Classification Of Mushrooms

(Using Machine Algorithms)

Project By:

J.S.S.Anvesh

K.Vivek

P.Sandeep

INDEX

I) Introduction

What is Machine Learning?

- Machine learning, a branch of artificial intelligence, concerns the construction and study of systems that can learn from data.
- "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performances at tasks in T, as measured by P, improves with experience E." Tom M.Mitchell

Why Machine Learning is important?

- The amount of knowledge available about certain tasks might be too large for explicit encoding by humans(e.g., Environments change over time).
- New knowledge about tasks is constantly being discovered by humans. It may be difficult to continuously re-design systems "by hand".

II) Abstract

- This project is mainly about classifying a 23 species of gilled mushrooms into 2 categories as Poisonous or Edible.
- Although this dataset was originally contributed to the UCI Machine Learning repository nearly 30 years ago, the dataset was downloaded from www.Kaggle.com
- This dataset includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family

Mushroom drawn from The Audubon Society Field Guide to North American Mushrooms (1981).

- Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one.

III) Dataset Feature Description

- ***Class***: edible=e, poisonous=p
- ***cap-shape:*** bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s
- ***cap-surface:*** fibrous=f, grooves=g, scaly=y, smooth=s
- ***cap-color:*** brown=n, buff=b, cinnamon=c, grey=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y
- ***Bruises:*** bruises=t, no=f
- ***Odour:*** almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s
- ***gill-attachment:*** attached=a, descending=d, free=f, notched=n
- ***gill-spacing:*** close=c, crowded=w, distant=d
- ***gill-size:*** broad=b, narrow=n
- ***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
- ***stalk-shape:*** enlarging=e, tapering=t

- ***stalk-root:*** bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?
- ***stalk-surface-above-ring:*** fibrous=f, scaly=y, silky=k, smooth=s
- ***stalk-surface-below-ring:*** fibrous=f, scaly=y, silky=k, smooth=s
- ***stalk-color-above-ring:*** brown=n, buff=b, cinnamon=c, grey=g, orange=o, pink=p, red=e, white=w, yellow=y
- ***stalk-color-below-ring:*** brown=n, buff=b, cinnamon=c, grey=g, orange=o, pink=p, red=e, white=w, yellow=y
- ***veil-type:*** partial=p, universal=u
- ***veil-color:*** brown=n, orange=o, white=w, yellow=y
- ***ring-number:*** none=n, one=o, two=t
- ***ring-type:*** cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z
- ***spore-print-color:*** black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y
- ***population:*** abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y
- ***habitat:*** grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d

IV) Libraries

a) Pandas:

Pandas is an opensource library that allows to you perform data manipulation in **Python**. **Pandas** library is built on top of Numpy, meaning **Pandas** needs Numpy to operate.

b) Numpys:

NumPy is a **Python** package which stands for 'Numerical **Python**'. It is the core library for scientific computing, which contains a powerful n-dimensional array object, provide tools for integrating C, C++ etc. It is also useful in linear algebra, random number capability etc.

c) Sklearn:

Scikit-learn (formerly **scikits.learn** and also known as **sklearn**) is a free software machine learning library for the Python programming language.[3] It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN.

d) Matplotlib:

Matplotlib is a plotting library for the **Python** programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.

e) Seaborn:

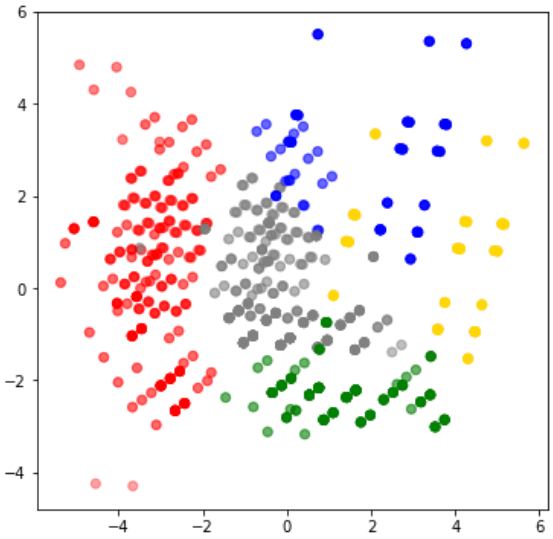
Seaborn: statistical data visualization. **Seaborn** is a **Python** data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

V) Data Visualizaion

Data visualization is the graphic representation of data. It involves producing images that communicate relationships among the represented data to viewers of the images.

a) Scatter Plot:

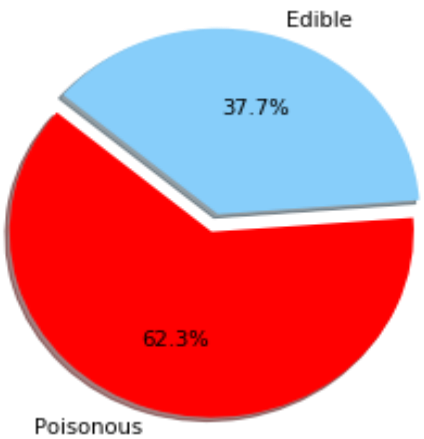
A scatter plot is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data.



Scatter Plot Using Clustering

b) Pie Plot:

A pie chart is a circular statistical graphic, which is divided into slices to illustrate numerical proportion. In a pie chart, the arc length of each slice, is proportional to the quantity it represents.



VI) Methods Used In Pre-Processing

a) Removal Of Null Values:

- Initially checked for any null values presence in the dataset.

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

```
stalk-shape                  0
stalk-root                2480
stalk-surface-above-ring     0
```

- As we found some we removed those null values using a command dropna().

b) LabelEncoder:

Another approach to **encoding** categorical values is to use a technique called **label encoding**. **Label encoding** is simply converting each value in a column to a number. For example, the body_style column co ntains 5 different values.

```
le=LabelEncoder()
```

```
data['class']=list(le.fit_transform(data.iloc[:,0]))
data['cap-shape']=list(le.fit_transform(data.iloc[:,1]))
data['cap-surface']=list(le.fit_transform(data.iloc[:,2]))
data['cap-color']=list(le.fit_transform(data.iloc[:,3]))
data['bruises']=list(le.fit_transform(data.iloc[:,4]))
```

c) Splitting:

Data splitting is an approach to protecting sensitive **data** from unauthorized access by encrypting the **data** and storing different portions of a file on different servers. When **split data** is accessed, the dataset will be divided into two splits i.e. Training Data and Testing Data.

VII) Algorithms Performed

a) Logistic Regression:

Logistic regression is a statistical **model** that in its basic form uses a **logistic** function to **model** a binary dependent variable, although many more complex extensions exist. In **regression** analysis, **logistic regression** (or **logit regression**) is estimating the parameters of a **logistic model** (a form of binary **regression**).

LOGISTIC

```
In [52]: print(accuracy_score(y_train, train_pred))
         print(accuracy_score(y_test,test_pred))

         1.0
         1.0
```

b) Knn:

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.

KNN

```
In [35]: # Predicting the Test set results
         y_pred = knn.predict(x_test)
         train_pred=knn.predict(x_train)
         print('train=',accuracy_score(train_pred,y_train))
         print('test=',accuracy_score(y_pred,y_test))

         train= 0.9996569468267581
         test= 1.0
```

c) Naïve Bayes:

Naïve Bayes is a classification technique based on **Bayes**' Theorem with an assumption of independence among predictors. In simple terms, a **Naive Bayes Classifier** assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Naive Bayes

In [39]:
```
# Predicting the Test and train sets results
y_pred = GNB.predict(x_test)
train_pred=GNB.predict(x_train)
print('train=',accuracy_score(train_pred,y_train))
print('test=',accuracy_score(y_pred,y_test))
```

```
train= 0.7718696397941681
test= 0.8175582990397805
```

d) Decision Tree:

Decision tree learning is one of the predictive modeling approaches used in statistics, data mining and machine learning. It uses a decision tree to go from observations about an item to conclusions about the item's target value.

Decision Tree

In [41]:
```
# Predicting the Test and train sets results
y_pred = dt.predict(x_test)
train_pred=dt.predict(x_train)
print('train=',accuracy_score(train_pred,y_train))
print('test=',accuracy_score(y_pred,y_test))
```

```
train= 1.0
test= 1.0
```

e) Random Forest:

Random forest is a supervised learning **algorithm** which is used for both classification as well as regression. ... Similarly, **random forest algorithm** creates **decision** trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting.

Random Forest

In [49]:
```
# Predicting the Test and train sets results
y_pred = rfc.predict(x_test)
train_pred=rfc.predict(x_train)
print('train=',accuracy_score(train_pred,y_train))
print('test=',accuracy_score(y_pred,y_test))
```

```
train= 1.0
test= 1.0
```

f) Svm:

A Support Vector Machine (**SVM**) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the **algorithm** outputs an optimal hyperplane which categorizes new examples.

SVM

In [37]:

```
# Predicting the Test set results
y_pred = svm.predict(x_test)
train_pred=svm.predict(x_train)
print('train=',accuracy_score(train_pred,y_train))
print('test=',accuracy_score(y_pred,y_test))
```

```
train= 1.0
test= 1.0
```

g) XGBoosting:

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a **gradient boosting** framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks.

XGBOOSTING

In [46]:

```
# Predicting the Test and train sets results
y_pred = model1.predict(x_test)
train_pred=model1.predict(x_train)
print('train=',accuracy_score(train_pred,y_train))
print('test=',accuracy_score(y_pred,y_test))
```

```
train= 1.0
test= 1.0
```

VIII) Model Selection

- As we can see that almost every algorithm gave an 100% accuracy except for Gaussian Naive Bayes algorithm which gave only 77.8%.
- As the classification of the given dataset is into only two categories (Poisonous or Edible).
- Which is after encoding gets as 0's or 1's.
- So here I prefer to choose Logistic Regression model as it is much suitable for these kind of classification.

LOGISTIC

```
In [52]: print(accuracy_score(y_train, train_pred))
         print(accuracy_score(y_test,test_pred))

         1.0
         1.0
```

Outcome:

The main aim of this classification is to predict whether the mushroom is **EDIBLE** or **POISONOUS.** Hence, this can be achieved by the **Logistic Regression**. This Model can be used for this type of datasets with 100% Accuracy.