MUSHROOM DATASET PROJECT

A project report submitted to ICT Academy of Kerala

in partial fulfillment of the requirements

for the certification of

CERTIFIED PROFESSIONAL

IN

DATA SCIENCE & ANALYTICS

submitted by

Aravind M J

ICT ACADEMY OF KERALA

THIRUVANANTHAPURAM, KERALA, INDIA

DECEMBER 2019

Abstract

In this project, we are going to analyze the mushroom dataset given by Data Science Project coordinators of ICT Academy.As part of the project First, we gained some domain knowledge on mushrooms , It help us to understand more about the dataset features and for getting into better conclusions while doing the project.

Afterwards,we will build models to classify mushrooms as edible or poisoned by using the Python, we apply logistic regression, decision trees,SVM,KNN and random forest techniques for getting better classification model. We implemented models in Python and find out the accuracy for each model. Finally we reach out best conclusions based on the various models we applied for train and test the mushroom dataset.

CONTENTS

FIGURE INDEX

GLOSSARY

KNN	-	K- NEAREST NEIGHBOUR
SVM	-	Support-Vector Machines

Problem Statement

We three people planned for a trip to the US unfortunately out plane crashed and we lost in the woods and we ran out of food. But we were surrounded by a lot of mushrooms. But we don't know these mushrooms are edible or poisonous, suddenly some thought comes we have a data set of 8000 kinds of mushrooms and whether they are safe to eat or not. Unfortunately, the latest mushroom we were surrounded is not in your our dataset, so we need to decide whether it is safe to eat based on similarity to other mushrooms.

The dataset has 22 descriptors for mushrooms, such as the cap shape, which can be convex, bell, flat, or sunken. Similarly, the cap color can be brown, yellow, white, or gray. Using all these descriptors, called *features* in machine learning, we will decide if we should eat this mushroom.

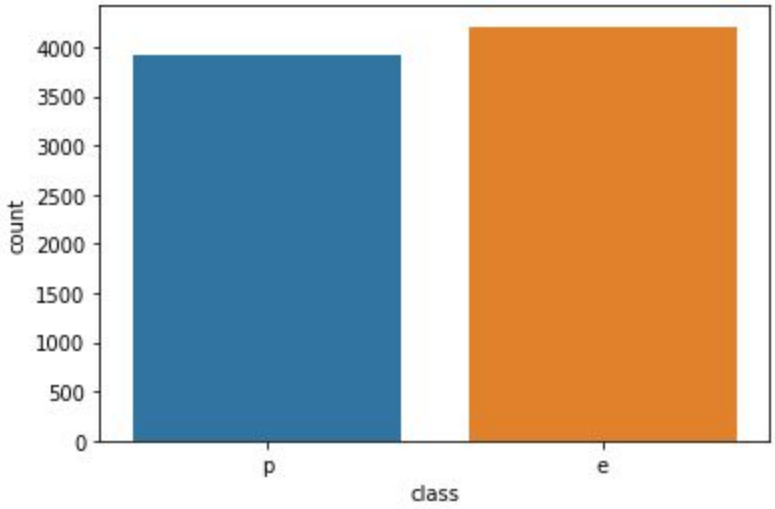
METHODS

Preprocessing Steps

Step 1 : to know the amount of unique values in each every column/features by using `.unique()`

Step 2 :There is no missing values in the form of blank space (using `.isna().sum()`)but ,there are some missing values in the form of '?'.so it is import to check the unique values of each and every columns.

Step 3 : The target feature 'class' contains categorical value . It is seen that the categorical values are normally distributed, by plotting suitable graphs .



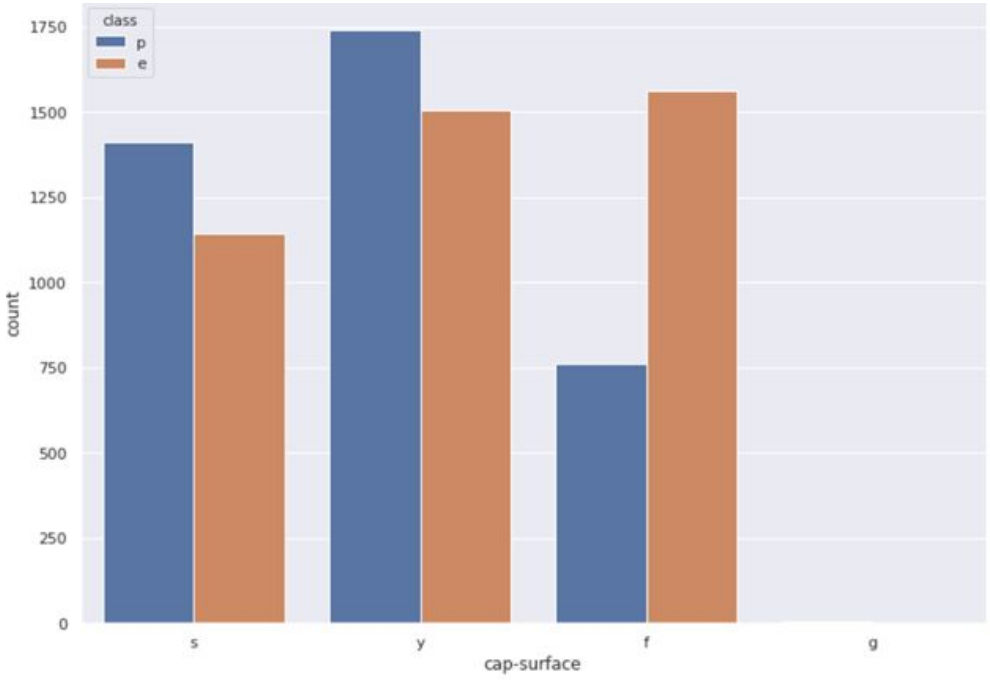
Step 4 : Each column contains categorical variables ,so encode the variable by using " one hot encoding",except the 'class' column

Step 5 : Dropped an unwanted column 'veil-type'

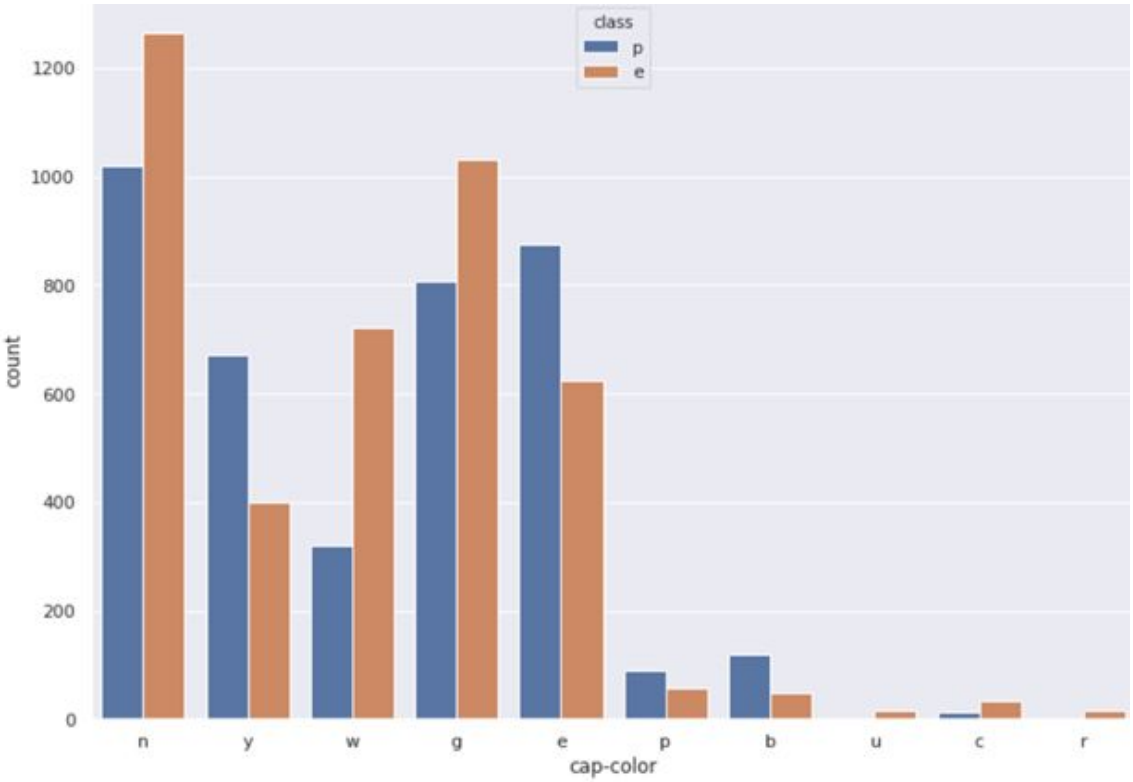
Step 6 : The ‘class’ column is target variable ,therefore impossible to ‘one hot’ . so , map the variable into 1 and 0

Step 7 : To see how each feature affects the target, for each feature, we made a bar plot of all possible values separated by the class of mushroom. We build this helper function for plotting all 22 bar plots and randomly five plots given below.

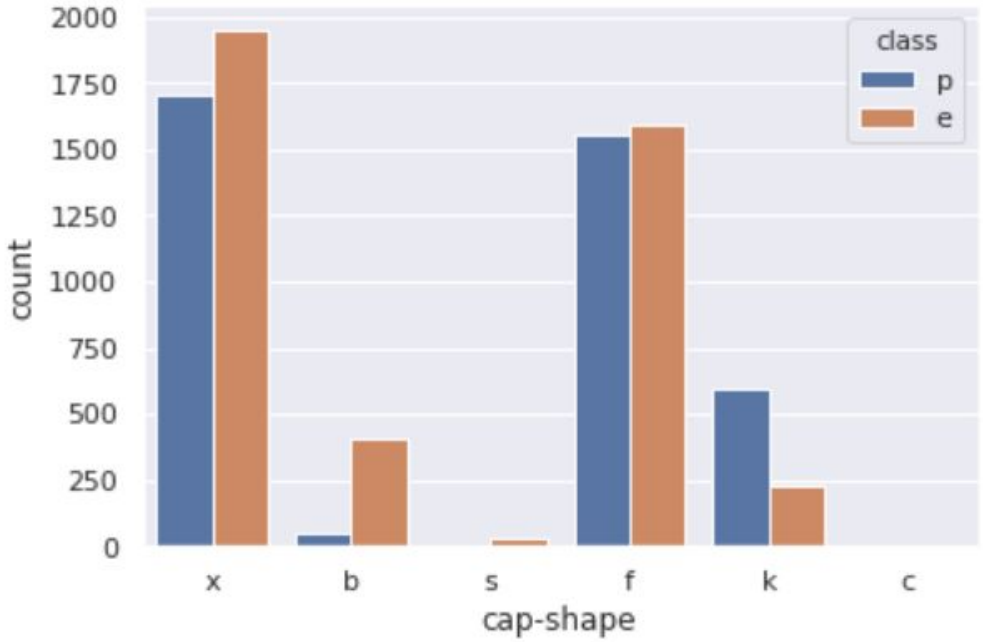
A)



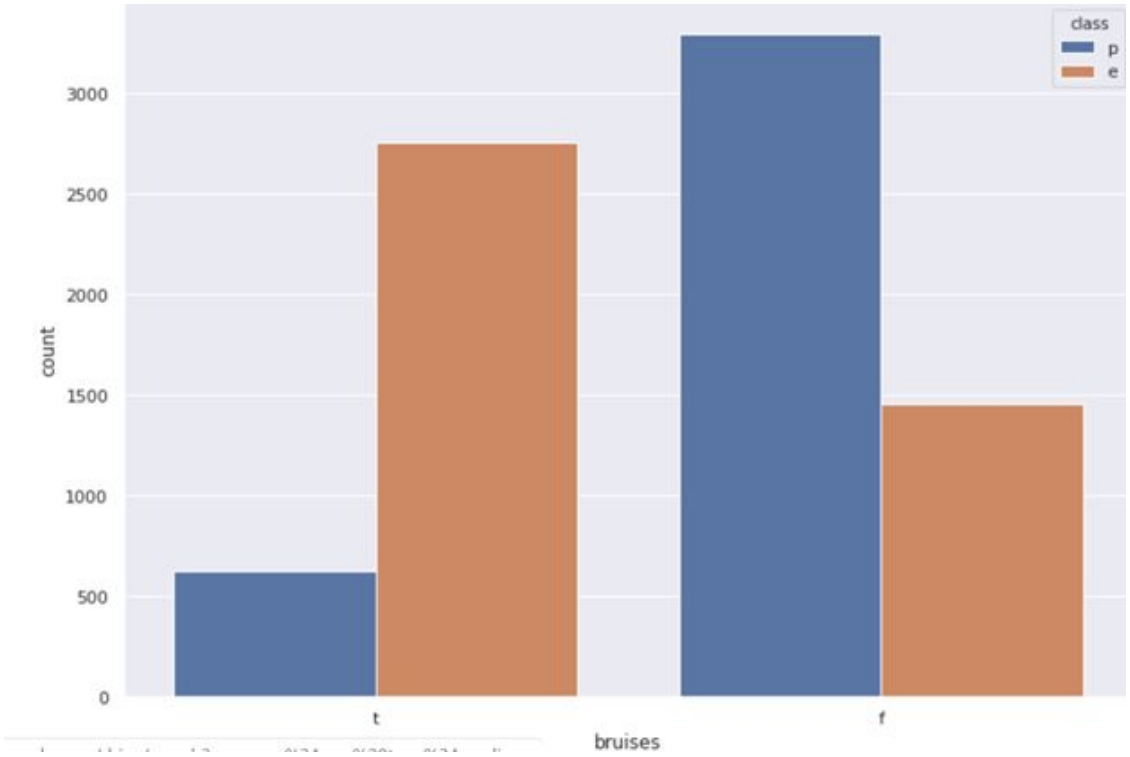
B)


class
p
e
1200
1000
800
600
400
200
0
count
n
y
w
g
e
p
b
u
c
r
cap-color

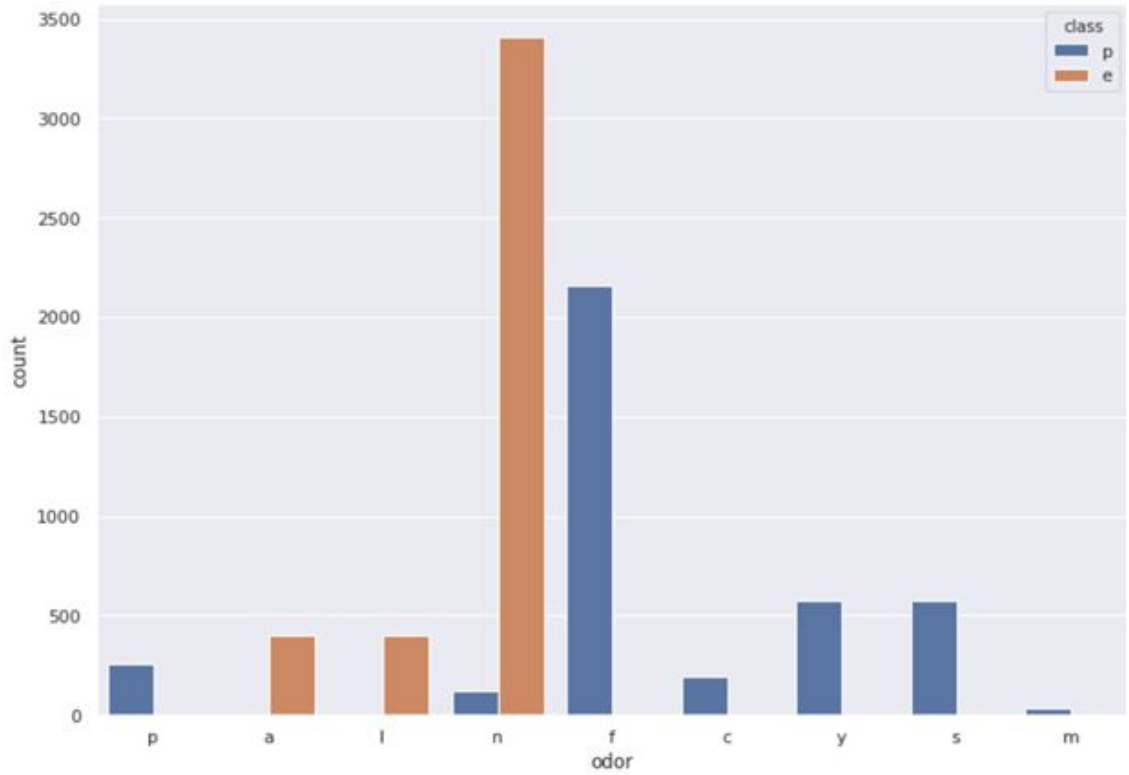
C)


class
p
e
2000
1750
1500
1250
1000
750
500
250
0
count
x
b
s
f
k
c
cap-shape

D)


class
p
e
3000
2500
2000
1500
1000
500
0
count
t
f
bruises

E)


class
p
e
3500
3000
2500
2000
1500
1000
500
0
count
p
a
l
n
f
c
y
s
m
odor

Algorithms we used

1) Logistic regression

In statistics, the **logistic model** is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick.

2) SVM

In machine learning, **support-vector machines** are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

3) Decision Tree

A **decision tree** is a decision support tool that uses a tree like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

4) Random forests

Random forests or **random decision forests** are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees ‘habit of overfitting to their training set’.

5) KNN

In *k-NN classification*, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbor.

TOOLS & TECHNOLOGIES

1. Google Colab
2. Microsoft Word
3. Google Sheets
4. Jupiter Notebook

RESULT

Output or Results we gained by applying various machine learning algorithms in our Mushroom Dataset

1) LOGISTIC REGRESSION

```
#LOGISTIC REGRESSION

from sklearn.linear_model import LogisticRegression
classifier1= LogisticRegression(random_state=42,solver='lbfgs')
classifier1.fit(X_train,y_train)
```

```
[ ] y_pred_logistic = classifier1.predict(X_test)
```

```
[ ] Accuracy_values=[]
    from sklearn.metrics import precision_score,recall_score,f1_score,accuracy_score
    print('Precision is:',precision_score(y_test,y_pred_logistic))
    print('F score is:',f1_score(y_test,y_pred_logistic))
    print('accuracy is:',accuracy_score(y_test,y_pred_logistic))
```

```
Precision is: 1.0
F score is: 1.0
accuracy is: 1.0
```

Accuracy: 1.0
F1 score: 1.0

2) SVM

```
#SVM
from sklearn.svm import SVC
classifier2= SVC(kernel = 'rbf' ,random_state= 0)
classifier2.fit(X_train,y_train)
```

```
[ ] y_pred_svm = classifier2.predict(X_test)
    print('Precision is:',precision_score(y_test,y_pred_svm))
    print('F score is:',f1_score(y_test,y_pred_svm))
    print('accuracy i s:',accuracy_score(y_test,y_pred_svm))
```

```
Precision is: 0.9976190476190476
F score is: 0.99880810488677
accuracy i s: 0.9987694831829368
```

Accuracy : 0.9987694831829368
F score : 0.99880810488677

3)RANDOM FOREST

```
# RandomForest
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(X_train,y_train)
y_pred_rfc=rfc.predict(X_test)
y_pred_rfc
```

```
[ ] from sklearn.metrics import *
    print('confusion matrix is' ,confusion_matrix(y_test,y_pred_rfc))
    print('Precision is',precision_score(y_test,y_pred_rfc))
    print('recall is', recall_score(y_test,y_pred_rfc))
    print('accuracy is',accuracy_score(y_test,y_pred_rfc))
    print('f1 score is',f1_score(y_test,y_pred_rfc))
```

```
confusion matrix is [[1181    0]
 [   0 1257]]
Precision is 1.0
recall is 1.0
accuracy is 1.0
f1 score is 1.0
```

Accuracy : 1.0
f1 score : 1.0

4)DECISION TREE

```
[ ]  #DECISION TREE

     from sklearn.tree import DecisionTreeClassifier
     dtc=DecisionTreeClassifier()
     dtc.fit(X_train,y_train)
     y_pred_dtc=dtc.predict(X_test)
     y_pred_dtc
```

```
array([1, 0, 0, ..., 0, 1, 1])
```

```
[ ]  from sklearn.metrics import *
     print('confusion matrix is' ,confusion_matrix(y_test,y_pred_dtc))
     print('Precision is',precision_score(y_test,y_pred_dtc))
     print('recall is', recall_score(y_test,y_pred_dtc))
     print('accuracy is',accuracy_score(y_test,y_pred_dtc))
     print('f1 score is',f1_score(y_test,y_pred_dtc))
```

```
confusion matrix is [[1181    0]
 [   0 1257]]
Precision is 1.0
recall is 1.0
accuracy is 1.0
f1 score is 1.0
```

Accuracy : 1.0
f1 score : 1.0

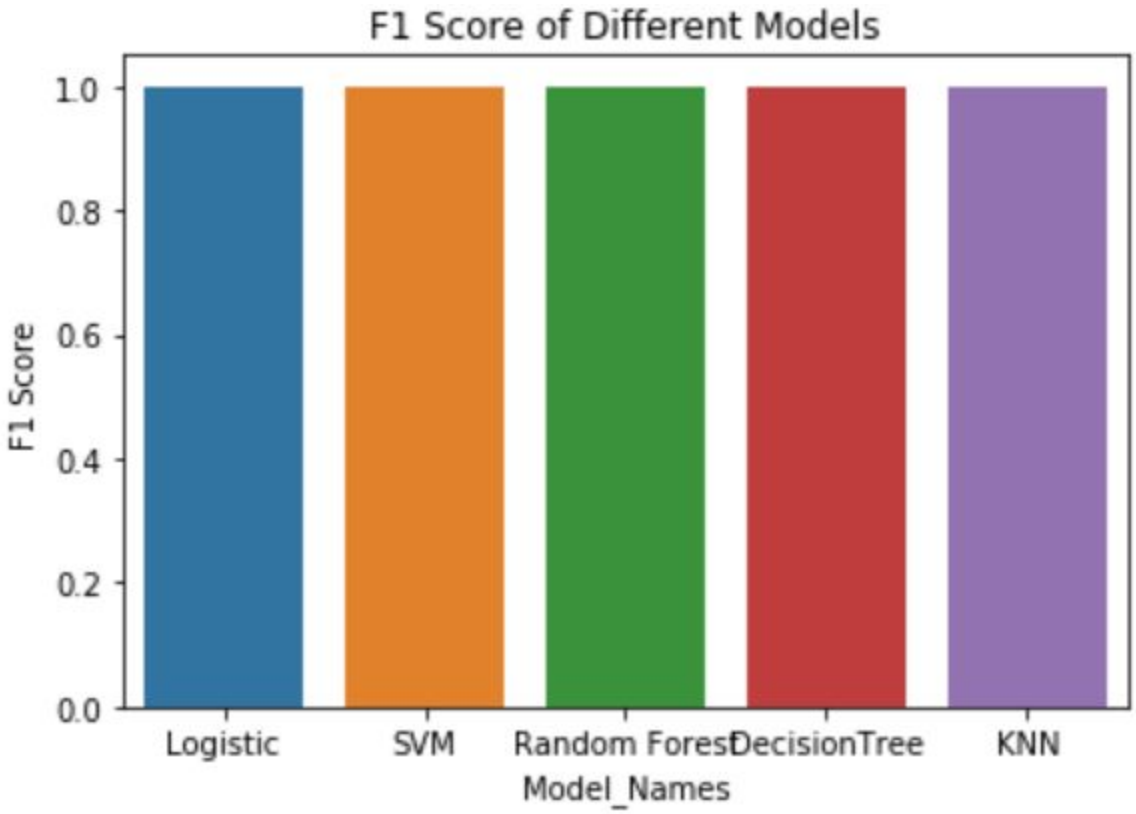
5) KNN

```
[ ]  #KNN
     from sklearn.neighbors import KNeighborsClassifier
     classifier= KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=2)
     classifier.fit(X_train,y_train)
     y_pred_knn=classifier.predict(X_test)
```

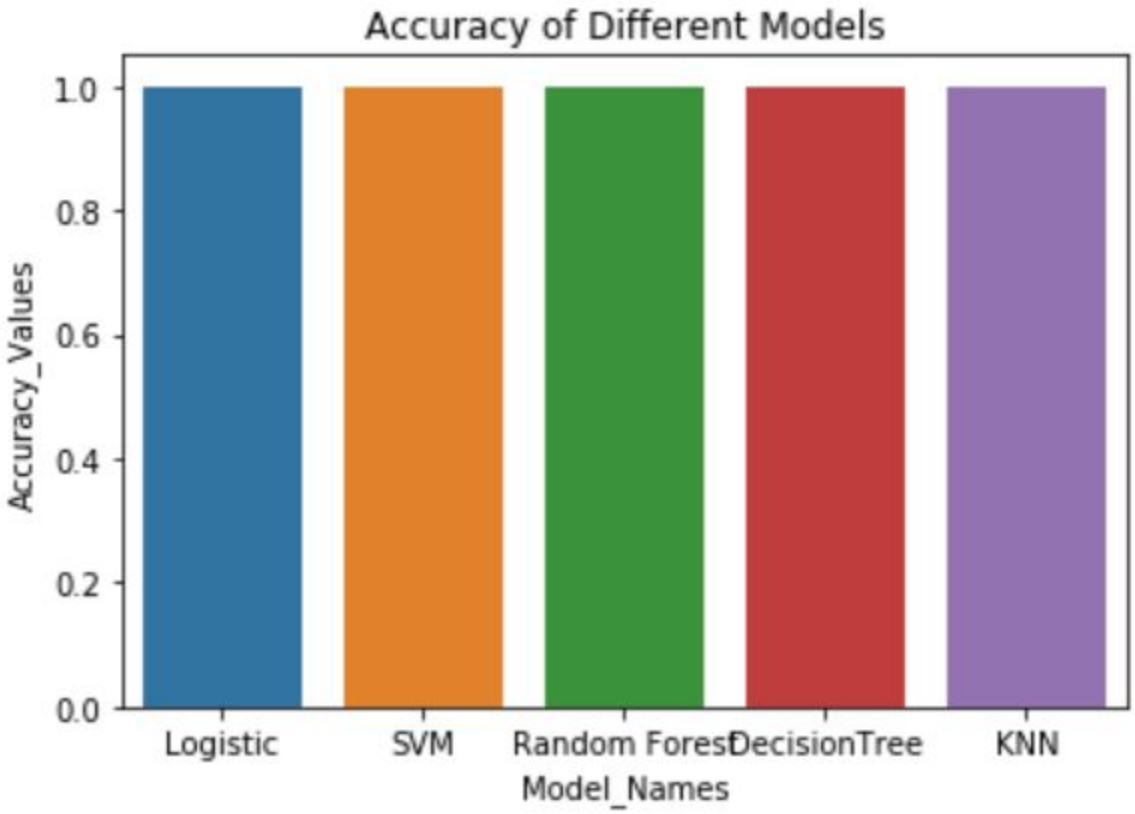
```
[ ]  from sklearn.metrics import *
     print('confusion matrix is' ,confusion_matrix(y_test,y_pred_knn))
     print('Precision is',precision_score(y_test,y_pred_knn))
     print('recall is', recall_score(y_test,y_pred_knn))
     print('accuracy is',accuracy_score(y_test,y_pred_knn))
     print('f1 score is',f1_score(y_test,y_pred_knn))
```

```
confusion matrix is [[1181    0]
 [   0 1257]]
Precision is 1.0
recall is 1.0
accuracy is 1.0
f1 score is 1.0
```

accuracy : 1.0
f1 score : 1.0



Here we plotted the F1 score of all the models we applied in this project. All the five models are giving 100% accuracy except SVM. SVM F1 score is **0.99880810488677** which is very much close to 100.



Here we plotted accuracy score of the five models we applied. Same like F1 score,all models are giving 100% accuracy except SVM. SVM accuracy value is 0.9987694831829368 which is very much close to 100.

CONCLUSION

Based on the data, we generated some high accuracy mushroom classifiers, by applying various machine learning models such as logistic regression, decision tree,knn and random forest, which all came up with good results with low prediction error.

Finally we created better machine learning models for predicting whether the mushrooms are edible or poisonous.

References

1. www.datacamp.org
2. www.kaggle.com
3. www.wikipedia.com
4. www.towarsdatascience.com