Data Science & Artificial Intelligence Final Examination Alternative Continuous Assessment

Student Name: Molin Sun

Student Number: C00266170

1 Machine Learning

1.1 Introduction

Machine learning is a branch of artificial intelligence and computer science.It is to study and construct an algorithm that allows computers to learn from data itself to make predictions. In other words, a user can input a large amount of data to a computer algorithm and then the computer can analyze these data and make data-driven recommendations and decisions based on the input data. For example, write a program that can recognize animals according to their facial features and body type.[1]

1.2 How machine learning works

Machine learning mainly consists of the following three parts [1]:

- A decision process: a computational method or other steps that takes data to make a prediction or classification and return a predicted outcome about the type of the data that the algorithm is looking for.
- An error function: to measure the accuracy of predicted results.
- An updating or optimization process: update and optimize the decision process according to errors, so that the next time the errors are less serious.

1.3 Machine learning methods

Supervised learning

Supervised learning [2] is to use labeled data to train algorithms to categorize data or predict results. When input data passes the model, it will adjust its weights until the model is fitted properly. This is a part of the cross-validation process which can avoid overfitting or underfitting of the model. The commonly used supervised learning algorithm are neural networks, native bayes, random forest, support vector machine(SVM),etc.

Supervised learning can solve the following two types of problem:

- Regression problems: to predict future values and the model was trained by historic data
- Classification problems: to recognize the category of target items.

Unsupervised learning

In unsupervised learning [2], input data are not labeled and there is no accurate outcome. Because unsupervised learning is to find underlying phenomena by input data set. Unsupervised learning can solve the following two kind of problem:

- Clustering: bundle input variables with the same characteristics. For example, a user reads a news article and then the software will recommend the same type of news to the user.

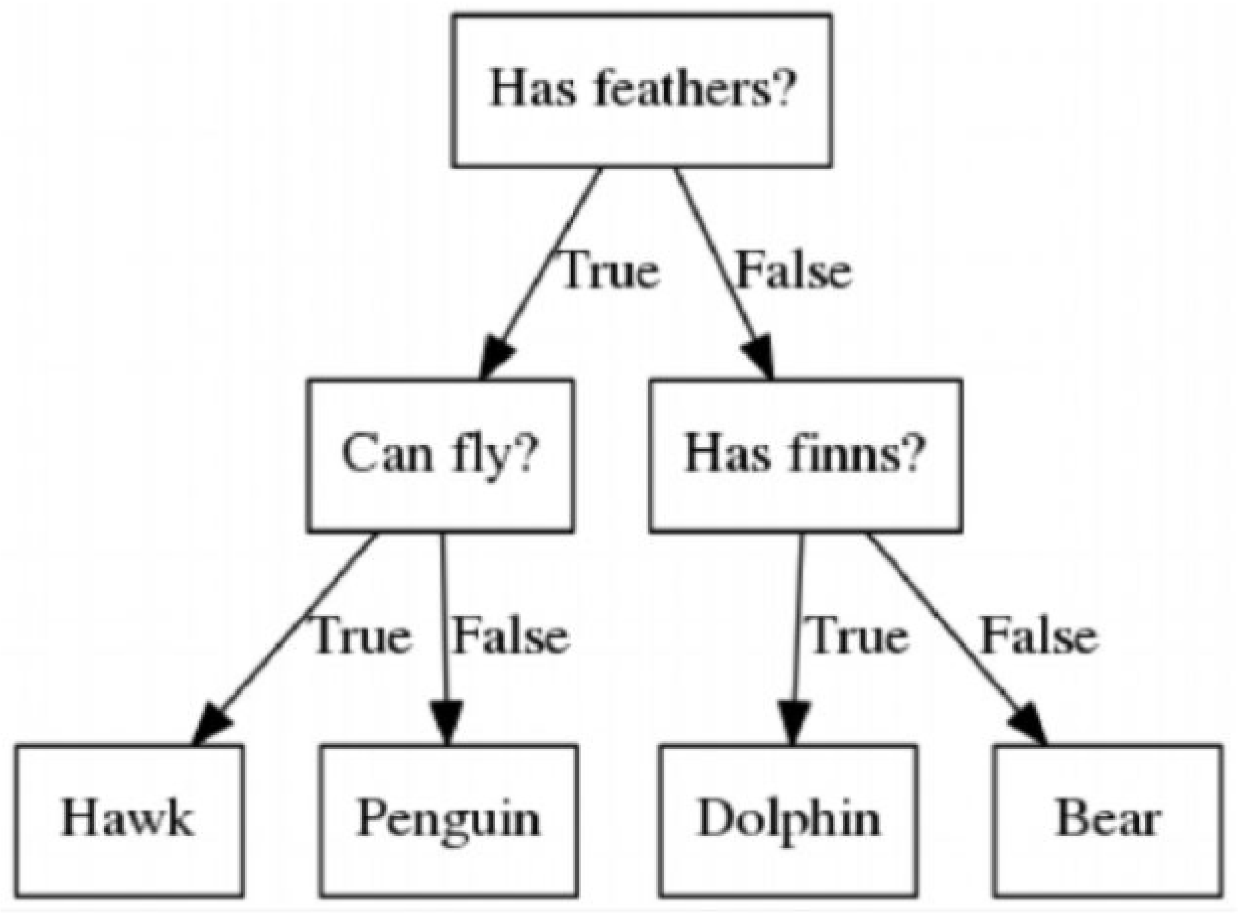
- Association: discover meaningful associations among the dataset. For example, a user buys a desk on a shopping app, and then the app will recommend chairs to the user.

2 Decision Trees

2.1 Introduction

- Decision trees are a supervised learning method that creates a tree-like model to divide a large heterogeneous data set into smaller data sets based on different conditions.
- Decision trees are used for classification and regression
- The goal of Decision trees is to create a model that can predict the value of the target variable by learning decision rules.
- The figure shown below is an example of decision tree

animal_tree



- For this simple example, it uses some rules to predict the species of animals. The model uses 3 attributes from the data set, namely feather, fly and finn.

- As shown above, a decision tree is drawn upside down with its root at the top. The sentences with question marks represent conditions or internal node to split a node into branches. And the end of branch is leaf node representing class label.

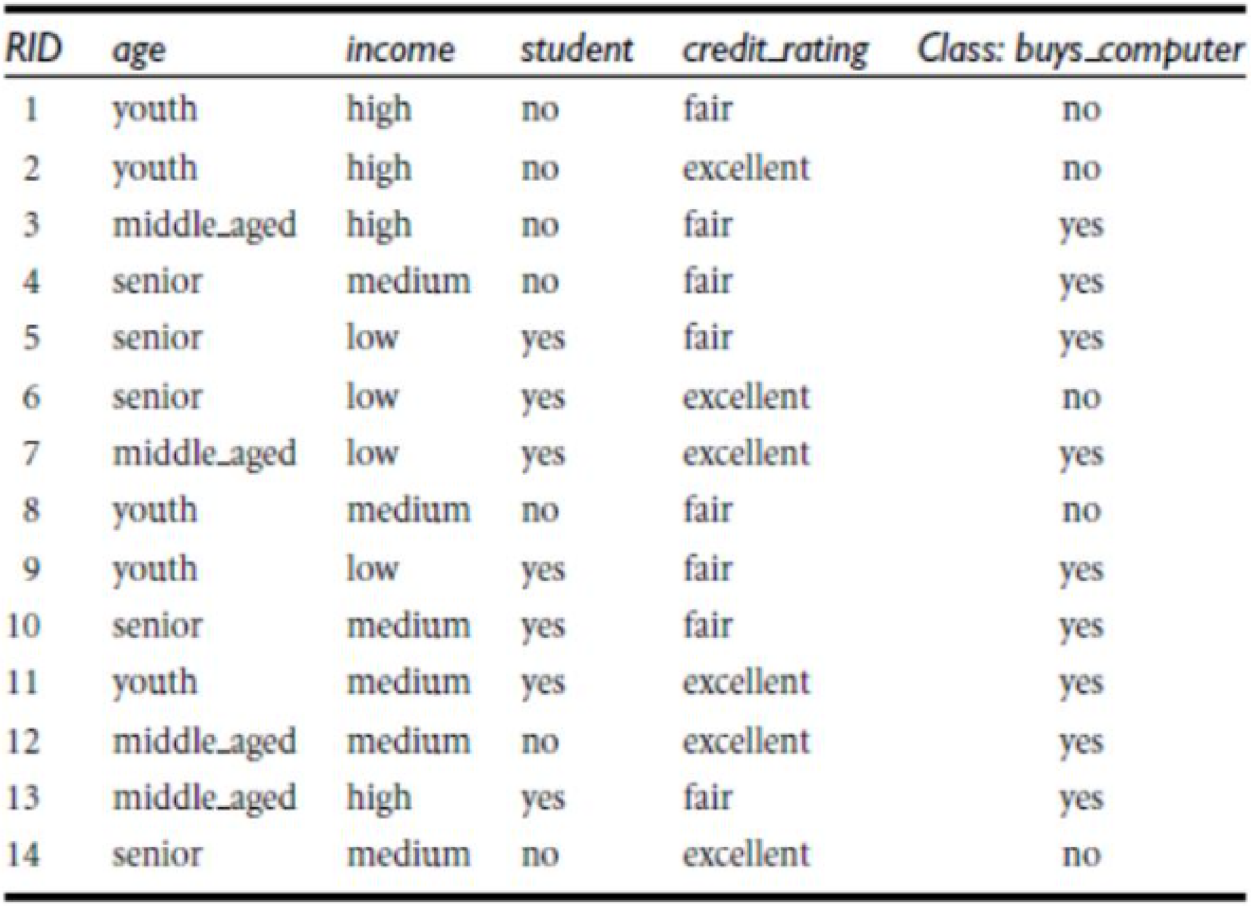
2.2 ID3 algorithm

2.2.1 Introduction

- Is an algorithm used to generate a decision tree from a data set in a decision tree.
- Is used to select the best attributes which can best split the data set.
- Is a greedy algorithm.
- The steps of the algorithm:
    - select the best attribute using the decreasing speed of information entropy as the standard
    - Use the selected best attributes as the decision node and use the different value of the attributes to split remaining instances
    - Repeat the previous step recursively for each child node
    - If all instances are perfectly classified, stop the iteration.

2.2.2 Entropy

- Entropy is a measure of order in a system. The more ordered a system is, the lower entropy the system has.
    - Expression: Entropy(S) = ∑(i=1 to l)-|Si|/|S| * log2(|Si|/|S|)
    - S is the set of examples
    - Si is a subset of S whose value is vi under the target attribute
    - l is the size of the range of the target attribute
- Example
    - In the example dataset, there is one special attribute(whether the person will buy the computer) and four regular attributes(age, income, student and credit).



RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

    - Calculate entropy : as shown in the table, there are 14 examples, 9 positive and 5 negatives.So the entropy of S relative to this classification is
        - Entropy(S) = -(9/14)log2(9/14)-(5/14)log2(5/14) = 0.9403
    - Calculate entropy of other attributes using expression :
        - ∑(i=1 to k)|Si|/|S|Entropy(Si) (k refers to the range of the attribute we are testing)

    - Age
        - Entropy(young) = -(2/5)log2(2/5)-(3/5)log2(3/5) = 0.970
        - Entropy(medium) = -(4/4)log2(4/4)-(0/4)log2(0/4) = 0
        - Entropy(old) = -(3/5)log2(3/5)-(2/5)log2(2/5) = 0.970
        - Entropy(Age|S) = 5/14_Entropy(young) + 4/14_Entropy(medium)+5/14*Entropy(old) = 0.6935
    - Income
        - Entropy(high) = -(2/4)log2(2/4)-(2/4)log2(2/4) = 1
        - Entropy(medium) = -(4/6)log2(4/6)-(2/6)log2(2/6) = 0.9183
        - Entropy(low) = -(3/4)log2(3/4)-(1/4)log2(1/4) = 0.8113
        - Entropy(Income|S) = 4/14_Entropy(high) + 6/14_Entropy(medium)+4/14*Entropy(low) = 0.9111
    - Student
        - Entropy(yse) = -(6/7)log2(6/7)-(1/7)log2(1/7) = 0.5917
        - Entropy(no) = -(4/7)log2(4/7)-(3/7)log2(3/7) = 0.9852
        - Entropy(Student|S) = 7/14_Entropy(yes) + 7/14_Entropy(no) = 0.7885
    - Credit
        - Entropy(good) = -(6/8)log2(6/8)-(2/8)log2(2/8) = 0.8113
        - Entropy(excellent) = -(3/6)log2(3/6)-(3/6)log2(3/6) = 1
        - Entropy(Credit|S) = 8/14_Entropy(good) + 6/14_Entropy(excellent)=0.8922

2.2.3 Information Gain

- The best attribute has the greatest reduction in entropy. So information gain is the expected reduction in entropy using A attribute to split the data set.

$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- The expression shown above equal: Gain(S，A) = Entropy(S) - Entropy(A|S)
- Calculate Information gain(using the example mentioned above)
    - Gain(S,Age)= Entropy(S) - Entropy(Age|S) = 0.9403 - 0.6935 = 0.2468
    - Gain(S,Income)= Entropy(S) - Entropy(Income|S) = 0.9403 - 0.9111 = 0.0292
    - Gain(S,Student)= Entropy(S) - Entropy(Student|S) = 0.9403 - 0.7885 = 0.1518
    - Gain(S,Credit)= Entropy(S) - Entropy(Credit|S) = 0.9403 - 0.8922 = 0.0481
- As a result, age is the best attribute to split the data set.

2.3 Implementation

2.3.1 Dataset

- Titanic[3] is the dataset we use in this part. This dataset is from the sinking of the Titanic. We know that the sinking of the Titanic is one of the world's worst maritime accidents, resulting in a large number of casualties. We will use this dataset to do survivor prediction.
- Size of dataset: 891
- Number of attributes: 12
    - PassengerId
    - Survived: 0 means not survived and 1 means survived
    - Pclass: ticket class, 1 means 1st, 2 means 2nd and 3 means 3rd
    - Name
    - Sex

- Age: age in years
- SibSp: siblings or spouses aboard the Titanic
- Parch: parents or children aboard the Titanic
- Ticket:ticket number
- Fare: passenger fare
- Cabin: cabin number
- Embarked: port of embarkation. C means Cherbourg, Q means Queenstown and S means Southampton

2.3.2 Process

- Data preprocessing
- Model building
- Check the model we build on the test set [4]

1. Firstly, we start with some basic understanding of the dataset.

In [1]:

```python
import pandas as pd
import numpy as np

# read data
data=pd.read_csv('train.csv')
data.head()
```

Out[1]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [2]:

```python
data.columns
```

Out[2]:

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

1. We need to preprocess the dataset.

In [3]:

```python
# Delete unnecessary features
del data['PassengerId']
del data['Name']
del data['Ticket']
del data['Cabin']
```

In [4]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Survived  891 non-null    int64  
 1   Pclass    891 non-null    int64  
 2   Sex       891 non-null    object 
 3   Age       714 non-null    float64
 4   SibSp     891 non-null    int64  
 5   Parch     891 non-null    int64  
 6   Fare      891 non-null    float64
 7   Embarked  889 non-null    object 
dtypes: float64(2), int64(4), object(2)
memory usage: 55.8+ KB
```

In [5]:

```
# check whether data is missing
data.isnull().sum()
```

Out[5]:

```
Survived      0
Pclass        0
Sex           0
Age         177
SibSp         0
Parch         0
Fare          0
Embarked      2
dtype: int64
```

According to the result show above, the columns 'Age' and 'Embarked' have missing value. So we need to fill these two columns.For column 'Age', we use mean padding. And for column 'Embarked', there are only two missing values. So we just delete these two samples.

In [6]:

```
data['Age']=data['Age'].fillna(data['Age'].mean())
data=data.dropna()
```

For ease of operation, we change the columns 'Sex' and 'Embarked' to numeric type.

In [7]:

```
data['Sex']=data['Sex'].map({'male':1,'female':0})
labels = data["Embarked"].unique().tolist()
data["Embarked"] = data["Embarked"].apply(lambda x: labels.index(x))
```

In [7]:

```
data.head()
```

Out[7]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	22.0	1	0	7.2500	0
1	1	1	0	38.0	1	0	71.2833	1
2	1	3	0	26.0	0	0	7.9250	0
3	1	1	0	35.0	1	0	53.1000	0
4	0	3	1	35.0	0	0	8.0500	0

In [8]:

```
data.isnull().sum()
```

Out[8]:

```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    0
dtype: int64
```

1. Build the model

In [9]:

```
# Extract the label and feature matrix
data_x = data.drop(['Survived'], axis=1)
data_y = data['Survived']
```

In [10]:

```
# Modified index
for i in [data_x,data_y]:
    i.index = range(i.shape[0])
```

In [11]:

```
# Cross validation
from sklearn import tree
from sklearn.model_selection import cross_val_score

clf = tree.DecisionTreeClassifier(criterion="entropy",random_state=25)
cross_val_score(clf, data_x, data_y, cv=10).mean()
```

Out[11]:

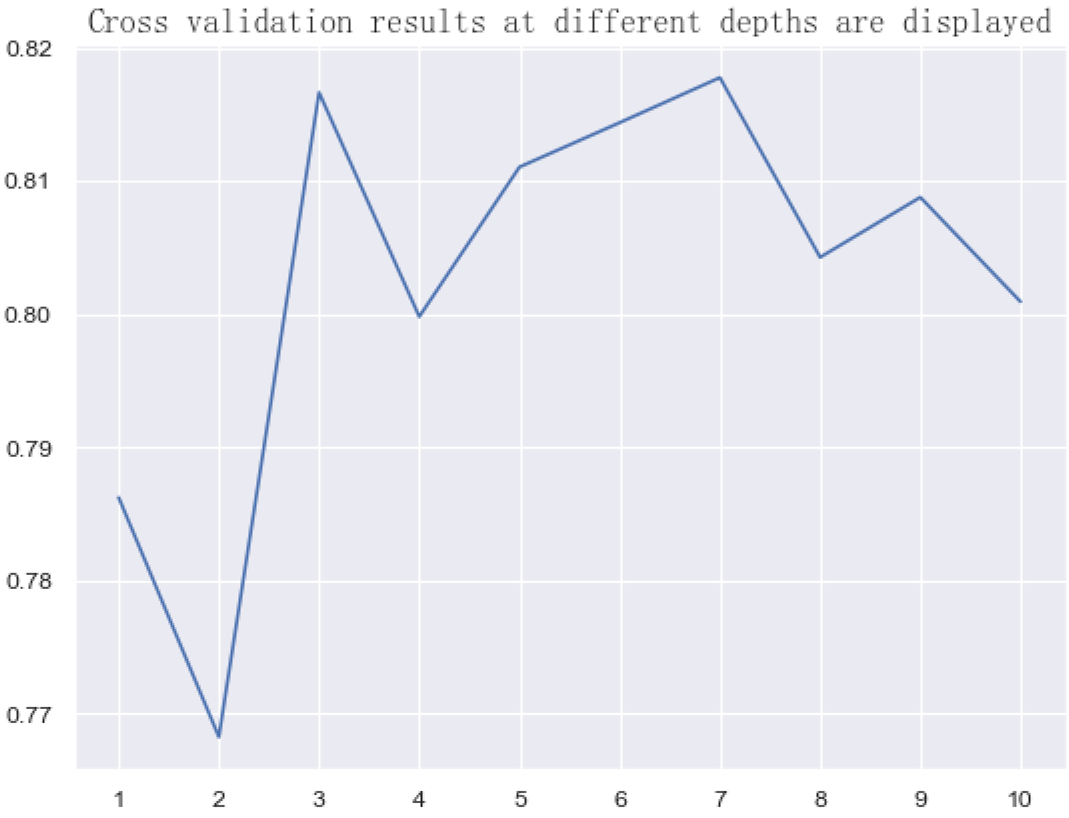
```
0.7818181818181819
```

The code below is to check the fitting status of the model observed at different max_depth. And we need to take out the highest value.

In [12]:

```
import matplotlib.pyplot as plt
%matplotlib inline
import matplotlib
import seaborn as sns
sns.set()
import warnings
warnings.filterwarnings('ignore')
plt.figure(figsize=(8,6),dpi=80,num=4)
myfont = matplotlib.font_manager.FontProperties(fname='C:\Windows\Fonts\simsun.ttc')
te = []
for i in range(10):
    clf = tree.DecisionTreeClassifier(random_state=25,max_depth=i+1,criterion="entropy")
    score_te = cross_val_score(clf, data_x, data_y, cv=10).mean()
    te.append(score_te)
print("The highest accuracy occurred in the {}, the highest value{:.2%}".format(te.index(m
plt.plot(range(1,11),te)
plt.xticks(range(1,11))
plt.title('Cross validation results at different depths are displayed',FontProperties=myf
plt.show()
```

```
The highest accuracy occurred in the 7, the highest value81.78%
```

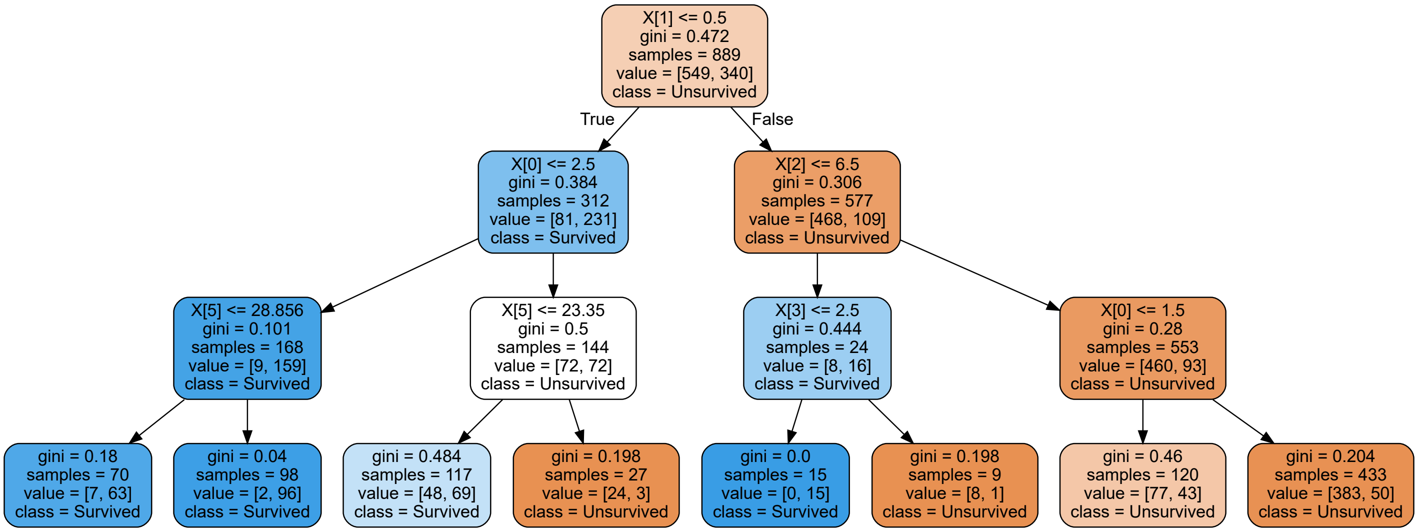


And then we set the max_depth to 3 and draw a simple decision tree.

In [13]:

```
import graphviz
clf = tree.DecisionTreeClassifier(random_state=25,min_samples_leaf= 6,splitter= 'best',max
clf = clf.fit(data_x,data_y)
dot_data = tree.export_graphviz(clf
,out_file = None
# ,feature_names= feature_name
,class_names=["Unsurvived","Survived"]
,filled=True
,rounded=True
)
graph = graphviz.Source(dot_data)
graph
```

Out[13]:



1. Test the model with the test set.

In [17]:

```
# import the tase sett and preprocess it
data_test = pd.read_csv('test.csv')
```

```
data_test.drop(["PassengerId","Cabin","Name","Ticket"],inplace=True,axis=1)
data_test["Age"] = data_test["Age"].fillna(data_test["Age"].mean())
data_test = data_test.dropna()
data_test["Sex"] = (data_test["Sex"]== "male").astype("int")
labels = data_test["Embarked"].unique().tolist()
data_test["Embarked"] = data_test["Embarked"].apply(lambda x: labels.index(x))
```

In [18]:

```
# Use the predict interface to predict the test set
data_test['Survived']=pd.DataFrame(clf.predict(data_test))
```

In [19]:

```
# Check the prediction result of the model in the test set
data_survive = data_test['Survived'].value_counts(normalize=True)
n0_sample = data_survive[0]
n1_sample = data_survive[1]
print('death rate{:.2%}; survived rate{:.2%}'.format(n0_sample,n1_sample))
```

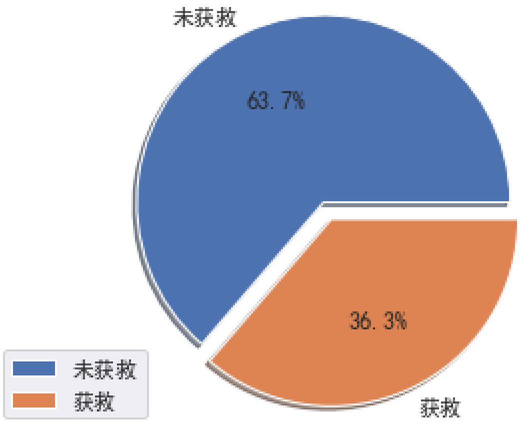
```
death rate63.70%; survived rate36.30%
```

In [20]:

```
# Visualize
data_counts = data_test['Survived'].value_counts()
plt.rcParams['font.sans-serif']=['SimHei']
plt.pie(data_counts, explode=(0, 0.1),labels=['未获救','获救'],
         shadow=True, autopct='%1.1f%%')
plt.axis('equal')
plt.legend()
```

Out[20]:

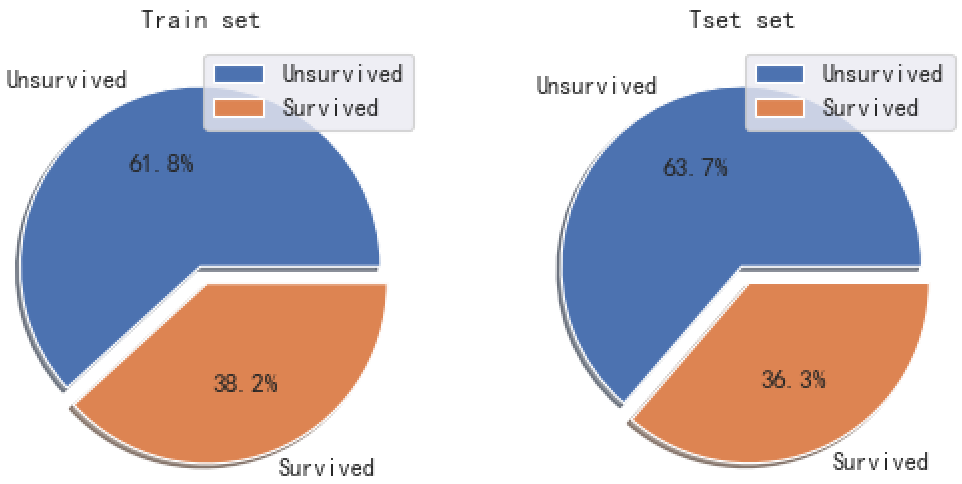
```
<matplotlib.legend.Legend at 0x1ed47feb710>
```



In [21]:

```
# Visualize the result of train set and test set
plt.figure(figsize=(8,6),dpi=80,num=4)
df_count = data['Survived'].value_counts()
plt.rcParams['font.sans-serif']=['SimHei']
plt.subplot(1,2,1)
plt.pie(df_count, explode=(0, 0.1),labels=['Unsurvived','Survived'],
         shadow=True, autopct='%1.1f%%')
plt.title('Train set')
plt.legend()

df_counts = data_test['Survived'].value_counts()
plt.rcParams['font.sans-serif']=['SimHei']
plt.subplot(1,2,2)
plt.pie(df_counts, explode=(0, 0.1),labels=['Unsurvived','Survived'],
         shadow=True, autopct='%1.1f%%',)
plt.title('Tset set')
plt.legend()
```

```
Out[21]: <matplotlib.legend.Legend at 0x1ed4a313c88>
```



According to the graph show above, The prediction result of the model in the test set is close to the result in the train set, which means that the prediction result of our model is reliable.

2.4 Application

- In real life, if we encounter some complicated or uncertainty factors, we can use the decision tree to help us make scientific, objective and rational decisions. the decision tree algorithm is also used in many apps. For example, decision trees can be used to predict user's preferences in some social apps.

2.5 Conclusion

- Decision tree algorithm mainly refers to the algorithm that selects the optimal feature when splitting the dataset. Its main purpose is to select the best feature that can separate the data as neatly as possible. In other words, make unordered data more ordered.
- Advantages
  - Decision trees is easy to understand, interpret and visualize
  - Work well on discrete and categorical variables
- Disadvantages
  - When a learner creates an over-complex tree, which will lead to overfitting.

3 Bayesian Classifier

3.1 Introduction

3.1.1 Conditional Probability

Conditional probability[4]is the probability of event A happening when event B has already happened.

- $P(A \mid B) = P(AB) / P(B) = P(A \mid B)\ \mathit{P(B) / P(B) = P(B \mid A)}\ P(A) / P(B)$

3.1.2 Bayes Theorem

The theory behind Bayesian algorithm is Bayes Theorem. So let's start with Bayes theorem [5].

- $P(H \mid X) = P(X \mid H) * P(H) / P(X)$
- H and E are events and $P(H)!=0$
- $P(H)$ is the possibility of event H happening

- P(X) is the possibility of event X happening
- P(H | X) is the conditional probability of H happening given X as ture
- P(X | H) is the conditional probability of X happening given H as ture

And the names of terms used in Bayes Theorem equations depend on the context in which the equation is used.

- P(H | X) is posterior probability
- P(H) refers to class prior probability
- P(X | H) is likelihood
- P(X) is prediction prior probability

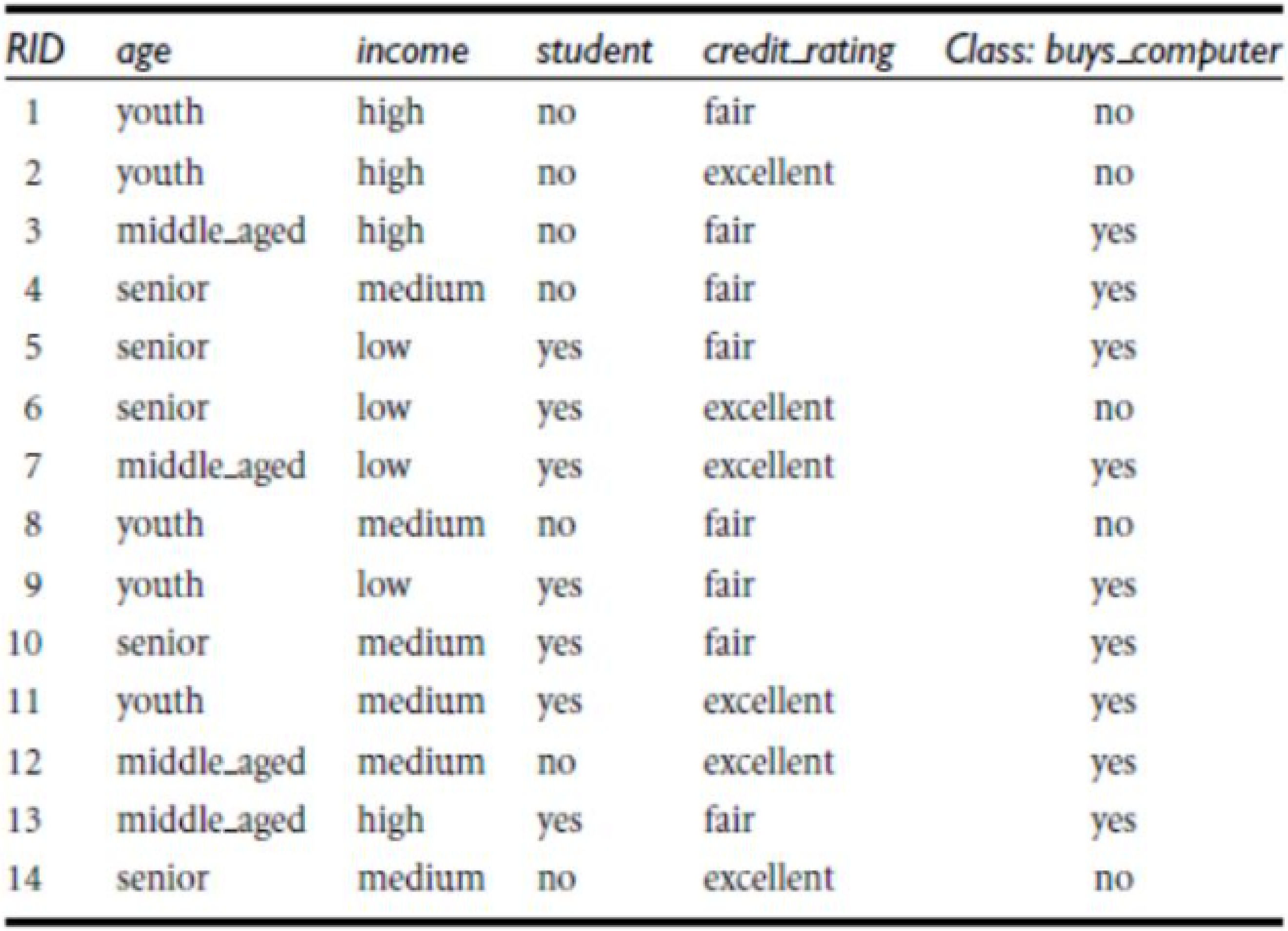
3.1.3 Bayes Classifier

- Bayes algorithm is a supervised learning algorithm.
- Bayes classifier is a probabilistic machine learning model for classification model for classification task. And it is based on Bayes theorem.
- Bayes classifier is a simple and effective classification algorithm, which is helpful for building the fast machine learning models that can make quick predictions. [5]

3.2 How does Bayesian Classifier work

In this part, we will have an example that can help us to know how Bayes classifiers work [6].

And the example dataset is show below:

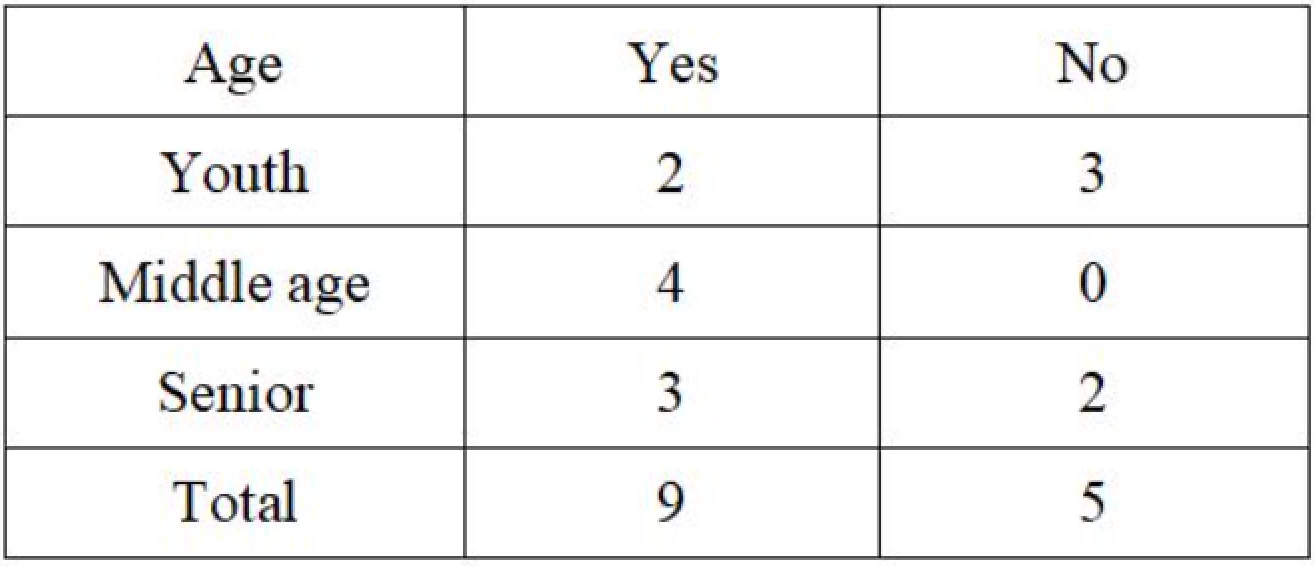


RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

So according to the dataset, will young people buy a computer? How to solve this problem. We need to following steps:

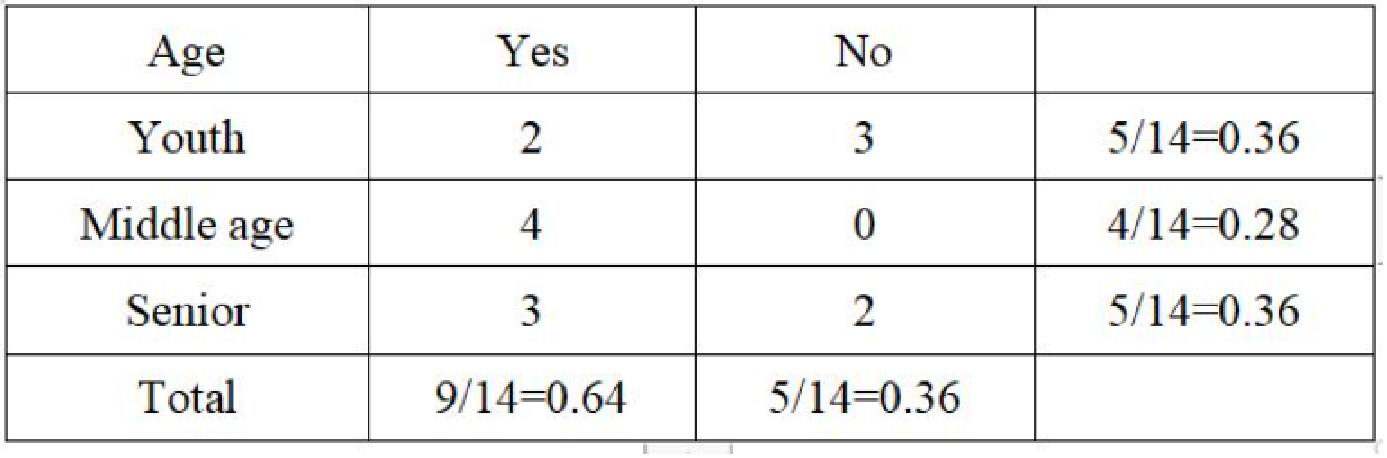
- Converts a given data set to a frequency table
- Generate a likelihood table by looking up probabilities of given features
- use Bayes theorem to calculate the posterior probability

The frequency table for age conditions.



Age	Yes	No
Youth	2	3
Middle age	4	0
Senior	3	2
Total	9	5

Likelihood table for age condition.



Age	Yes	No	
Youth	2	3	5/14=0.36
Middle age	4	0	4/14=0.28
Senior	3	2	5/14=0.36
Total	9/14=0.64	5/14=0.36	

And then using the Bayes theorem to calculate the posterior probability.

- P(Yes | Youth) = P(Youth | Yes) * P(Yes) / P(Youth)
  - P(Youth | Yes) = 2/9 = 0.22
  - P(Yes) = 0.64
  - P(Youth) = 0.36
  - So P(Yes | Youth) = 0.22*0.64/0.36 = 0.39
- P(No | Youth) = P(Youth | No) * P(No) / P(Youth)
  - P(Youth | No) = 3/5 = 0.6
  - P(No) = 0.36
  - P(Youth) = 0.36
  - So P(Yes | Youth) = 0.6*0.36/0.36 = 0.6

As we can see that P(Yes | Youth) < P(No | Youth). Hence, young people don't buy a computer.

3.3 Implementation

3.3.1 Dataset

- The dataset we use in this implementation is Iris Species[7]. This dataset includes three iris species and 50 samples of each. We need to use features of iris to predict the species of iris.
- Size of dataset: 150
- Numbers of attributes: 6
    - Id
    - SepalLengthCm: the length of sepal
    - SepalWidthCm: the width of sepal
    - PetalLengthCm: the length of petal
    - PetalWidthCm: the width of petal
    - Species: Iris-setosa, Iris-versicolor, Iris-virginica

3.3.2 Process

- Data preprocessing
- Train the model
- Predict
- Check the performance of the model on test set [8]

1.Data preprocessing:

Firstly, we need to import all the libraries we need and then read the data.

In [6]:

```
# Import libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns

iris = pd.read_csv("Iris.csv")
iris.head()
```

Out[6]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Before we build the Bayes classifier model, we need to do some preprocessing. The code below we divide the dataset into train and test sets by using the train_test_split method of sklearn. And the test_size means that we use 25% of the data to test.

In [7]:

```
x = iris.drop('Species',axis=1)
y = iris['Species']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=42)
```

2.Train the model. we use Gaussian Native Bayes.

In [8]:

```
model = GaussianNB()
model.fit(x_train, y_train)
```

Out[8]:

```
GaussianNB()
```

3.Predict. The model has been trained. We can use the model to predict and pass x_test as parameters to get the output pred.

In [9]:

```
pred = model.predict(x_test)
pred
```

Out[9]:

```
array(['Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa'], dtype='<U15')
```

Finally, we check the performance of the model on the teat set. The output accuracy represents the accuracy of the model.

In [10]:

```
accuracy = accuracy_score(y_test,pred)*100
accuracy
```

Out[10]:

```
100.0
```

3.4 Application

- text classification
- spam filtering

3.5 Conclusion

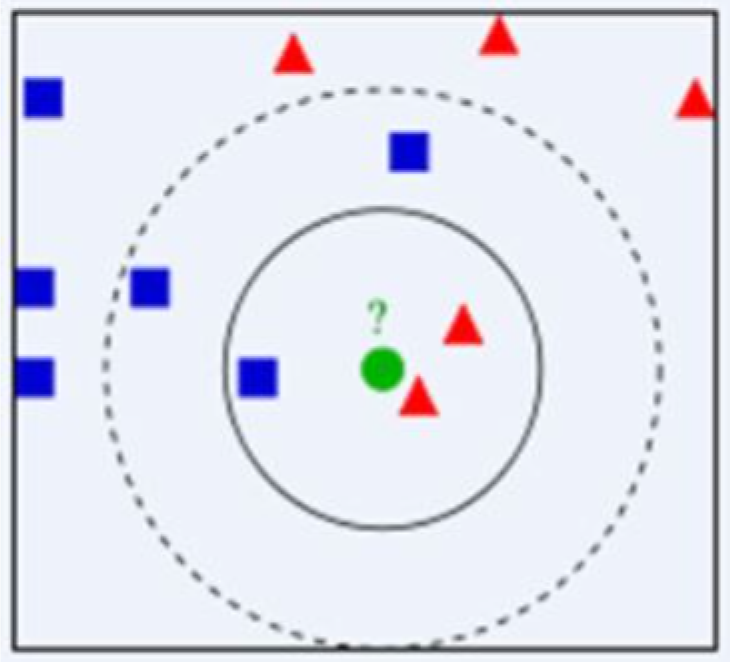
- Advantages
  - Bayes classifier algorithm is fast and easy to predict
  - It can be used for binary and multi-class classification.
  - It can solve text classification problems well.
- Limitation
  - Bayes algorithm only holds that all features are independent or unrelated. So it cannot cope with problems that have related features.

4 K-Nearest Neighbour Algorithm

4.1 Introduction

- K-Nearest Neighbour algorithm is a classification algorithm based on supervised learning.
- K-NN classify items by measuring the distance between features

- For example, as shown in the following figure, which class does the green circle belong to. In K-NN, we assign it to a group by looking at which group is its nearest neighbors.



4.2 How does K-NN work

The process of K-NN algorithm is shown below [9]:

- Choose the value of K
- Calculate the Euclidean distance between the unknown instances and known instances
- Select the K nearest known instances
- According to the rule of majority voting, let the unknown instance be classified as the most majority of the K nearest neighbor samples. The specific meaning of K value is K nearest neighbor data samples.

4.2.1 How to choose the value of K

The selection of K value is key to the K-NN algorithm, because its value has a significant impact on the result of the algorithm. Generally, cross validation is used to select the optimal K value [9].

4.2.2 Euclidean distance

Assume that there are two data points, x=(x1,x2,...,xn) and y(y1,y2,...,yn). The Euclidean distance between x and y is [9]:

$$d(x,y)=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\cdots+(x_n-y_n)^2}$$

4.3 Implementation

4.3.1 Dataset

This dataset[10] is for glass identification. The task in this implementation is to use 9 attributes which are included in the dataset to predict the classes of glass. Number of attributes: 10

- RI: refractive index
- Na: Sodium

- Mg: Magnesium
- Al: Aluminum
- Si: Silicon
- K: Potassium
- Ca: Calcium
- Ba: Barium
- Fe: Iron
- Type of glass:
    - 1 -> buildingwindowsfloatprocessed
    - 2 -> buildingwindowsnonfloatprocessed
    - 3 -> vehiclewindowsfloatprocessed
    - 4 -> vehiclewindowsnonfloatprocessed (none in this database)
    - 5 -> containers
    - 6 -> tableware
    - 7 -> headlamps

4.3.2 Process

- Read dataset
- Normalize the data
- Divide the dataset into train set and test set
- Train the model
- Predict
- Evaluate the accuracy of prediction
- Calculate the best K values
- Adjust K value and retrain model [11]

1. Firstly, we import the necessary library and read the dataset.

In [20]:

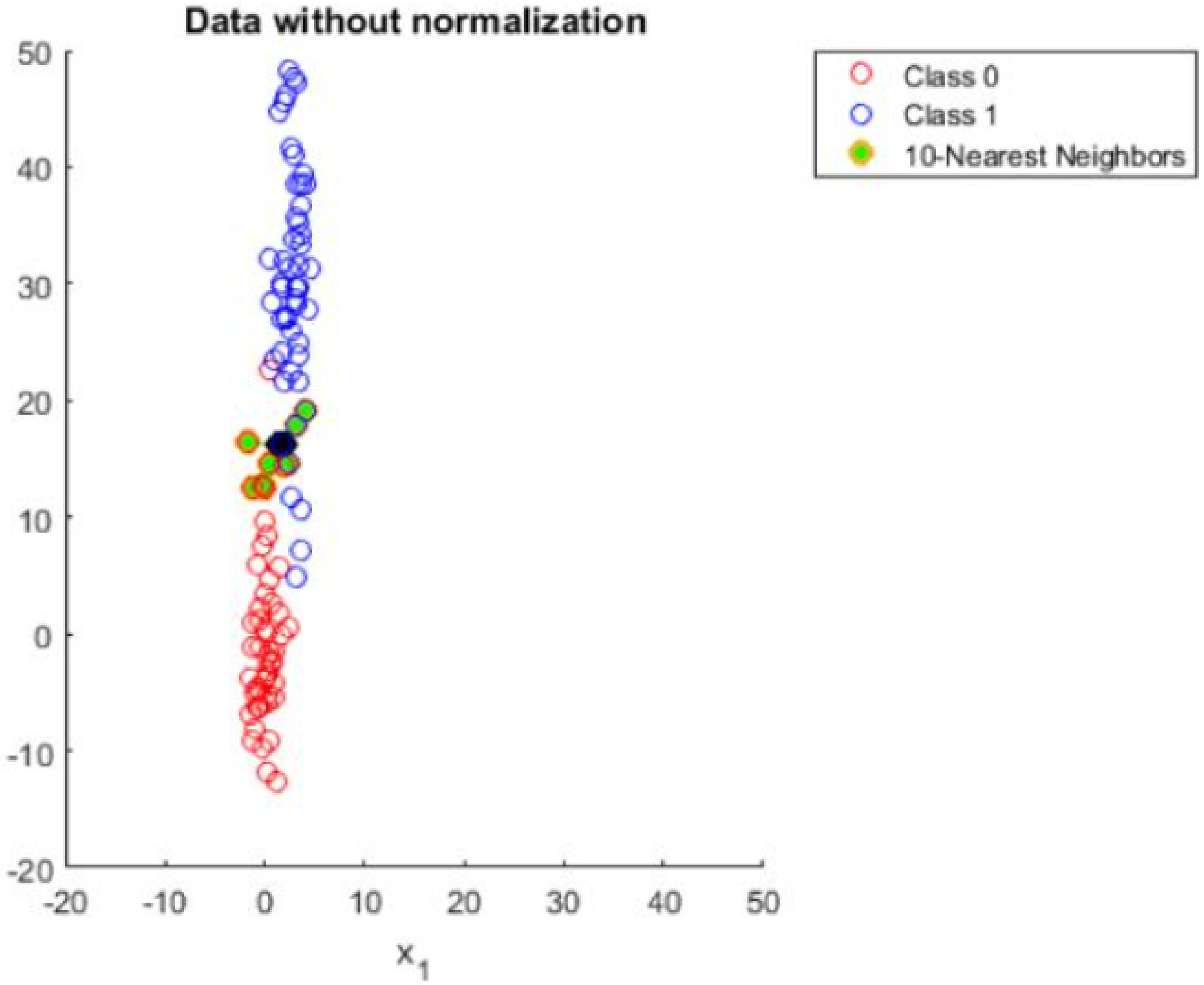
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

df = pd.read_csv('glass.csv')
df.head()
```

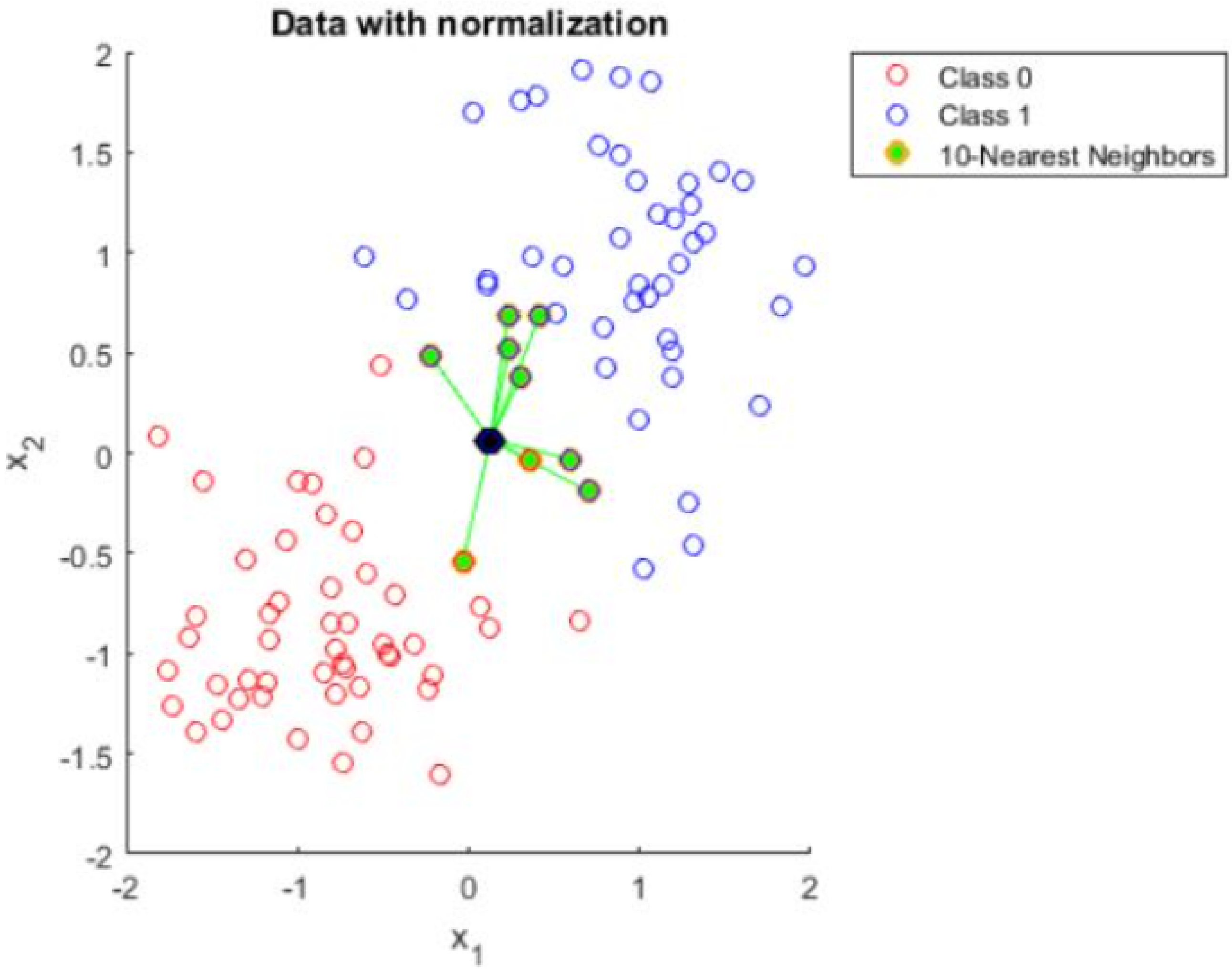
Out[20]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0	1

1. Next, we need to scale the data. So why do we need to normalize data in the KNN? Because the KNN algorithm needs to calculate the Euclidean distance between the samples, the larger the range of a feature will affect the prediction. This explanation is abstract and difficult to understand. The figures shown below can help us to understand. In the first picture, the data is not normalized. But in the second picture, the data

is normalized. We can see in the first picture all of the nearest neighbors are aligned along the narrow axis, which leads to the wrong classification [12].


Data without normalization
50
40
30
20
10
0
-10
-20
-20
-10
0
10
20
30
40
50
x_1
Class 0
Class 1
10-Nearest Neighbors



In [52]:

```
# Import module to standardize the scale
from sklearn.preprocessing import StandardScaler

# Create instance (i.e. object) of the standard scaler
scaler = StandardScaler()

# Fit the object to all the data except the Target Class
# use the .drop() method to gather all features except Target Class
# axis -> argument refers to columns; a 0 would represent rows
scaler.fit(df.drop('Type', axis=1))
```

Out[52]:

```
StandardScaler()
```

In [37]:

```
# Use scaler object to conduct a transforms
scaled_features = scaler.transform(df.drop('Type',axis=1))

# Review the array of values generated from the scaled features process
scaled_features
```

Out[37]:

```
array([[ 0.87286765,  0.28495326,  1.25463857, ..., -0.14576634,
        -0.35287683, -0.5864509 ],
       [-0.24933347,  0.59181718,  0.63616803, ..., -0.79373376,
        -0.35287683, -0.5864509 ],
       [-0.72131806,  0.14993314,  0.60142249, ..., -0.82894938,
        -0.35287683, -0.5864509 ],
       ...,
       [ 0.75404635,  1.16872135, -1.86551055, ..., -0.36410319,
         2.95320036, -0.5864509 ],
       [-0.61239854,  1.19327046, -1.86551055, ..., -0.33593069,
         2.81208731, -0.5864509 ],
       [-0.41436305,  1.00915211, -1.86551055, ..., -0.23732695,
         3.01367739, -0.5864509 ]])
```

In [38]:

```
df_frame = pd.DataFrame(scaled_features, columns = df.columns[:-1])
df_frame.head()
```

Out[38]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe
0	0.872868	0.284953	1.254639	-0.692442	-1.127082	-0.671705	-0.145766	-0.352877	-0.586451
1	-0.249333	0.591817	0.636168	-0.170460	0.102319	-0.026213	-0.793734	-0.352877	-0.586451
2	-0.721318	0.149933	0.601422	0.190912	0.438787	-0.164533	-0.828949	-0.352877	-0.586451
3	-0.232831	-0.242853	0.698710	-0.310994	-0.052974	0.112107	-0.519052	-0.352877	-0.586451
4	-0.312045	-0.169205	0.650066	-0.411375	0.555256	0.081369	-0.624699	-0.352877	-0.586451

1. Split the dataset into a train set and a test set. We use 30% of data to test.

In [39]:

```
# Import module to split the data
from sklearn.model_selection import train_test_split
# Set the X and ys
x = df_frame
y = df['Type']
# Use the train_test_split() method to split the data into respective sets
# test_size -> argument refers to the size of the test subset
# random_state -> argument ensures guarantee that the output of Run
# 1 will be equal to the output of Run 2, i.e. your split will be always the same
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=101)
```

1. Create the KNN model and use fit() method to train the model.

In [65]:

```
# Import module for KNN
from sklearn.neighbors import KNeighborsClassifier

# Create KNN instance
# n_neighbors -> argument identifies the amount of neighbors used to ID classification
knn = KNeighborsClassifier(n_neighbors=5)

# Fit (i.e. traing) the model
knn.fit(x_train, y_train)
```

Out[65]:

```
KNeighborsClassifier()
```

1. Using the test set to predict.

In [67]:

```
# Use the .predict() method to make predictions from the x_test subset
pred = knn.predict(x_test)

# Show predictions
pred
```

Out[67]:

```
array([2, 5, 1, 6, 1, 2, 2, 7, 1, 2, 2, 1, 1, 1, 2, 2, 5, 2, 1, 1, 1, 1,
       1, 7, 1, 1, 1, 5, 2, 2, 2, 2, 2, 1, 1, 1, 2, 2, 1, 6, 1, 2, 1, 2,
       1, 1, 2, 5, 1, 1, 6, 6, 1, 2, 2, 2, 1, 2, 1, 1, 1, 2, 1, 1, 1],
      dtype=int64)
```

1. Evaluate the accuracy of predictions. According to the output,the model is not that accurate, about 60%.

In [68]:

```
# Import classification report and confusion matrix to evaluate predictions
from sklearn.metrics import classification_report, confusion_matrix

# Print out classification report and confusion matrix
print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

           1       0.66      0.78      0.71        27
           2       0.57      0.81      0.67        16
           3       0.00      0.00      0.00         9
           5       1.00      0.80      0.89         5
           6       0.75      0.75      0.75         4
           7       1.00      0.50      0.67         4

    accuracy                           0.66        65
   macro avg       0.66      0.61      0.61        65
weighted avg       0.60      0.66      0.62        65

c:\users\momoko\appdata\local\programs\python\python37\lib\site-packages\sklearn\metrics\_
classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to con
trol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\users\momoko\appdata\local\programs\python\python37\lib\site-packages\sklearn\metrics\_
classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to con
trol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\users\momoko\appdata\local\programs\python\python37\lib\site-packages\sklearn\metrics\_
classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to con
trol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

1. Select the best K values from 1 to 39. error_rate is an empty list to collect the error rate at various K values. According to the output picture, we can see the error rate continues to increase as the K value increases. And the lowest error rate is occur when K value is around 1

In [69]:

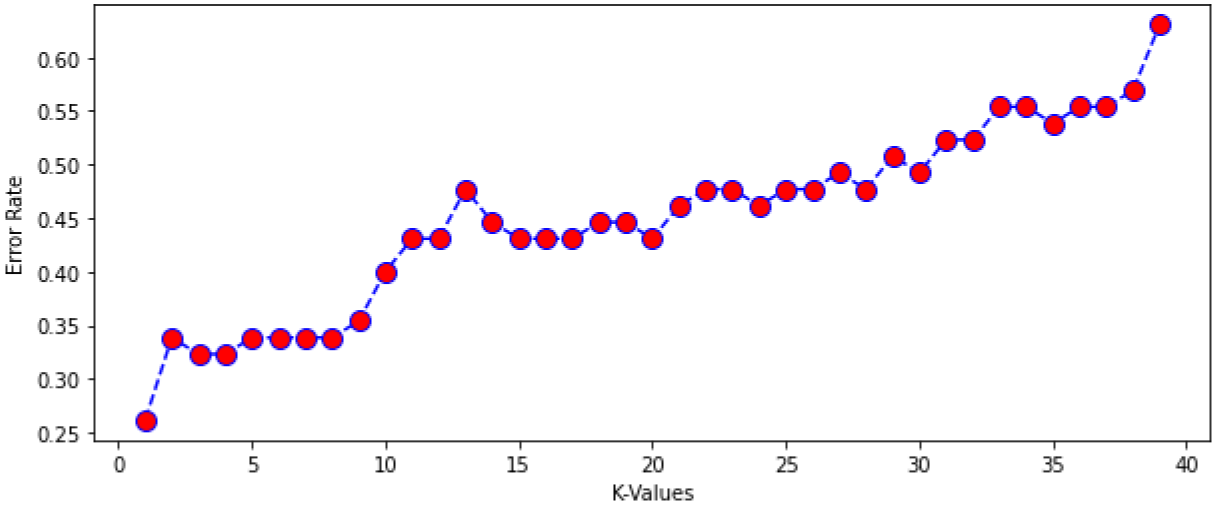
```
error_rate = []
for i in range(1,40):

    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train, y_train)
    pred_i = knn.predict(x_test)
    error_rate.append(np.mean(pred_i != y_test))
```

In [75]:

```
plt.figure(figsize=(10,4))
plt.plot(range(1,40), error_rate, color='blue', linestyle='dashed', marker='o', markerface
plt.title('Error Rate vs. K-Values')
plt.xlabel('K-Values')
plt.ylabel('Error Rate')
```

Out[75]:

```
Text(0, 0.5, 'Error Rate')
```



1. Retrain the model using the best K value 1. And we can see that the acccuracy of the prediction has improved.

In [72]:

```
# Retrain model using optimal k-value

knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(x_train, y_train)
pred = knn.predict(x_test)
```

In [73]:

```
# Print out classification report and confusion matrix
print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

           1       0.77      0.85      0.81        27
           2       0.62      0.81      0.70        16
           3       0.67      0.22      0.33         9
           5       0.80      0.80      0.80         5
           6       1.00      1.00      1.00         4
           7       1.00      0.50      0.67         4

    accuracy                           0.74        65
   macro avg       0.81      0.70      0.72        65
weighted avg       0.75      0.74      0.72        65
```

4.4 Application

KNN algorithm is simple and effective. And now KNN algorithm has been successfully used in text classification, image classification and other classification problems.

4.5 Conclusion

- Advantage [9]
  - it's easy to implement
  - The algorithm has good robustness to noise training data
  - IF the size of training data is large, the results will be even better.
- Disadvantage [9]
  - Need to determine the value of K which sometimes is complex.

- Due to the need to calculate the distance between the data points of all the training samples, the calculation cast is high.

5 Support vector machine

5.1 Introduction

5.2 How does SVM work

5.2.1 Linear Separators

5.2.2 Margin

5.3 Implementation

5.4 Application

5.5 Conclusion

5.6 Presentation

6 Convolutional Neural Network

7 References

[1] IBM Cloud Education(2020), "Machine Learning", https://www.ibm.com/cloud/learn/machine-learning#toc-how-machin-NoVMSZI_

[2] Hussain Mujtaba(2021), "Machine Learning Tutorial For Complete Beginners | Learn Machine Learning with Python" https://www.mygreatlearning.com/blog/machine-learning-tutorial/

[3] Kaggle," Titanic - Machine Learning from Disaster", https://www.kaggle.com/c/titanic

[4] Patrick Triest(2016), "Would You Survive the Titanic? A Guide to Machine Learning in Python Part 1", https://www.kdnuggets.com/2016/07/titanic-machine-learning-guide-part-2.html

[5] Jason Brownlee(2019), "A Gentle Introduction to Bayes Theorem for Machine Learning", https://machinelearningmastery.com/bayes-theorem-for-machine-learning/

[6] javapoint, "Native Bayes Algorithm", https://www.javatpoint.com/machine-learning-naive-bayes-classifier

[7] Kaggle, "Iris Species", https://www.kaggle.com/uciml/iris

[8] Dhiraj K(2020), "Naive Bayes Classifier in Python Using Scikit-learn",https://heartbeat.fritz.ai/naive-bayes-classifier-in-python-using-scikit-learn-13c4deb83bcf

[9] javapoint,"K-Nearest Neighbor(KNN) algorithm for Machine Learning", https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

[10] Kaggle, "Glass Classification", https://www.kaggle.com/uciml/glass

[11] Keith Brooks(2018), "Day (11) — Machine Learning — Using KNN (K Nearest Neighbors) with scikit-learn", https://medium.com/@kbrook10/day-11-machine-learning-using-knn-k-nearest-neighbors-with-scikit-learn-350c3a1402e6

[12] StackExchange, "Why do you need to scale data in KNN"https://stats.stackexchange.com/questions/287425/why-do-you-need-to-scale-data-in-knn

```
In [ ]:
```