Problem Statement:

Choosing machine learning domain, our dataset is tailored for classification tasks. With labelled data available, our objective is to leverage supervised learning algorithms, such as Logistic Regression, Random Forest, Support Vector Machine (SVM), and others, to effectively categorize instances.

Basic Dataset Info:

The dataset comprises 399 rows and 28 columns.

Pre-processing Method:

The pre-processing involves converting categorical variables into numerical format, particularly addressing nominal data. This is achieved using the get_dummies method from the pandas library, implementing one-hot encoding to transform categorical variables into binary numerical format, ensuring compatibility with machine learning algorithms.

Best Model - Logistic Regression:

- Utilized hyperparameter tuning via GridSearchCV.
- Best parameters used were: LogisticRegression(C=1, max_iter=300, solver='saga').
- The optimal parameters utilized for Logistic Regression were: C=1, max_iter=300, and solver='saga'. With these settings, the model achieved an impressive accuracy of 0.99. Furthermore, it exhibited balanced F1-scores and a perfect ROC AUC Score of 1.0.

Model Evaluation Results:

- **Random Forest Classification Report and AUC & ROC Score:**
  - The classification report shows high precision, recall, and F1-score for both classes (0 and 1), indicating good performance.
  - The ROC AUC score is very high (close to 1.0), indicating excellent performance in distinguishing between positive and negative classes.

```
#Classification report
classification_rep = classification_report(y_test, test_predictions)
print("Classification Report:")
print(classification_rep)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        45
           1       0.99      0.99      0.99        75

    accuracy                           0.98       120
   macro avg       0.98      0.98      0.98       120
weighted avg       0.98      0.98      0.98       120
```

```
# ROC AUC score
test_probabilities = best_random_forest.predict_proba(X_test)[:, 1]
roc_auc = roc_auc_score(y_test, test_probabilities)
print(f"ROC AUC Score: {roc_auc}")
```

```
ROC AUC Score: 0.9997037037037038
```

- **Gaussian Naive Bayes Classification Report and AUC & ROC Score:**
  - The classification report also shows high precision, recall, and F1-score for both classes.
  - The ROC AUC score is perfect (1.0), indicating perfect performance in distinguishing between positive and negative classes.

```
[[45  0]
 [ 2 73]]
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        45
           1       1.00      0.97      0.99        75

    accuracy                           0.98       120
   macro avg       0.98      0.99      0.98       120
weighted avg       0.98      0.98      0.98       120

ROC AUC Score:: 1.0
```

- **Decision Tree Classification Report and AUC & ROC Score:**
  - The classification report shows decent performance with lower precision, recall, and F1-score compared to Random Forest and Naive Bayes.
  - The ROC AUC score is good but slightly lower than the other models.

```
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.96      0.92        45
           1       0.97      0.93      0.95        75

    accuracy                           0.94       120
   macro avg       0.93      0.94      0.94       120
weighted avg       0.94      0.94      0.94       120
```

```
1 # ROC AUC score
2 test_probabilities = best_decision_tree.predict_proba(X_test)[:, 1]
3 roc_auc = roc_auc_score(y_test, test_probabilities)
4 print(f"ROC AUC Score: {roc_auc}")
```

```
ROC AUC Score: 0.9444444444444445
```

- **SVM Classification Report and AUC & ROC Score:**
  - The classification report and ROC AUC score indicate excellent performance, like Random Forest and Naive Bayes.

```
#Classification report
classification_rep = classification_report(y_test, test_predictions)
print("Classification Report:")
print(classification_rep)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        45
           1       1.00      0.97      0.99        75

    accuracy                           0.98       120
   macro avg       0.98      0.99      0.98       120
weighted avg       0.98      0.98      0.98       120
```

```
test_scores = best_svm_classifier.decision_function(X_test)
```

```
# Calculating ROC AUC score
roc_auc = roc_auc_score(y_test, test_scores)
print(f"ROC AUC Score: {roc_auc}")
```

```
ROC AUC Score: 1.0
```

  -
- **KNN Classification Report and AUC & ROC Score:**
  - The classification report shows good performance, but the ROC AUC score is slightly lower compared to Random Forest and Naive Bayes.

```
#Classification report
classification_rep = classification_report(y_test, test_predictions)
print("Classification Report:")
print(classification_rep)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.88      1.00      0.94        45
           1       1.00      0.92      0.96        75

    accuracy                           0.95       120
   macro avg       0.94      0.96      0.95       120
weighted avg       0.96      0.95      0.95       120
```

```
# ROC AUC score
test_probabilities = best_knn_neighbour.predict_proba(X_test)[:, 1]
roc_auc = roc_auc_score(y_test, test_probabilities)
print(f"ROC AUC Score: {roc_auc}")
```

```
ROC AUC Score: 0.9995555555555555
```

- **Logistic Regression Classification Report and AUC & ROC Score:**
  - The classification report and ROC AUC score indicate excellent performance, like Random Forest and Naive Bayes.

```
#Classification report
classification_rep = classification_report(y_test, test_predictions)
print("Classification Report:")
print(classification_rep)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        45
           1       1.00      0.99      0.99        75

    accuracy                           0.99       120
   macro avg       0.99      0.99      0.99       120
weighted avg       0.99      0.99      0.99       120
```

```
# ROC AUC score
test_probabilities = best_logistic_regression.predict_proba(X_test)[:, 1]
roc_auc = roc_auc_score(y_test, test_probabilities)
print(f"ROC AUC Score: {roc_auc}")
```

```
ROC AUC Score: 1.0
```

Conclusion:

Conclusion: In conclusion, after evaluating multiple classification models including Random Forest, Gaussian Naive Bayes, Support Vector Machine (SVM), KNN, and Logistic Regression, it is evident that Random Forest and Logistic Regression stand out as strong candidates for model deployment.

Logistic Regression demonstrates exceptional precision and recall for both classes, with an accuracy of 0.99. Its balanced F1-scores, along with a perfect ROC AUC Score of 1.0, underscore its superior performance among all evaluated models. Therefore,

Logistic Regression emerges as the recommended model for deployment in this scenario.