Project Report
Mushroom Classification
Date : 11/1/2022

Contents

1. Introduction

Mushroom is found to be one of the best nutritional foods with high proteins, vitamins, and minerals. It contains antioxidants that prevent people from heart disease and cancer. Around 45000 species of mushroom are found to be existing worldwide. Among these, only some of the mushroom varieties were found to be edible. Some of them are really dangerous to consume. In order to distinguish between the edible and poisonous mushrooms in the mushroom dataset which was obtained from UCI Machine Learning Repository, some data mining techniques are used. Weka is a data mining tool with various machine learning algorithms that can pre-process, analyze, classify, visualize and predict the given data. Thus, to select the attributes that help better classify mushrooms, the Wrapper method and Filter method in Weka is used to identify the best attributes for the classification. The attributes ‘odor’ and ‘spore_print_color’ were chosen to be the best ones that contributed to the better classification of edible and poisonous mushrooms. After identifying the key attributes, classification is performed, a decision tree is constructed based on those attributes, and its Precision, Recall, and F-Measure values are analyzed.

2. Problem Statement

The Audubon Society Field Guide to North American Mushrooms contains descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family Mushroom (1981). Each species is labeled as either definitely edible, definitely poisonous, or maybe edible but not recommended. This last category was merged with the toxic category. The Guide asserts unequivocally that there is no simple rule for judging a mushroom's edibility, such as "leaflets three, leave it be" for Poisonous Oak and Ivy. The main goal is to predict which mushroom is poisonous & which is edible.

3. Data Information

3.1 Data Requirements

Although this dataset was originally contributed to the UCI Machine Learning repository nearly 30 years ago, mushroom hunting (otherwise known as "shrooming") is enjoying new peaks in popularity. Learn which features spell certain death and which are most palatable in this dataset of mushroom characteristics. And how certain can your model be?

3.2 Data Content

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. The Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule like "leaflets three, let it be" for Poisonous Oak and Ivy.

3.3 About this File

Attribute Information: (classes: 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,gray=g,green=r,pink=p,purple=u, red=e,white=w,yellow=y
- bruises: bruises=t,no=f
- odor:almond=a,anise=l,creosote=c,fishy=y,foul=f,musty=m,none=n,pung e nt=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,gray=g,orange=o,pink=p,red=e,white=w,yell ow=y
- stalk-color-below-ring:brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yell ow=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

4. Code part

The project is about classification, so for classification I used several classification models like logistic regression, SVM, KNeighbours, Ensemble methods like RandomForest, AdaboostClassifier, GradientBoosting etc.

Following steps i used in coding:

1. Importing the necessary libraries
2. Data cleaning
3. Data analysing
4. One hot coding
5. Used GridSearchCV for finding best hyperparameters
6. Training models
7. Testing models
8. Finding accuracy

Code:

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")


data=pd.read_csv('mushrooms.csv')


data.head()

[data.columns]

data.info()

# assigning 'p' as 0 and 'e' as 1
x=data['class'].map({'p':0,'e':1})

#independent variables
y=data.loc[:,data.columns!='class']
y=pd.get_dummies(y)
y.head()


from sklearn.model_selection import train_test_split,
GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier ,
AdaBoostClassifier , GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.metrics import accuracy_score


final={}
train_x, test_x, train_y, test_y=
train_test_split(y,x,test_size=0.4, random_state=10)
train_x.shape, test_x.shape, train_y.shape, test_y.shape


# ### LogisticRegression

model1= LogisticRegression()
param1={"C":np.logspace(-3,3,7),
"penalty":["l1","l2"],'solver':['newton-cg', 'lbfgs',
'liblinear', 'sag', 'saga']}
model1_GCV= GridSearchCV(model1,param1, cv=10, verbose=0)
model1_GCV.fit(train_x, train_y)

# to get best estimator, predict, and getting accuracy
model1_GCV.best_estimator_

p1=model1_GCV.predict(test_x)

final[type(model1).__name__]=accuracy_score(p1, test_y)

accuracy_score(p1, test_y)


# ### RandomForestClassifier

model2= RandomForestClassifier()
param2={'criterion':['gini', 'entropy'],
        'n_estimators':[100,140,180,200,300],
        'max_depth':[1,2,3],
        'bootstrap':[True, False]
```

```
       }
model2_GCV= GridSearchCV(model2,param2, cv=10, verbose=0)
model2_GCV.fit(train_x, train_y)

# to get best estimator, predict, and getting accuracy
model2_GCV.best_estimator_


p2=model2_GCV.predict(test_x)

final[type(model2).__name__]=accuracy_score(p2,
test_y)

accuracy_score(p2, test_y)


# ### AdaBoostClassifier

model3= AdaBoostClassifier()
param3=
{'base_estimator':[DecisionTreeClassifier(max_depth=1)],
         'n_estimators':[100,140,180,200,300],
         'learning_rate':[0.01, 0.1, 1, 10,100 ],
         'algorithm':['SAMME', 'SAMME.R']
        }
model3_GCV= GridSearchCV(model3,param3, cv=10, verbose=0)
model3_GCV.fit(train_x, train_y)

# to get best estimator, predict, and getting accuracy

model3_GCV.best_estimator_

p3=model3_GCV.predict(test_x)
final[type(model3).__name__]=accuracy_score(p3, test_y)
accuracy_score(p3, test_y)
```

```
# ### GradientBoostingClassifier

model4 =GradientBoostingClassifier()
param4= {'max_depth':[1,2,3],
         'n_estimators':[100,140,180,200,300],
         'learning_rate':[0.01, 0.1, 1, 10,100 ]
        }
model4_GCV= GridSearchCV(model4,param4, cv=10, verbose=0)
model4_GCV.fit(train_x, train_y)

# to get best estimator, predict, and getting accuracy

model4_GCV.best_estimator_

p4=model4_GCV.predict(test_x)
final[type(model4).__name__]=accuracy_score(p4, test_y)
accuracy_score(p4, test_y)



#  ### SVC

model5= SVC()
param5= {'C':[0.05,0.1,1,10,100],
         'gamma':[0.05,0.1,1,10,100]
        }
model5_GCV= GridSearchCV(model5,param5, cv=10, verbose=0)
model5_GCV.fit(train_x, train_y)

# to get best estimator, predict, and getting accuracy

model5_GCV.best_estimator_
```

```
p5=model5_GCV.predict(test_x)

final[type(model5).__name__]=accuracy_score(p5, test_y)

accuracy_score(p5, test_y)


# ### KNeighborsClassifier

model6=KNeighborsClassifier()
model6.fit(train_x, train_y)


# to get best estimator, predict, and getting accuracy
p6=model6.predict(test_x)
final[type(model6).__name__]=accuracy_score(p6, test_y)
accuracy_score(p6, test_y)

from matplotlib import pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')

print(final)


plt.figure(figsize=[8,6])
plt.bar(final.keys(), final.values())
plt.xticks(rotation=80,fontsize=15);
```

Accuracies of the models: