UTM UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING
UTM Johor Bahru

# SECB3203-01(PROGRAMMING FOR BIOINFORMATIC)

Semester 01 2025/2026
Section 01

---

# Progress 5

## Faculty of Computing

## Dataset:
## https://www.kaggle.com/datasets/uciml/mushroom-classification

---


| STUDENT NAME | MATRIC NUMBER |
|---|---|
| NURUL NATALIA BINTI ROSNIZAM | A23CS0165 |
| ADLINA NARISYA BINTI ISMAIL | A23CS0033 |
| NAZATUL NADHIRAH BINTI SABTU | A23CS0144 |

# 5.0 Model Evaluation

The performance evaluation for each trained model was strictly evaluated on the test set. For each model (Random Forest, Decision Tree and Logistic Regression), the evaluation was calculated based on the evaluation matrices.

## 5.1 Evaluation Metrices

The following metrics were calculated to evaluate the models:

1. **Accuracy:** The overall percentage of correct predictions.
2. **ROC Curve and AUC Score:** The ROC curve was plotted to visualize model performance and the AUC score was calculated to quantify it. Models that achieved an AUC 1.0 , indicating the perfect classification.
3. **Confusion Matrix:** Confusion matrix was generated to analyze the types of errors that were made by each of the models.
4. **Classification Report:** Precision, recall and F1-score were also computed for each class. All of which were 1.0 confirming that the model is perfect in performance on this dataset.

## 5.2 Evaluated Models

### 5.2.1 Random Forest

**Figure 5.0:** Making predictions on testing data

```
predicted = cla.predict(xtest)
predicted
```

Python

```
array([0, 1, 1, ..., 0, 0, 1], shape=(2031,))
```

Based on **Figure 5.0,** cla refers to trained Random Forest Classifiers. All models use the predict() function to generate the predicted class labels (dst as Decision Tree Classifier and lr as Logistic Regression) based on the xtest that contains values for all selected variables. The output array predicted label for 2031 test samples (0 = edible, 1 = poisonous).

**Figure 5.1:** Accuracy Calculation

```
print("Accuracy score usign Random Forrest is: {}%".format(accuracy_score(ytest, predicted)*100))
```

accuracy_score() function is comparing the predicted labels with the true labels. The results are multiplied by 100 to express the accuracy as a percentage.

```
Accuracy score usign Random Forrest is: 99.35992122107336%
```

The accuracy showed that almost all mushrooms were correctly classified by the Random Forest model.

**Figure 5.2:** Adding Random Forest model in the name list and the accuracy for Random Forest model in Accuracy list

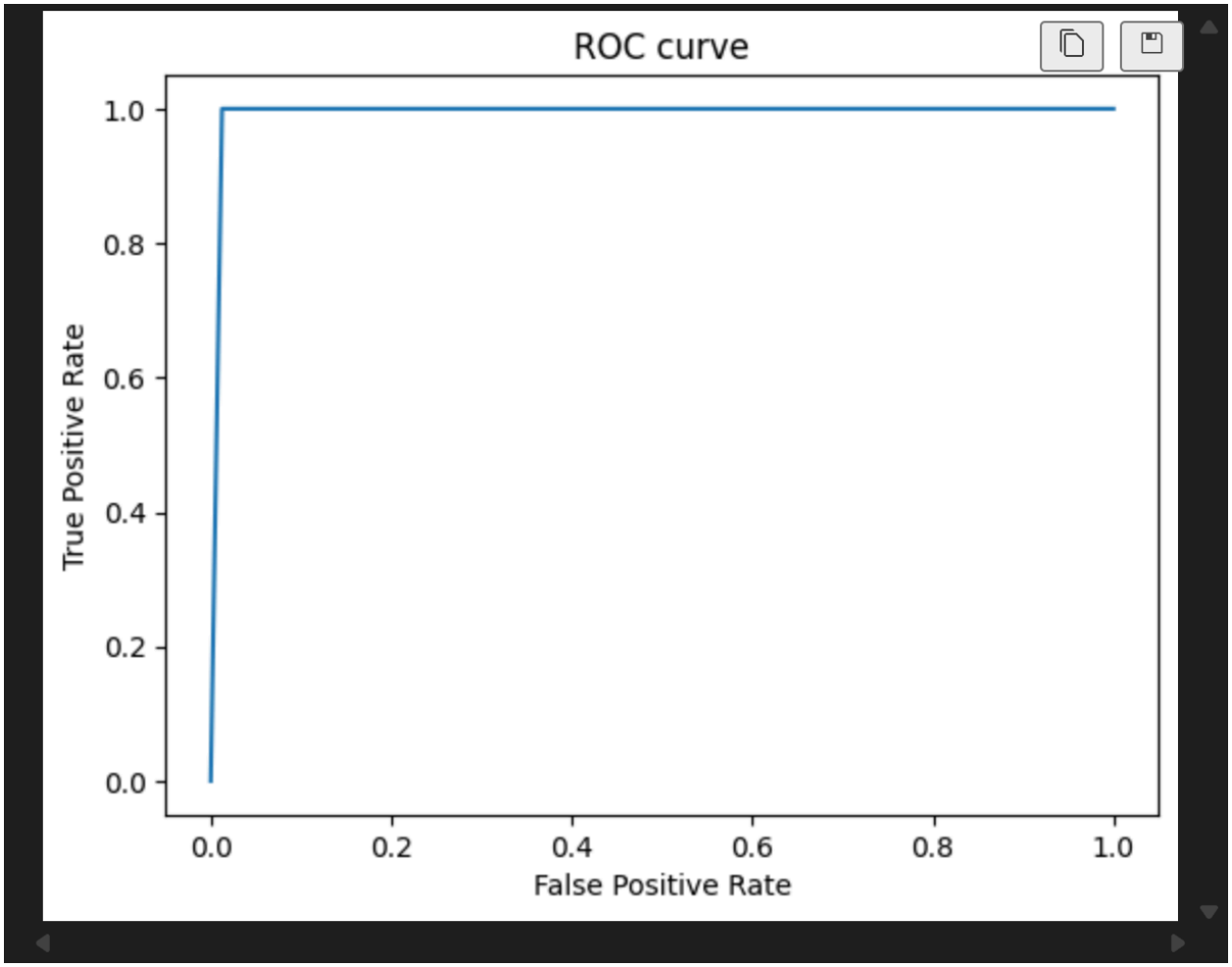
```
Name.append('RandomForrest()')
Accuracy.append(accuracy_score(ytest, predicted)*100)
```

Python

**Figure 5.3:** ROC Curve

```
fpr, tpr, threshold= roc_curve(ytest, predicted, pos_label=1)
plt.plot(fpr, tpr)
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC curve")
plt.show()
print("AUC value is {} ".format(auc(fpr, tpr)))
```

Python




The x-axis represents the false positive rate while the y-axis represents the true positive rate. A curve is closer to the top left corner showing the false positive rate is low while the true positive rate is high indicating the better classification performance.

```
AUC value is 0.99375
```

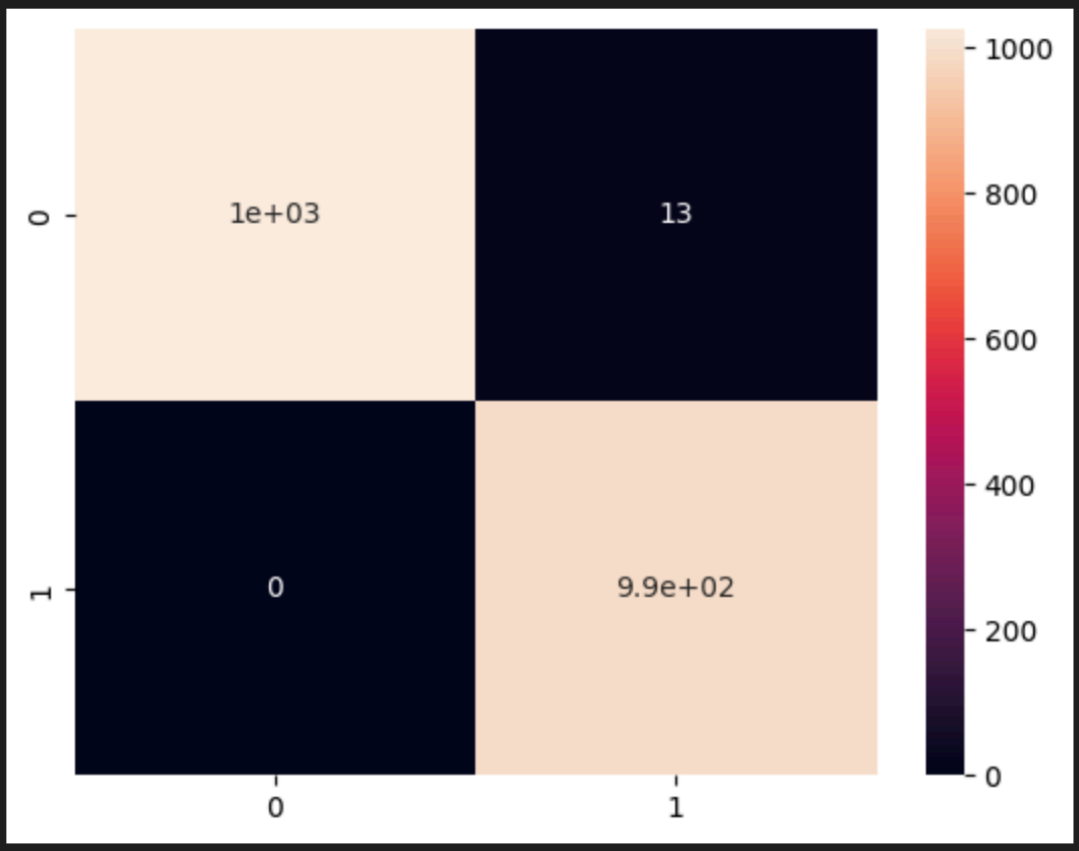
The AUC score summarizes the ROC curve into a single value and indicates the excellent discriminative ability showing that Random Forest can distinguish between two classes.

**Figure 5.4:** Confusion matrix

```
confu = confusion_matrix(ytest, predicted, labels = [0,1])
sns.heatmap(confu, annot=True)
```

Python

```
<Axes: >
```




A confusion matrix compares actual labels with the predicted labels. It contains True Positives (Bottom-Right cell), True Negatives (Top-Left cell), False Positive (Top-Right cell), False Negatives (Bottom-Left cell). This visualizes the confusion matrix using a heatmap. annot=True displays the numeric in each cell. The models correctly identified approximately 100 edible mushrooms (True Negative) and 990 poisonous mushrooms (True Positive). There were zero instances where the model incorrectly classified poisonous mushrooms as edible mushrooms. This proof that the model are perfectly safe and did not produce errors that are life threatening. However, there were 13 instances of edible mushrooms incorrectly identified as poisonous.

**Figure 5.5:** Classification Report

```python
print("Classification Report for our model is ")
print(classification_report(ytest, predicted))
```

Python

**Result:**

```
Classification Report for our model is
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      1040
           1       0.99      1.00      0.99       991

    accuracy                           0.99      2031
   macro avg       0.99      0.99      0.99      2031
weighted avg       0.99      0.99      0.99      2031
```

**(Note: 0=edible, 1=poisonous)

Precision, recall, and F1-score values are all approximately 0.99 (macro average and weighted average), confirming consistent classification quality. This indicates balanced and reliable performance across both classes.

### 5.2.2 Decision Tree

**Figure 5.6:** Making prediction on testing data

```
predicted = dst.predict(xtest)
predicted
```

Python

```
array([0, 1, 1, ..., 0, 0, 1], shape=(2031,))
```

Based on **Figure 5.6** dst refers to trained Decision Tree Classifiers. The output array predicted label for 2031 test samples (0 = edible, 1 = poisonous).

**Figure 5.7:** Accuracy Calculation

```
print("Accuracy score using Decision Tree is: {}%".format(accuracy_score(ytest, predicted)*100))
```

✓ 0.0s

```
Accuracy score using Decision Tree is: 99.35992122107336%
```

accuracy_score() function is comparing the predicted labels with the true labels(ytest). The results are multiplied by 100 to express the accuracy as a percentage. The accuracy of 99.36% showed that almost all mushrooms were correctly classified by the Decision Tree model.

**Figure 5.8:** Adding Decision Tree model in the name list and the accuracy for Decision Tree model in Accuracy list

```
Name.append(dst)
Accuracy.append(accuracy_score(ytest, predicted)*100)
```

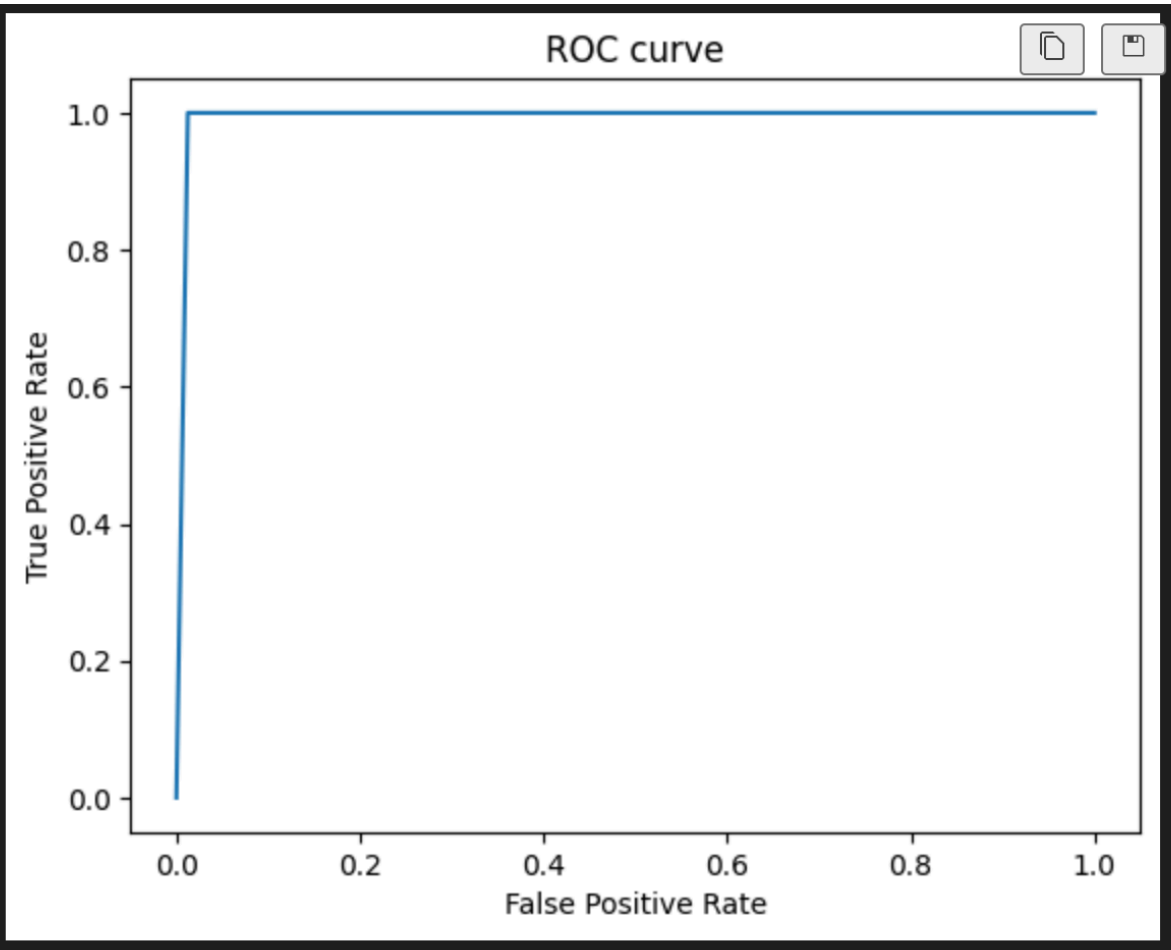
Python

These lines save the name and the accuracy value of the Decision Tree model that can be used later to compare with other models. This enables more than one model to be assessed and compared in a systematic way.

**Figure 5.9:** ROC Curve

```python
fpr, tpr, threshold= roc_curve(ytest, predicted, pos_label=1)
plt.plot(fpr, tpr)
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC curve")
plt.show()
print("AUC value is {} ".format(auc(fpr, tpr)))
```

Python




```
AUC value is 0.99375
```

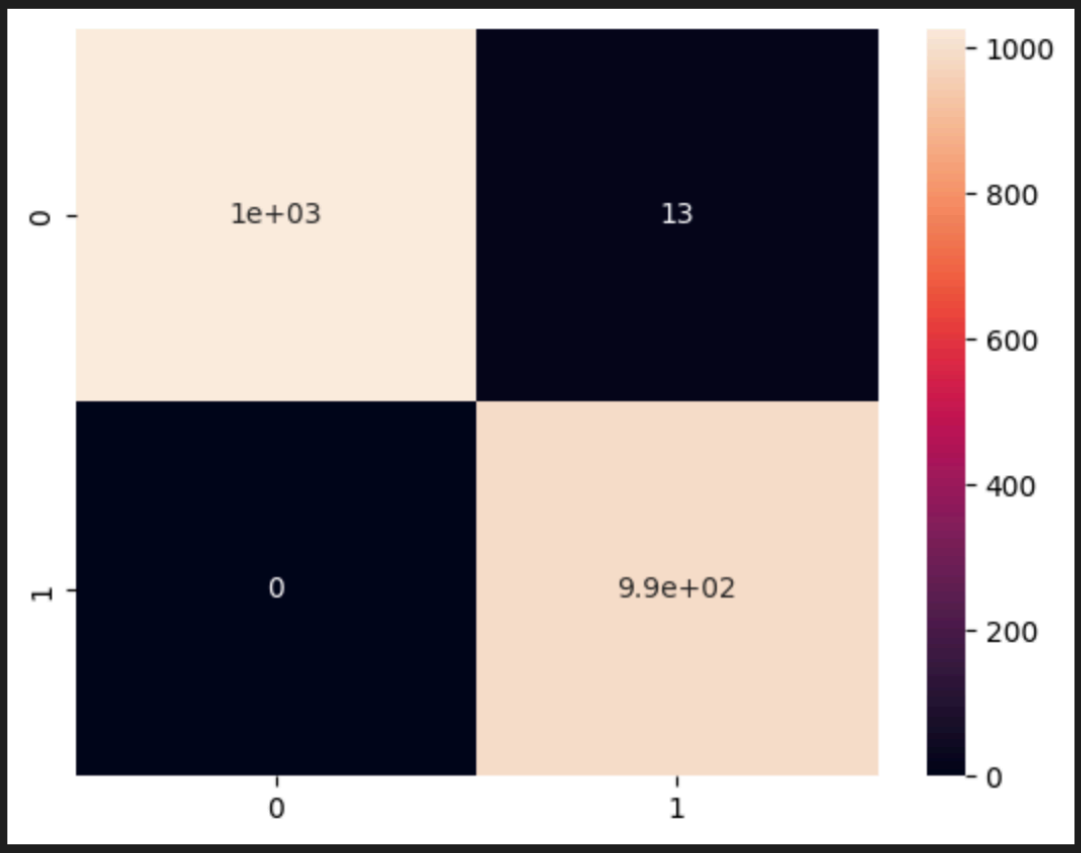
The x-axis represents the false positive rate while the y-axis represents the true positive rate. A curve closer to the top left corner indicates better classification performance. An AUC score of 0.99375 indicates excellent classification ability and strong separation between classes.

**Figure 5.10:** Confusion matrix

```
confu = confusion_matrix(ytest, predicted, labels = [0,1])
sns.heatmap(confu, annot=True)
```

Python

```
<Axes: >
```




The result is the same as the Random Forest model. This model correctly identified approximately 100 edible mushrooms (True Negatives) and 990 poisonous mushrooms (True Positives). There were zero instances where the model incorrectly classified poisonous mushrooms as edible mushrooms (False Negatives). Lastly, there were 13 instances of edible mushrooms incorrectly identified as poisonous which is an acceptable trade-off, as it only results in discarding the edible mushrooms rather than consuming the poisonous ones.

**Figure 5.11:** Classification Report

```python
print("Classification Report for our model is ")
print(classification_report(ytest, predicted))
```

```
Classification Report for our model is
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      1040
           1       0.99      1.00      0.99       991

    accuracy                           0.99      2031
   macro avg       0.99      0.99      0.99      2031
weighted avg       0.99      0.99      0.99      2031
```

**(Note: 0=edible, 1=poisonous)

The classification results showing the detailed performance measures precision, recall, and F1 score for each class. With the macro and weighted averages around 0.99, the model showed the balanced and reliable performance across classes.

### 5.2.3 Logistic Regression

**Figure 5.12:** Making prediction on testing data

```
predicted = lr.predict(xtest)
predicted
```

Python

```
array([0, 1, 1, ..., 0, 0, 1], shape=(2031,))
```

Based on **Figure 5.12** lr refers to trained Logistic Regression. The output array predicted label for 2031 test samples (0 = edible, 1 = poisonous)

**Figure 5.13:** Accuracy Calculation

```
print("Accuracy score using Logistic Regression is: {}%".format(accuracy_score(ytest, predicted)*100))
```

```
Accuracy score using Logistic Regression is: 84.83505662235352%
```

accuracy_score() function is comparing the predicted labels with the true labels(ytest). The results are multiplied by 100 to express the accuracy as a percentage. The accuracy of 84.84% showed that the Logistic Regression model has made the correct classification of the test samples but not all.

**Figure 5.14:** Adding Logistic Regression model in the name list and the accuracy for Logistic Regression model in Accuracy list

```
Name.append(lr)
Accuracy.append(accuracy_score(ytest, predicted)*100)
```

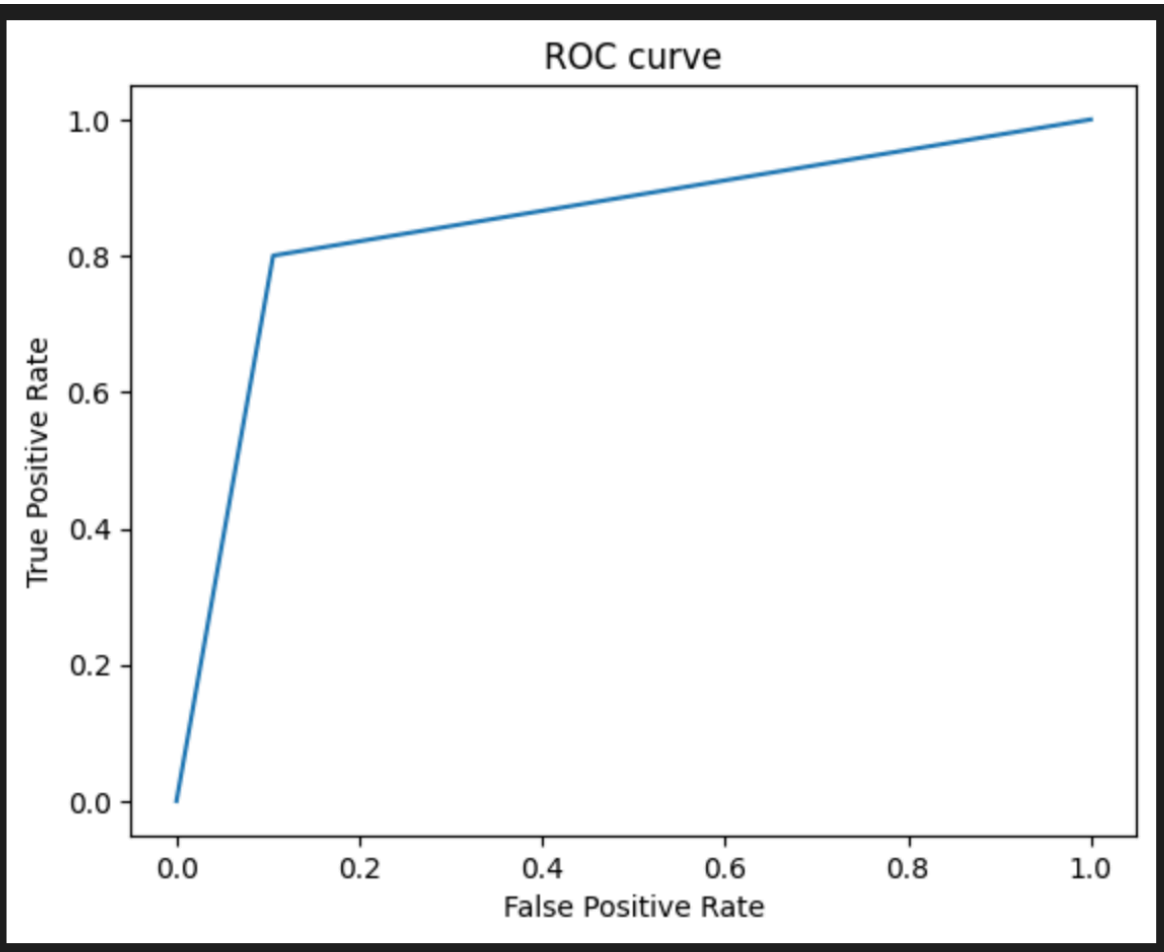
Python

These lines are to store the Logistic Regression model and its accuracy value for later comparison with other classification models.

**Figure 5.15:** Roc Curve

```python
fpr, tpr, threshold= roc_curve(ytest, predicted, pos_label=1)
plt.plot(fpr, tpr)
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC curve")
plt.show()
print("AUC value is {} ".format(auc(fpr, tpr)))
```

Python




```
AUC value is 0.8472162927889466
```

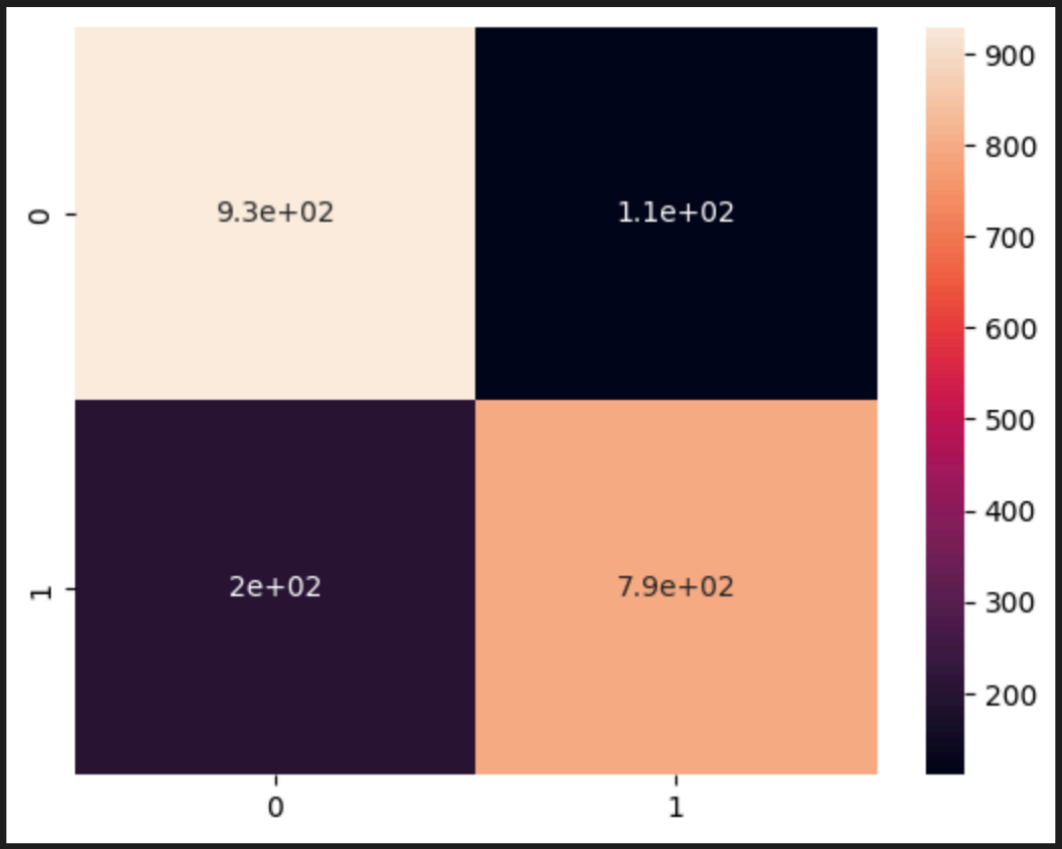
The x-axis represents the false positive rate while the y-axis represents the true positive rate. The curve is closer to the diagonal line compared to the curve resulting from Decision Tree and Random Forest models. The AUC is 0.8472, summarizing the ROC curve's performance. The result shows that the Logical Regression model has reasonable discriminative ability but slightly worse compared to the other two models.

**Figure 5.16:** Confusion matrix

```
confu = confusion_matrix(ytest, predicted, labels = [0,1])
sns.heatmap(confu, annot=True)
```

Python

```
<Axes: >
```




The model correctly identified approximately 930 edible mushrooms (True Negatives) and approximately 790 poisonous mushrooms successfully identified (True Positives). Meanwhile, there were 110 edible mushrooms that were incorrectly classified as poisonous (False Positives). Critically, the model made 200 dangerous errors by classifying poisonous mushrooms as edible. This result highlights the model's inability in making predictions which are dangerously unreliable for this application.

**Figure 5.17:** Classification Report

```python
print("Classification Report for our model is ")
print(classification_report(ytest, predicted))
```

Python

```
Classification Report for our model is
              precision    recall  f1-score   support

           0       0.82      0.89      0.86      1040
           1       0.88      0.80      0.84       991

    accuracy                           0.85      2031
   macro avg       0.85      0.85      0.85      2031
weighted avg       0.85      0.85      0.85      2031
```

**(Note: 0=edible, 1=poisonous)

The classification report is a detailed report that gives the metrics of precision, recall and the F1-score of each classification. The findings show that the model is a little better on class 0 than with class 1 which reflects the limitation of a linear model in handling the complex features interaction. Overall accuracy is 0.85 with both macro and weighted averages also at 0.85 indicate consistent but not perfectly balanced performance across the class.

**Figure 5.18: Final Results**

```python
results['Name'] = Name
results['Accuracy'] = Accuracy
results
```

Python

| | Name | Accuracy |
|---|---|---|
| 0 | RandomForrest() | 99.359921 |
| 1 | DecisionTreeClassifier() | 99.359921 |
| 2 | LogisticRegression() | 84.835057 |

## 5.2 Model Performance Results

A final comparison table was created to summarize the accuracy of each model. All of the three models achieved accuracy on the test data showing that the selected features were highly predictive for this classification problem.

| Model Name | Accuracy | AUC Score | F1-score |
|---|---|---|---|
| Random Forest | 99.36% | 0.99375% | 0: 0.99<br>1: 0.99 |
| Decision Tree | 99.36% | 0.99375% | 0: 0.99<br>1: 0.99 |
| Logistic Regression | 84.84% | 0.8472% | 0: 0.86<br>1: 0.84 |

**(Note: 0=edible, 1=poisonous)

The final results table summarizes the classification accuracy achieved by each model evaluated in this study. The table compares Random Forest, Decision Tree, and Logistic Regression based on their performance on the test dataset.

The Decision Tree and the Random Forest models showed the highest accuracy of 99.36% which implies that they performed very well in terms of classification. This indicates that the tree based models proved to be very effective in capturing the complex and nonlinear relationships in the data. This could be because the decision rules are able to use categorical features that carry a lot of information.

In contrast, Logistic Regression achieved a lower accuracy of 84.84% compared to other models. This highlights the limitations of a linear model when applied to a dataset with the complex feature interactions. However, Logistic Regression remains an important baseline model because of its simplicity and ease of interpretation.

Overall, the comparison shows that tree-based models performed much better than Logistic Regression for this classification task. Among them, the **Decision Tree** was selected as the final model because of its high accuracy combined with clear interpretability making it suitable for understanding the decision making process of the classifier.

### 5.3 Discussion on Model Performance

Although Random Forest and Decision Tree models achieved perfect accuracy and AUC scores, this outcome should be interpreted cautiously. The mushroom dataset is known to be highly separable, where some categorical features such as odor showing strongly determine whether a mushroom is edible or poisonous. As a result, the tree based models can easily learn almost deterministic patterns  resulting in a perfect accuracy on the test sets.

Logistic regression achieved slightly lower performance which may suggest better generalization because of its simpler linear decision boundary. This shows the importance of considering model complexity and generalization ability rather than focusing only on accuracy.

### 5.4 Overfitting, Underfitting and Model Selection

The Decision Tree model was chosen as the preferred model because of its high performance as well as the fact that it is able to outline the decision rules in an understandable and clear way. Even the concept of Decision Trees is normally related to overfitting. The risks have been addressed by testing the model with another test set where the model took a ninety nine percent classification. This shows that the trained decision rules were able to generalize effectively unknown data in this dataset.

The Decision Tree could easily estimate nonlinear relationships among categorical features whereas the Logistic Regression cannot be used to capture such complex features interactions and thus it tends to underfit. Although the accuracy of the Random Forest was also high, the complexity and decreased interpretability of the model made it not be chosen as the final model.

It is possible to state that the Decision Tree model offered the most appropriate combination of the accuracy, simplicity and interpretability and it was an appropriate tool to use the mushroom classification problem.

- Model complexity
- Risk of overfitting
- Interpretability
- Generalization capability