2022

Jannik Scheider

iT Engineering Software Innovations GmbH

Project presentation

MUSHROOM CLASSIFICATION

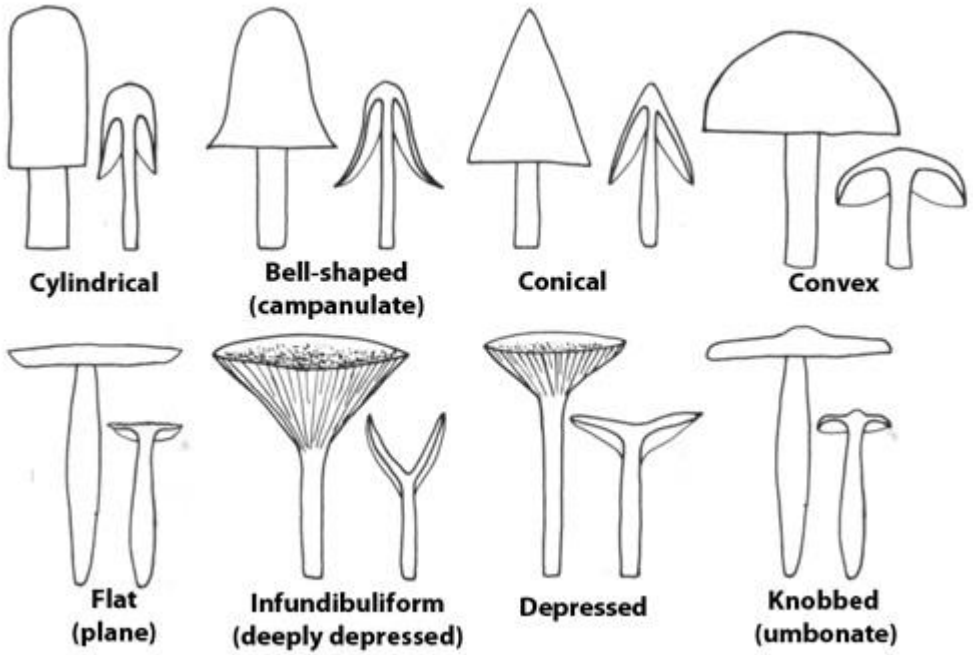
Table of contents

Introduction Dataset

- Kaggle Mushroom Classification
- Originally contributed to the UCI Machine Learning repository
- Includes desriptions of hypothetical samples corresponding to 23 species of gilled mushrooms
- Each species is identified as definitely edible, poisonous or of unknown edibility and not recommended
- 8124 entries
- 22 attributes

Introduction Dataset

Column cap-shape



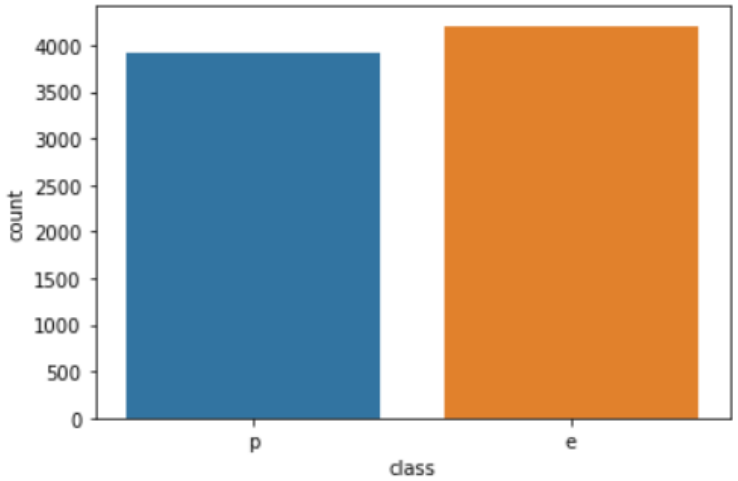
Important the dataset does not consider Cylindrical and Infundibuliform

EDA

- .head()
- .info()
- .describe()

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring
0	p	x	s	n	t	p	f	c	n	k	...	s
1	e	x	s	y	t	a	f	c	b	k	...	s
2	e	b	s	w	t	l	f	c	b	n	...	s
3	p	x	y	w	t	p	f	c	n	n	...	s
4	e	x	s	g	f	n	f	w	b	k	...	s

5 rows × 23 columns



Preparing Dataset

```
df = df.apply(lambda x: pd.factorize(x)[0])
X = df.drop(columns="class")
y = df["class"]
```

Classification Models

	Logistic Regression	Support Vector Machine
standard	0.986	0.998
Fine tuned	1.0	1.0

*values show the accuracy score

Classification Models

```python
from sklearn import svm

#Create a svm Classifier
clf = svm.SVC(kernel='linear', probability=True) # Linear Kernel

#Train the model using the training sets
clf.fit(X_train, y_train)

#Predict the response for test dataset
y_prob = clf.predict_proba(X_test)[:,1] # positive class prediction probabilities
y_pred = np.where(y_prob > 0.5, 1, 0) # This will threshold the probabilities to give class predictions.
```

```python
# Model Accuracy
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9980305268340719
```

Classification Models

Hyperparameter tuning

```
# defining parameter range
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf']}

grid = GridSearchCV(svm.SVC(), param_grid, refit = True, verbose = 3)

# fitting the model for grid search
grid.fit(x_train, y_train)
```

```
# print best parameter after tuning
print(grid.best_params_)

# print how our model looks after hyper-parameter tuning
print(grid.best_estimator_)

# print best accuracy score
print(grid.best_score_)
```

```
{'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}
SVC(C=1, gamma=0.1)
1.0
```

Classification Models

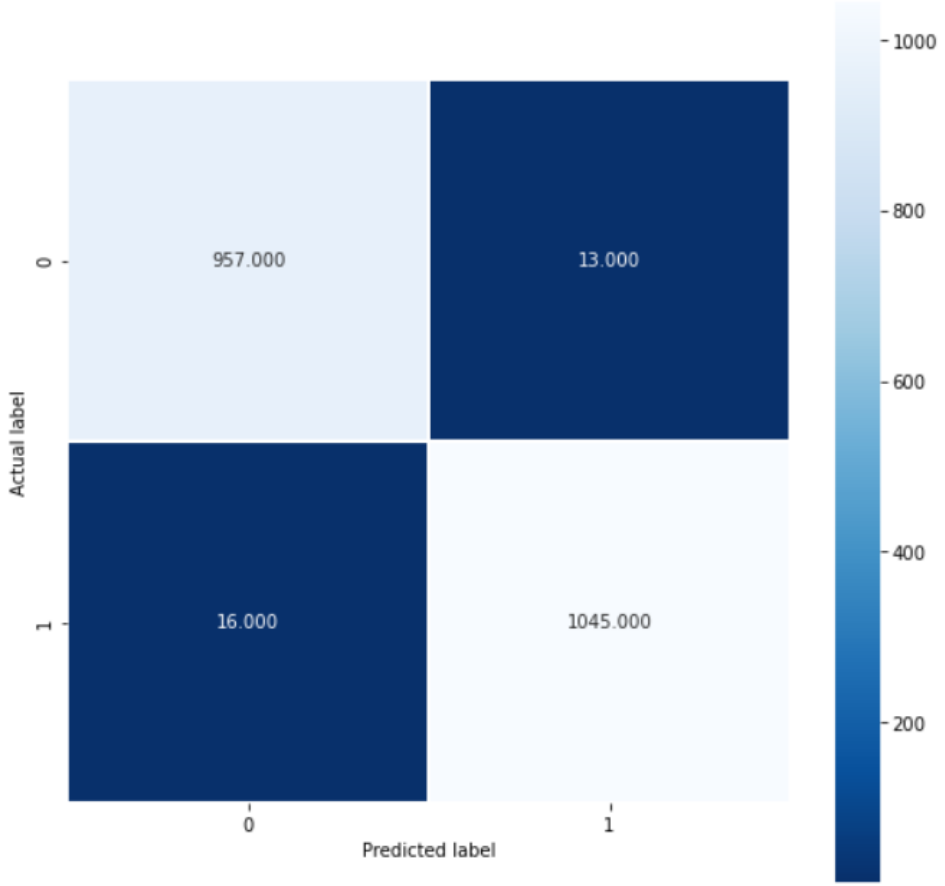
Logistic Regression Confusion Matrix

```
from sklearn import metrics
```

```
# confusion matrix for performance of the classification model
cm = metrics.confusion_matrix(y_test, y_pred)
print(cm)
```

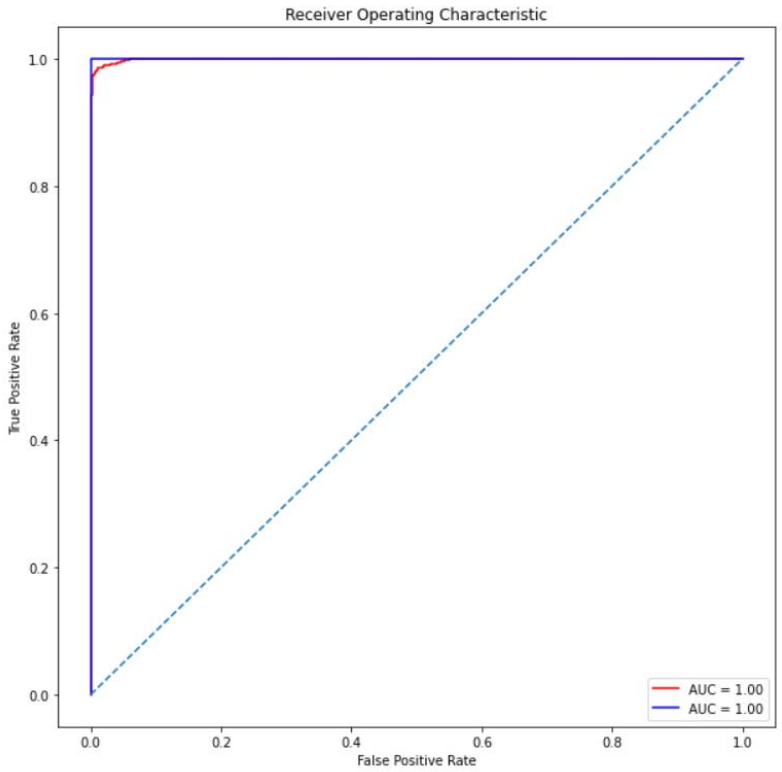
```
[[ 957   13]
 [  16 1045]]
```

```
plt.figure(figsize=(9,9))
sns.heatmap(cm, annot=True, fmt=".3f", linewidths=.5, square=True, cmap="Blues_r")
plt.ylabel("Actual label")
plt.xlabel("Predicted label")
all_sample_title = "Accuracy Score: {0}".format(score)
```

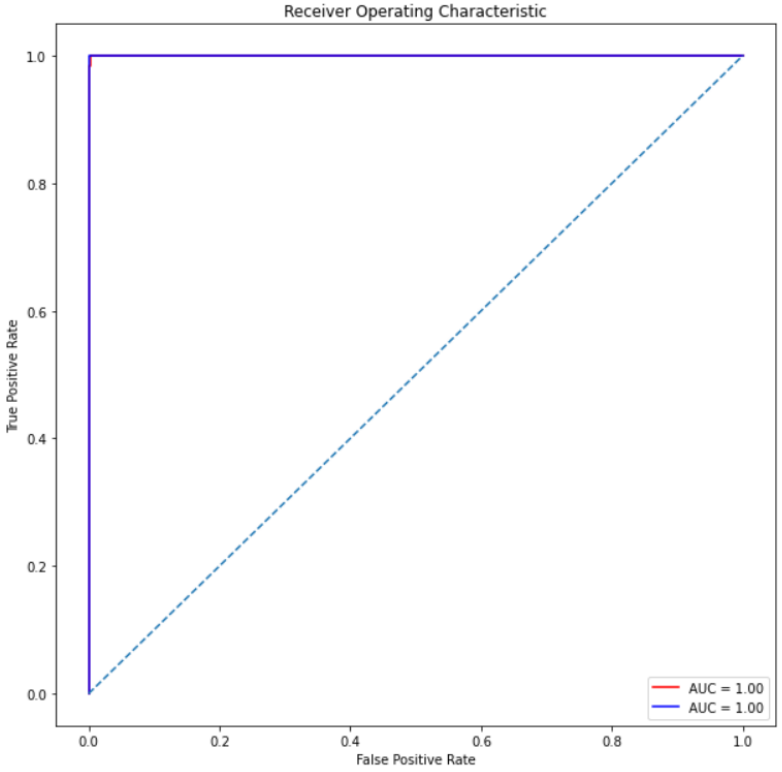


Evaluation
AUC - ROC Curve

Logistic Regression



SVM



Evaluation
AUC - ROC Curve

```
false_positive_rate_SVM, true_positive_rate_SVM, thresholds_SVM = roc_curve(y_test, y_prob)
roc_auc_SVM = auc(false_positive_rate_SVM, true_positive_rate_SVM)
```

```
false_positive_rate_SVM_tuned, true_positive_rate_SVM_tuned, thresholds_SVM_tuned = roc_curve(y_test, y_prob)
roc_auc_SVM_tuned = auc(false_positive_rate_SVM_tuned, true_positive_rate_SVM_tuned)
```

```
plt.figure(figsize=(10,10))
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate_SVM,true_positive_rate_SVM, color='red',label = 'AUC = %0.2f' % roc_auc_SVM)
plt.plot(false_positive_rate_SVM_tuned,true_positive_rate_SVM_tuned, color='blue',label = 'AUC = %0.2f' % roc_auc_SVM_tuned)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],linestyle='--')
plt.axis('tight')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
```

THANK YOU!