**INFO6105 Data Sci Eng Methods - Midterm Exam**

**Full Name: Xijing Zhang**

1. In a decision tree, the accuracy of the model on training data is 98% but the accuracy on test data is 65%, what is the problem? Explain possible solutions for this problem.

The accuracy of test data may be caused by overfitting, small size of the training data.

Solution 1: Stop Growth (Pre-pruning) techniques(Pruning), Solve the overfitting problem

Solution 2: Post pruning techniques(Pruning) ), Solve the overfitting problem

Solution 3: Increase Training Data, , increasing the size of the training data can help the model to generalize better by learning from a more diverse set of examples.

Solution 4: Regularization, Incorporating regularization techniques can penalize the complexity of the tree, encouraging simpler models that generalize better.

1. Consider a **Random Forest** using Majority Vote with the following decision trees.

A diagram of a mathematical equation

Description automatically generated

Calculate **accuracy** of the model on the following test data:

( Accuracy = Number of correct predicted labels / Total number of predictions )

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

(x1, x2, x3: features, y: label)

Accuracy of T1 = (1+1+1)/3

Accuracy of T2 = (1+1+0)/3

Accuracy of T3 = (0+0+0)/3, Accuracy of model = 2/3

1. **Classify** (assign a label to) a test data with features of x1=4 and x2=4 using a **3-NN** and Euclidean distance on the following training data:

(Euclidean distance between a test and a train data)

|  |  |  |
| --- | --- | --- |
| x1 | x2 | Label |
| 4 | 1 | 0 |
| 4 | 2 | 1 |
| 3 | 4 | 1 |
| 2 | 4 | 0 |

Distance to point 1 = =

Distance to point 2 = =

Distance to point 3 = =

Distance to point 4 = =

3-NN (3-Nearest Neighbors) is point 3(1), point 2(1), and point 4(0)

So x1=4 and x2=4 would be classified as label "1" since two out of three closest neighbors are labeled as "1"

1. Show steps of **Apriori** algorithm to find **frequent itemsets** with length of one, two and three (if there is any) using Support Threshold of 0.5 (50%) on the following transactions:

( Support = Number of transactions containing an itemset / Total number of transactions )

{A, B, D}

{A, C, D}

{A, B, C, D}

{A, C}

{A, D}

**Step 1: Get Support for length 1**

{A} with support = 5/5 = 1

{B} with support = 2/5 = 0.4 (not frequent)

{C} with support = 3/5 = 0.6

{D} with support = 4/5 = 0.8

**Step 2: Get Support for length 2**

{A, C} with support = 3/5 = 0.6

{A, D} with support = 4/5 = 0.8

{C, D} with support = 2/5 = 0.4(not frequent)

**Step 3: Get Support for length 3**

Since all frequent itemsets of length two contain at least one infrequent item, we cannot generate frequent itemsets of length three.

**Result:**

Frequent itemsets:

Length one: {A}, {C}, {D}

Length two: {A, C}, {A, D}

Step For code:

**# Step 1 Input Dateset**

org\_df = pd.read\_csv("amr\_horse\_ds.csv")

org\_df= pd.get\_dummies(org\_df.loc[:,org\_df.columns!='Age'])

**# Step 2 Extract Association Rules and set the min\_support=0.5**

frequent\_patterns\_df = fpgrowth(org\_df, min\_support=0.5,use\_colnames=True)

rules\_df = association\_rules(frequent\_patterns\_df, metric = "confidence", min\_threshold = 0.9)

high\_lift\_rules\_df = rules\_df[rules\_df['lift'] > 1.5]

**# Step 3 Save Association Rules**

high\_lift\_rules\_df.to\_csv('arules.csv')

1. Create a **bottom-up hierarchical clustering** (agglomerative) on the following training data using Euclidean distance and single linkage (nearest distance between two clusters):

( Euclidean distance between and = )

|  |  |
| --- | --- |
|  | Score |
| Student1 | 70 |
| Student2 | 75 |
| Student3 | 90 |
| Student4 | 60 |

import pandas as pd

import numpy as np

from sklearn.cluster import KMeans

from scipy.cluster.hierarchy import dendrogram, linkage

import matplotlib.pyplot as plt

# Create a DataFrame with the scores of the four students

data = {

'Student': ['Student1', 'Student2', 'Student3', 'Student4'],

'Score': [70, 75, 90, 60]

}

df = pd.DataFrame(data)

train\_feat = df[['Score']]

# Agglomerative clustering

linkage\_data = linkage(train\_feat, method='single', metric='euclidean')

dendrogram(linkage\_data)

plt.show()

A graph with blue and orange squares

Description automatically generated