National College of Ireland

# Leveraging Advanced Machine Learning Models to Analyse Mental Health in the Era of Social Media and Digital Platforms

MSc Research Project
Data Analytics

Shivani Nityanand Shedole
Student ID: X22218688

School of Computing
National College of Ireland

Supervisor: Dr Abid Yaqoob

National College of Ireland
Project Submission Sheet
School of Computing

| | |
|---|---|
| **Student Name:** | Shivani Nityanand Shedole |
| **Student ID:** | X22218688 |
| **Programmer:** | Data Analytics |
| **Year:** | 2024 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Dr Abid Yaqoob |
| **Submission Due Date:** | 12/12/2024 |
| **Project Title:** | Leveraging Advanced Machine Learning Models to Analyse Mental Health in the Era of social media and Digital Platforms |
| **Word Count:** | 982 |
| **Page Count:** | 12 |

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

**ALL** internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

| | |
|---|---|
| **Signature:** | |
| **Date:** | 12/12/2024 |

## PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies). | ☐ |
| **Attach a Moodle submission receipt of the online project submission**, to each project (including multiple copies). | ☐ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | ☐ |

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

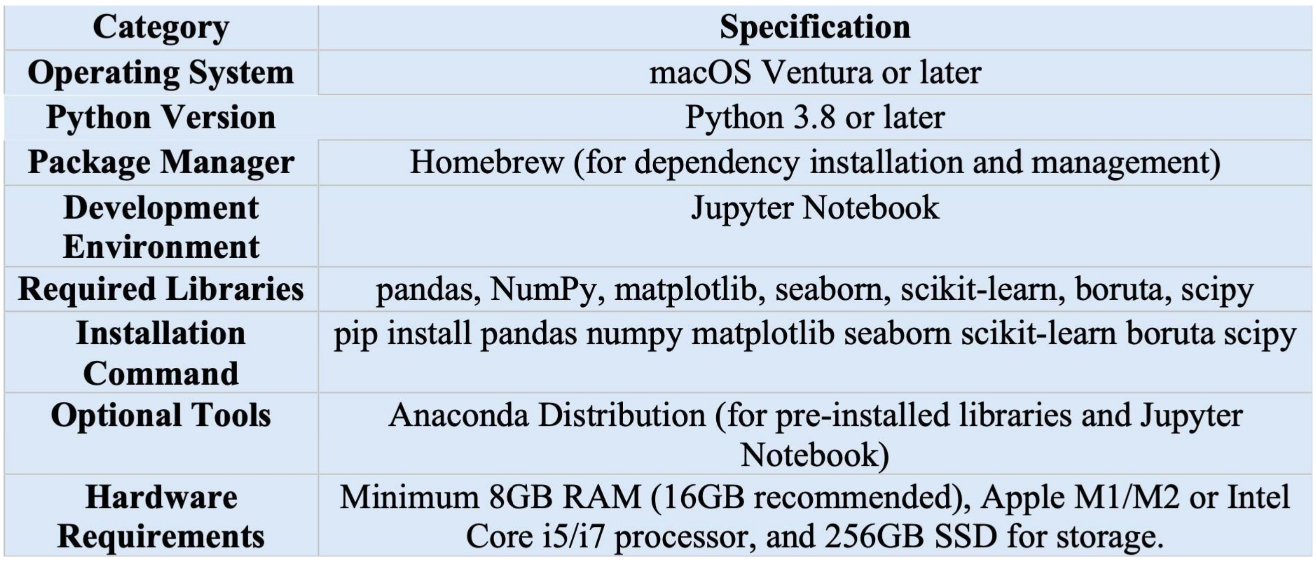
| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Leveraging Advanced Machine Learning Models to Analyse Mental Health in the Era of Social Media and Digital Platforms

Shivani Nityanand Shedole
X22218688

## 1 Hardware Specifications

The hardware used for this research study is an Apple MacBook air laptop with 8Gb ram and MacOS operating system as shown in the figure.

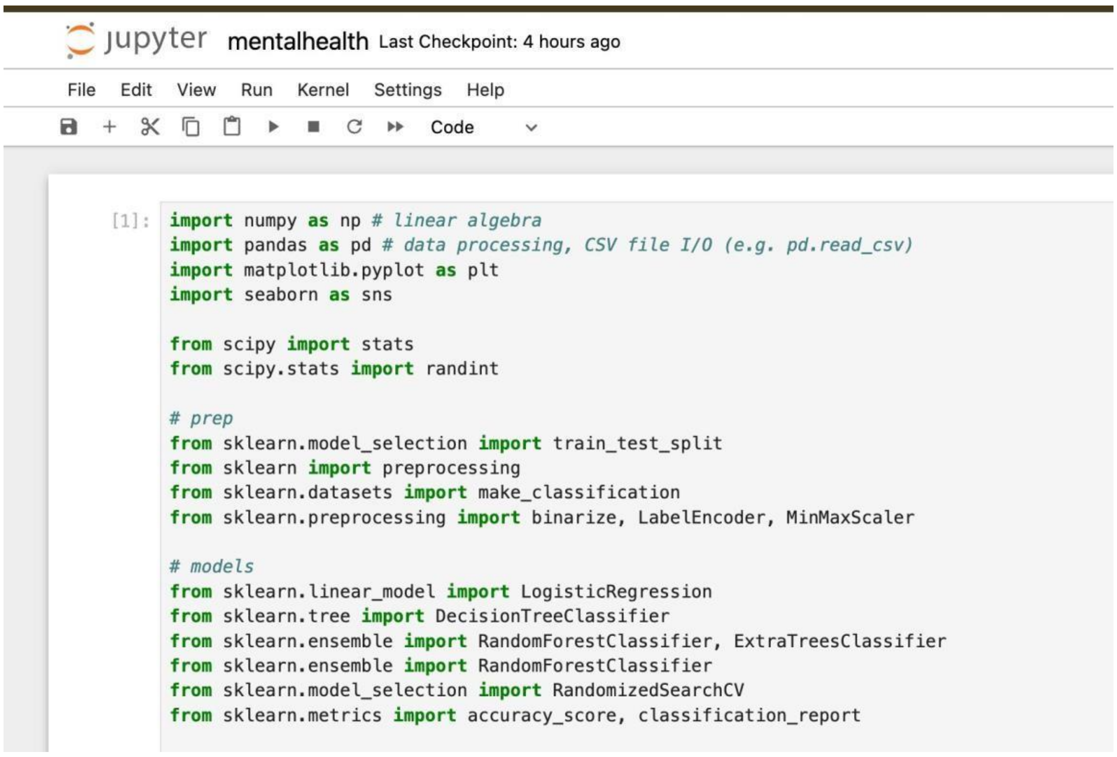


| Category | Specification |
|---|---|
| Operating System | macOS Ventura or later |
| Python Version | Python 3.8 or later |
| Package Manager | Homebrew (for dependency installation and management) |
| Development Environment | Jupyter Notebook |
| Required Libraries | pandas, NumPy, matplotlib, seaborn, scikit-learn, boruta, scipy |
| Installation Command | pip install pandas numpy matplotlib seaborn scikit-learn boruta scipy |
| Optional Tools | Anaconda Distribution (for pre-installed libraries and Jupyter Notebook) |
| Hardware Requirements | Minimum 8GB RAM (16GB recommended), Apple M1/M2 or Intel Core i5/i7 processor, and 256GB SSD for storage. |

Figure 1: Hardware Requirements

## 2 Software Requirements

The entire implementation of this research project was done in Jupyter Notebook using Python Programming language. As shown in the figure 2, Jupyter is browser-based service to create and execute notebook with python and there is no need of installation on your computer.



```
[1]: import numpy as np # linear algebra
     import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
     import matplotlib.pyplot as plt
     import seaborn as sns

     from scipy import stats
     from scipy.stats import randint

     # prep
     from sklearn.model_selection import train_test_split
     from sklearn import preprocessing
     from sklearn.datasets import make_classification
     from sklearn.preprocessing import binarize, LabelEncoder, MinMaxScaler

     # models
     from sklearn.linear_model import LogisticRegression
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.model_selection import RandomizedSearchCV
     from sklearn.metrics import accuracy_score, classification_report
```

Figure 2: Programming Software

## 3 Implementation

The implementation of the entire research project is performed in 5 python notebooks which are as under.

- Menatlhealth. ipynb
- social media. ipynb

The following libraries were used in implementation of this research study.

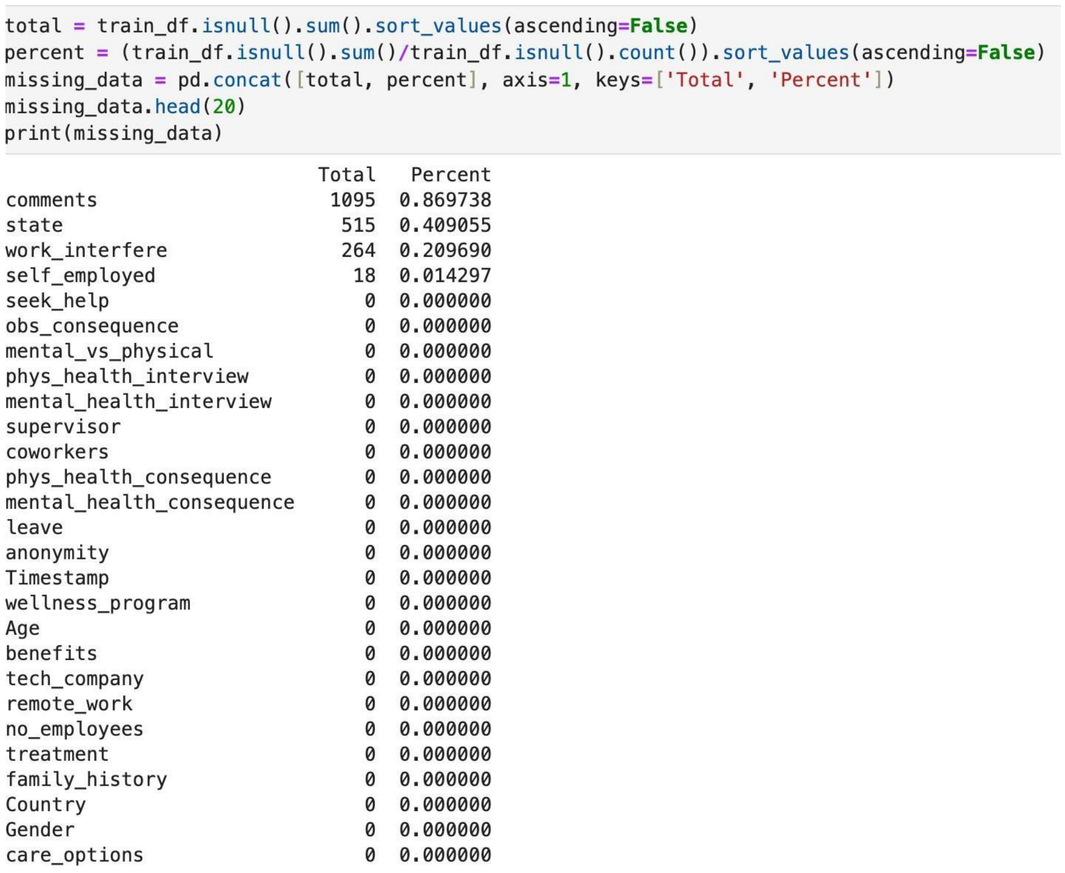
- SciPy. Stats.
- scipy
- numpy.
- matplotlib.
- pandas.
- datetime.
- sklearn.

## 4 Dataset Overview

- Data from two reliable sources were employed in the analysis and each was presented in tabular form. The first data set, included in DS1, is formed by 5000 records connected to some factors including mental health diseases, work life balance, and estimations.
- The second data set, DS2, consists of 481 data records for social media.

## 5 Data pre-processing

- It was an exercise in processing and modeling two datasets – DS1 and DS2 – which addressed work, stress/mental health, and social media use. The machine learning models building was done substages, at first with no data augmentation, then using such techniques like random sampling, ways to address class imbalance, and feature scaling. Some of the changes made included in and out of sample adjustments of the learning rate, batch size and iterations to avoid over and under fitting. These models were useful in finding patterns and relations and provide useful information concerning the complex link between mental health and social media.



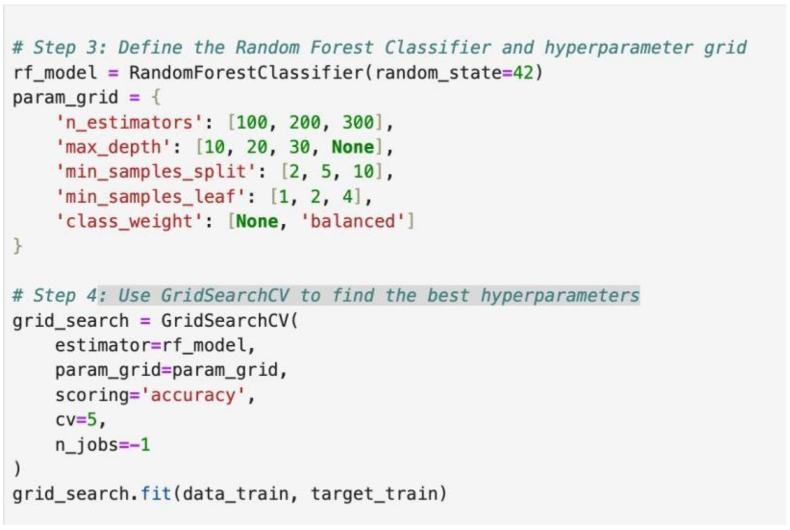
```
total = train_df.isnull().sum().sort_values(ascending=False)
percent = (train_df.isnull().sum()/train_df.isnull().count()).sort_values(ascending=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(20)
print(missing_data)
```

```
                           Total   Percent
comments                    1095  0.869738
state                        515  0.409055
work_interfere               264  0.209690
self_employed                 18  0.014297
seek_help                      0  0.000000
obs_consequence                0  0.000000
mental_vs_physical             0  0.000000
phys_health_interview          0  0.000000
mental_health_interview        0  0.000000
supervisor                     0  0.000000
coworkers                      0  0.000000
phys_health_consequence        0  0.000000
mental_health_consequence      0  0.000000
leave                          0  0.000000
anonymity                      0  0.000000
Timestamp                      0  0.000000
wellness_program               0  0.000000
Age                            0  0.000000
benefits                       0  0.000000
tech_company                   0  0.000000
remote_work                    0  0.000000
no_employees                   0  0.000000
treatment                      0  0.000000
family_history                 0  0.000000
Country                        0  0.000000
Gender                         0  0.000000
care_options                   0  0.000000
```

Figure 3: Missing values

## 6 Data Generation

- GridSearchCV is an internal tool to define the best hyperparameters or Machine learning model by trying all the possible combination present in some predefined grid. It is utilizing a cross-validation technique and checks every single combination and finally, the combination which has the best selected performance, say accuracy or F1-score is selected. This helps to check that the model has a good capacity to fit the model by not either over fitting or under fitting. For instance, Random Forest, GridSearchCV can try and set features such as the number of Trees or Maximum depth of Trees with resultant best setup.

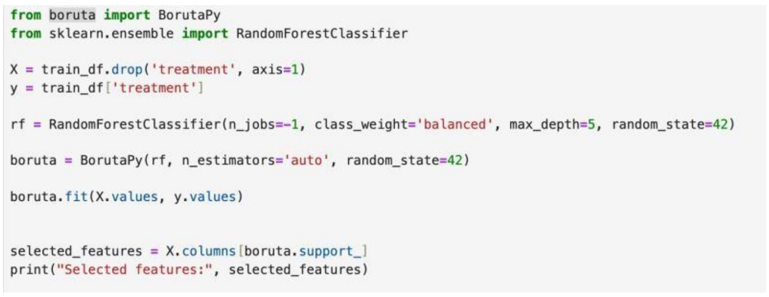


```
# Step 3: Define the Random Forest Classifier and hyperparameter grid
rf_model = RandomForestClassifier(random_state=42)
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'class_weight': [None, 'balanced']
}

# Step 4: Use GridSearchCV to find the best hyperparameters
grid_search = GridSearchCV(
    estimator=rf_model,
    param_grid=param_grid,
    scoring='accuracy',
    cv=5,
    n_jobs=-1
)
grid_search.fit(data_train, target_train)
```

Figure 4: Model Building

- Boruta is a procedure the selects feature by comparing their significance with random data for choosing the most efficient variables for a specified model.

- RandomizedSearchCV is used for hyperparameter tuning where the model randomly samples certain number of hyperparameters from the sample space and then tests their accuracy this makes the process faster than GridSearchCV However.



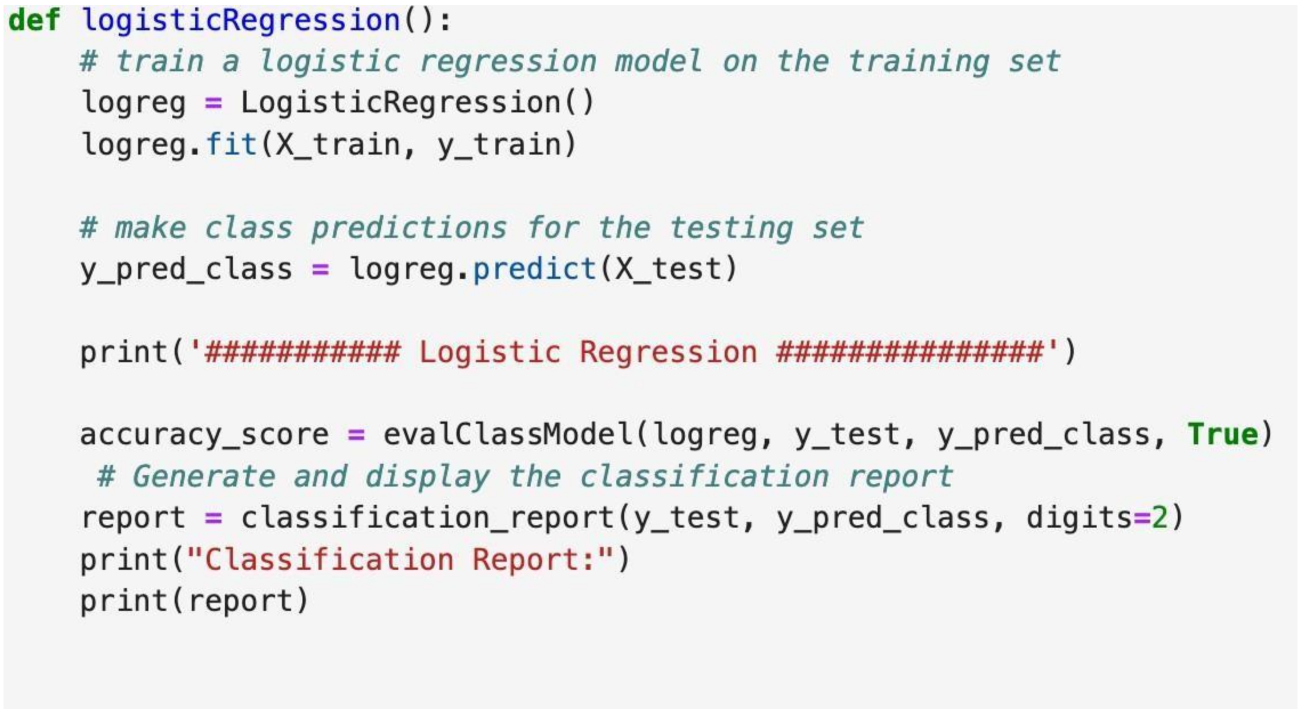
```
from boruta import BorutaPy
from sklearn.ensemble import RandomForestClassifier

X = train_df.drop('treatment', axis=1)
y = train_df['treatment']

rf = RandomForestClassifier(n_jobs=-1, class_weight='balanced', max_depth=5, random_state=42)

boruta = BorutaPy(rf, n_estimators='auto', random_state=42)

boruta.fit(X.values, y.values)


selected_features = X.columns[boruta.support_]
print("Selected features:", selected_features)
```

Figure 5. Boruta Algorithm

## 7 Modelling

Before data is balanced, modelling is performed on imbalanced data using algorithms like Logistic Regression, and Random Forest. This phase is implemented in both mental health. ipynb and social media. ipynb notebook.
The logistic Regression function executes operations to develop and assess a Logistic Regression model model. First the model gets built then the training takes place using input features (X_train) alongside corresponding labels (y_train). After receiving training data, the model prepares outcomes

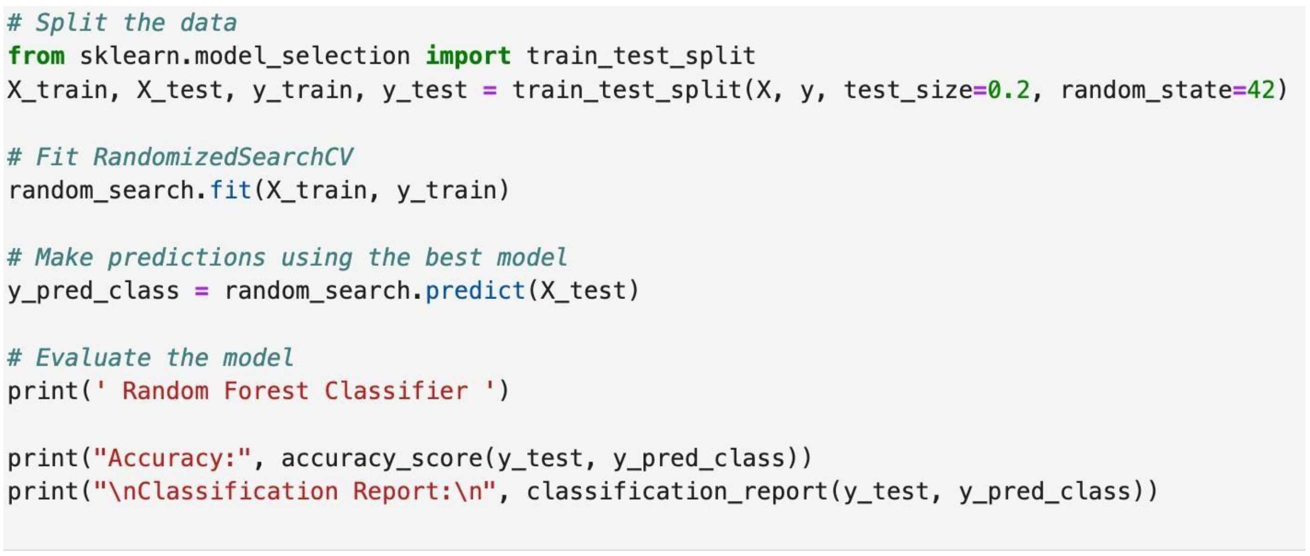
predictions for unknown X_test observations. After prediction completion the saved results are evaluated to measure model performance. A classification report ideologically displays important metrics regarding accuracy, precision along with recall information to aid model performance assessment.



```python
def logisticRegression():
    # train a logistic regression model on the training set
    logreg = LogisticRegression()
    logreg.fit(X_train, y_train)

    # make class predictions for the testing set
    y_pred_class = logreg.predict(X_test)

    print('########### Logistic Regression ###############')

    accuracy_score = evalClassModel(logreg, y_test, y_pred_class, True)
     # Generate and display the classification report
    report = classification_report(y_test, y_pred_class, digits=2)
    print("Classification Report:")
    print(report)
```

Figure 6: Test_Train_Data for DS1

Mixed Integer encoding allows users to observe how RandomizedSearchCV optimizes a Random Forest Classifier through different hyperparameter values for model evaluation. The analysis starts with a train_test_split operation which partitions the X features and y labels dataset. The distributed training set takes 80% as its learning data and 20% as test data by using the exact reproducibility seed value of 42 for the random state variable. The training process for the model takes place by applying fit () to random search using X_train alongside y_train data. After training completes the random search. predict () function predicts labels on X_test data which stores the results as y_pred_class. The model evaluation prints both accuracy score alongside a classification report with precision, recall and F1-score metrics to evaluate performance effectively.



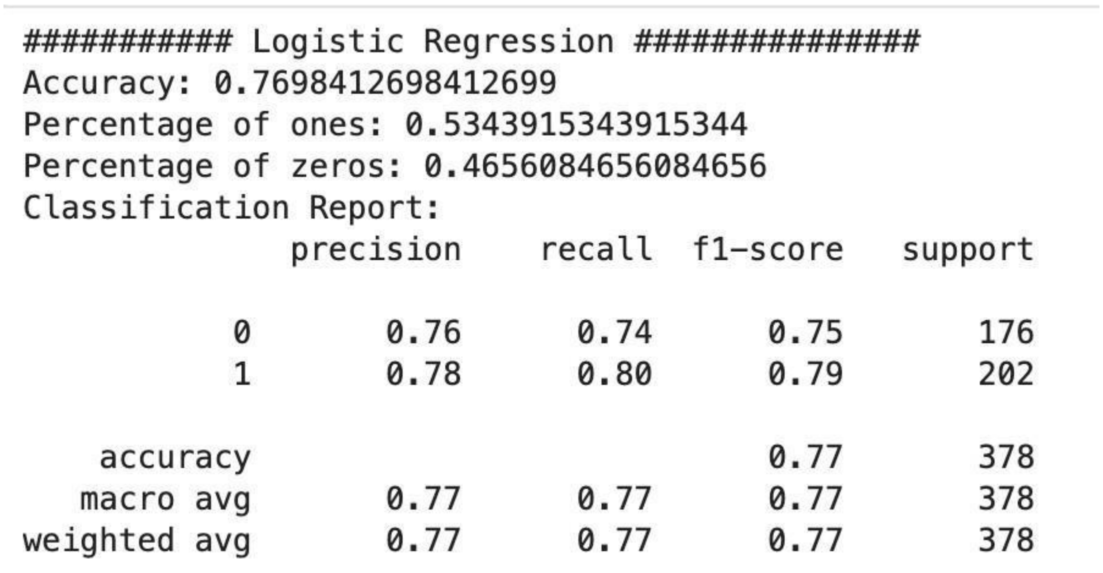
```python
# Split the data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Fit RandomizedSearchCV
random_search.fit(X_train, y_train)

# Make predictions using the best model
y_pred_class = random_search.predict(X_test)

# Evaluate the model
print(' Random Forest Classifier ')

print("Accuracy:", accuracy_score(y_test, y_pred_class))
print("\nClassification Report:\n", classification_report(y_test, y_pred_class))
```

Figure 7: Test_Train_Data for DS2

## 8 Evaluation of Implemented Methods

- The evaluation metrics—accuracy, precision, recall, and F1 score—were applied to assess model performance on two datasets. Accuracy measured overall performance, while precision and recall evaluated classifier performance, especially for imbalanced data. The F1 score, as the harmonic mean of precision and recall, was effective for skewed distributions. The confusion matrix highlighted true positives, true negatives, false positives, and false negatives, while the ROC curve and AUC assessed classification accuracy between true and false positives.

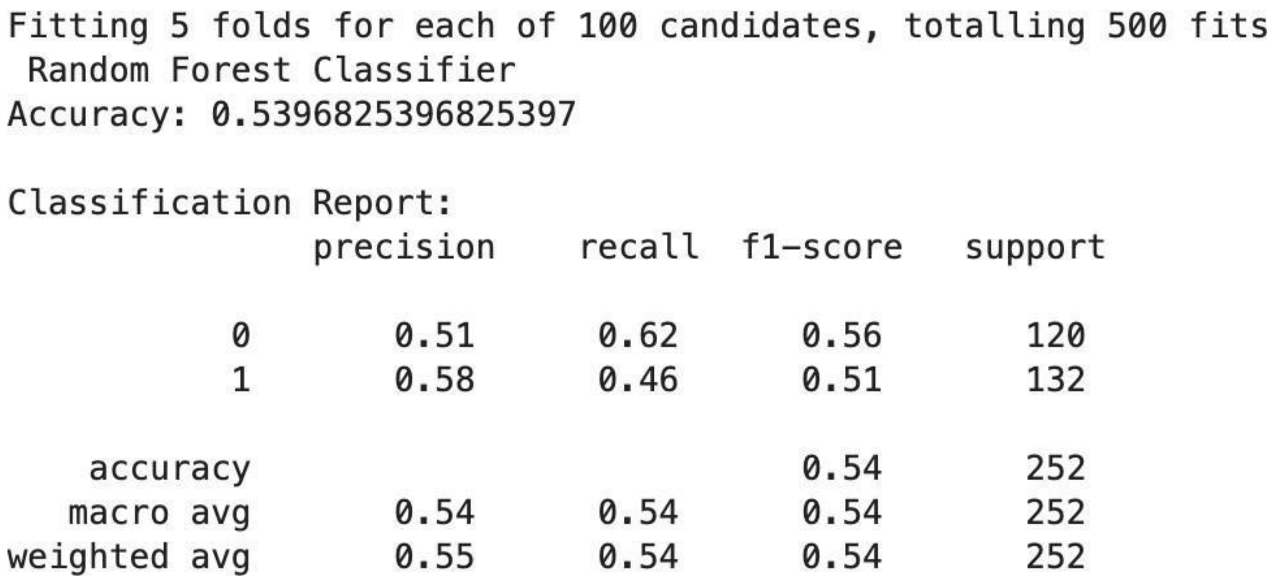
The output shows the performance metrics which evaluate a Logistic Regression model. Under testing conditions, the predictive model demonstrated 76.98% accuracy by maintaining correct predictions for 77% of all samples. The distribution of data reveals a small bias between positive class samples (53.44%) and negative class samples (46.56%). A classification report shows comprehensive metrics about both classification categories. Across 176 negative class (0) samples the model demonstrates precision at 0.76 along with recall at 0.74 and F1-score at 0.75. Model results indicate a precision rate of 0.78 and recall rate of 0.80 and F1-score of 0.79 when identifying positive class data (1) across 202 samples. A model assessment using equal-weight macro average and weighted average metrics shows consistent performance results of 0.77 precision along with recall and F1-score. The model demonstrates adequate performance throughout both positive and negative samples although it more effectively determines the presence of the positive class.



```
########### Logistic Regression ###############
Accuracy: 0.7698412698412699
Percentage of ones: 0.5343915343915344
Percentage of zeros: 0.4656084656084656
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.74      0.75       176
           1       0.78      0.80      0.79       202

    accuracy                           0.77       378
   macro avg       0.77      0.77      0.77       378
weighted avg       0.77      0.77      0.77       378
```

Figure 8: Logistic Regression for Menatlhealth DS

Random Forest Classifier demonstrates an overall accuracy level of 53.97% throughout its evaluation indicating a 54% correct classification rate for samples. Detailed quantitative information about both classification groups appears in the report. The negative class output reveals that the model shows moderate identification skill with an F1-score of 0.56 alongside precision 0.51 and recall at 0.62. Models experience higher difficulty classifying examples belonging to class 1 (positive class) because precision stands at 0.58 and recall reaches 0.46 and F1-score measures 0.51. All three metrics in the macro average stand at 0.54 indicating reasonable performance that remains below average across each class. Class-balancing calculations using weighted averages produce results of 0.55 compared to the simple average. The presented model reveals constrained prediction capabilities while requiring additional parameter optimization and modified feature treatment or novel computational methodologies.



```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
 Random Forest Classifier
Accuracy: 0.5396825396825397

Classification Report:
              precision    recall  f1-score   support

           0       0.51      0.62      0.56       120
           1       0.58      0.46      0.51       132

    accuracy                           0.54       252
   macro avg       0.54      0.54      0.54       252
weighted avg       0.55      0.54      0.54       252
```

Figure 9: Random Forest Classifier for Menatlhealth DS

The illustrated curve demonstrates Receiver Operating Characteristic (ROC) capability to evaluate classification model effectiveness. When threshold values change the orange curve shows how True Positive Rate (sensitivity) relates to False Positive Rate. Better model performance appears when the ROC curve locates near the top-left part of the chart. The analysis shows a random classifier performing along the diagonal dashed line while the model's efficiency to differentiate positive from negative classes reaches an area under the curve value of 0.92. The predictive abilities of the model are reflected in its 0.92 AUC value.

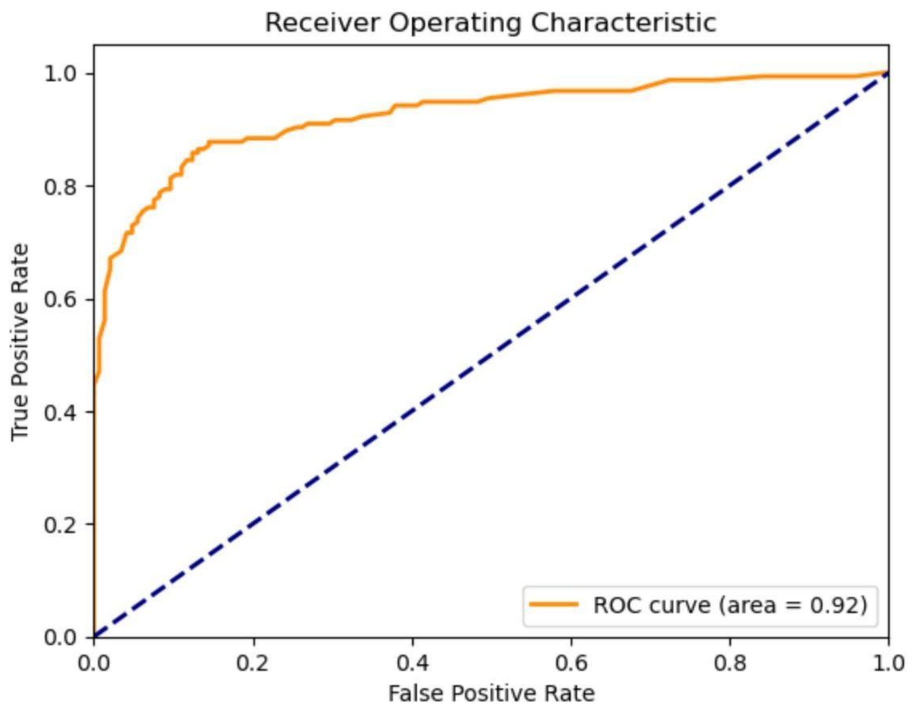
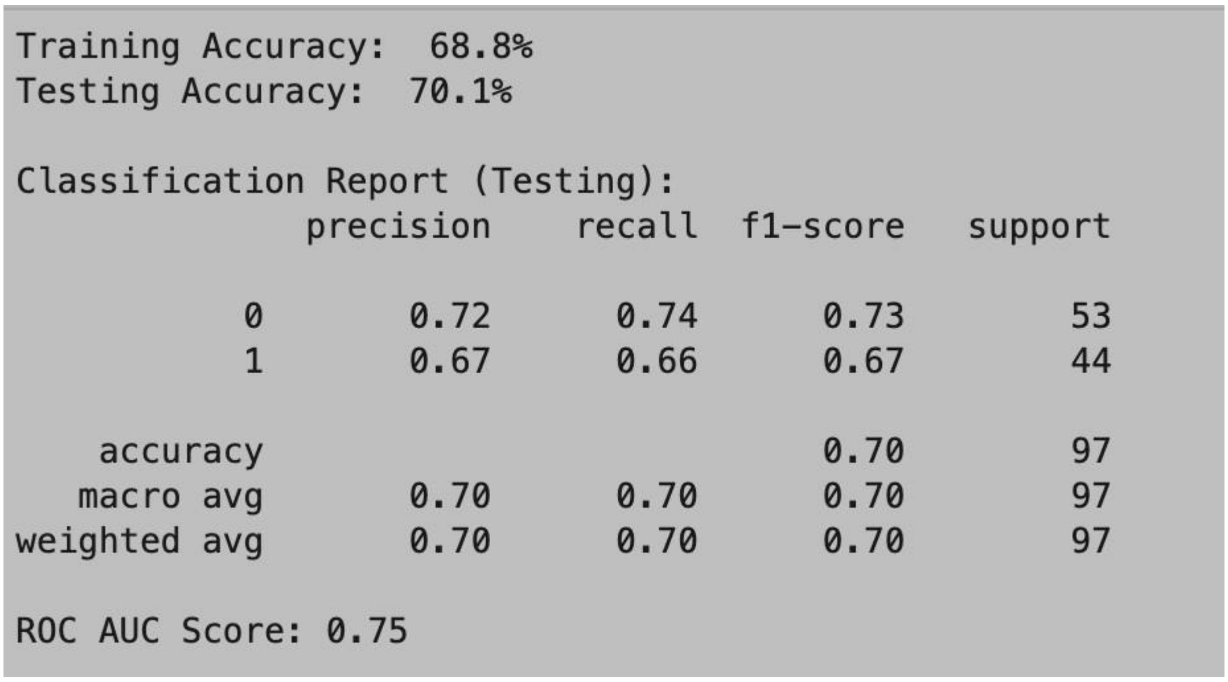


Figure 10: ROC Curve

The evaluation metrics for a classification model appear in the first visual display. Training accuracy stands at 68.8% while testing operations reach 70.1% thus demonstrating good generalization potential. Testing data results show complete performance metrics for class values 0 and 1. The model demonstrates precision values of 0.72 and 0.67 alongside recall values of 0.74 and 0.66 for class 0 and class 1 respectively and achieves an F1-Score of 0.73 and 0.67 across the two classes. Precision together with recall and F1-score all show a consistent measurement of 0.70 through both macro average and weighted average calculation methods. The model demonstrates moderate discriminatory ability through its ROC AUC metric resulting in 0.75 points.

The ROC curve in this image shows how well the model distinguishes between different classes. The blue curve demonstrates the sensitivity (True Positive Rate) and False Positive Rate relationship across different classification thresholds. A model prediction accuracy measurement of 0.75 for AUC (Area Under the Curve) indicates moderate capability to identify positive and negative classes. The model's predictive capability becomes evident through its distance from the diagonal reference line which reflects random classification performance. These combined metrics demonstrate a relatively strong model performance yet indicate potential improvements concerning class 1 precision and recall levels.



```
Training Accuracy:  68.8%
Testing Accuracy:  70.1%

Classification Report (Testing):
              precision    recall  f1-score   support

           0       0.72      0.74      0.73        53
           1       0.67      0.66      0.67        44

    accuracy                           0.70        97
   macro avg       0.70      0.70      0.70        97
weighted avg       0.70      0.70      0.70        97

ROC AUC Score: 0.75
```

Figure 11: Logistics Regression for social media DS2

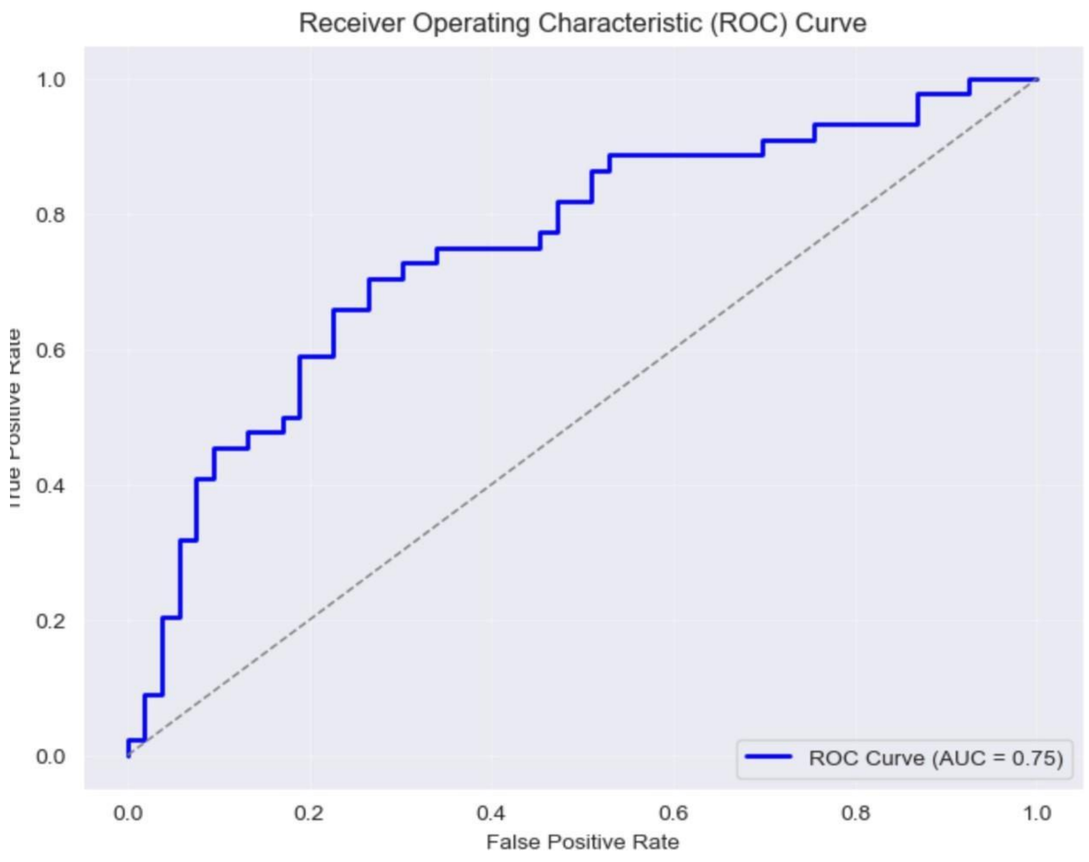
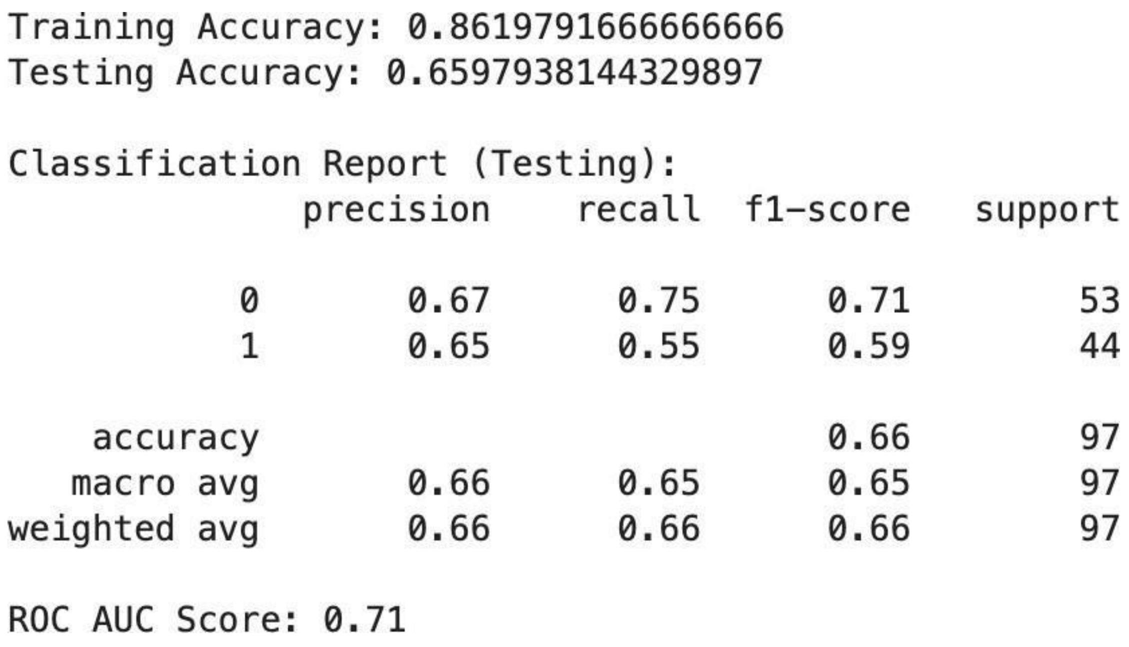


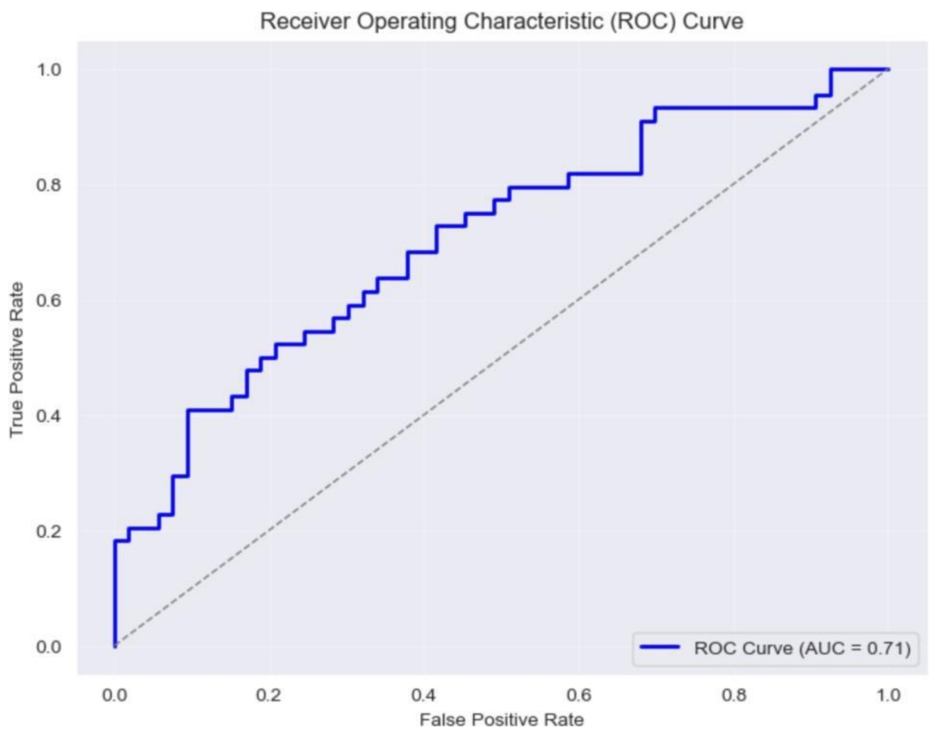
Figure 12: ROC Curve for Social media Ds

The graphics in this first display illustrate how a classification model performs based on its training statistics and testing platform. The building model shows 86.2% training precision while demonstrating testing results at 65.9% due to potential fitting problems and generic adaptation barriers. The testing data classification report presents a set of detailed performance metrics to the user. Testing results show that class 0 accuracy measures at 0.67 precision alongside 0.75 recall which produces 0.71 F1-score, yet class 1 outcomes generate 0.65 precision, 0.55 recall, and 0.59 F1-score. The calculated metrics using macro and weighted averages demonstrate stable performance of 0.66 precision while reaching equivalent recall and F1-score values at 0.66, indicating moderate model effectiveness for all categories. The ROC AUC model score reached 0.71 demonstrating reasonable discrimination power.
The second image presents the Receiver Operating Characteristic (ROC) curve that checks how well the model distinguishes between two different classes. A blue curve in this illustration shows how varying thresholds introduce trade-offs between True Positive Rate and False Positive Rate. Testing data revealed the model outperformed random guessing (diagonal) because its AUC score reached 0.71 which shows moderate predictive accuracy. The model demonstrates respectable results; however, its testing accuracy decline coupled with a lower-than-ideal AUC score confirms that additional feature optimization could enhance its operational effectiveness.



```
Training Accuracy: 0.8619791666666666
Testing Accuracy: 0.6597938144329897

Classification Report (Testing):
              precision    recall  f1-score   support

           0       0.67      0.75      0.71        53
           1       0.65      0.55      0.59        44

    accuracy                           0.66        97
   macro avg       0.66      0.65      0.65        97
weighted avg       0.66      0.66      0.66        97

ROC AUC Score: 0.71
```

Figure 13: Random Forest Classifier for social media DS2


Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC Curve (AUC = 0.71)

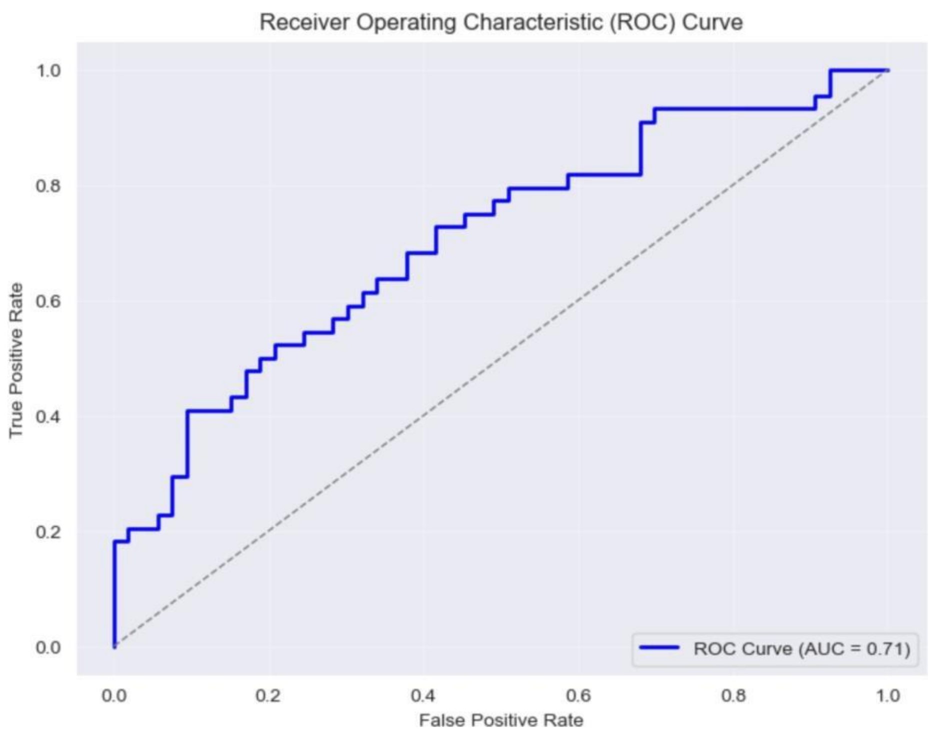


Figure 14: ROC for Randomforestclassifer