heart disease based on features such as gender, age, blood pressure, and other criteria.

Dataset introduction link:

https://archive.ics.uci.edu/ml/datasets/Heart+Disease

Download link for the dataset:

https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data

1.1. Data Input

• Read the data from the desired file and store it in a dataframe.

• Enter the column names as follows:

Columns = ['age', 'sex', 'cp', 'restbp', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak',

'slope', 'ca', 'thal', 'hd']

• Perform the necessary preprocessing on the data.

• Select the column that indicates the presence of heart disease ('hd') as the column you want to predict.

- Select the other columns as features to be used for prediction.

1.2. One-Hot Encoding

- Use the "get_dummies()" function to perform one-hot encoding on non-binary columns.

2. Model Creation

Next, we will proceed with creating two models based on Support Vector Machines (SVM) and Decision Trees. In each step, if there are multiple options, please state your assumptions, and the thoroughness of your investigation and reporting will be reflected in your grade.

2.1. Support Vector Machines (SVM)

2.1.1. Data Scaling

- Split the data into training and testing sets (using random_state=42).
- Use the "scale" function to scale the training and testing data.

2.1.2. Initial Model Creation

- Create an initial SVM model.
- Specify the differences in four kernels (linear, poly, rbf, sigmoid) and their respective parameters.
- Compare the results by varying the parameter values.
- Evaluate the performance of the SVM model using the Confusion Matrix and the testing data.
- Visualize the Confusion Matrix

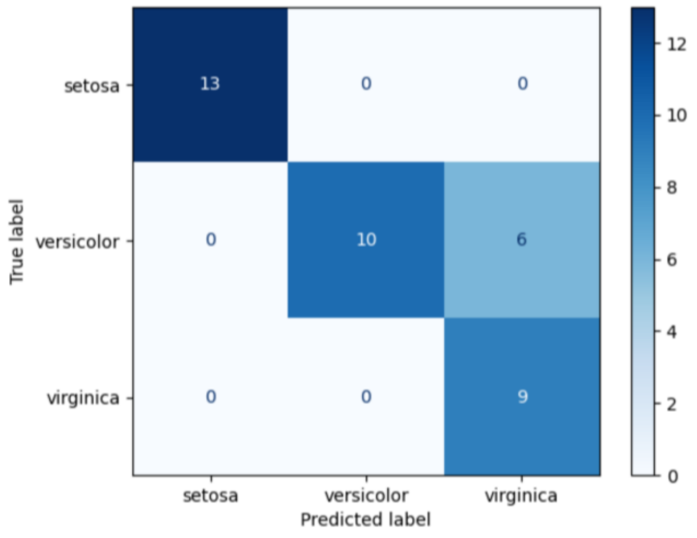


Figure 1: example of confusion matrix

2.1.3.Parameter Optimization

- Use the GridSearchCV() function to find the optimal values for the parameters.

2.1.4.Model Creation, Evaluation, Visualization, and Interpretation of the Latest SVM Model

- Create a new model using the optimized parameters.
- Compare the evaluation results of this model with the one where parameters were not optimized.
- Plot the ROC curve for both models.
- Use PCA to reduce the features to 2 dimensions and use it for plotting.
- Plot an image of the classifier similar to Figure 2.

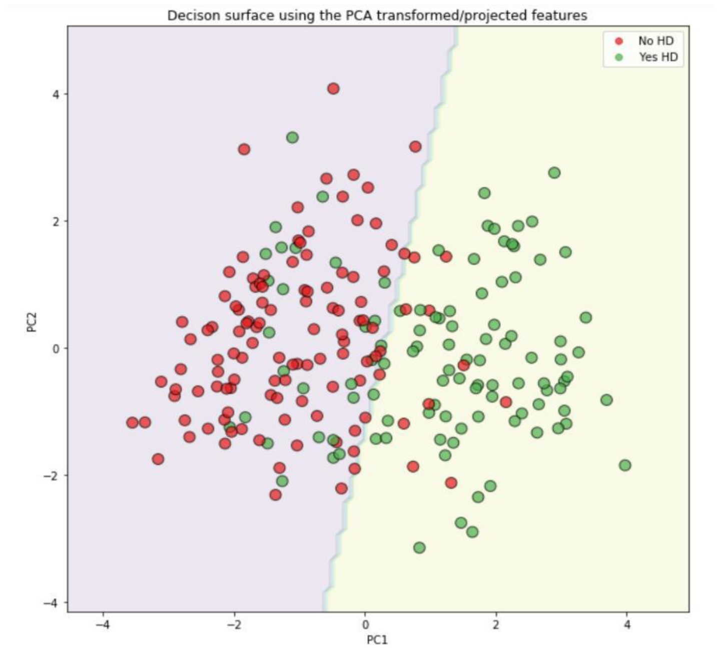


Figure 2: result of classification