Classification of Mushrooms

Elif Çiçek Şensöz
Computer Engineering, BAU
elifcicek.sensoz@bahcesehir.edu.tr

***Abstract*—This project be done for classification of mushrooms. Gaussian Naïve Bayes and Random Forest algorithms are used for comparison.**

Keywords:Gaussian Naïve Bayes, Random Forest, classification, ROC Curves, machine learning, big data, big data analysis

I. INTRODUCTION *(HEADING 1)*

Mushrooms are rich as protein and have high quality nutrition especially vegans unless they are not poisonous. Many people have a hobby to collect edible mushrooms and this activity is called as "shrooming". The major problem about shrooming is identify edible mushrooms and poisonous mushrooms. In this project, I tried to classify edible and poisonous mushrooms from each other.

The background domain of this project is prediction of poisonous mushrooms and differentiate them from non-poisonous mushrooms. In 1981 data is gathered by The Audubon Society Field Guide to help shroomers for finding non-poisonous mushrooms. In 1987, data is donated by Jeff Schlimmer to UCI Machine Learning Repository. This data has been used by enthusiastic machine learning engineers since then. Since I am an outdoor person and I like to be in nature this domain gets my interest.

I analyzed poisonous and non-poisonous mushrooms. Some people are learning by child to separate poisonous mushrooms than others. Even though human brain can separate instinctively, there is a high risk for misclassification. In this project I will try to classify with several machine learning algorithms and analyze the results.

II. ANALYSIS OF DATA

Dataset is retrieved from UCI Machine Learning Repository and details can be found from this link: https://archive.ics.uci.edu/ml/datasets/Mushroom

There are 8124 instance in dataset. This means the dataset is not too big but is also is enough to generalize. 4208 of them are edible and 3916 of them are poisonous. In this aspect, dataset looks balanced.
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,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,gray=g,orange=o,pink=p, red=e,white=w,yellow=y
stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=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

In this dataset there isn't any missing value as it can be seen in the output below:

```
class                       0
cap-shape                   0
cap-surface                 0
cap-color                   0
bruises                     0
odor                        0
gill-attachment             0
gill-spacing                0
gill-size                   0
gill-color                  0
stalk-shape                 0
stalk-root                  0
stalk-surface-above-ring    0
stalk-surface-below-ring    0
stalk-color-above-ring      0
stalk-color-below-ring      0
veil-type                   0
veil-color                  0
ring-number                 0
ring-type                   0
spore-print-color           0
population                  0
habitat                     0
dtype: int64
```

III. Analysis of Features

In this dataset there are 23 columns, that means one of them is for class labels and the others are features. All features are categorical as it can be seen below, so data should be encoding:

```
class                       object
cap-shape                   object
cap-surface                 object
cap-color                   object
bruises                     object
odor                        object
gill-attachment             object
gill-spacing                object
gill-size                   object
gill-color                  object
stalk-shape                 object
stalk-root                  object
stalk-surface-above-ring    object
stalk-surface-below-ring    object
stalk-color-above-ring      object
stalk-color-below-ring      object
veil-type                   object
veil-color                  object
ring-number                 object
ring-type                   object
spore-print-color           object
population                  object
habitat                     object
dtype: object
```

IV. Preprocessing

Since our all data are categorical, we need to transform all categorical data to numerical data. In this project LabelEncoder is used for this transformation. It is a function from preprocessing module of sklearn library in Python.

Result:

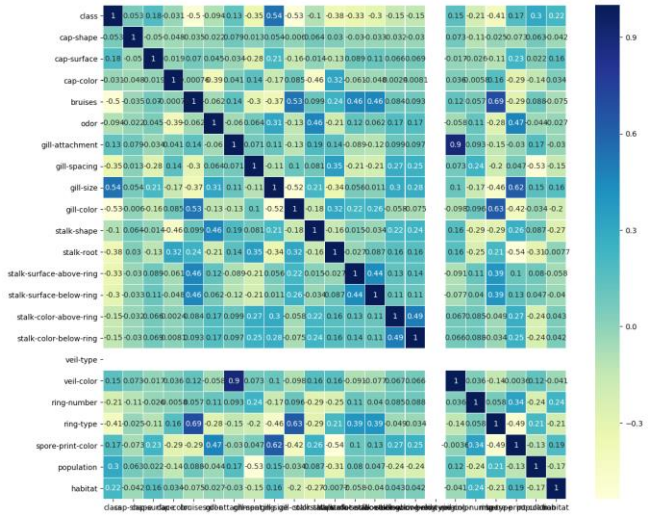
```
class                        int64
cap-shape                    int64
cap-surface                  int64
cap-color                    int64
bruises                      int64
odor                         int64
gill-attachment              int64
gill-spacing                 int64
gill-size                    int64
gill-color                   int64
stalk-shape                  int64
stalk-root                   int64
stalk-surface-above-ring     int64
stalk-surface-below-ring     int64
stalk-color-above-ring       int64
stalk-color-below-ring       int64
veil-type                    int64
veil-color                   int64
ring-number                  int64
ring-type                    int64
spore-print-color            int64
population                   int64
habitat                      int64
dtype: object
```

Since the output is scattered, and working with small numbers is easier than the working with big numbers, I scale the dataset. StandardScaler function from preprocessing module of sklearn library in Python is used to scale data. This function is removing the means of a feature and then scaling with variance. It is done for every feature.

Data is get small as it can be seen below:

```
array([[ 1.02971224,  0.14012794, -0.19824983, ..., -0.67019486,
        -0.5143892 ,  2.03002809],
       [ 1.02971224,  0.14012794,  1.76587407, ..., -0.2504706 ,
        -1.31310821, -0.29572966],
       [-2.08704716,  0.14012794,  1.37304929, ..., -0.2504706 ,
        -1.31310821,  0.86714922],
       ...,
       [-0.8403434 ,  0.14012794, -0.19824983, ..., -1.50964337,
        -2.11182722,  0.28570978],
       [-0.21699152,  0.95327039, -0.19824983, ...,  1.42842641,
         0.28432981,  0.28570978],
       [ 1.02971224,  0.14012794, -0.19824983, ...,  0.16925365,
        -2.11182722,  0.28570978]])
```

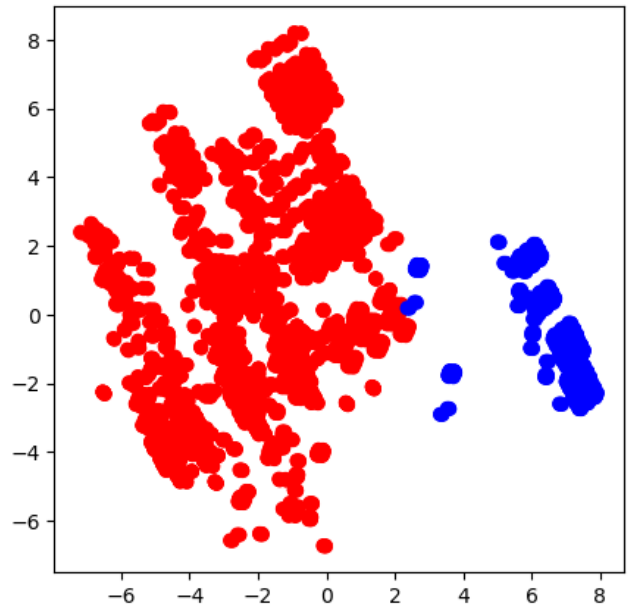
Then, correlations of features are checked and visualized with seaborn library in Python:

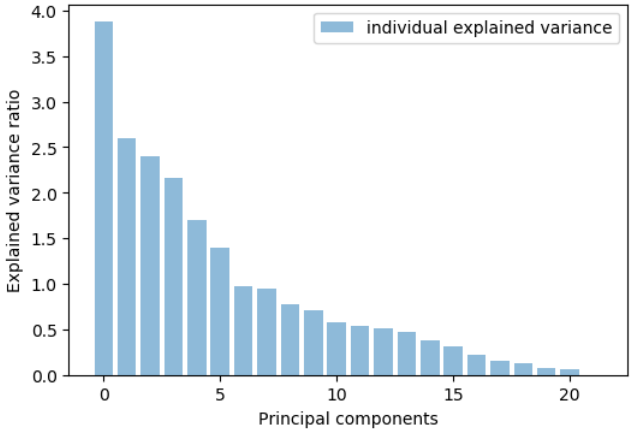


In this graphic, more blue means has more correlation to the outputs. As we can see above, it is hard to analyze because we have so many features.

To decide which features are important, I used explained variance via Principle Component Analysis to compare features and their contributions. Principle Component Analysis is also good for visualize the data.

Let’s visualize data with PCA:





According to this graphic, if explained variance is summed from 1st attribute to 17th, the result is more than 90, that means first 17 features can explain 90% of data, and these 17 features can give a good result in output. Reducing dimensions of dataset can improve the output.

V. Model Selection

In this project two machine learning algorithms are selected for comparison. Since the problem is classification, Gaussian Naïve Bayes and Random Forest algorithms are used with help of threshold (>0.5).

A. *Gaussian Naïve Bayes*

Strengths: Very simple, easy to implement and fast, needed less training data, scalable.

Weaknesses: it can’t learn interactions between features

B. *Random Forest:*

Strengths: It is very good for large datasets, it can be used for both classification and regression tasks. It can be run in parallel to speed up training, reduces variance caused by decision trees by combining multiple decision trees, Random Forest classifier can be modeled for categorical values.

Weaknesses: The main disadvantage of Random forests is their complexity. They are much harder and time-consuming to construct than decision trees. They also require more computational resources and are also less intuitive. When you have a large collection of decision trees it is hard to have an intuitive grasp of the relationship existing in the input data and also the prediction process using random forests is time-consuming than other algorithms.

Candidacy: Random Forest gives good performance when there are categorical variables. It means that it works correctly for a large range of data items than single decision trees.

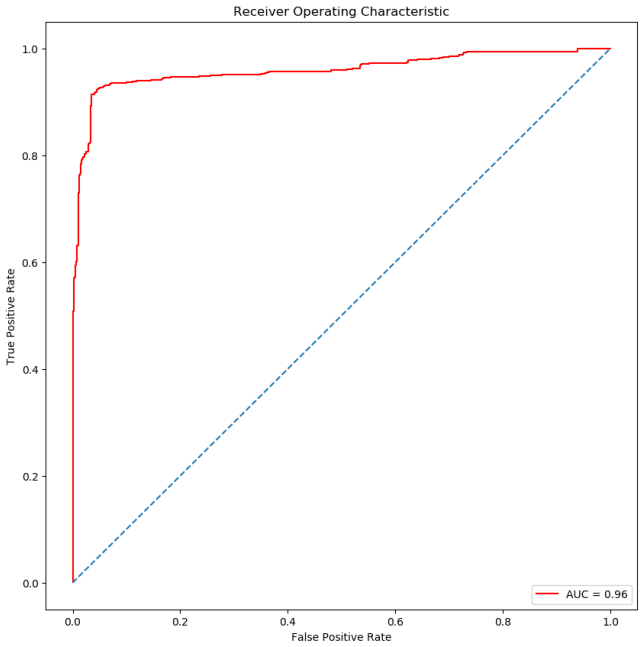
VI. Classification

In this dataset, instances are separated as edible and poisonous mushrooms, so the aim of classification is finding which one is edible and which one is poisonous. Gaussian Naïve Bayes and Random Forest is used to overcome this problem. These two machine learning algorithm is compared with learning and training time, accuracy and ROC curves.

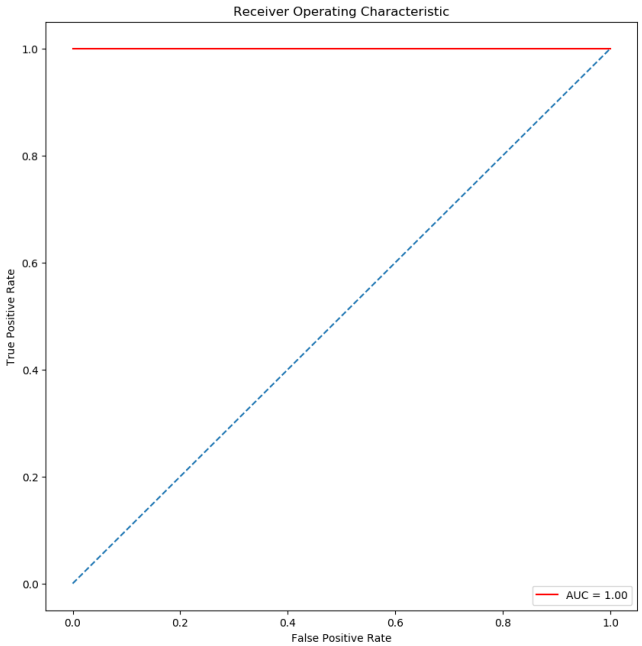
Gaussian learning time: 0.00600004196167
Gaussian prediction time: 0.0019998550415
Random Forest learning time: 0.249000072479
Random Forest prediction time: 0.00499987602234

('Gaussian accuracy: ', '0.9316923076923077')
('Random forest accuracy: ', '1.0')

ROC Curve of Gaussian Naïve Bayes:



ROC Curve of Random Forest:



VII. CONCLUSION

According to ROC curve and accuracy, Random Forest is giving better result. But if running time is checked, Random Forest is incredibly slower than Gaussian Naïve Bayes. Since datasets which smaller than 10 MB is considered as small dataset, this dataset is small. These running times can be neglect in this dataset, but in big datasets, running time of Random Forest is going be increase exponentially and in a point it will be useless.

REFERENCES

[1] http://scikit-learn.org/stable/auto_examples/decomposition/plot_pca_iris.html

[2] https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60

[3] https://www.wikiwand.com/en/Naive_Bayes_classifier

[4] https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d