

FINAL-TERM PROJECT - CS 634 DATA MINING

Instructed by Professor Jason T.L. Wang

OPTION 1

**SUPERVISED DATA MINING
(CLASSIFICATION)**

Naive Bayes and Decision Tree Classification

By : Roopali Rajaram Sarode

NJIT ID : 31570495

Email Id : rs366@njit.edu

Table of contents

Contents	Page Number
Introduction	3
Input Dataset	9
Guide to run the program	11
Source Code for Naive Bayes algorithm	12
Output for Naive Bayes algorithm	14
Source Code for Decision Tree algorithm	16
Output for Decision Tree algorithm	18
Decision Tree visualization	20
K-fold cross validation	21
Confusion matrix	22
Classifier comparison & Conclusion	24

Introduction

What is supervised data mining ?

Supervised data mining is applying algorithms or techniques that learn from a training data set which is labeled. The trained algorithm is then used to classify the data or predict outcome accurately with utmost accuracy.

What is Classification ?

Classification is a data mining technique or function that uses supervised learning to identify the class to which the data belongs.

Classification algorithms can be divided into 2 categories:

- Generative
- Discriminative

Generative classifiers : It includes the distribution of the data itself, and tells how likely a given example is. These classifiers try to learn the model which creates the data through estimation of distributions and assumptions of the model. When a new observation is given to these classifiers, it tries to predict which class would have most likely generated the given observation. Such methods try to learn about the environment. A great example of generative algorithms is **Naive Bayes Classifier**.

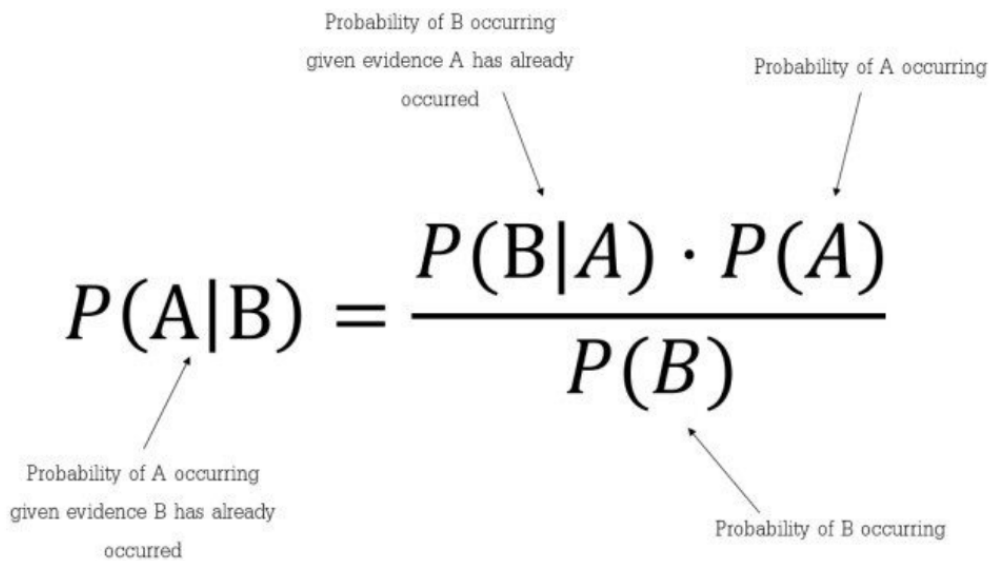
Discriminative classifiers : these kinds of classifiers try to find boundaries that distinguish a class from another. They learn what the features in the input are most useful to distinguish between the various possible classes. Discriminative classifier tries to model just by depending on the observed data, and depends heavily on the quality of data rather than on distributions. Logistic regression, **decision trees** are excellent examples of discriminative classifiers.

What is the Naive Bayes algorithm ?

Naive Bayes is a probabilistic machine learning algorithm based on the Bayes Theorem, used in a wide variety of classification tasks.

Conditional probability : It is a measure of the probability of an event occurring given that another event has (by assumption, presumption, assertion, or evidence) occurred.

$$P(A,B) = P(A|B)P(B) = P(B|A)P(A)$$

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$


Probability of B occurring
given evidence A has already
occurred

Probability of A occurring

Probability of A occurring
given evidence B has already
occurred

Probability of B occurring

Above mentioned is the Bayes rule. Here, A represents the class eg. cat, dog, bat etc and B represents features calculated individually. The fundamental assumption made in the algorithm is that all the variables in the dataset are “Naive” i.e not correlated to each other or independent of each other.

Terminology related to Naive Bayes:

$P(A|B)$: posterior probability for A

$P(Y)$: prior probability

$P(X|Y)$: class-conditional probability

$P(X)$: evidence

Bayes theorem (Bayes rule) allows us to calculate the posterior probability $P(Y|X)$ using the prior probability $P(Y)$, the class-conditional probability $P(X|Y)$ and the evidence $P(X)$ (which is constant and ignored).

Types of Naive Bayes Classifiers:

1. **Gaussian Naive Bayes Classifier** : It assumes that the features follow a normal distribution. This algorithm is used when the features are continuous values.
2. **Multinomial Naive Bayes Classifier**: This algorithm is generally used in Natural Language Processing (NLP). Feature vectors represent the frequencies with which certain

events have been generated by a multinomial distribution. For eg. Document Classification.

3. **Complement Naive Bayes classifier** : In complement Naive Bayes, instead of calculating the probability of an item belonging to a certain class, the probability of the item belonging to all the classes is calculated.
4. **Bernoulli Naive Bayes classifier** : This model is used when the feature vectors are binary.

Few advantages of Naive Bayes algorithm :

- It can perform better in comparison with other models like logistic regression when the assumption of independence holds and less training data is required.
- Prediction of the class of the training dataset is easy and fast. It also performs well in multi- class prediction.
- It is better suited for categorical input variables than numerical variables.

Few disadvantages of Naive Bayes algorithm

- Due to its assumption that all features are independent, which is rarely the case in real life which limits the applicability of this algorithm in real-world use cases.
- This algorithm faces the 'zero-frequency problem' where it assigns zero probability to a categorical variable whose category in the test dataset was not available in the training dataset.

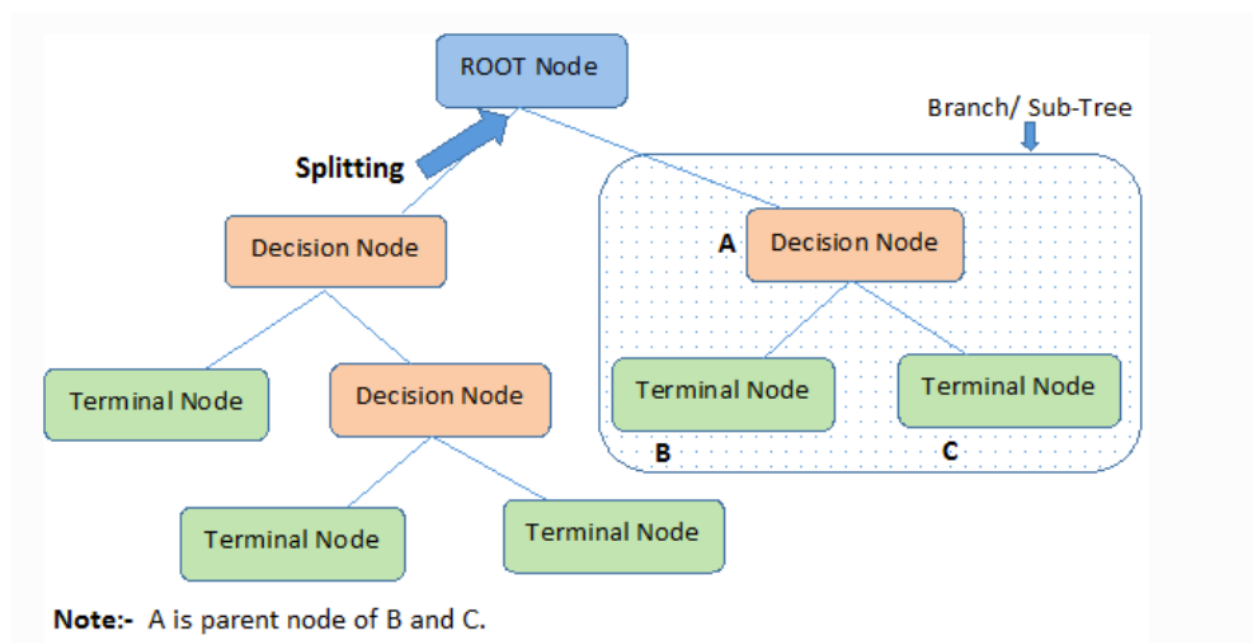
What is the Decision Tree algorithm ?

Decision tree is another supervised learning technique that can be used for both classification and regression problems. It is a tree-structured classifier, where internal nodes represent the features of the dataset, branches represent the decision rules and each leaf node represents the result.

It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. Similar to a tree it starts at the root node, expanding on branches till the leaf nodes.

Terminology related to Decision Tree :

- **Root Node** : It represents the entire population or sample and this further gets divided into two or more homogeneous sets.
- **Decision Node** : When a sub-node splits into further sub-nodes, then it is called the decision node.
- **Leaf Node/Terminal Node** : It represents the result of the branches and does not split any further.
- **Branch** : A subsection of the entire tree is called a branch or a sub-tree
- **Parent and Child Node** : A node, which is divided into sub-nodes is called a parent node whereas the sub-nodes connected to a parent node are called child nodes.



The decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes.

Few more terminology used for calculating the decision tree :

- **Entropy** : Entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information. The smaller the value of entropy the better selection it turns out to be. A branch with an entropy of zero is a leaf node and A branch with entropy more than zero needs further splitting.

$$\sum \frac{p_i + n_i}{p + n} * IG(A)$$

Where

p = positive class

n= negative class

IG = Information Gain

- **Information Gain** : It is a measure of how well a given attribute separates a training data according to its target classification. While constructing a decision tree, the attribute with the highest value of Information Gain should be selected. Information gain is a decrease in entropy. It computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values. Information gain computes the difference between entropy before and after split and specifies the impurity in class elements.

$$\frac{-p}{p+n} * \log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n} * \log_2\left(\frac{n}{p+n}\right)$$

Where

p = positive class

n= negative class

- **Gini Index** : Gini Impurity is a measurement used to build Decision Trees to determine how the features of a data set should split nodes to form the tree. Higher value of Gini index implies higher inequality, higher heterogeneity.

$$Gini = 1 - \sum_{i=1}^n (p_i)^2$$

Where

n = number of classes in the dataset

P_i = probability of an object being classified to a particular class

- **Gain** : It is a measure of overall gain of the node and based on this value the dominant node is decided. Node with maximum gain is chosen as the dominant node.

Formula for Gain:

$$Gain(A) = I(p, n) - E(A)$$

Where

I = Information Gain

p = positive class

n= negative class

E(A) = Entropy of an attribute A

Input Dataset

The Mushroom dataset consists of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family

Number of Instances: 8124
Dataset Source : [UCI Machine Learning Repository: Mushroom Data Set](#)
Number of Attributes: 22 predictor attributes+1 column for the class (edible or poisonous)
Attribute Information: classes: edible=e, poisonous=p
Class Distribution:
-- edible: 4208 (51.8%)
-- poisonous: 3916 (48.2%)
-- total: 8124 instances

Input Datasource : [mushroom.csv](#)

1. cap-shape	bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
2. cap-surface	fibrous=f,grooves=g,scaly=y,smooth=s
3. cap-color	brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y
4. bruises?	bruises=t,no=f
5. odor	almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s
6. gill-attachment	attached=a, descending=d, free=f, notched=n
7. gill-spacing	close=c, crowded=w, distant=d
8. gill-size	broad=b, narrow=n
9. gill-color	black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y
10. stalk-shape	enlarging=e, tapering=t
11. stalk-root	bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?
12. stalk-surface-above-ring	fibrous=f, scaly=y, silky=k, smooth=s
13. stalk-surface-below-ring	fibrous=f, scaly=y, silky=k, smooth=s
14. stalk-color-above-ring	brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y

15. stalk-color-below-ring	brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
16. veil-type	partial=p, universal=u
17. veil-color	brown=n, orange=o, white=w, yellow=y
18. ring-number	none=n, one=o, two=t
19. ring-type	cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z
20. spore-print-color	black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y
21. population	abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y
22. habitat	grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d

Snippet of the input file:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	class	cap-shape	cap-surf	cap-color	bruises	odor	gill-attach	gill-spacing	gill-size	gill-color	stalk-shap	stalk-root	stalk-surf	stalk-surf	stalk-color	stalk-color	veil-type	veil-color	ring-numb	ring-type	spore-prin	population	habitat
2	p	x	s	n	t	p	f	c	n	k	e	e	s	s	w	w	p	w	o	p	k	s	u
3	e	x	s	y	t	a	f	c	b	k	e	c	s	s	w	w	p	w	o	p	n	n	g
4	e	b	s	w	t	l	f	c	b	n	e	c	s	s	w	w	p	w	o	p	n	n	m
5	p	x	y	w	t	p	f	c	n	n	e	e	s	s	w	w	p	w	o	p	k	s	u
6	e	x	s	g	f	n	f	w	b	k	t	e	s	s	w	w	p	w	o	e	n	a	g
7	e	x	y	y	t	a	f	c	b	n	e	c	s	s	w	w	p	w	o	p	k	n	g
8	e	b	s	w	t	a	f	c	b	g	e	c	s	s	w	w	p	w	o	p	k	n	m
9	e	b	y	w	t	l	f	c	b	n	e	c	s	s	w	w	p	w	o	p	n	s	m
10	p	x	y	w	t	p	f	c	n	p	e	e	s	s	w	w	p	w	o	p	k	v	g
11	e	b	s	y	t	a	f	c	b	g	e	c	s	s	w	w	p	w	o	p	k	s	m
12	e	x	y	y	t	l	f	c	b	g	e	c	s	s	w	w	p	w	o	p	n	n	g
13	e	x	y	y	t	a	f	c	b	n	e	c	s	s	w	w	p	w	o	p	k	s	m
14	e	b	s	y	t	a	f	c	b	w	e	c	s	s	w	w	p	w	o	p	n	s	g
15	p	x	y	w	t	p	f	c	n	k	e	e	s	s	w	w	p	w	o	p	n	v	u
16	e	x	f	n	f	n	f	w	b	n	t	e	s	f	w	w	p	w	o	e	k	a	g
17	e	s	f	g	f	n	f	c	n	k	e	e	s	s	w	w	p	w	o	p	n	y	u
18	e	f	f	w	f	n	f	w	b	k	t	e	s	s	w	w	p	w	o	e	n	a	g
19	p	x	s	n	t	p	f	c	n	n	e	e	s	s	w	w	p	w	o	p	k	s	g
20	p	x	y	w	t	p	f	c	n	n	e	e	s	s	w	w	p	w	o	p	n	s	u
21	p	x	s	n	t	p	f	c	n	k	e	e	s	s	w	w	p	w	o	p	n	s	u
22	e	b	s	y	t	a	f	c	b	k	e	c	s	s	w	w	p	w	o	p	n	s	m
23	p	x	y	n	t	p	f	c	n	n	e	e	s	s	w	w	p	w	o	p	n	v	g
24	e	b	y	y	t	l	f	c	b	k	e	c	s	s	w	w	p	w	o	p	n	s	m
25	e	b	y	w	t	a	f	c	b	w	e	c	s	s	w	w	p	w	o	p	n	n	m
26	e	b	s	w	t	l	f	c	b	g	e	c	s	s	w	w	p	w	o	p	k	s	m
27	p	f	s	w	t	p	f	c	n	n	e	e	s	s	w	w	p	w	o	p	n	v	g
28	e	x	y	y	t	a	f	c	b	n	e	c	s	s	w	w	p	w	o	p	n	n	m
29	e	x	y	w	t	l	f	c	b	w	e	c	s	s	w	w	p	w	o	p	n	n	m

Guide to run the program :

Programs for both the classification algorithms have been written in separate python files. Visual Studio Code software has been used to write the code and compile it. Both the source code files are stored under a folder named src which is located in the main folder of the project 'Classification_Algorithms'.

This PC > Nihal (D:) > ROOPALI FILES > SPRING 2022 COURSES > DATA MINING > Classification_Algorithms > src				
Name	Date modified	Type	Size	
decision_tree_algo	7/20/2022 6:03 PM	Python Source File	5 KB	
Naive_bayes_algo	7/20/2022 6:06 PM	Python Source File	4 KB	

This PC > Nihal (D:) > ROOPALI FILES > SPRING 2022 COURSES > DATA MINING > Classification_Algorithms				
Name	Date modified	Type	Size	
data	7/20/2022 6:21 PM	File folder		
image	7/20/2022 6:05 PM	File folder		
result	7/20/2022 6:21 PM	File folder		
src	7/20/2022 6:22 PM	File folder		

Execution of both the program files has been kept simple and uncluttered inorder for better understanding. To run the program, open the desired algorithm file in Visual Studio Code. Simply right click on the code file and select 'run python file in terminal'.

Source Code for Naive Bayes algorithm :

```
import csv
import pandas as pd
from sklearn import metrics
from sklearn.model_selection import cross_validate, train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB as NB

# Function for 10 fold cross validation

def cross_validation(model, _X, _Y, _cv=10):
    '''Function to perform 10 Folds Cross-Validation
    Parameters
    -----
    model: Python Class, default=None
        This is the machine learning algorithm to be used for training.
    _X: array
        This is the matrix of features.
    _y: array
        This is the target variable.
    _cv: int, default=5
        Determines the number of folds for cross-validation.
    Returns
    -----
    The function returns a dictionary containing the metrics 'accuracy',
    'precision',
    'recall', 'f1' for both training set and validation set.
    '''
    _scoring = ['accuracy', 'precision', 'recall', 'f1']
    results = cross_validate(estimator=model,
                            X=_X,
                            y=_Y,
                            cv=_cv,
                            scoring=_scoring,
                            return_train_score=True)

    return {
        "Mean Training Accuracy": results['train_accuracy'].mean()*100,
        "Mean Training Precision": results['train_precision'].mean(),
        "Mean Training Recall": results['train_recall'].mean(),
        "Mean Training F1 Score": results['train_f1'].mean(),
        "Mean Validation Accuracy": results['test_accuracy'].mean()*100,
        "Mean Validation Precision": results['test_precision'].mean(),
        "Mean Validation Recall": results['test_recall'].mean(),
        "Mean Validation F1 Score": results['test_f1'].mean()
    }

#Function for writing predicted results into a csv file

def output_writer(path, result):
```

```

with open(path, 'w') as f:
    file = csv.writer(f, delimiter=',', quotechar='\r')
    for item in result:
        file.writerow([int(item)])
print('Results have been successfully saved to: %s' % (path))

def main():

    # Reading and importing the data file
    data = pd.read_csv('../data/mushroom.csv')

    # Printing dimensions of the input data file
    print("\nThe dataset contains " + str(data.shape[0]) + " rows and " +
str(data.shape[1]) + " columns")

    # Pre-processing the input data
    # Separating the response from predictors
    # 'class' is response in this dataset and and rest columns are
predictors
    X = data.drop(['class'],axis=1)
    Y = data['class']

    #Encoding the categorical data via Label Encoding
    X = pd.get_dummies(X)
    encoder= LabelEncoder()
    Y = encoder.fit_transform(Y)

    # Splitting data into training and testing dataset
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size =
0.3,random_state = 42)

    #Using Naive Bayes Classifier from Sklearn library
    classifier = NB()

    # Fitting Decision Tree classifier to the Training set
    classifier.fit(X_train,y_train)

    # Predicting the values for test dataset
    y_pred=classifier.predict(X_test)

    # Writing the predicted values to a file
    print("\n")
    output_writer("../result/Naive_Bayes_Prediction_Result.csv", y_pred)
    print("\n")

    # Performing 10 fold cross validation for accuracy
    naive_bayes_result = cross_validation(classifier, X, Y, 10)
    for key,value in naive_bayes_result.items():
        print(key,value)

    # Creating confusion matrix for test dataset
    print("\n Confusion Matrix\n")

```

```
print(metrics.confusion_matrix(y_test, y_pred))
print("\n")

main()
```

Output for Naive Bayes Algorithm :

```
PS D:\ROOPALI FILES\SPRING 2022 COURSES\DATA MINING\Classification_Algorithms\src> & C:/Users/roopa/AppData/Local
/Microsoft/WindowsApps/python3.9.exe "d:/ROOPALI FILES/SPRING 2022 COURSES/DATA MINING/Classification_Algorithms/
src/Naive_bayes_algo.py"

The dataset contains 8124 rows and 23 columns

Results have been successfully saved to: ../result/Naive Bayes Prediction Result.csv

Mean Training Accuracy 96.34958765410595
Mean Training Precision 0.9314489672098911
Mean Training Recall 0.999177168111672
Mean Training F1 Score 0.9638166034559438
Mean Validation Accuracy 91.47033428462363
Mean Validation Precision 0.9053803927932467
Mean Validation Recall 0.9334183673469388
Mean Validation F1 Score 0.9001751332772514

Confusion Matrix

[[1171  86]
 [  1 1180]]

PS D:\ROOPALI FILES\SPRING 2022 COURSES\DATA MINING\Classification_Algorithms\src> █
```

The program creates a csv file which contains the predicted results of test data. The file gets stored in the result folder which is under the main folder (Classification_Algorithms). The csv file is stored with the name "Naive_Bayes_Prediction_result". The training to test ratio used here is 70% to 30% respectively.

Snippet from Naive_Bayes_Prediction_result.csv

	A	B	C	D	E	F	G	H	
1	0								
2									
3	1								
4									
5	1								
6									
7	0								
8									
9	1								
10									
11	1								
12									
13	1								
14									
15	1								

The above file stores the classification result of the test dataset. Here 0 represents the “edible” class and 1 represents the “poisonous” class.

Source Code for Decision Tree algorithm :

```
import csv
import graphviz
import pandas as pd
from sklearn import metrics, tree
from sklearn.model_selection import cross_validate, train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier as DT

# Function for 10 fold cross validation
def cross_validation(model, _X, _Y, _cv=10):
    '''Function to perform 10 Folds Cross-Validation
    Parameters
    -----
    model: Python Class, default=None
        This is the machine learning algorithm to be used for training.
    _X: array
        This is the matrix of features.
    _y: array
        This is the target variable.
    _cv: int, default=5
        Determines the number of folds for cross-validation.
    Returns
    -----
    The function returns a dictionary containing the metrics 'accuracy',
    'precision',
    'recall', 'f1' for both training set and validation set.
    '''
    _scoring = ['accuracy', 'precision', 'recall', 'f1']
    results = cross_validate(estimator=model,
                            X=_X,
                            y=_Y,
                            cv=_cv,
                            scoring=_scoring,
                            return_train_score=True)

    return {
        "Mean Training Accuracy": results['train_accuracy'].mean()*100,
        "Mean Training Precision": results['train_precision'].mean(),
        "Mean Training Recall": results['train_recall'].mean(),
        "Mean Training F1 Score": results['train_f1'].mean(),
        "Mean Validation Accuracy": results['test_accuracy'].mean()*100,
        "Mean Validation Precision": results['test_precision'].mean(),
        "Mean Validation Recall": results['test_recall'].mean(),
        "Mean Validation F1 Score": results['test_f1'].mean()
    }

#Function for writing predicted results into a csv file
def output_writer(path, result):
```



```

with open(path, 'w') as f:
    file = csv.writer(f, delimiter=',', quotechar='\"')
    for item in result:
        file.writerow([int(item)])
print('Results have been successfully saved to: %s' % (path))

def main():
    # Reading and importing the data file
    data = pd.read_csv('../data/mushroom.csv')
    data.head()

    # Printing dimensions of the input data file
    print("\nThe dataset contains " + str(data.shape[0]) + " rows and " +
str(data.shape[1]) + " columns")

    # Pre-processing the input data
    # Separating the response from predictors
    # 'class' is response in this dataset and rest columns are
predictors
    X = data.drop(['class'],axis=1)
    Y = data['class']

    #Encoding the categorical data via Label Encoding
    X = pd.get_dummies(X)
    encoder = LabelEncoder()
    Y = encoder.fit_transform(Y)

    # Splitting data into training and testing dataset
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size =
0.3,random_state = 42)

    #Using Decision Tree Classifier from Sklearn library
    classifier = DT(criterion="entropy", min_samples_split=5,
random_state=0)

    # Fitting Decision Tree classifier to the Training set
    classifier.fit(X_train,y_train)

    # Predicting the values for test dataset
    y_pred=classifier.predict(X_test)

    # Writing the predicted values to a file
    print("\n")
    output_writer("../result/Decision_Tree_Prediction_Result.csv", y_pred)
    print("\n")

    # Performing 10 fold cross validation for accuracy
    decision_tree_result = cross_validation(classifier, X, Y, 10)
    for key,value in decision_tree_result.items():
        print(key,value)

    # Creating confusion matrix for test dataset
    print("\n Confusion Matrix\n")
    print(metrics.confusion_matrix(y_test, y_pred))

```

```

print("\n")

# Visualization for Decision Tree
dot_data = tree.export_graphviz(classifier, out_file = None,
filled=True, rounded=True, feature_names =X.columns, class_names =
['edible', 'poisonous'])
graph = graphviz.Source(dot_data)
graph.format = 'png'
graph.render('../image/decision_tree_1')
print('>> [Decision Tree Runner] - Tree visualization complete.')
print("\n")

main()

```

Output for Decision Tree Algorithm :

```

PS D:\ROOPALI FILES\SPRING 2022 COURSES\DATA MINING\Classification_Algorithms\src> & C:/Users/roopa/AppData/Local
/Microsoft/WindowsApps/python3.9.exe "d:/ROOPALI FILES/SPRING 2022 COURSES/DATA MINING/Classification_Algorithms/
src/decison_tree_algo.py"

The dataset contains 8124 rows and 23 columns

Results have been successfully saved to: ../result/Decision_Tree_Prediction_Result.csv

Mean Training Accuracy 100.0
Mean Training Precision 1.0
Mean Training Recall 1.0
Mean Training F1 Score 1.0
Mean Validation Accuracy 96.85116851168513
Mean Validation Precision 1.0
Mean Validation Recall 0.9346938775510203
Mean Validation F1 Score 0.9515151515151515

Confusion Matrix
[[1257   0]
 [   0 1181]]

>> [Decision Tree Runner] - Tree visualization complete.

```

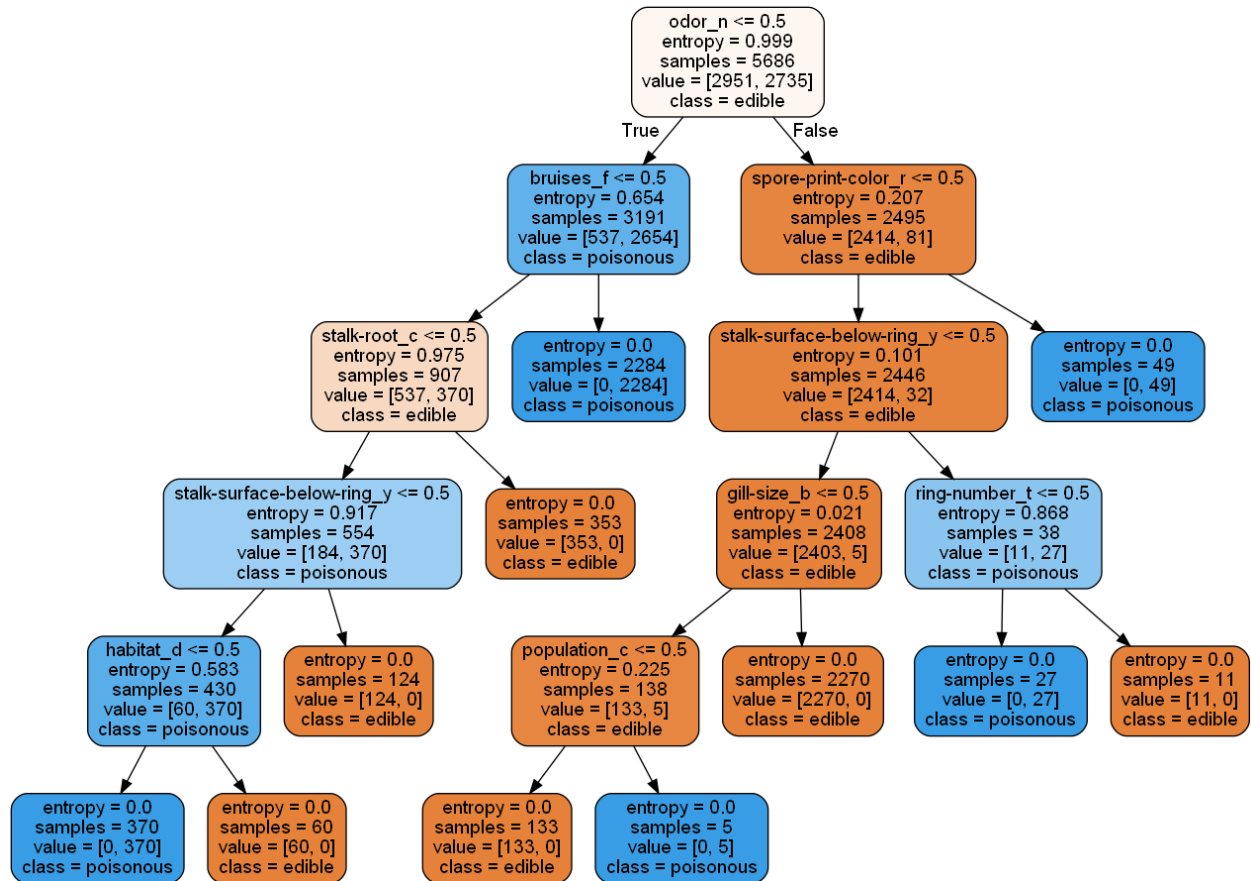
Just like Naive Bayes, this code for Decision Tree also creates a csv file of the predicted values for the test dataset. In addition to the csv file it also creates a png image showing the resulting decision tree.

The csv file named “Decision_Tree_Prediction_Result” can be found under the folder “results” and the decision tree image can be found under the folder “image” with name “Decision_Tree_Image”.

Snippet from Decision_Tree_Prediction_result.csv

	A	B	C	D	E	F	G	H	I	J	K
1	0										
2											
3	1										
4											
5	1										
6											
7	0										
8											
9	1										
10											
11	1										
12											
13	1										
14											
15	1										

Snippet from Decision_Tree_Image.png



For visualization of decision tree, Graphviz tool has been used. Each node contains the column or attribute name, entropy value, number of samples and the resulting class. The blue nodes indicate that the mushroom is of class “poisonous” whereas the orange nodes represent “edible” mushrooms.

Few points on K-fold Cross-validation method

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k -fold cross-validation. In this project the value of k used is 10.

Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.

It is a popular method because it is simple to understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split.

The general procedure is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
 1. Take the group as a hold out or test data set
 2. Take the remaining groups as a training data set
 3. Fit a model on the training set and evaluate it on the test set
 4. Retain the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores

What is a confusion matrix ?

A Confusion matrix is an $N \times N$ matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives a holistic view of how well our classification model is performing and what kinds of errors it is making.

For a binary classification problem, it would consist of a 2×2 matrix as shown below with 4 values:

		ACTUAL VALUES	
		POSITIVE	NEGATIVE
PREDICTED VALUES	POSITIVE	TP	FP
	NEGATIVE	FN	TN

- The target variable has two values: Positive or Negative
- The columns represent the actual values of the target variable
- The rows represent the predicted values of the target variable

What is TP, TN, FP, FN ?

True Positive (TP)

- The predicted value matches the actual value
- The actual value was positive and the model predicted a positive value

True Negative (TN)

- The predicted value matches the actual value
- The actual value was negative and the model predicted a negative value

False Positive (FP) – Type 1 error

- The predicted value was falsely predicted
- The actual value was negative but the model predicted a positive value
- Also known as the Type 1 error

False Negative (FN) – Type 2 error

- The predicted value was falsely predicted
- The actual value was positive but the model predicted a negative value
- Also known as the Type 2 error

Confusion matrix for Naive Bayes :

PREDICTED VALUES	ACTUAL VALUES		
		POSITIVE	NEGATIVE
	POSITIVE	1171	86
	NEGATIVE	1	1180

Confusion matrix for Decision Tree :

PREDICTED VALUES	ACTUAL VALUES		
		POSITIVE	NEGATIVE
	POSITIVE	1257	0
	NEGATIVE	0	1181

Comparison of Naive Bayes and Decision Tree classifier of the mushroom dataset

Parameter Name	Value for Naive Bayes	Value for Decision Tree
Mean Training Accuracy	96.34	100
Mean Training Precision	93.14	100
Mean Training Recall	99.91	100
Mean Training F1 Score	96.38	100
Mean Validation Accuracy	91.47	96.85
Mean Validation Precision	90.53	100
Mean Validation Recall	93.34	93.46
Mean Validation F1 Score	90.01	95.15

Conclusion :

The Decision Tree classifier performed better than Naive Bayes Classifier for the chosen dataset.

