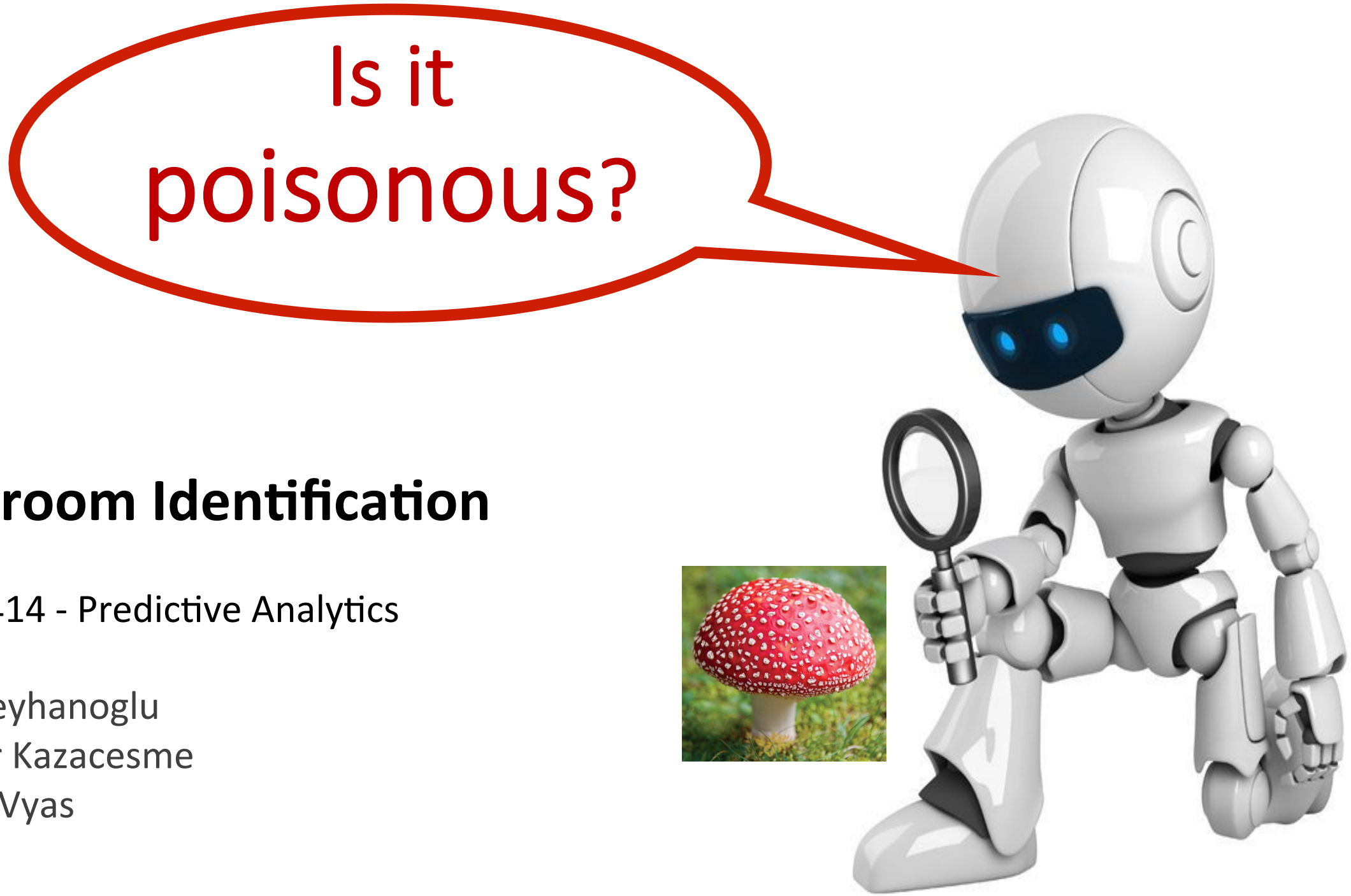


Mushroom Identification

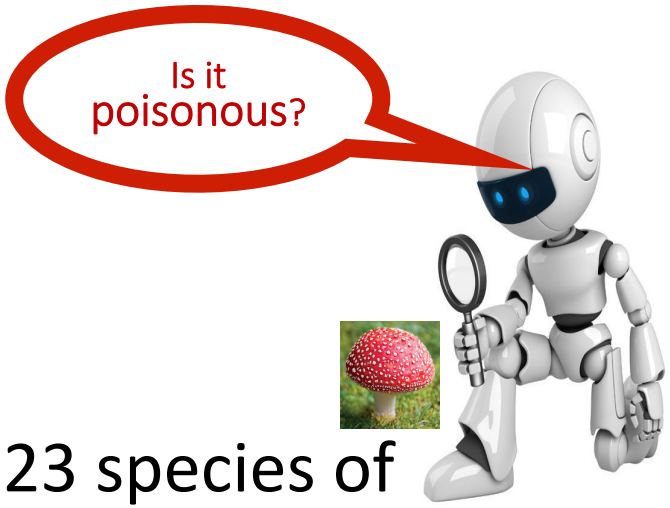
DBDA.X414 - Predictive Analytics

Gonul Reyhanoglu
Hale Nur Kazacesme
Reshma Vyas

Content

- Data Description
- Objective
- Descriptive Data Analysis
- Machine Learning Models
- Conclusion

Data Description

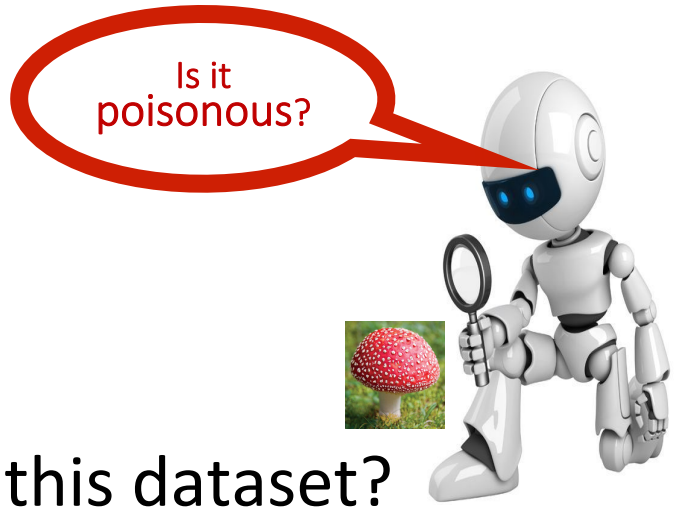


- Descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms
- 2 classes: edible, poisonous
- UCI Machine Learning repository

Content

- Data Description
- Objective
- Descriptive Data Analysis
- Machine Learning Models
- Conclusion

Objective

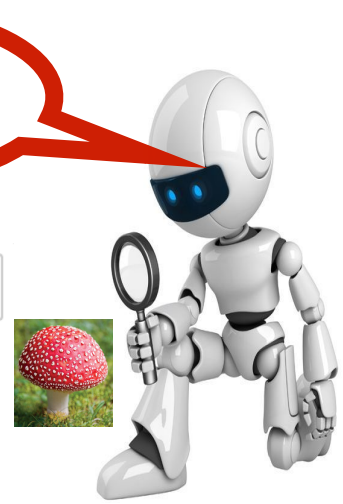


- What types of machine learning models perform best on this dataset?
- Which features are most indicative of a poisonous mushroom?

Content

- Data Description
- Objective
- Descriptive Data Analysis
- Machine Learning Models
- Conclusion

Descriptive Data Analysis 1

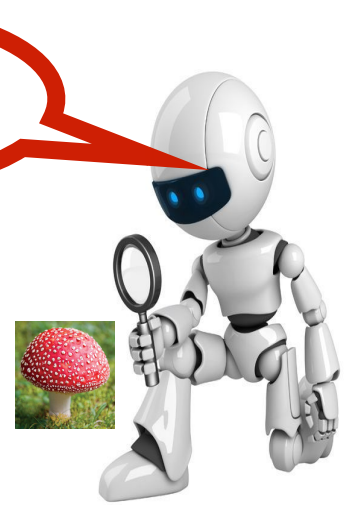



```
mushrooms.describe()
```

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	popul
count	8124	8124	8124	8124	8124	8124	8124	8124	8124	8124	...	8124	8124	8124	8124	8124	8124	8124	8124	
unique	2	6	4	10	2	9	2	2	2	12	...	4	9	9	1	4	3	5	9	
top	e	x	y	n	f	n	f	c	b	b	...	s	w	w	p	w	o	p	w	
freq	4208	3656	3244	2284	4748	3528	7914	6812	5612	1728	...	4936	4464	4384	8124	7924	7488	3968	2388	

4 rows × 23 columns

Descriptive Data Analysis 2



Cap

The top part of the mushroom. The shape, color and texture of the cap can be helpful in identification. But, these can vary and change over time.

Ring (or annulus)

Leftover from the development of the mushroom, when the cap spread out. Some mushrooms have them, some don't, and some loose them with age so look closely and at different stages.

Stem (or stalk)

Most (but not all) mushrooms have stems and these can vary by:

- Shape and size
- Texture (chalk-like? string cheese texture?)
- Color (some even bruise when touched)
- Presence of remnant ring or volva

Volva

Present on some mushrooms and not on others. Useful in identification. Because this is at the base, collect mushrooms by carefully digging them up not cutting the stem.

Gills, pores, tubes, veins, teeth, etc…

On the underside of the cap you will find structures that produce the spores that mushrooms use to spread to new places. These can come in different forms including:

Spores

You can look at the spores by making "spore prints:"

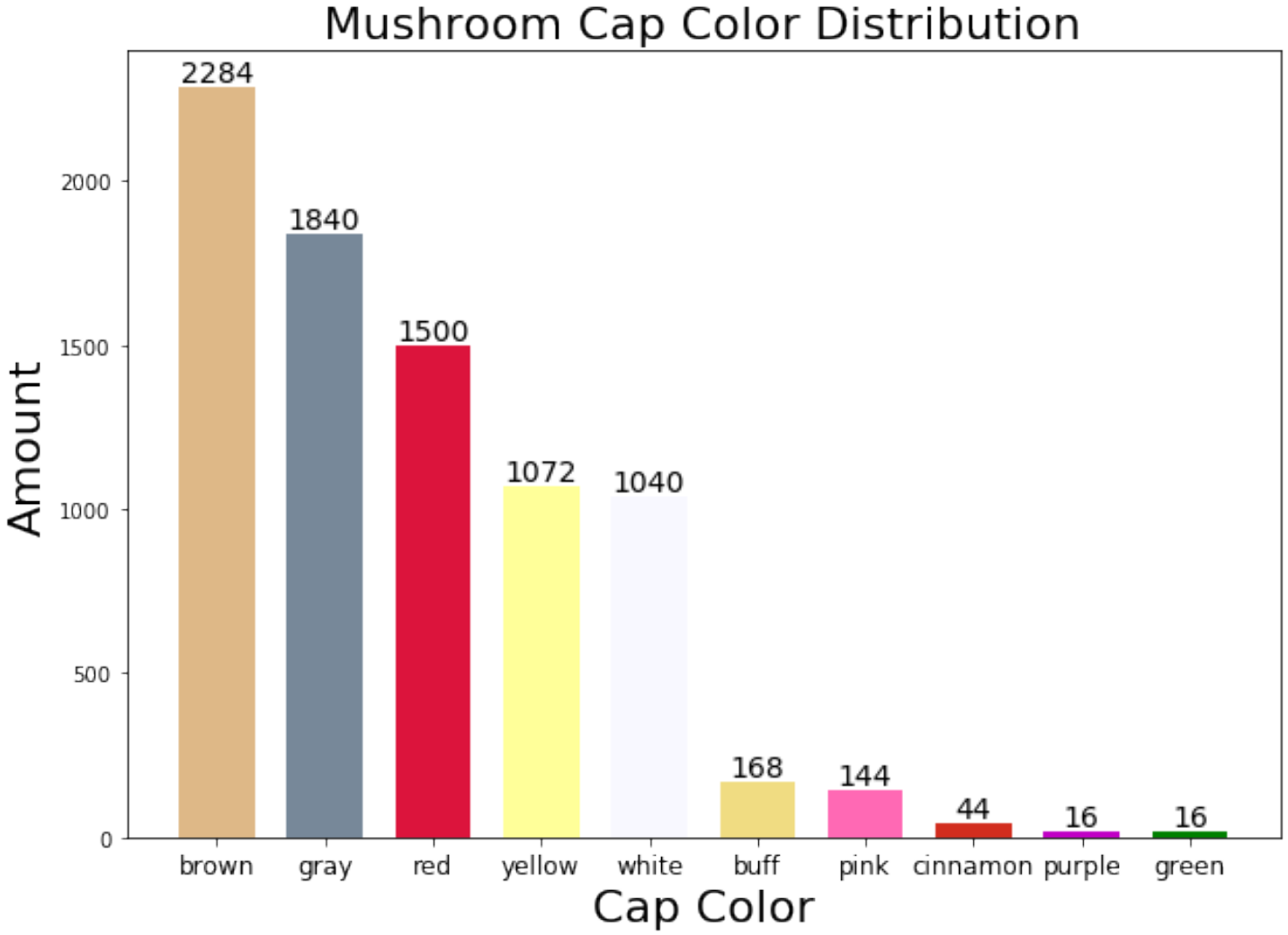
- Place cap gill (or pore) side down on a sheet of paper
- Cover with a bowl or cloth overnight
- The next morning, check the spore print for color

Descriptive Data Analysis 3

The 4 features under magnifier :

- Cap-color
- Odor
- Population Type
- Habitat Type

Descriptive Data Analysis 1

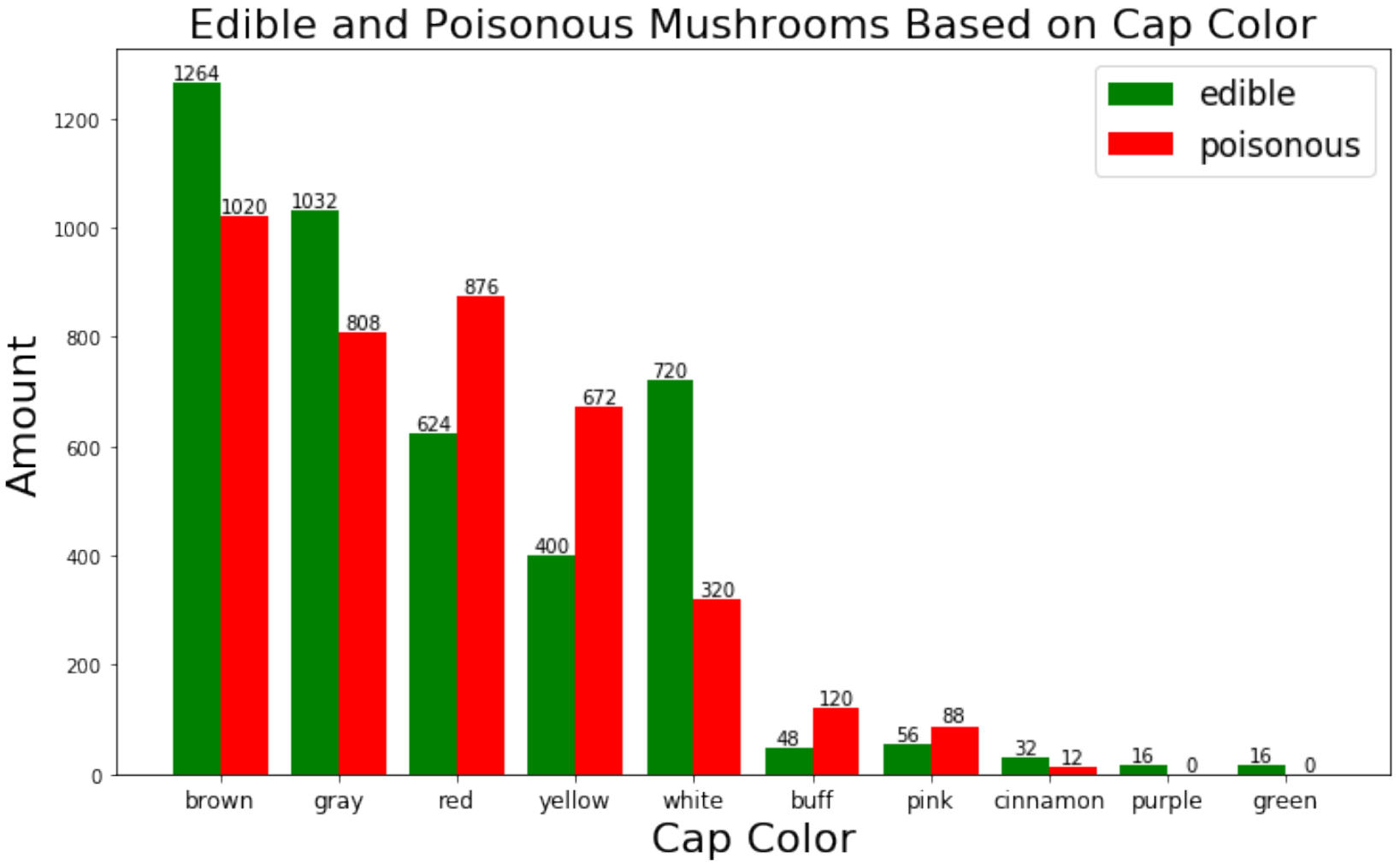


Top tree colors:

- Brown
- Gray
- Red

Descriptive Data Analysis 2

Cap-color does not make a clear distinction



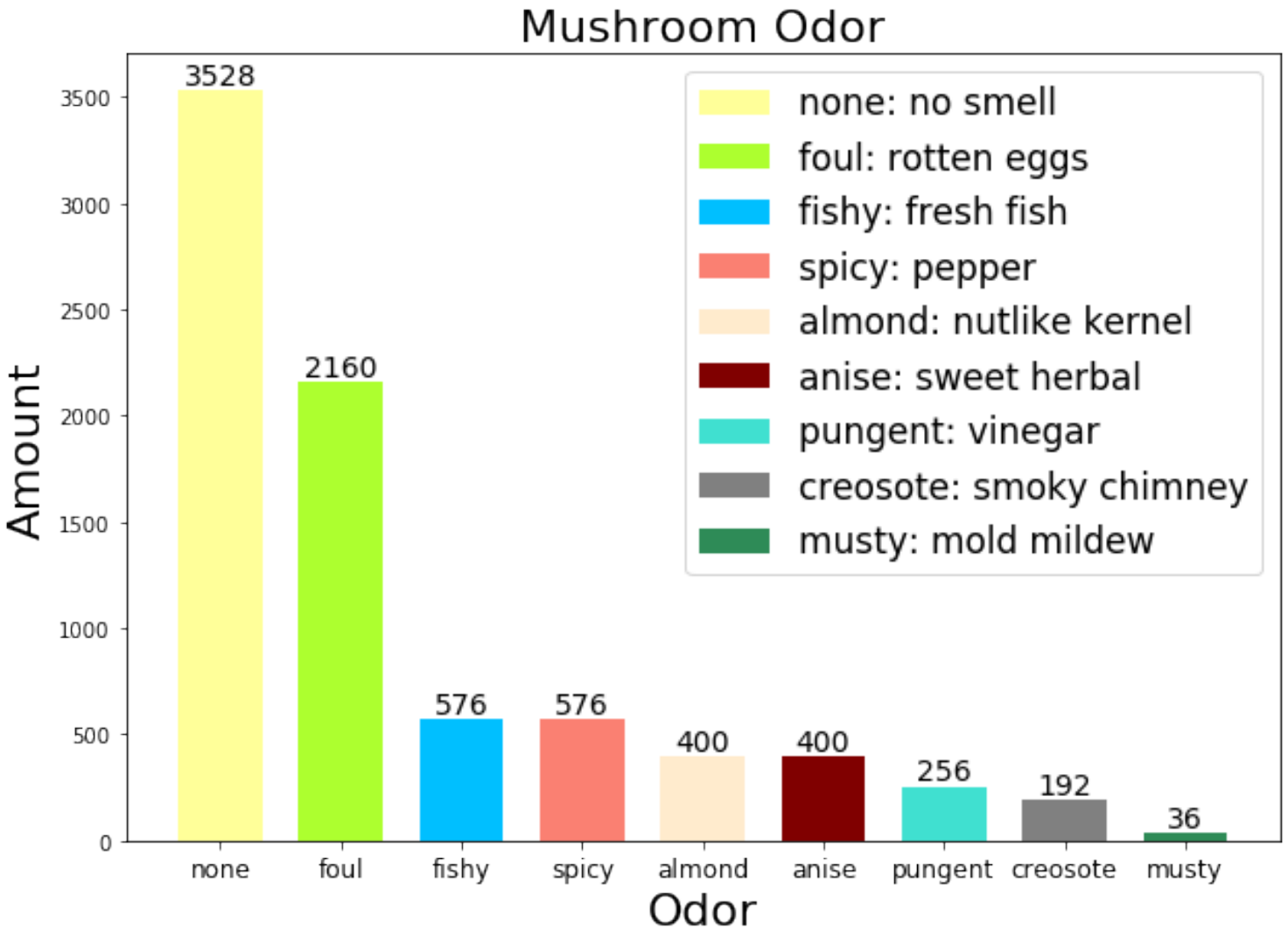
More poisonous colors:

- Red
- Yellow
- Buff
- Pink

More edible colors:

- Brown
- Gray
- White
- Cinnamon
- Purple
- Green

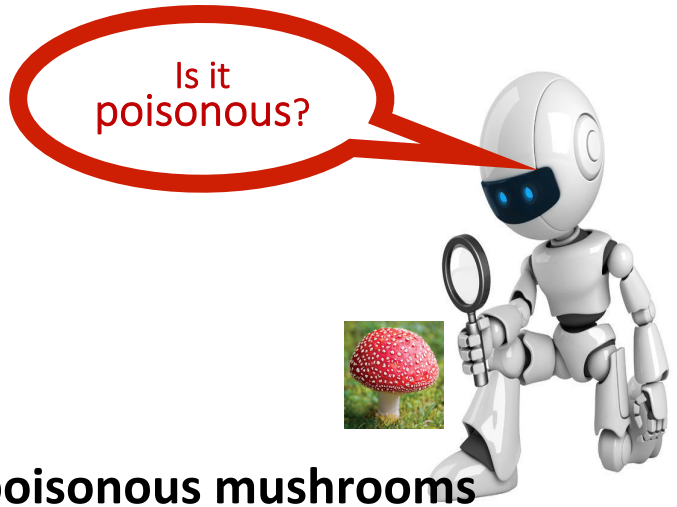
Descriptive Data Analysis 3



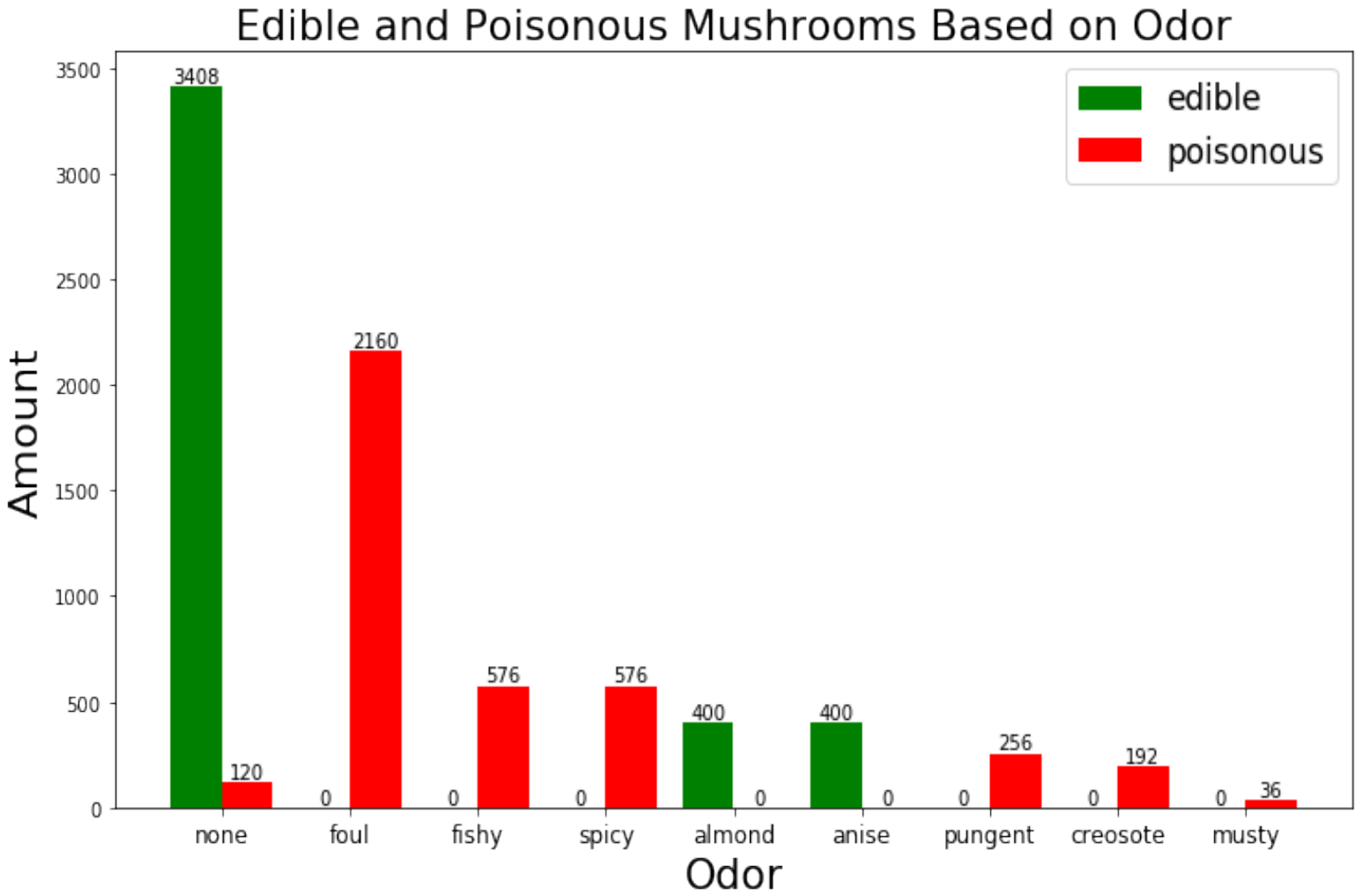
Top 3odor:

- Odorless
- Foul
- Fishy

Descriptive Data Analysis 4



Smell of mushrooms gives the clue



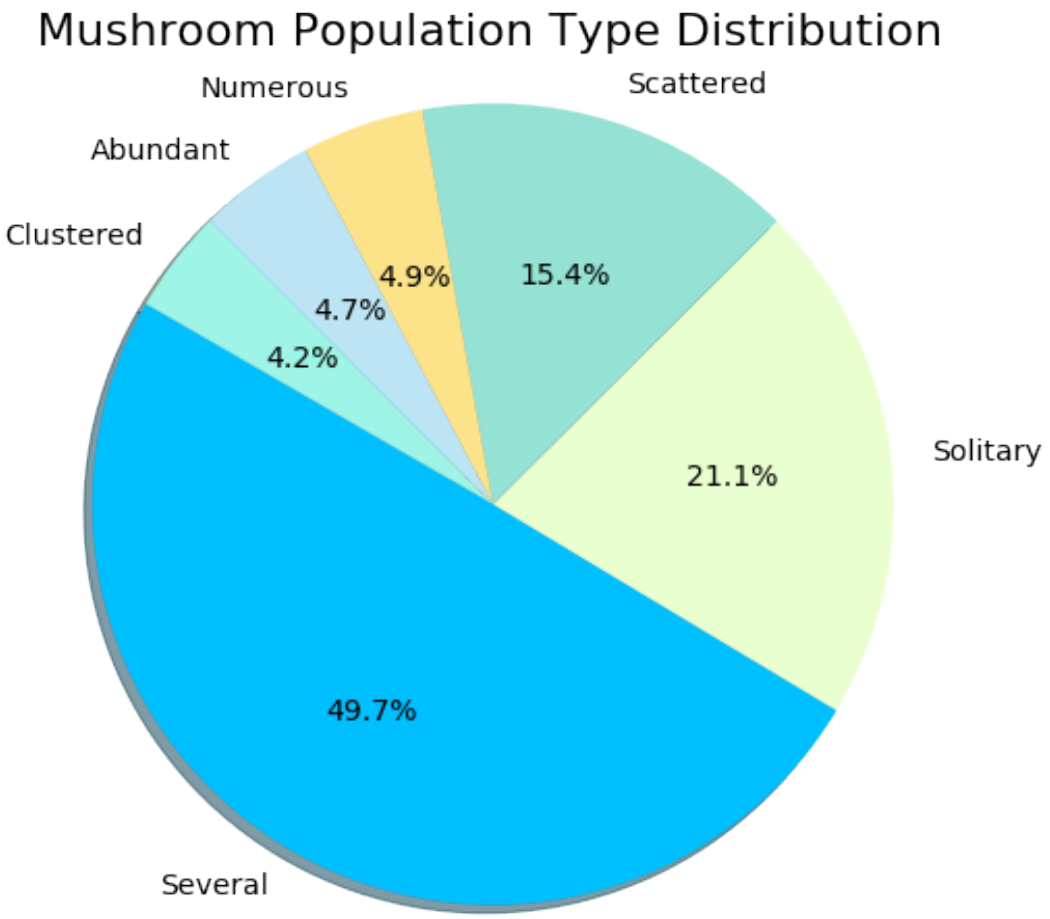
Odor of poisonous mushrooms

- Foul
- Fishy
- Spicy
- Pungent
- Creosote
- Musty

Odor of edible mushrooms

- None
- Almond
- Anise

Descriptive Data Analysis 5

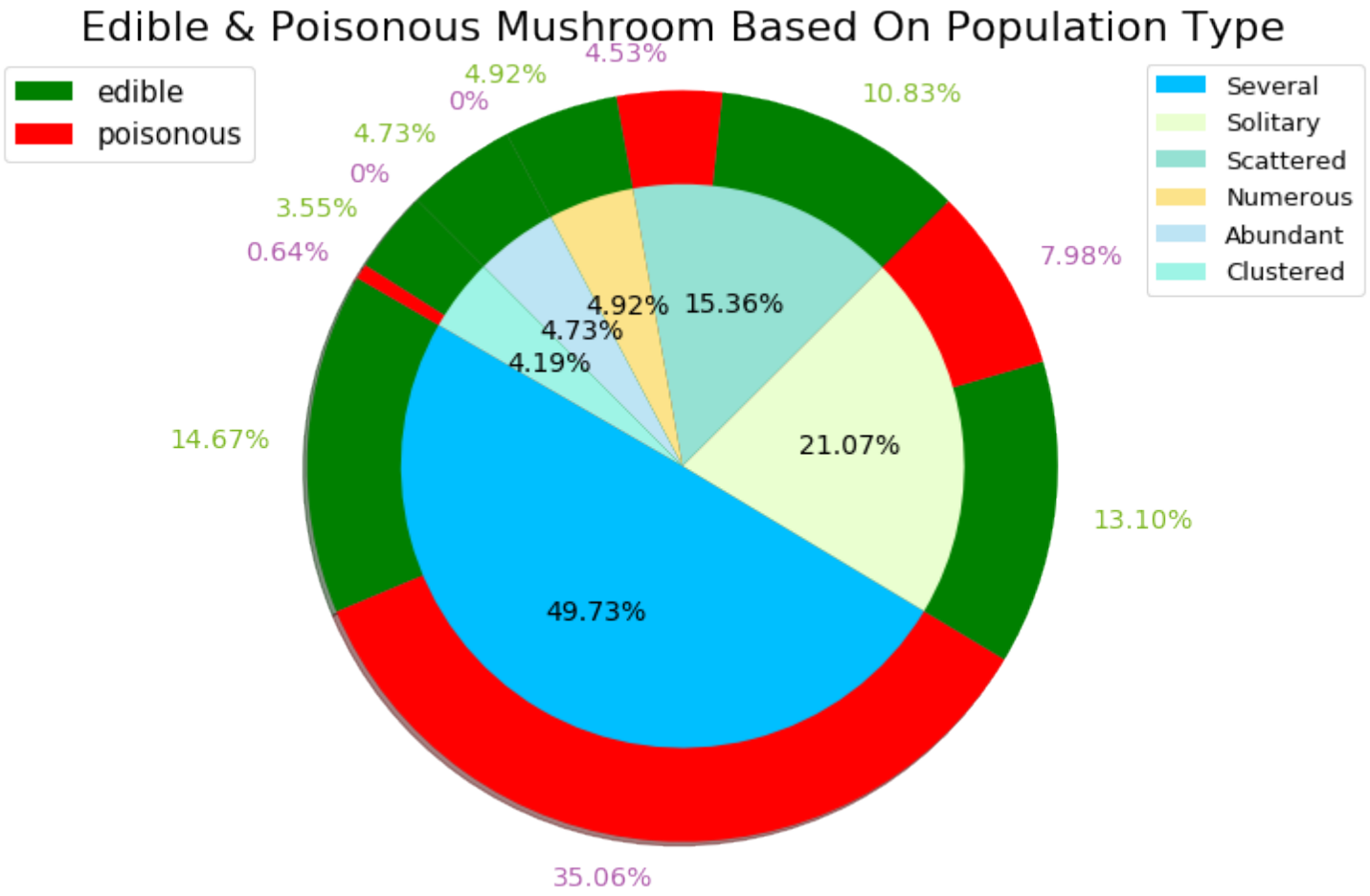


Top 3 population Type

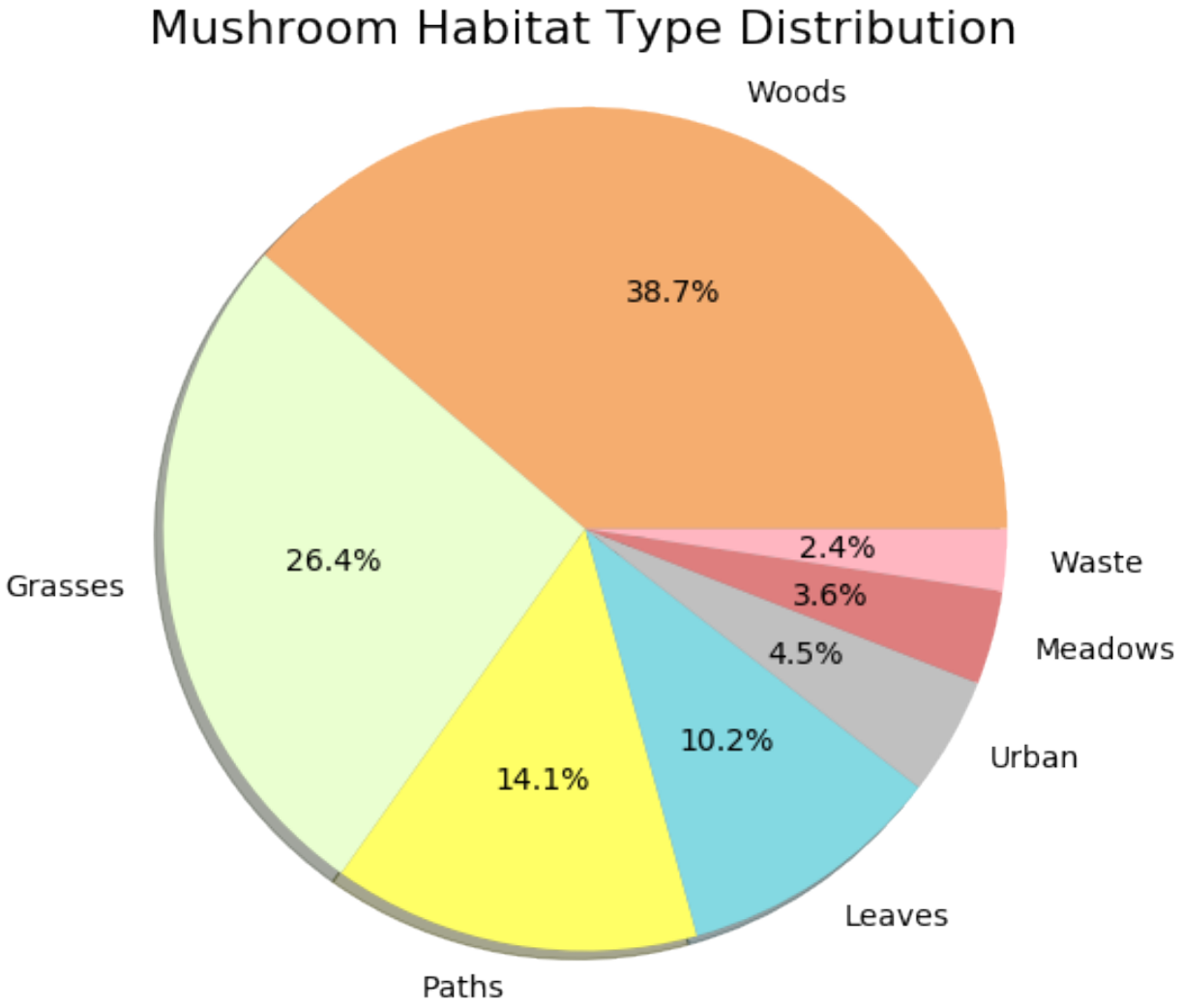
- Several
- Solitary
- Scattered

Descriptive Data Analysis 6

Population type does not give strong evidence on mushroom edibility



Descriptive Data Analysis 7

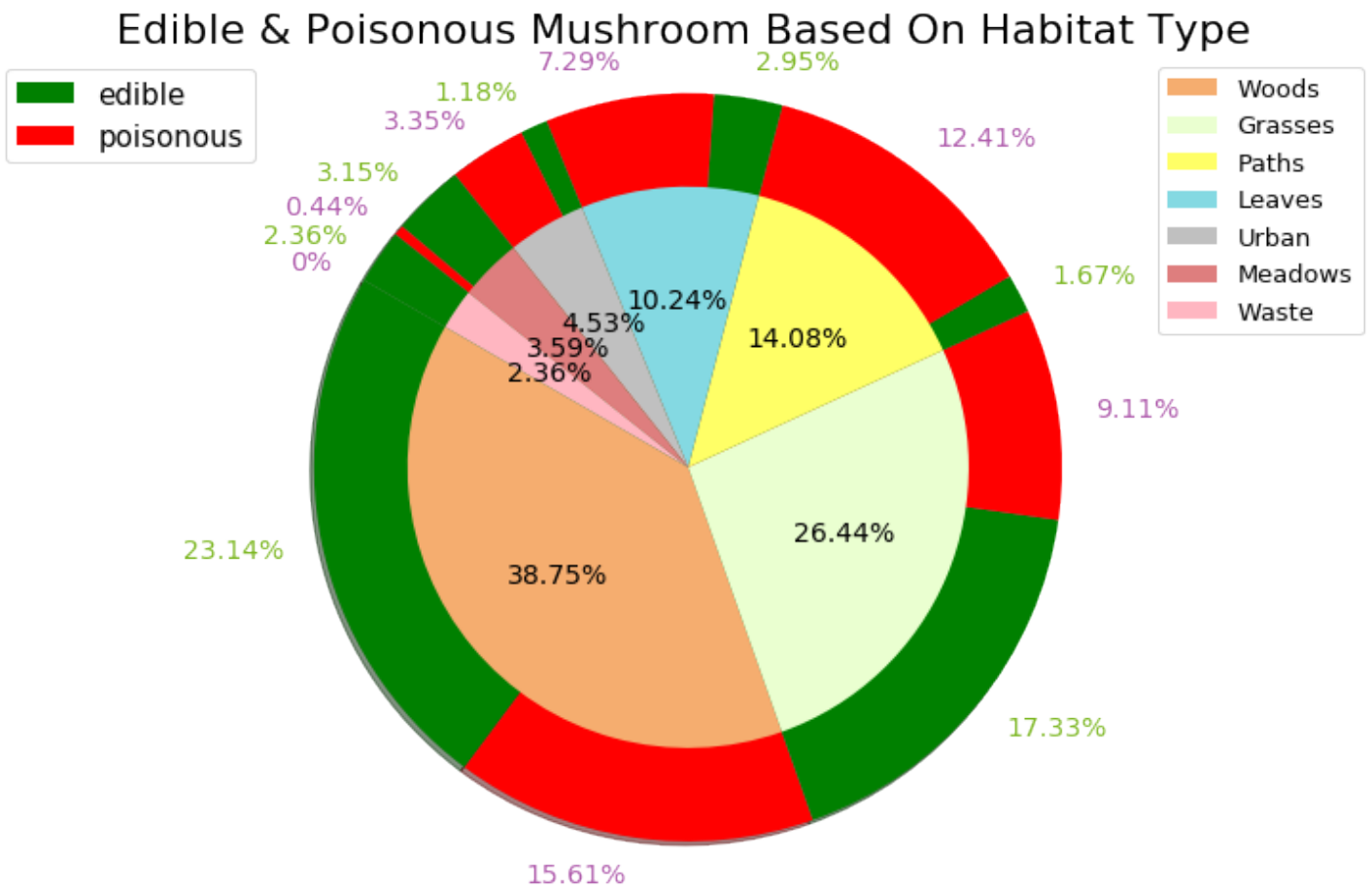


Main habitats of mushroom

- Woods
- Grasses
- Paths

Descriptive Data Analysis 8

Mushrooms in waste are 100% edible!



More poisonous habitats

- Leaves
- Paths
- Urban

More edible habitats

- Waste
- Meadow
- Woods
- Grasses

Content

- Data Description
- Objective
- Descriptive Data Analysis
- **Machine Learning Models**
- Conclusion

Data Preparation


```
from sklearn.preprocessing import LabelEncoder
labelencoder=LabelEncoder()
for col in mushrooms.columns:
    mushrooms[col] = labelencoder.fit_transform(mushrooms[col])
mushrooms.head()
```

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	population
0	1	5	2	4	1	6	1	0	1	4	...	2	7	7	0	2	1	4	2	3
1	0	5	2	9	1	0	1	0	0	4	...	2	7	7	0	2	1	4	3	2
2	0	0	2	8	1	3	1	0	0	5	...	2	7	7	0	2	1	4	3	2
3	1	5	3	8	1	6	1	0	1	5	...	2	7	7	0	2	1	4	2	3
4	0	5	2	3	0	5	1	1	0	4	...	2	7	7	0	2	1	0	3	0

- Data transformation from categorical to numerical

```
# Scale the data to be between -1 and 1
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X=scaler.fit_transform(X)
X
```

```
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]])
```

- Data normalization

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=4)
```

- Train/ test split

Models

Strategy :

- Apply default model with no tuning of the hyper parameter to keep the models simple
- Plot ROC curve for each algorithm to select the best machine learning model

Models:

- Logistic Regression
- Gaussian Naive Bayes
- Random Forest
- Decision Tree model
- KNN

Logistic Regression (default)


```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn import metrics

model_LR= LogisticRegression()
```

```
model_LR.fit(X_train,y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

```
print("Training accuracy:", 100*model_LR.score(X_train,y_train), "%")
print("Test Accuracy:", 100* model_LR.score(X_test,y_test), "%")
```

```
Training accuracy: 95.7685797815 %
Test Accuracy: 95.8153846154 %
```

```
scores_LR = cross_val_score(model_LR, X, y, cv=10, scoring='accuracy')
print(scores_LR)
```

```
[ 0.6703567   0.85854859  0.98154982  0.98523985  0.89544895  0.86100861
  0.99876847  0.99630542  0.62762022  0.93958076]
```

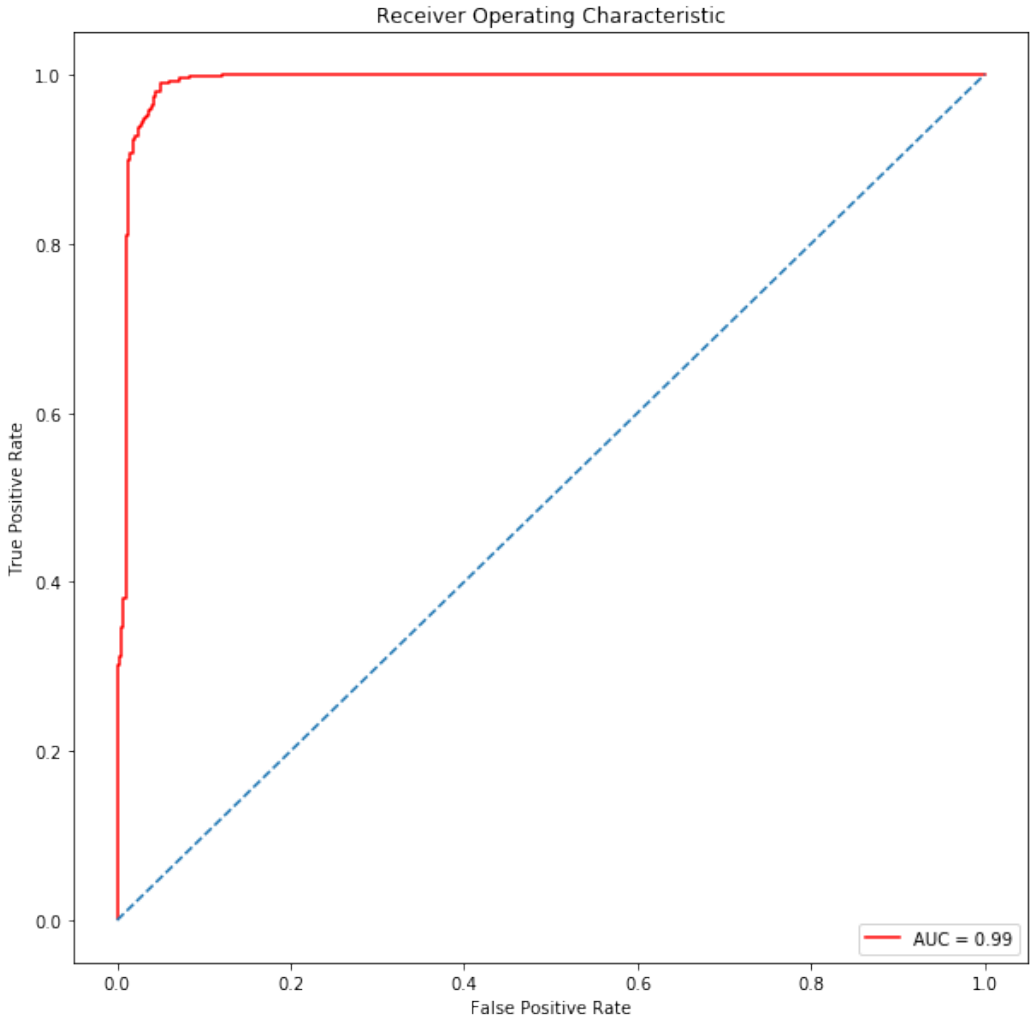
```
print("Accuracy with 10 fold cross validation:", 100*scores_LR.mean(), "%")
```

```
Accuracy with 10 fold cross validation: 88.1442739959 %
```

Accuracy = 95.8%

Accuracy w/ 10 fold CV: 88.1%

Logistic Regression (default)



Area under curve = 0.99

Gaussian Naive Bayes


```
from sklearn.naive_bayes import GaussianNB
model_naive = GaussianNB()
model_naive.fit(X_train, y_train)
```

```
GaussianNB(priors=None)
```

```
print("Training accuracy:", 100*model_naive.score(X_train,y_train), "%")
print("Test Accuracy:", 100* model_naive.score(X_test,y_test), "%")
```

```
Training accuracy: 91.9372211109 %
Test Accuracy: 93.1692307692 %
```

```
scores_NB = cross_val_score(model_naive, X, y, cv=10, scoring='accuracy')
print(scores_NB)
```

```
[ 0.59778598  0.78843788  0.97908979  0.9803198   0.84870849  0.81303813
  0.81034483  0.81650246  0.85326757  0.97533909]
```

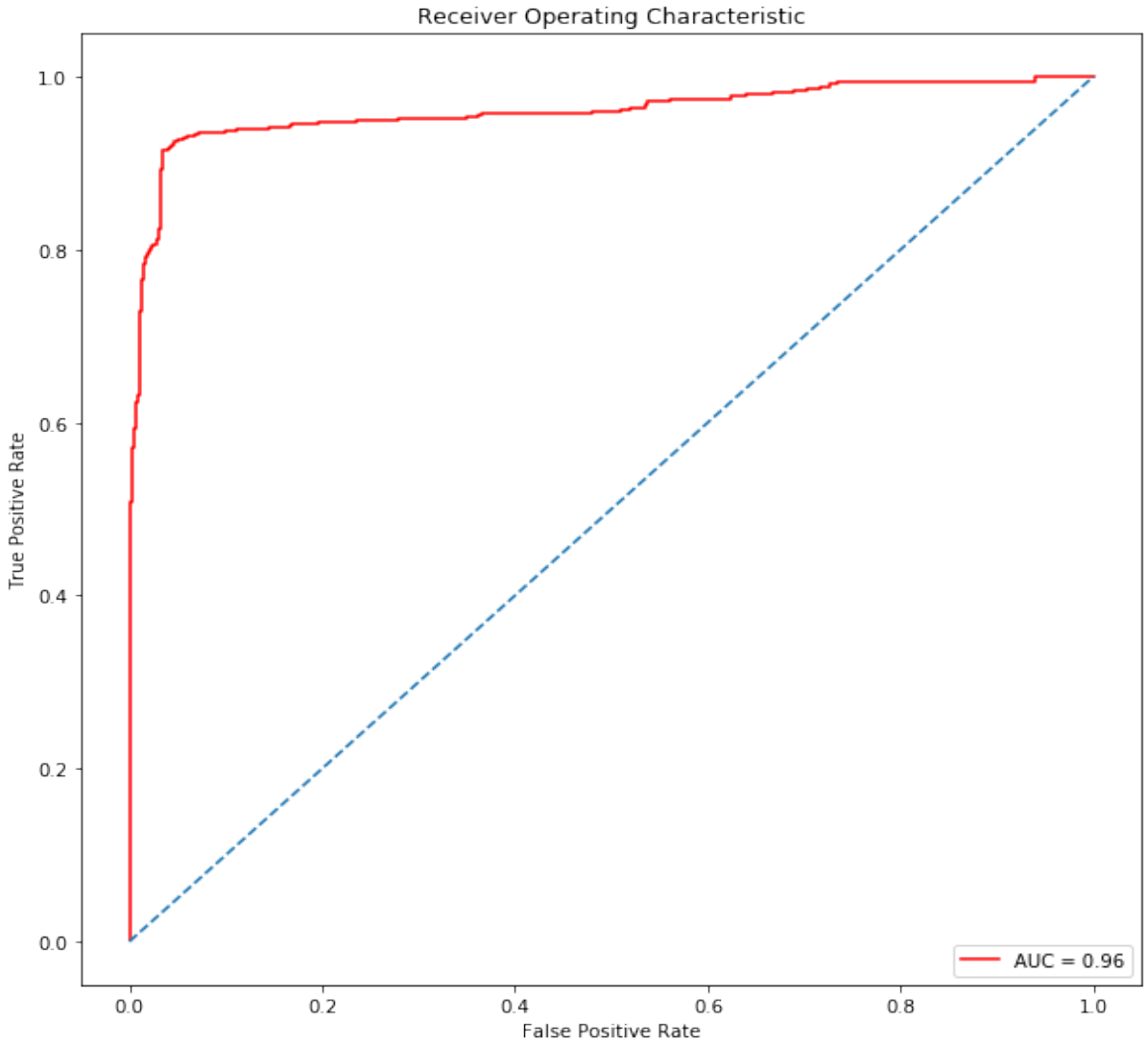
```
print("Accuracy with 10 fold cross validation:", 100*scores_NB.mean(), "%")
```

```
Accuracy with 10 fold cross validation: 84.6283402289 %
```

Accuracy = 93.2%

Accuracy w/ 10 fold CV: 84.6%

Gaussian Naive Bayes



Area under curve = 0.96

Random Forest

```
from sklearn.ensemble import RandomForestClassifier

model_RF=RandomForestClassifier()
```

```
model_RF.fit(X_train,y_train)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

```
print("Training accuracy:", 100*model_RF.score(X_train,y_train), "%")
print("Test Accuracy:", 100* model_RF.score(X_test,y_test), "%")
```

```
Training accuracy: 100.0 %
Test Accuracy: 100.0 %
```

```
scores_RF = cross_val_score(model_RF, X, y, cv=10, scoring='accuracy')
print(scores_RF)
```

```
[ 0.68511685  1.          1.          1.          1.          1.          1.
  1.          0.99013564  1.        ]
```

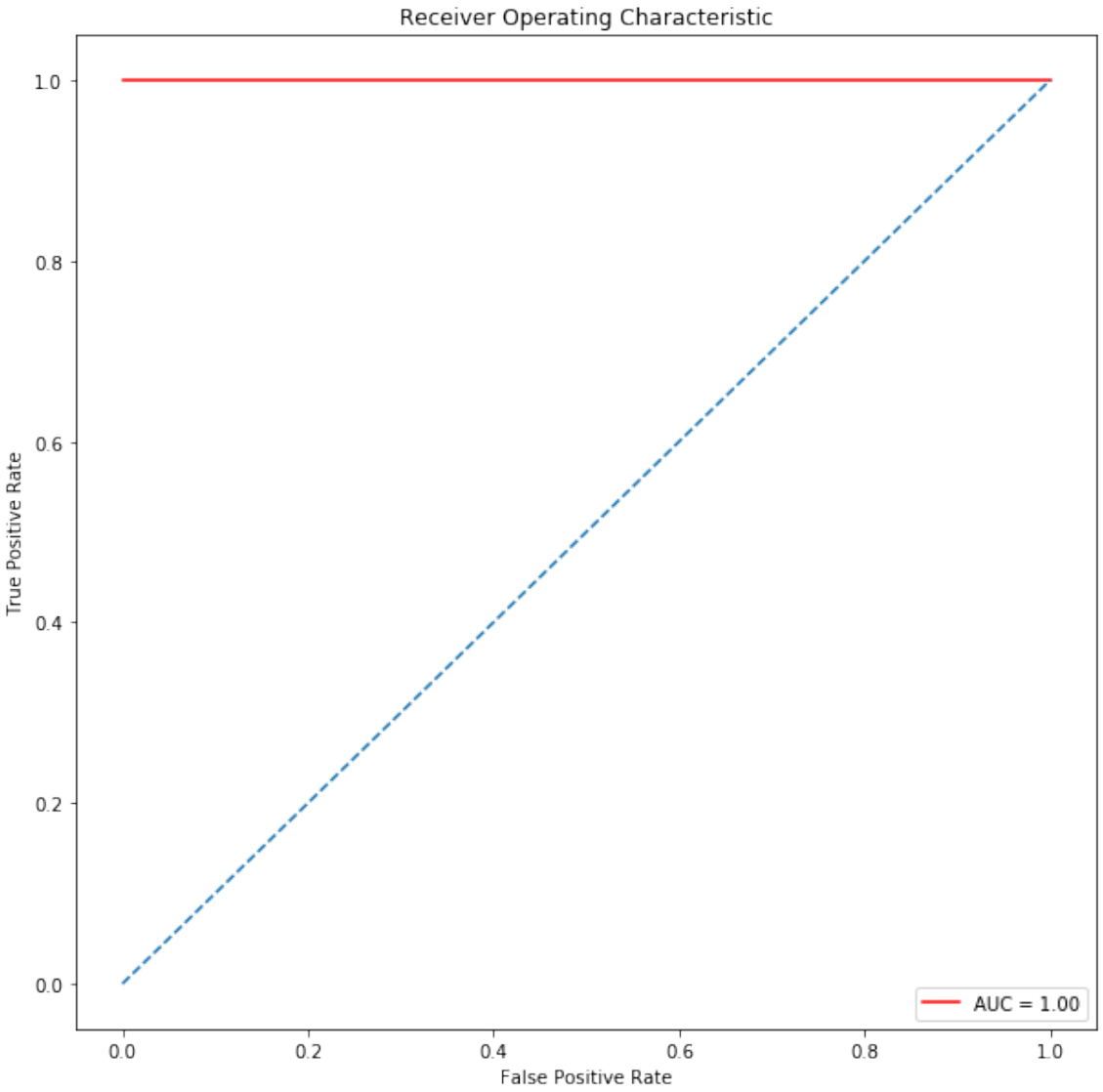
```
print("Accuracy with 10 fold cross validation:", 100*scores_RF.mean(), "%")
```

```
Accuracy with 10 fold cross validation: 96.7525248619 %
```

Accuracy = 100%

Accuracy w/ 10 fold CV: 96.8%

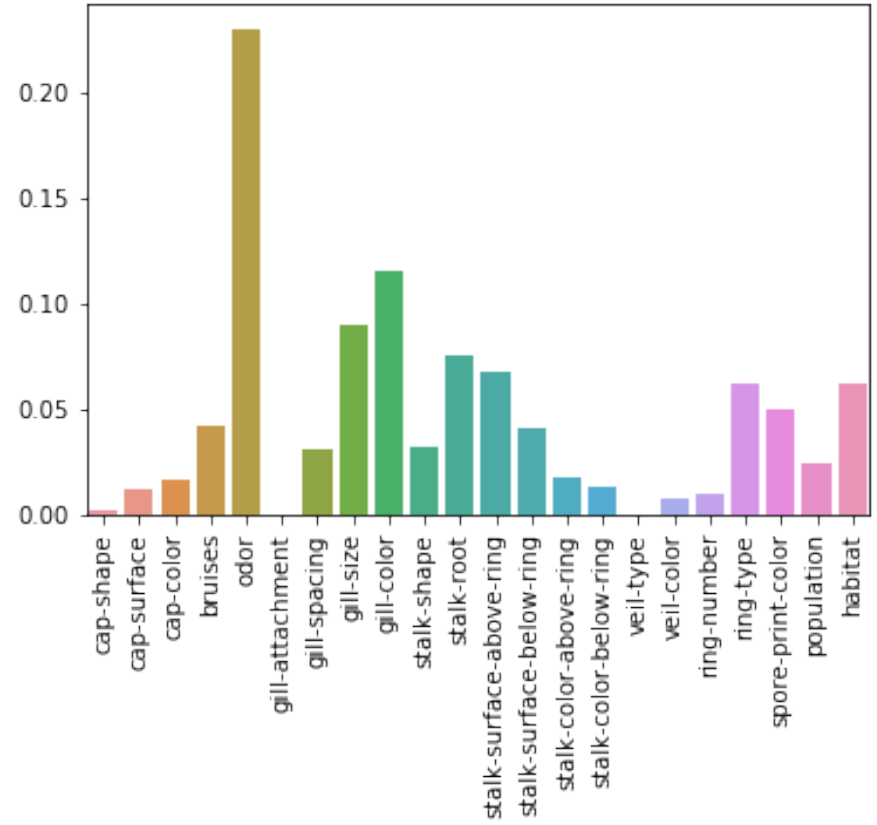
Random Forest

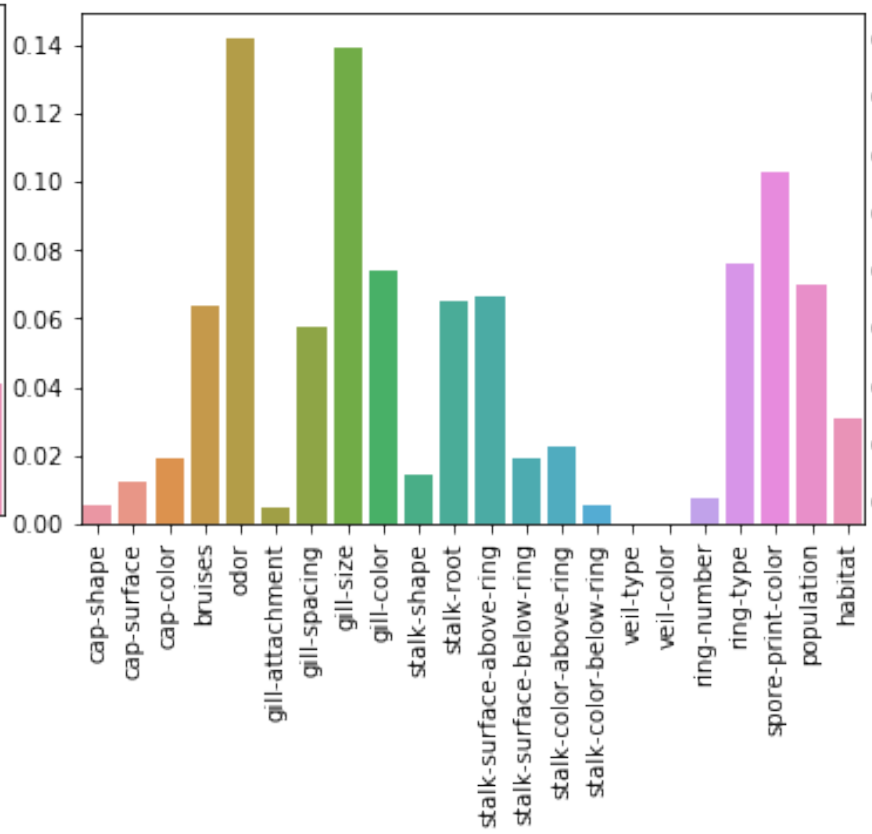


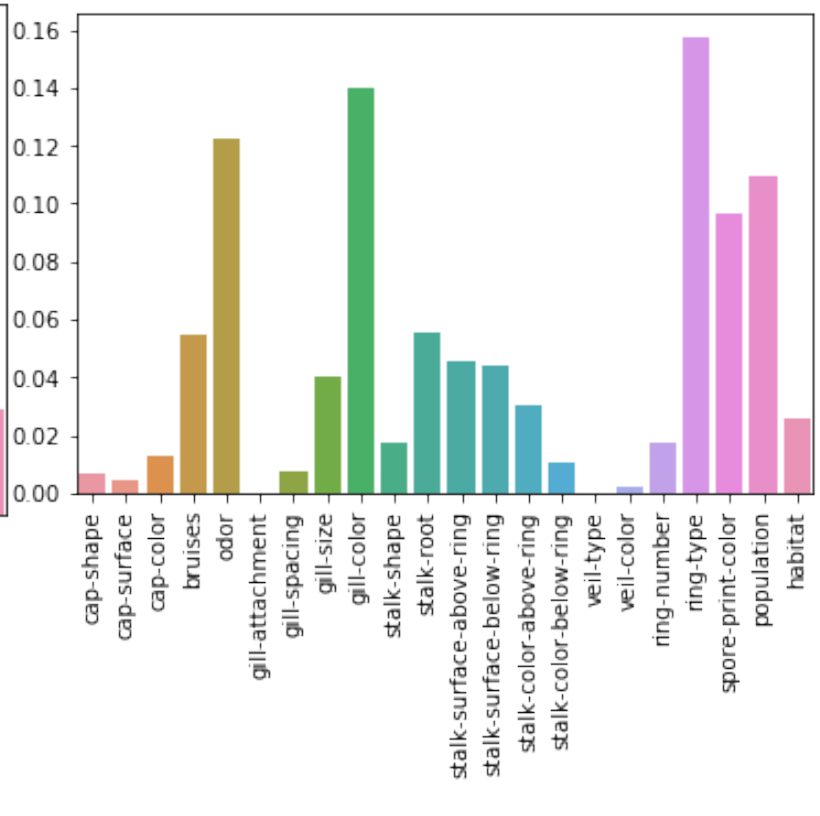
Area under curve = 1

Random Forest

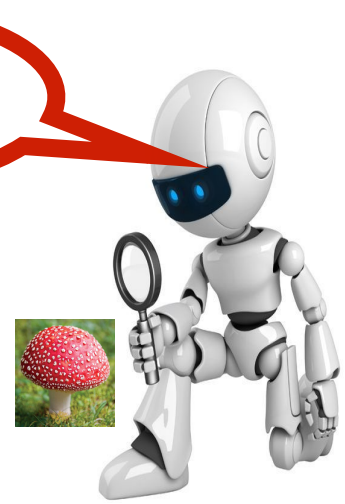
Most important features in 3 different runs of the model







Decision Tree model (default)




```
from sklearn.tree import DecisionTreeClassifier

model_tree = DecisionTreeClassifier()
```

```
model_tree.fit(X_train, y_train)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

```
print("Training accuracy:", 100*model_tree.score(X_train,y_train), "%")
print("Test Accuracy:", 100* model_tree.score(X_test,y_test), "%")
```

```
Training accuracy: 100.0 %
Test Accuracy: 100.0 %
```

```
scores_T = cross_val_score(model_tree, X, y, cv=10, scoring='accuracy')
print(scores_T)
```

```
[ 0.68511685  1.          1.          1.          1.          1.          1.
  1.          0.93341554  1.        ]
```

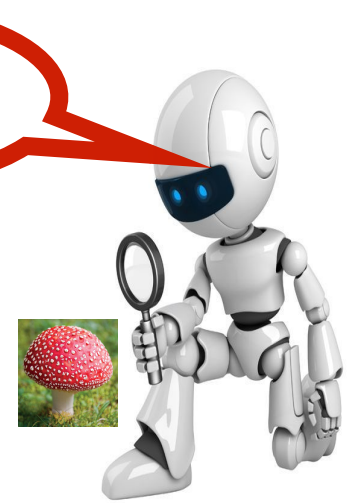
```
print("Accuracy with 10 fold cross validation:", 100*scores_T.mean(), "%")
```

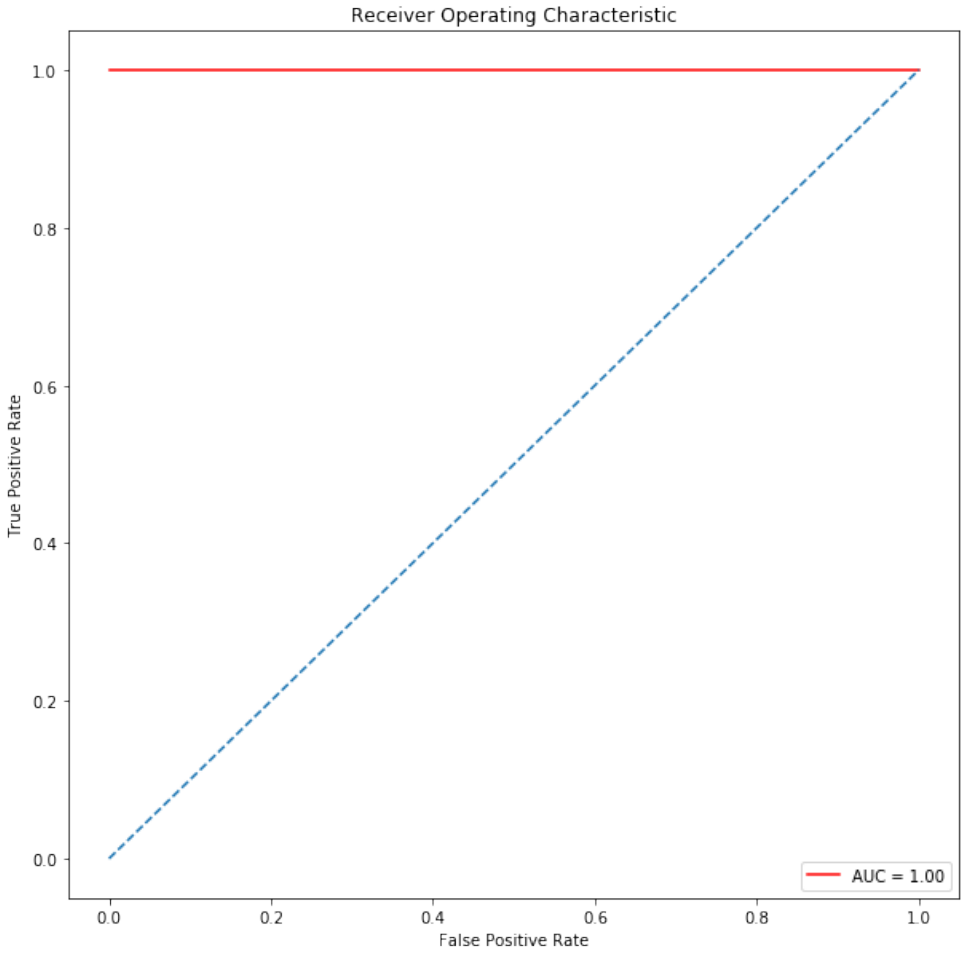
```
Accuracy with 10 fold cross validation: 96.1853238754 %
```

Accuracy = 100%

Accuracy w/ 10 fold CV: 96.1%

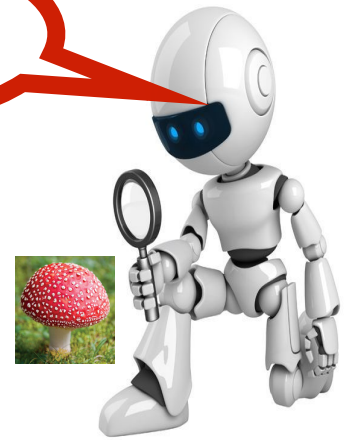
Decision Tree model (default)



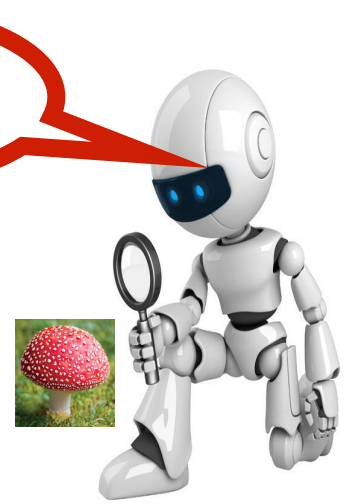


Area under curve = 1

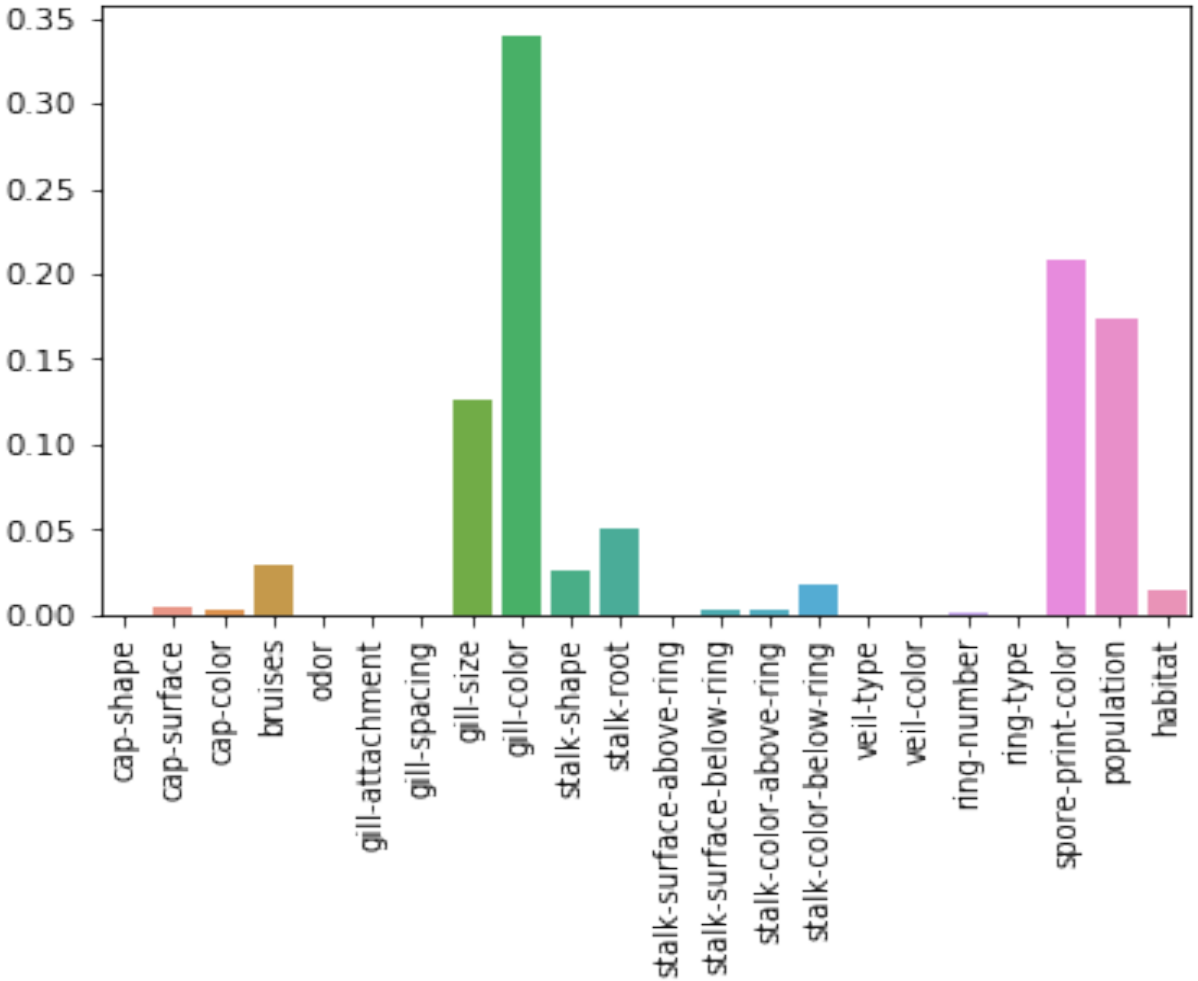
Decision Tree model (default)



Decision Tree model (default)



Feature importance in Decision Tree Model



KNN (Euclidean dist.)

```
from sklearn.neighbors import KNeighborsClassifier
```

```
model_knn = KNeighborsClassifier()
```

```
model_knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

```
print("Training accuracy:", 100*model_knn.score(X_train,y_train), "%")
print("Test Accuracy:", 100* model_knn.score(X_test,y_test), "%")
```

```
Training accuracy: 100.0 %
Test Accuracy: 100.0 %
```

```
scores_knn = cross_val_score(model_knn, X, y, cv=10, scoring='accuracy')
print(scores_knn)
```

```
[ 0.68634686  0.99753998  1.          0.99753998  0.99261993  1.          1.
  1.          0.7891492   0.99753391]
```

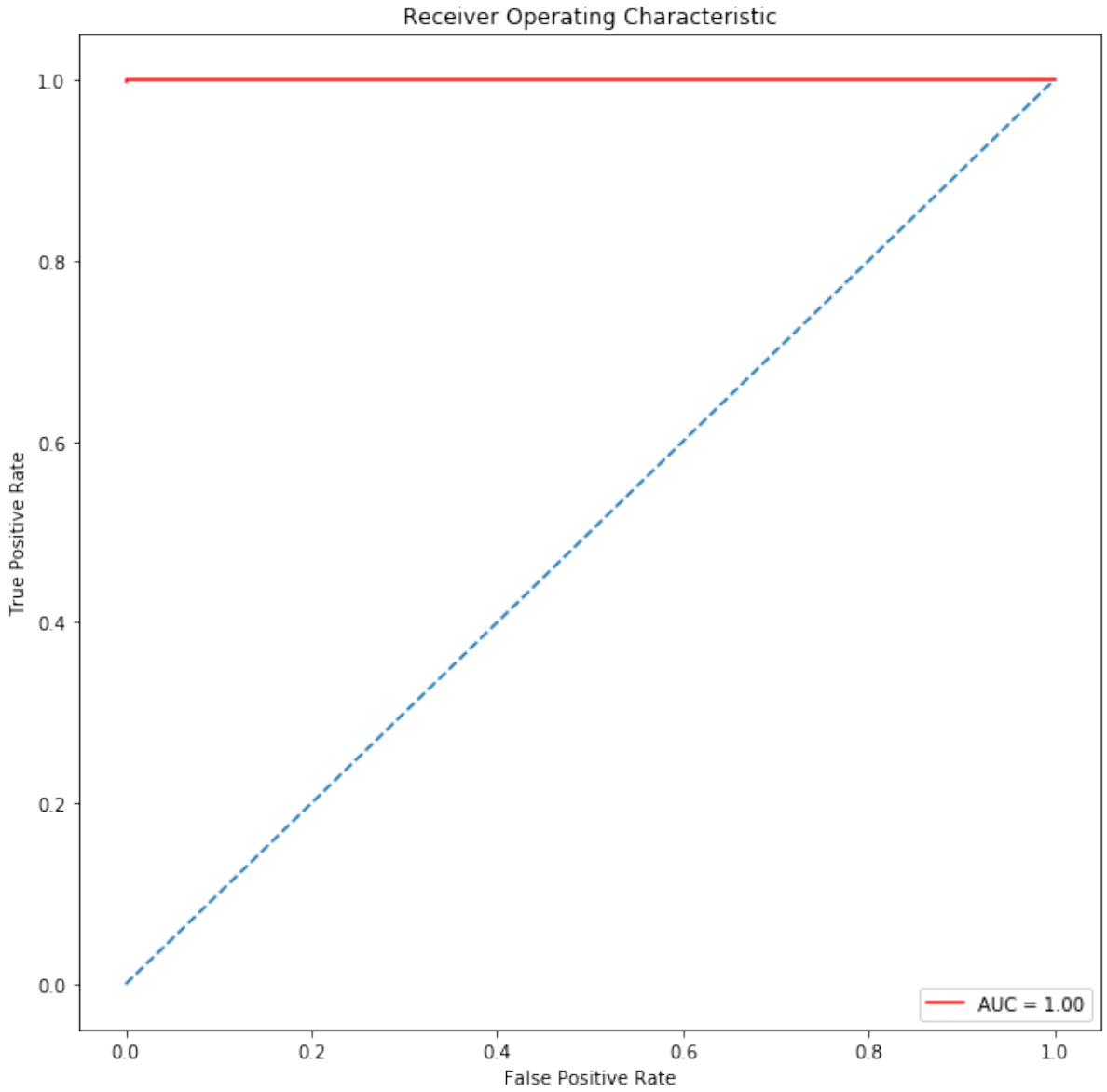
```
print("Accuracy with 10 fold cross validation:", 100*scores_knn.mean(), "%")
```

```
Accuracy with 10 fold cross validation: 94.6072984774 %
```

Accuracy = 100%

Accuracy w/ 10 fold CV: 94.6%

KNN



Area under curve = 1

KNN (Hamming dist.)


```
model_knnH = KNeighborsClassifier(metric='hamming')
```

```
model_knnH.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='hamming',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

```
print("Training accuracy:", 100*model_knnH.score(X_train,y_train), "%")
print("Test Accuracy:", 100* model_knnH.score(X_test,y_test), "%")
```

```
Training accuracy: 100.0 %
Test Accuracy: 100.0 %
```

```
scores_knnH = cross_val_score(model_knnH, X, y, cv=10, scoring='accuracy')
print(scores_knnH)
```

```
[ 0.68757688  1.          1.          1.          1.          1.          1.
  1.          0.86806412  1.        ]
```

```
print("Accuracy with 10 fold cross validation:", 100*scores_knnH.mean(), "%")
```

```
Accuracy with 10 fold cross validation: 95.5564099414 %
```

Accuracy = 100%

Accuracy w/ 10 fold CV: **95.6%**

Content

- Data description
- Objective
- Descriptive Data Analysis
- Machine Learning Models
- Conclusion

Conclusion

Model	Accuracy (%)	Accuracy w/ 10 fold Cross Validation (%)
Logistic Regression	95.8	88.1
Gaussian Naive Bayes	93.2	84.6
Random Forest	100	96.8
Decision Tree model	100	96.1
KNN (Euclidean dist.)	100	94.6
KNN (Hamming dist.)	100	95.6