Final Project Report

Predicting Diabetes Using Supervised Data Mining (LSTM, Gaussian Naive Bayes, and Random Forest Algorithm)

Name: *Benyamin Plaksienko*

NJIT UCID: *bp446*

Email Address: *bp446@njit.edu*

Professor: *Yasser Abdullah*

Course: *CS 634 Data Mining*

1 Abstract

In this project, I explore supervised data mining techniques for binary classification by implementing and comparing three algorithms: Long Short-Term Memory (LSTM), Gaussian Naive Bayes (GNB), and Random Forest (RF). Using the Diabetes Dataset from Kaggle, I preprocess the data, apply feature scaling, and employ a 10-fold cross-validation approach to evaluate model performance. Each algorithm's classification accuracy, precision, recall, F1-score, and other key metrics are manually computed to ensure thorough analysis. The LSTM model is constructed using TensorFlow/Keras, while the GNB and RF models leverage Scikit-learn's implementations. By comparing the results, I assess the strengths and weaknesses of each approach in predicting diabetes, providing insights into their effectiveness and applicability to medical data classification.

2 Introduction

Data mining is a critical technique in the field of machine learning and data science, enabling the extraction of meaningful patterns from vast amounts of data. One key application of data mining is in the medical field, where it is used to predict diseases, analyze patient data, and support clinical decision-making. Supervised learning, a subfield of machine learning, plays a crucial role in classification tasks such as disease prediction. In this project, I applied supervised data mining techniques to predict diabetes using three different classification algorithms: Long Short-Term Memory (LSTM), Gaussian Naive Bayes, and Random Forest.

My primary goal was to assess the effectiveness of these algorithms in accurately predicting diabetes based on medical datasets. Specifically, I aimed to explore:

- The performance of each algorithm using standard evaluation metrics such as accuracy, precision, recall, and F1-score.
- The impact of 10-fold cross-validation on the consistency of the results and the robustness of each model.
- A comparative analysis of traditional machine learning models (Gaussian Naive Bayes and Random Forest) versus a deep learning-based approach (LSTM) in handling medical data.

The dataset I used for this analysis is the Diabetes Dataset retrieved from Kaggle. It consists of various medical attributes, including blood glucose levels, insulin levels, BMI, age, and other health indicators, which are key factors in determining the likelihood of diabetes. By applying the three selected classification models to this dataset, I was able to gain insights into which approach is most suitable for diabetes prediction.

3 Source of Data

The dataset used in this project is publicly available on Kaggle. It can be accessed at the following link, and downloaded as a csv:

https://www.kaggle.com/datasets/mathchi/diabetes-data-set

The dataset used in this project contains several health-related attributes collected from individuals, with the goal of predicting diabetes occurrence. Each row in the dataset represents an individual, and each column corresponds to a specific medical or demographic feature. Below is a description of the features:

- **Pregnancies**: The number of times the individual has been pregnant.
- **Glucose**: The plasma glucose concentration measured during an oral glucose tolerance test (mg/dL).
- **BloodPressure**: The diastolic blood pressure (mm Hg).
- **SkinThickness**: The thickness of the triceps skin fold (mm), which is used to estimate body fat.
- **Insulin**: The serum insulin level (U/mL) after a 2-hour glucose tolerance test.
- **BMI**: The Body Mass Index (BMI), calculated as weight (kg) divided by height squared (m^2).
- **DiabetesPedigreeFunction**: A score that estimates the likelihood of diabetes based on genetic heritage.
- **Age**: The age of the individual (years).
- **Outcome**: The target variable, where 1 indicates the presence of diabetes and 0 indicates its absence.

4 Project Workflow

4.1 Data Loading and Preprocessing

4.1.1 Load the Dataset

The first step in my project is loading the dataset from a CSV file, which contains multiple features and a target variable called "Outcome" (indicating whether the individual has diabetes).

4.1.2 Check for Data Skewness

Once I load the dataset, I examine the distribution of the target variable `Outcome`. This helps me determine if the dataset is imbalanced, which could potentially affect the performance of the models.

4.1.3 Feature Scaling

I standardize the features using the `StandardScaler`, which scales the features to have a mean of 0 and a standard deviation of 1.

4.1.4 Reshape Data for LSTM

Since Long Short-Term Memory (LSTM) models require the input data to be in a 3D format: `(samples, time steps, features)`, I reshape the data to satisfy this requirement, even though the data is not time-series.

4.2 Model Setup

I use three different machine learning models in this project:

4.2.1 Random Forest Classifier

The Random Forest model is an ensemble learning method that builds multiple decision trees and merges them to improve predictive performance. I use this model to predict whether an individual has diabetes based on the features.

4.2.2 Gaussian Naive Bayes

The Gaussian Naive Bayes classifier is a probabilistic model based on Bayes' theorem, which assumes that the features follow a Gaussian (normal) distribution. I use this model to see if it provides better performance for the task.

4.2.3 Long Short-Term Memory (LSTM)

LSTM is a type of Recurrent Neural Network (RNN) that is well-suited for sequential data. While my dataset does not involve explicit sequential data, I use LSTM to explore its potential in this project.

4.3 K-Fold Cross-Validation

To evaluate the performance of my models, I employ K-Fold Cross-Validation. The dataset is split into `K` subsets (folds). For each fold, I train the model on `K-1` folds and test it on the remaining fold. This process is repeated for all folds, and the results are averaged for a more reliable performance estimate.

For this project, I use 10-fold cross-validation, ensuring that each data point is used for both training and testing, which reduces the potential for overfitting.

4.4 Model Training and Evaluation

For each fold, I train the models on the training set and make predictions on the test set. The following evaluation metrics are computed for each model:

4.4.1 Confusion Matrix

The confusion matrix provides the following values:

- True Positives (TP)
- True Negatives (TN)
- False Positives (FP)
- False Negatives (FN)

4.4.2 Evaluation Metrics

From the confusion matrix, I derive the following metrics:

- **True Positive Rate (TPR)**: Measures the proportion of actual positives correctly identified by the model.
- **True Negative Rate (TNR)**: Measures the proportion of actual negatives correctly identified by the model.
- **False Positive Rate (FPR)**: Measures the proportion of actual negatives incorrectly classified as positives.
- **False Negative Rate (FNR)**: Measures the proportion of actual positives incorrectly classified as negatives.
- **Precision**: Measures the proportion of true positives among all predicted positives.
- **F1 Score**: The harmonic mean of precision and recall, providing a balance between the two.
- **Accuracy**: Measures the proportion of correct predictions (both true positives and true negatives).
- **Balanced Accuracy**: The average of TPR and TNR.
- **Error Rate**: The proportion of incorrect predictions (both false positives and false negatives).
- **True Skill Statistic (TSS)**: The difference between TPR and FPR, indicating the model's ability to distinguish between positive and negative cases.
- **Heidke Skill Score (HSS)**: A skill score that takes into account random chance in the predictions.

4.5 Metrics Calculation and Display

4.5.1 Calculate and Display Results for Each Fold

For each fold, I calculate the metrics and store them in a table for each model. This allows me to easily compare the models' performance across different folds.

4.5.2 Average Metrics Across All Folds

Once all folds have been processed, I compute the average of each evaluation metric across all folds for each model. This gives me a more generalized view of the model's performance, as it eliminates any biases caused by specific data splits.

5 Requirements

- **Python Version:** 3.8.20
- **Conda Version:** 24.11.3

5.1 Libraries

The following Python libraries need to be installed for the successful execution of the project. You can install them using `pip`:

- `scikit-learn` (sklearn): For machine learning algorithms and metrics.
  - `pip install scikit-learn`
- `tensorflow`: For building and training LSTM models.
  - `pip install tensorflow`
- `numpy`: For numerical operations and handling arrays.
  - `pip install numpy`
- `pandas`: For data manipulation and reading CSV files.
  - `pip install pandas`

6 Running the Code

Run the Jupyter Notebook:

```
jupyter notebook
```

Run the Python script:

```
python Benyamin_Plaksienko_Final_Project.py
```

7 Conclusion

7.1 Which Algorithm Performs Better and Why?

To understand which algorithm performed better, we need to analyze the results.

7.1.1 LSTM (Long Short-Term Memory)

Pros:

- LSTM leads with the highest F1 score (0.8384), Accuracy (0.780), and Balanced Accuracy (0.7403). This suggests that LSTM provides the best balance between precision and recall, achieving the highest overall accuracy.
- It has the lowest Error Rate (0.2200), indicating that it is the most accurate model in terms of minimizing misclassifications.
- LSTM has the lowest False Negative Rate (FNR) (0.114), suggesting it is better at avoiding false negatives compared to other models. For instance, Random Forest has an FNR of 0.148, and GNB (Gaussian Naive Bayes) has an FNR of 0.158.
- It also has the highest True Skill Statistic (TSS) (0.4807) and Heidke Skill Score (HSS) (0.5047), reflecting the best overall model performance in terms of distinguishing between classes.

7.1.2 Random Forest

Pros:

- Random Forest has the lowest False Positive Rate (FPR) (0.388), meaning it is less likely to incorrectly classify a negative instance as positive.

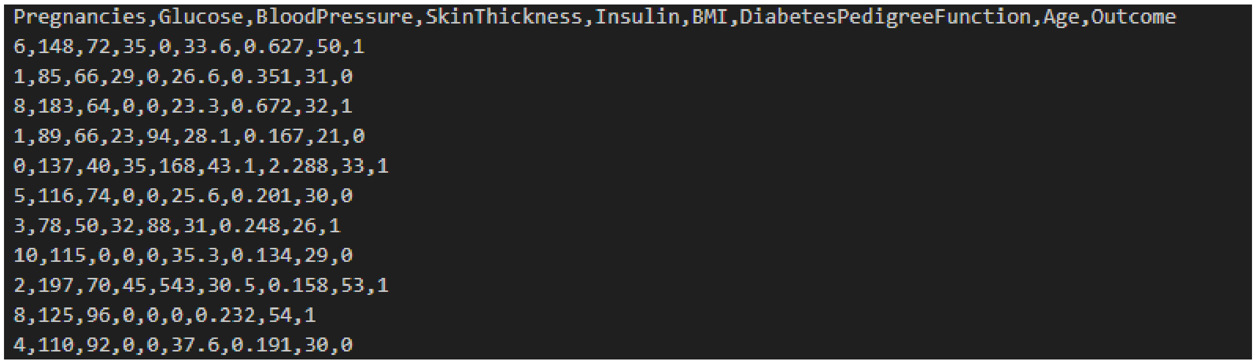
7.1.3 Results

Based on the analysis, LSTM is the best-performing model overall, with superior metrics in Accuracy, Balanced Accuracy, and Error Rate. It excels at minimizing misclassifications and false negatives, as well as distinguishing between classes, making it the most reliable model. Random Forest performs well in terms of the False Positive Rate but falls short in other areas like FNR and Accuracy. GNB, while useful in certain cases, has the highest FNR and lower overall performance, placing it in the third position.

7.1.4 Discussion

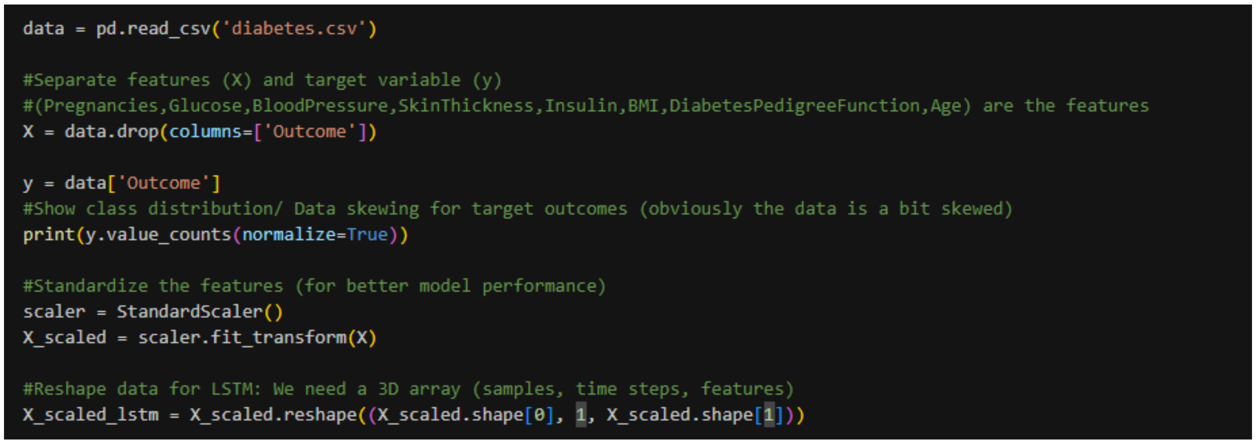
In general, I was expecting Random Forest to perform better than LSTM because my data isn't sequential, so it doesn't benefit from Long Short-Term Memory in that regard. Naive Bayes doesn't handle non-linear data well, and due to the high dimensionality of this dataset, I expected it to be the worst-performing algorithm. I believe the way I organized the LSTM, with 1 time step for each sample, is partially the reason it performs better than Random Forest. It kind of works like a Dense Neural Network, and due to the amount of data I'm inputting, it can outperform Random Forest.

This is how diabetes.csv looks like, and there are 769 rows.



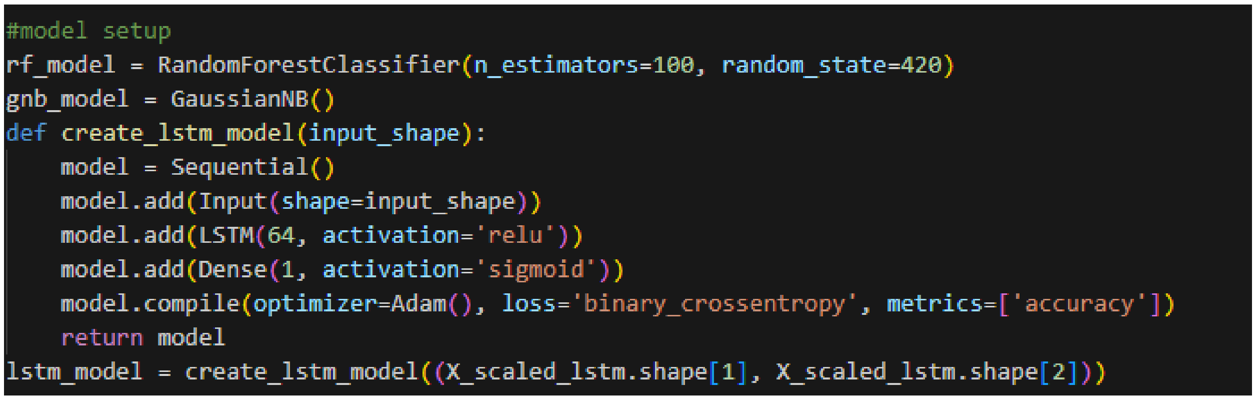
```
Pregnancies,Glucose,BloodPressure,SkinThickness,Insulin,BMI,DiabetesPedigreeFunction,Age,Outcome
6,148,72,35,0,33.6,0.627,50,1
1,85,66,29,0,26.6,0.351,31,0
8,183,64,0,0,23.3,0.672,32,1
1,89,66,23,94,28.1,0.167,21,0
0,137,40,35,168,43.1,2.288,33,1
5,116,74,0,0,25.6,0.201,30,0
3,78,50,32,88,31,0.248,26,1
10,115,0,0,0,35.3,0.134,29,0
2,197,70,45,543,30.5,0.158,53,1
8,125,96,0,0,0,0.232,54,1
4,110,92,0,0,37.6,0.191,30,0
```

This is data preprocessing



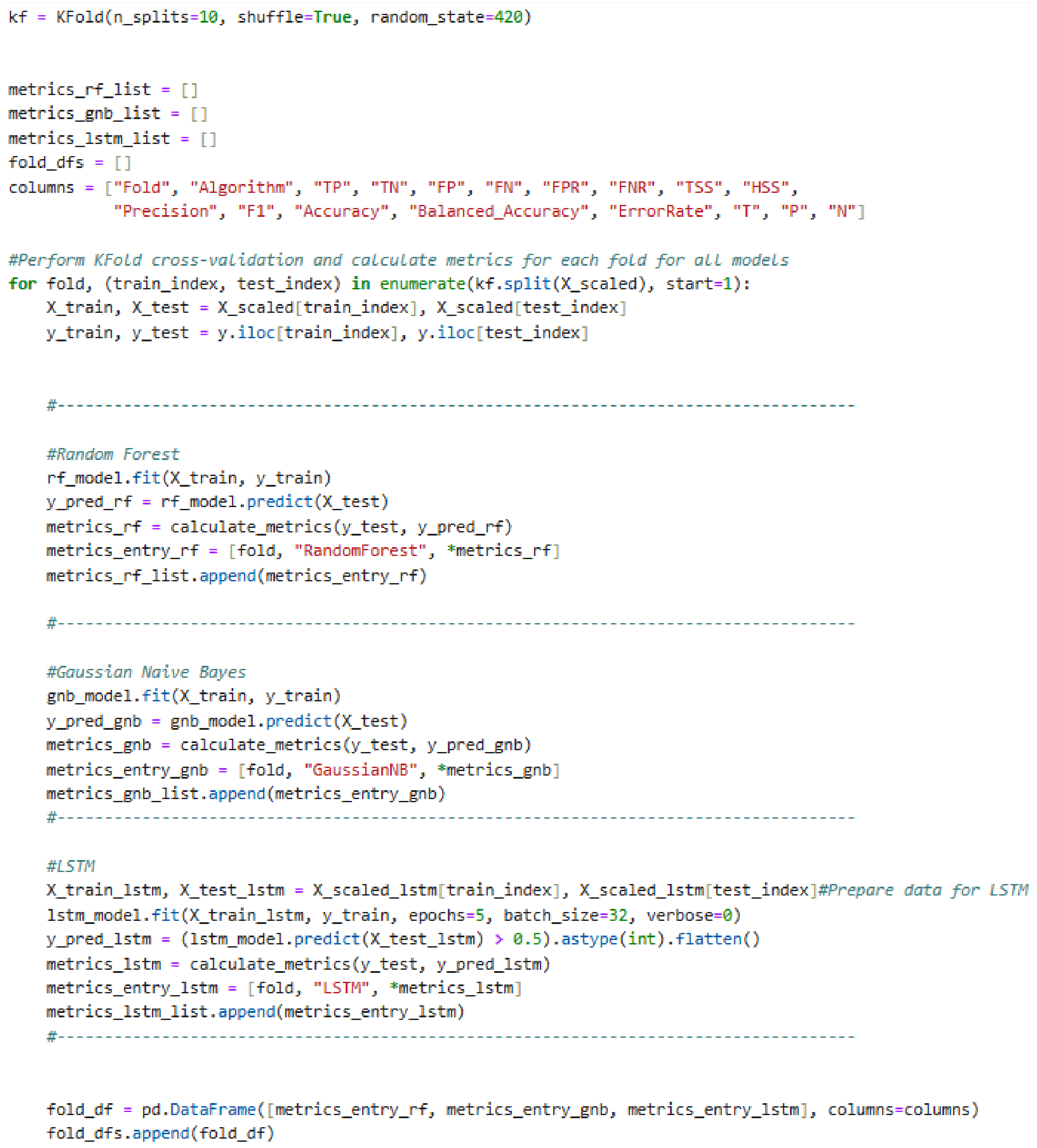
```
data = pd.read_csv('diabetes.csv')

#Separate features (X) and target variable (y)
#(Pregnancies,Glucose,BloodPressure,SkinThickness,Insulin,BMI,DiabetesPedigreeFunction,Age) are the features
X = data.drop(columns=['Outcome'])

y = data['Outcome']
#Show class distribution/ Data skewing for target outcomes (obviously the data is a bit skewed)
print(y.value_counts(normalize=True))

#Standardize the features (for better model performance)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

#Reshape data for LSTM: We need a 3D array (samples, time steps, features)
X_scaled_lstm = X_scaled.reshape((X_scaled.shape[0], 1, X_scaled.shape[1]))
```

This is model setup



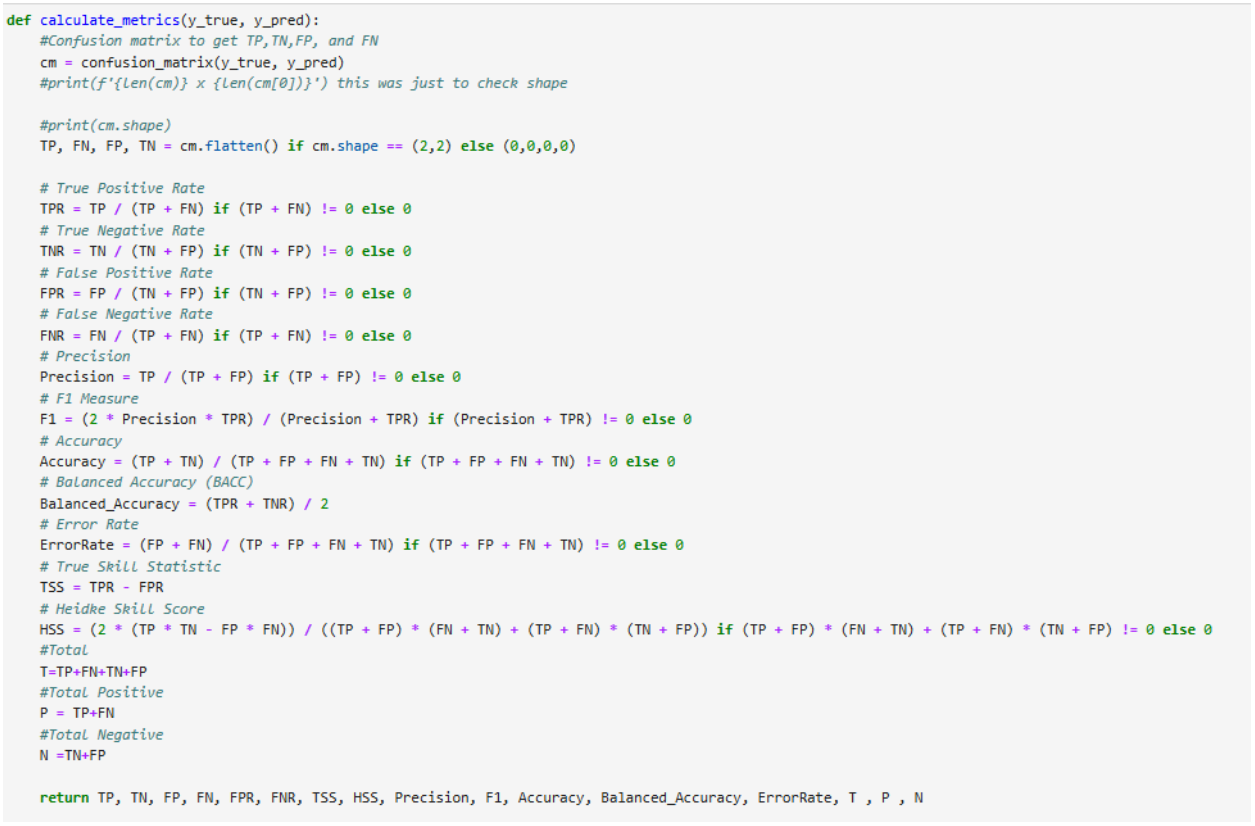
```
#model setup
rf_model = RandomForestClassifier(n_estimators=100, random_state=420)
gnb_model = GaussianNB()
def create_lstm_model(input_shape):
    model = Sequential()
    model.add(Input(shape=input_shape))
    model.add(LSTM(64, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer=Adam(), loss='binary_crossentropy', metrics=['accuracy'])
    return model
lstm_model = create_lstm_model((X_scaled_lstm.shape[1], X_scaled_lstm.shape[2]))
```

This is setting up the folds and running the models through all of them while calling a metrics function i made



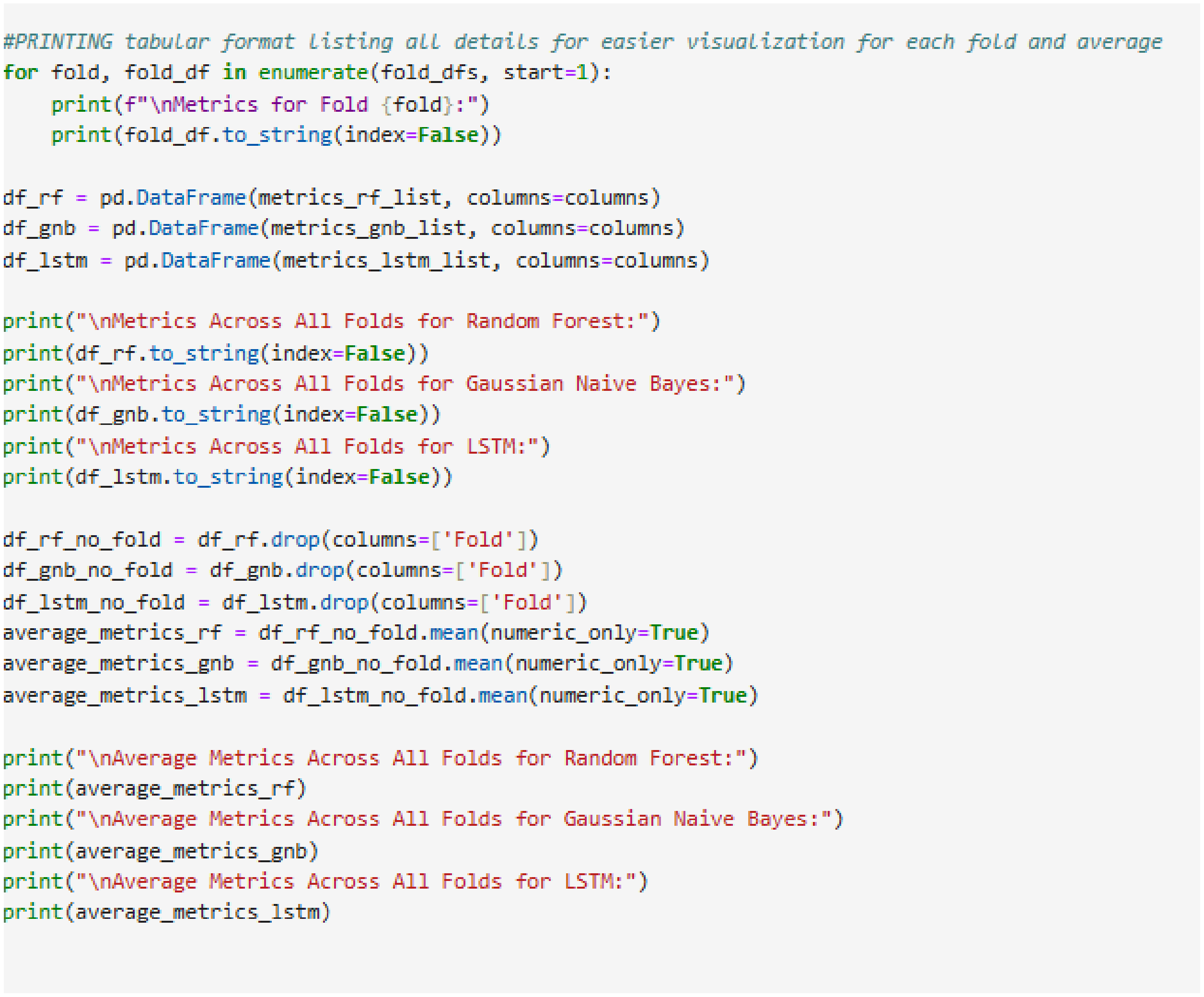
```python
kf = KFold(n_splits=10, shuffle=True, random_state=420)


metrics_rf_list = []
metrics_gnb_list = []
metrics_lstm_list = []
fold_dfs = []
columns = ["Fold", "Algorithm", "TP", "TN", "FP", "FN", "FPR", "FNR", "TSS", "HSS",
           "Precision", "F1", "Accuracy", "Balanced_Accuracy", "ErrorRate", "T", "P", "N"]

#Perform KFold cross-validation and calculate metrics for each fold for all models
for fold, (train_index, test_index) in enumerate(kf.split(X_scaled), start=1):
    X_train, X_test = X_scaled[train_index], X_scaled[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]


    #-----------------------------------------------------------------------------------

    #Random Forest
    rf_model.fit(X_train, y_train)
    y_pred_rf = rf_model.predict(X_test)
    metrics_rf = calculate_metrics(y_test, y_pred_rf)
    metrics_entry_rf = [fold, "RandomForest", *metrics_rf]
    metrics_rf_list.append(metrics_entry_rf)

    #-----------------------------------------------------------------------------------

    #Gaussian Naive Bayes
    gnb_model.fit(X_train, y_train)
    y_pred_gnb = gnb_model.predict(X_test)
    metrics_gnb = calculate_metrics(y_test, y_pred_gnb)
    metrics_entry_gnb = [fold, "GaussianNB", *metrics_gnb]
    metrics_gnb_list.append(metrics_entry_gnb)
    #-----------------------------------------------------------------------------------

    #LSTM
    X_train_lstm, X_test_lstm = X_scaled_lstm[train_index], X_scaled_lstm[test_index]#Prepare data for LSTM
    lstm_model.fit(X_train_lstm, y_train, epochs=5, batch_size=32, verbose=0)
    y_pred_lstm = (lstm_model.predict(X_test_lstm) > 0.5).astype(int).flatten()
    metrics_lstm = calculate_metrics(y_test, y_pred_lstm)
    metrics_entry_lstm = [fold, "LSTM", *metrics_lstm]
    metrics_lstm_list.append(metrics_entry_lstm)
    #-----------------------------------------------------------------------------------


    fold_df = pd.DataFrame([metrics_entry_rf, metrics_entry_gnb, metrics_entry_lstm], columns=columns)
    fold_dfs.append(fold_df)
```

This is my metrics function i made



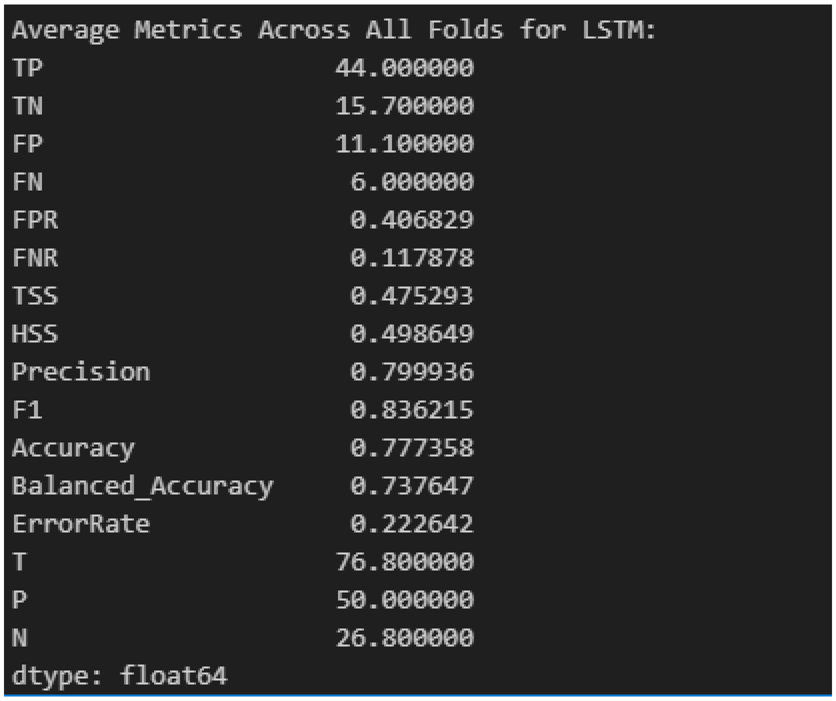
```python
def calculate_metrics(y_true, y_pred):
    #Confusion matrix to get TP,TN,FP, and FN
    cm = confusion_matrix(y_true, y_pred)
    #print(f'{len(cm)} x {len(cm[0])}') this was just to check shape

    #print(cm.shape)
    TP, FN, FP, TN = cm.flatten() if cm.shape == (2,2) else (0,0,0,0)

    # True Positive Rate
    TPR = TP / (TP + FN) if (TP + FN) != 0 else 0
    # True Negative Rate
    TNR = TN / (TN + FP) if (TN + FP) != 0 else 0
    # False Positive Rate
    FPR = FP / (TN + FP) if (TN + FP) != 0 else 0
    # False Negative Rate
    FNR = FN / (TP + FN) if (TP + FN) != 0 else 0
    # Precision
    Precision = TP / (TP + FP) if (TP + FP) != 0 else 0
    # F1 Measure
    F1 = (2 * Precision * TPR) / (Precision + TPR) if (Precision + TPR) != 0 else 0
    # Accuracy
    Accuracy = (TP + TN) / (TP + FP + FN + TN) if (TP + FP + FN + TN) != 0 else 0
    # Balanced Accuracy (BACC)
    Balanced_Accuracy = (TPR + TNR) / 2
    # Error Rate
    ErrorRate = (FP + FN) / (TP + FP + FN + TN) if (TP + FP + FN + TN) != 0 else 0
    # True Skill Statistic
    TSS = TPR - FPR
    # Heidke Skill Score
    HSS = (2 * (TP * TN - FP * FN)) / ((TP + FP) * (FN + TN) + (TP + FN) * (TN + FP)) if (TP + FP) * (FN + TN) + (TP + FN) * (TN + FP) != 0 else 0
    #Total
    T=TP+FN+TN+FP
    #Total Positive
    P = TP+FN
    #Total Negative
    N =TN+FP

    return TP, TN, FP, FN, FPR, FNR, TSS, HSS, Precision, F1, Accuracy, Balanced_Accuracy, ErrorRate, T , P , N
```

This is how i print out output, i do a bit of post processing

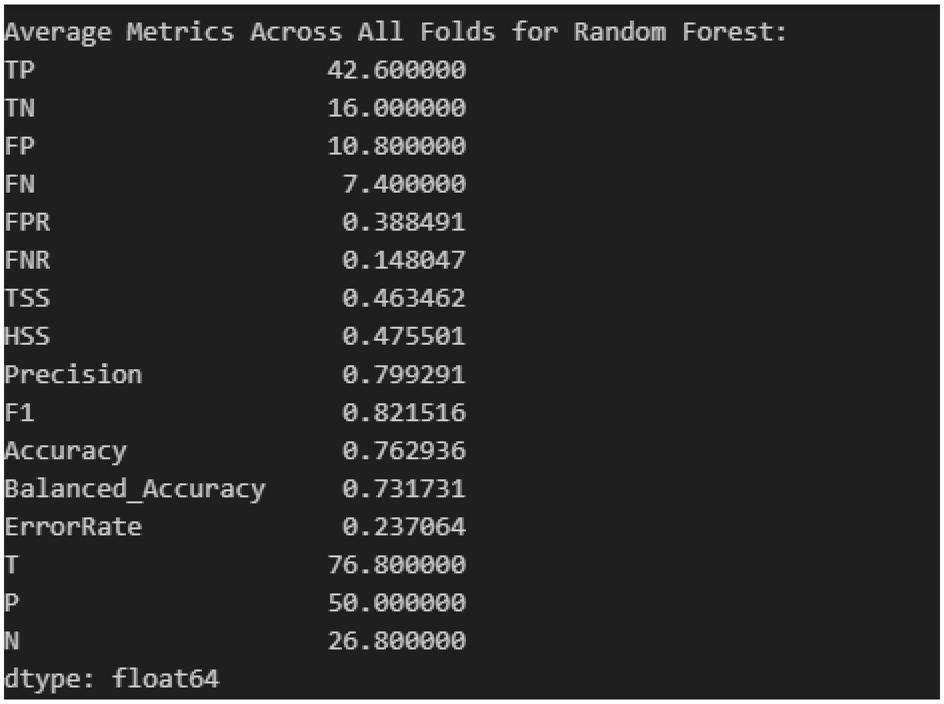


```python
#PRINTING tabular format listing all details for easier visualization for each fold and average
for fold, fold_df in enumerate(fold_dfs, start=1):
    print(f"\nMetrics for Fold {fold}:")
    print(fold_df.to_string(index=False))

df_rf = pd.DataFrame(metrics_rf_list, columns=columns)
df_gnb = pd.DataFrame(metrics_gnb_list, columns=columns)
df_lstm = pd.DataFrame(metrics_lstm_list, columns=columns)

print("\nMetrics Across All Folds for Random Forest:")
print(df_rf.to_string(index=False))
print("\nMetrics Across All Folds for Gaussian Naive Bayes:")
print(df_gnb.to_string(index=False))
print("\nMetrics Across All Folds for LSTM:")
print(df_lstm.to_string(index=False))

df_rf_no_fold = df_rf.drop(columns=['Fold'])
df_gnb_no_fold = df_gnb.drop(columns=['Fold'])
df_lstm_no_fold = df_lstm.drop(columns=['Fold'])
average_metrics_rf = df_rf_no_fold.mean(numeric_only=True)
average_metrics_gnb = df_gnb_no_fold.mean(numeric_only=True)
average_metrics_lstm = df_lstm_no_fold.mean(numeric_only=True)

print("\nAverage Metrics Across All Folds for Random Forest:")
print(average_metrics_rf)
print("\nAverage Metrics Across All Folds for Gaussian Naive Bayes:")
print(average_metrics_gnb)
print("\nAverage Metrics Across All Folds for LSTM:")
print(average_metrics_lstm)
```

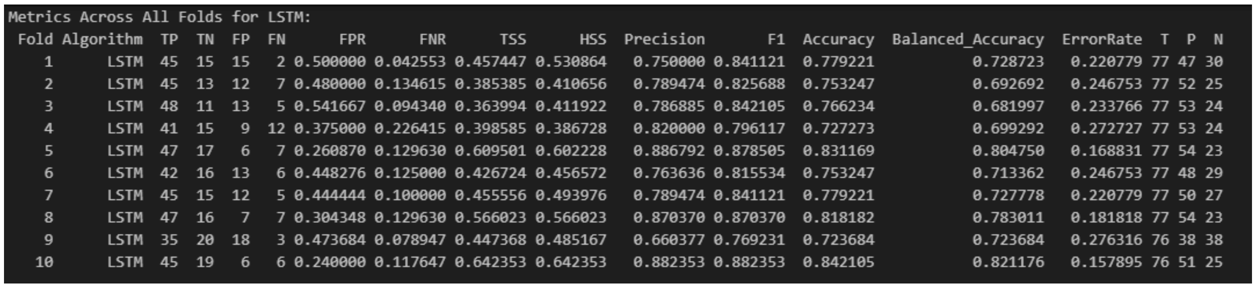
This is how results look like



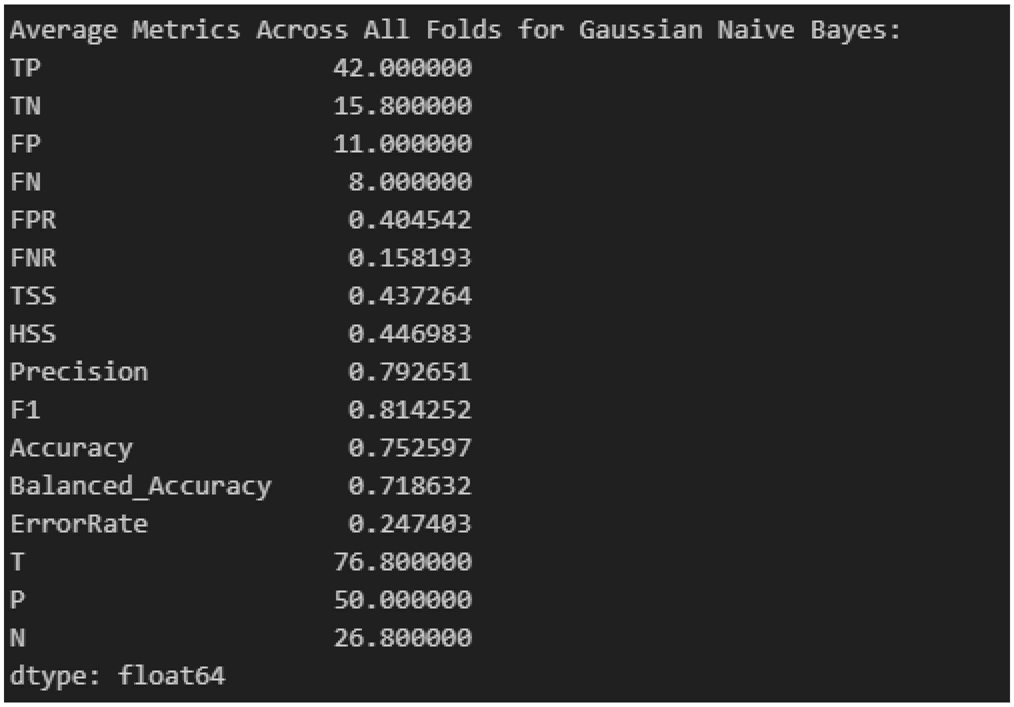
```
Average Metrics Across All Folds for LSTM:
TP                    44.000000
TN                    15.700000
FP                    11.100000
FN                     6.000000
FPR                    0.406829
FNR                    0.117878
TSS                    0.475293
HSS                    0.498649
Precision              0.799936
F1                     0.836215
Accuracy               0.777358
Balanced_Accuracy      0.737647
ErrorRate              0.222642
T                     76.800000
P                     50.000000
N                     26.800000
dtype: float64
```



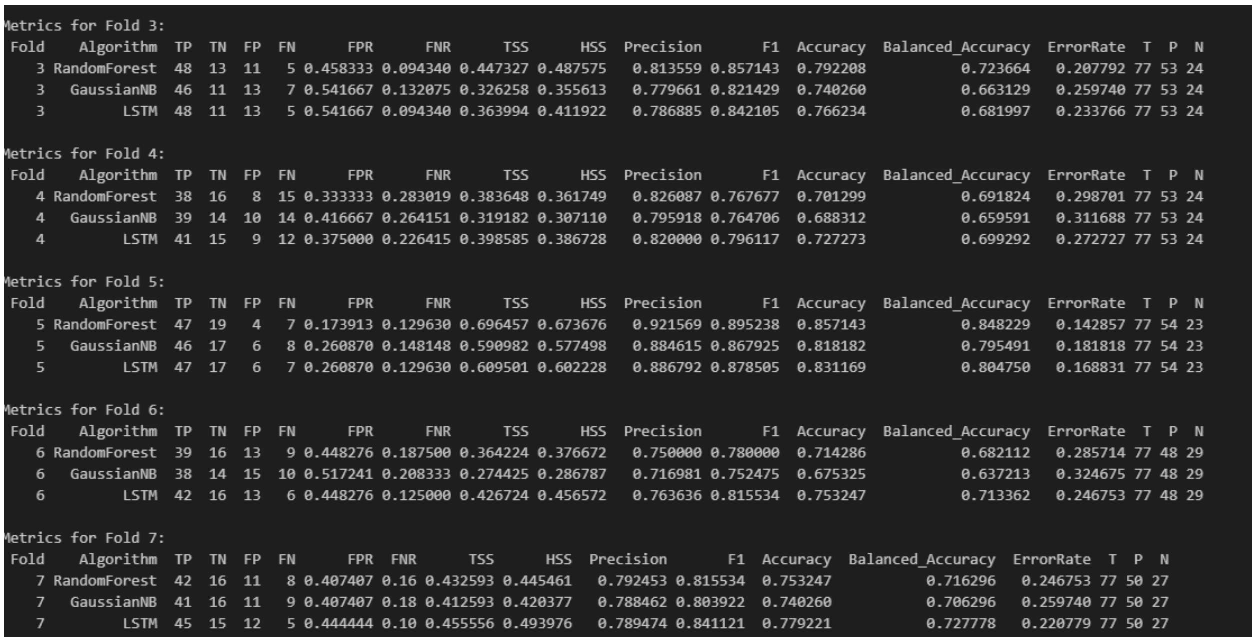
```
Average Metrics Across All Folds for Random Forest:
TP                    42.600000
TN                    16.000000
FP                    10.800000
FN                     7.400000
FPR                    0.388491
FNR                    0.148047
TSS                    0.463462
HSS                    0.475501
Precision              0.799291
F1                     0.821516
Accuracy               0.762936
Balanced_Accuracy      0.731731
ErrorRate              0.237064
T                     76.800000
P                     50.000000
N                     26.800000
dtype: float64
```



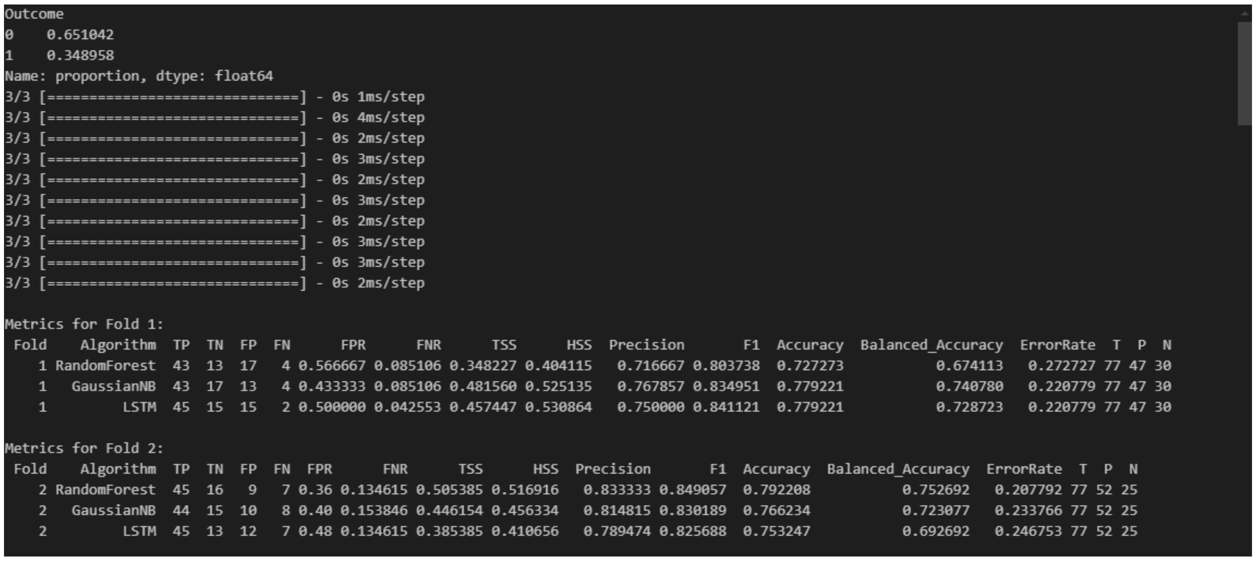
```
Metrics Across All Folds for LSTM:
 Fold Algorithm  TP  TN  FP  FN      FPR      FNR      TSS      HSS  Precision       F1  Accuracy  Balanced_Accuracy  ErrorRate  T  P  N
    1      LSTM  45  15  15   2 0.500000 0.042553 0.457447 0.530864   0.750000 0.841121  0.779221           0.728723   0.220779 77 47 30
    2      LSTM  45  13  12   7 0.480000 0.134615 0.385385 0.410656   0.789474 0.825688  0.753247           0.692692   0.246753 77 52 25
    3      LSTM  48  11  13   5 0.541667 0.094340 0.363994 0.411922   0.786885 0.842105  0.766234           0.681997   0.233766 77 53 24
    4      LSTM  41  15   9  12 0.375000 0.226415 0.398585 0.386728   0.820000 0.796117  0.727273           0.699292   0.272727 77 53 24
    5      LSTM  47  17   6   7 0.260870 0.129630 0.609501 0.602228   0.886792 0.878505  0.831169           0.804750   0.168831 77 54 23
    6      LSTM  42  16  13   6 0.448276 0.125000 0.426724 0.456572   0.763636 0.815534  0.753247           0.713362   0.246753 77 48 29
    7      LSTM  45  15  12   5 0.444444 0.100000 0.455556 0.493976   0.789474 0.841121  0.779221           0.727778   0.220779 77 50 27
    8      LSTM  47  16   7   7 0.304348 0.129630 0.566023 0.566023   0.870370 0.870370  0.818182           0.783011   0.181818 77 54 23
    9      LSTM  35  20  18   3 0.473684 0.078947 0.447368 0.485167   0.660377 0.769231  0.723684           0.723684   0.276316 76 38 38
   10      LSTM  45  19   6   6 0.240000 0.117647 0.642353 0.642353   0.882353 0.882353  0.842105           0.821176   0.157895 76 51 25
```



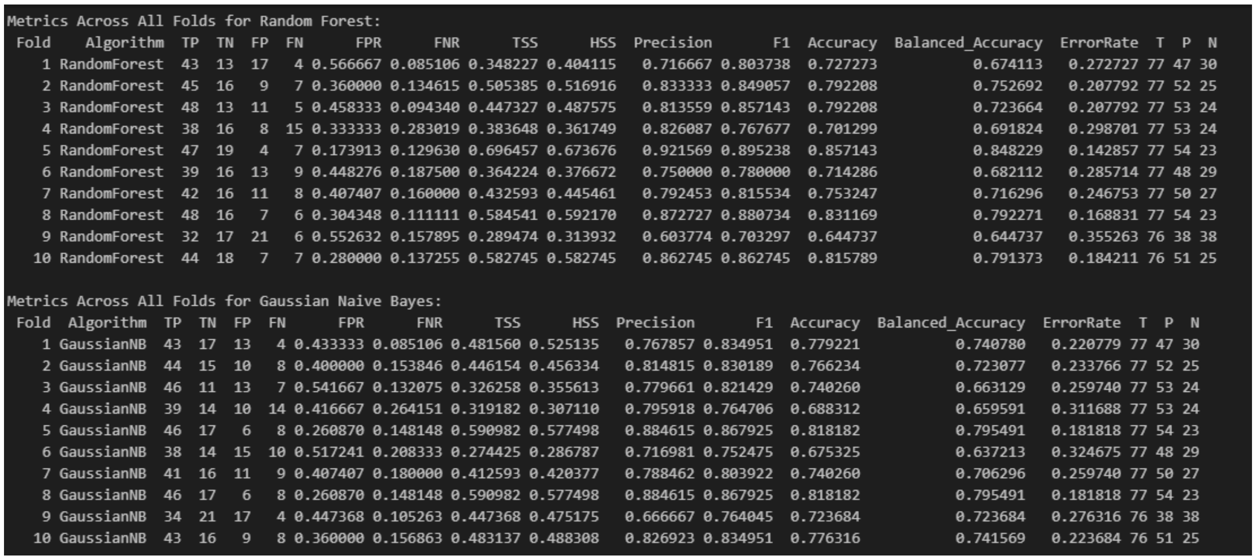
```
Average Metrics Across All Folds for Gaussian Naive Bayes:
TP                       42.000000
TN                       15.800000
FP                       11.000000
FN                        8.000000
FPR                       0.404542
FNR                       0.158193
TSS                       0.437264
HSS                       0.446983
Precision                 0.792651
F1                        0.814252
Accuracy                  0.752597
Balanced_Accuracy         0.718632
ErrorRate                 0.247403
T                        76.800000
P                        50.000000
N                        26.800000
dtype: float64
```



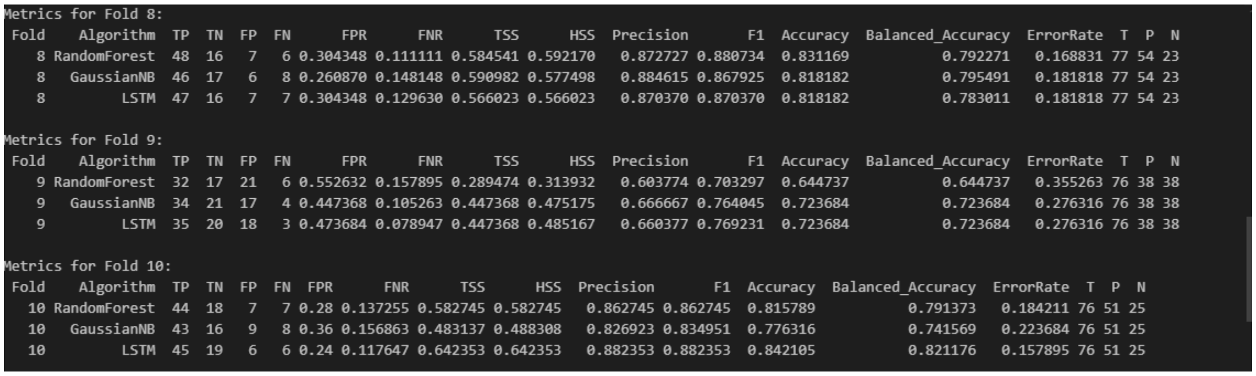
```
Metrics for Fold 3:
 Fold    Algorithm  TP  TN  FP  FN      FPR      FNR      TSS      HSS  Precision       F1  Accuracy  Balanced_Accuracy  ErrorRate  T  P  N
    3 RandomForest  48  13  11   5 0.458333 0.094340 0.447327 0.487575   0.813559 0.857143  0.792208           0.723664   0.207792 77 53 24
    3   GaussianNB  46  11  13   7 0.541667 0.132075 0.326258 0.355613   0.779661 0.821429  0.740260           0.663129   0.259740 77 53 24
    3         LSTM  48  11  13   5 0.541667 0.094340 0.363994 0.411922   0.786885 0.842105  0.766234           0.681997   0.233766 77 53 24

Metrics for Fold 4:
 Fold    Algorithm  TP  TN  FP  FN      FPR      FNR      TSS      HSS  Precision       F1  Accuracy  Balanced_Accuracy  ErrorRate  T  P  N
    4 RandomForest  38  16   8  15 0.333333 0.283019 0.383648 0.361749   0.826087 0.767677  0.701299           0.691824   0.298701 77 53 24
    4   GaussianNB  39  14  10  14 0.416667 0.264151 0.319182 0.307110   0.795918 0.764706  0.688312           0.659591   0.311688 77 53 24
    4         LSTM  41  15   9  12 0.375000 0.226415 0.398585 0.386728   0.820000 0.796117  0.727273           0.699292   0.272727 77 53 24

Metrics for Fold 5:
 Fold    Algorithm  TP  TN  FP  FN      FPR      FNR      TSS      HSS  Precision       F1  Accuracy  Balanced_Accuracy  ErrorRate  T  P  N
    5 RandomForest  47  19   4   7 0.173913 0.129630 0.696457 0.673676   0.921569 0.895238  0.857143           0.848229   0.142857 77 54 23
    5   GaussianNB  46  17   6   8 0.260870 0.148148 0.590982 0.577498   0.884615 0.867925  0.818182           0.795491   0.181818 77 54 23
    5         LSTM  47  17   6   7 0.260870 0.129630 0.609501 0.602228   0.886792 0.878505  0.831169           0.804750   0.168831 77 54 23

Metrics for Fold 6:
 Fold    Algorithm  TP  TN  FP  FN      FPR      FNR      TSS      HSS  Precision       F1  Accuracy  Balanced_Accuracy  ErrorRate  T  P  N
    6 RandomForest  39  16  13   9 0.448276 0.187500 0.364224 0.376672   0.750000 0.780000  0.714286           0.682112   0.285714 77 48 29
    6   GaussianNB  38  14  15  10 0.517241 0.208333 0.274425 0.286787   0.716981 0.752475  0.675325           0.637213   0.324675 77 48 29
    6         LSTM  42  16  13   6 0.448276 0.125000 0.426724 0.456572   0.763636 0.815534  0.753247           0.713362   0.246753 77 48 29

Metrics for Fold 7:
 Fold    Algorithm  TP  TN  FP  FN      FPR  FNR      TSS      HSS  Precision       F1  Accuracy  Balanced_Accuracy  ErrorRate  T  P  N
    7 RandomForest  42  16  11   8 0.407407 0.16 0.432593 0.445461   0.792453 0.815534  0.753247           0.716296   0.246753 77 50 27
    7   GaussianNB  41  16  11   9 0.407407 0.18 0.412593 0.420377   0.788462 0.803922  0.740260           0.706296   0.259740 77 50 27
    7         LSTM  45  15  12   5 0.444444 0.10 0.455556 0.493976   0.789474 0.841121  0.779221           0.727778   0.220779 77 50 27
```



```
Outcome
0    0.651042
1    0.348958
Name: proportion, dtype: float64
3/3 [==============================] - 0s 1ms/step
3/3 [==============================] - 0s 4ms/step
3/3 [==============================] - 0s 2ms/step
3/3 [==============================] - 0s 3ms/step
3/3 [==============================] - 0s 2ms/step
3/3 [==============================] - 0s 3ms/step
3/3 [==============================] - 0s 2ms/step
3/3 [==============================] - 0s 3ms/step
3/3 [==============================] - 0s 3ms/step
3/3 [==============================] - 0s 2ms/step

Metrics for Fold 1:
 Fold    Algorithm  TP  TN  FP  FN      FPR      FNR      TSS      HSS  Precision       F1  Accuracy  Balanced_Accuracy  ErrorRate  T  P  N
    1 RandomForest  43  13  17   4 0.566667 0.085106 0.348227 0.404115   0.716667 0.803738  0.727273           0.674113   0.272727 77 47 30
    1   GaussianNB  43  17  13   4 0.433333 0.085106 0.481560 0.525135   0.767857 0.834951  0.779221           0.740780   0.220779 77 47 30
    1         LSTM  45  15  15   2 0.500000 0.042553 0.457447 0.530864   0.750000 0.841121  0.779221           0.728723   0.220779 77 47 30

Metrics for Fold 2:
 Fold    Algorithm  TP  TN  FP  FN  FPR      FNR      TSS      HSS  Precision       F1  Accuracy  Balanced_Accuracy  ErrorRate  T  P  N
    2 RandomForest  45  16   9   7 0.36 0.134615 0.505385 0.516916   0.833333 0.849057  0.792208           0.752692   0.207792 77 52 25
    2   GaussianNB  44  15  10   8 0.40 0.153846 0.446154 0.456334   0.814815 0.830189  0.766234           0.723077   0.233766 77 52 25
    2         LSTM  45  13  12   7 0.48 0.134615 0.385385 0.410656   0.789474 0.825688  0.753247           0.692692   0.246753 77 52 25
```



```
Metrics Across All Folds for Random Forest:
 Fold    Algorithm  TP  TN  FP  FN      FPR      FNR      TSS      HSS  Precision       F1  Accuracy  Balanced_Accuracy  ErrorRate  T  P  N
    1 RandomForest  43  13  17   4 0.566667 0.085106 0.348227 0.404115   0.716667 0.803738  0.727273           0.674113   0.272727 77 47 30
    2 RandomForest  45  16   9   7 0.360000 0.134615 0.505385 0.516916   0.833333 0.849057  0.792208           0.752692   0.207792 77 52 25
    3 RandomForest  48  13  11   5 0.458333 0.094340 0.447327 0.487575   0.813559 0.857143  0.792208           0.723664   0.207792 77 53 24
    4 RandomForest  38  16   8  15 0.333333 0.283019 0.383648 0.361749   0.826087 0.767677  0.701299           0.691824   0.298701 77 53 24
    5 RandomForest  47  19   4   7 0.173913 0.129630 0.696457 0.673676   0.921569 0.895238  0.857143           0.848229   0.142857 77 54 23
    6 RandomForest  39  16  13   9 0.448276 0.187500 0.364224 0.376672   0.750000 0.780000  0.714286           0.682112   0.285714 77 48 29
    7 RandomForest  42  16  11   8 0.407407 0.160000 0.432593 0.445461   0.792453 0.815534  0.753247           0.716296   0.246753 77 50 27
    8 RandomForest  48  16   7   6 0.304348 0.111111 0.584541 0.592170   0.872727 0.880734  0.831169           0.792271   0.168831 77 54 23
    9 RandomForest  32  17  21   6 0.552632 0.157895 0.289474 0.313932   0.603774 0.703297  0.644737           0.644737   0.355263 76 38 38
   10 RandomForest  44  18   7   7 0.280000 0.137255 0.582745 0.582745   0.862745 0.862745  0.815789           0.791373   0.184211 76 51 25

Metrics Across All Folds for Gaussian Naive Bayes:
 Fold  Algorithm  TP  TN  FP  FN      FPR      FNR      TSS      HSS  Precision       F1  Accuracy  Balanced_Accuracy  ErrorRate  T  P  N
    1 GaussianNB  43  17  13   4 0.433333 0.085106 0.481560 0.525135   0.767857 0.834951  0.779221           0.740780   0.220779 77 47 30
    2 GaussianNB  44  15  10   8 0.400000 0.153846 0.446154 0.456334   0.814815 0.830189  0.766234           0.723077   0.233766 77 52 25
    3 GaussianNB  46  11  13   7 0.541667 0.132075 0.326258 0.355613   0.779661 0.821429  0.740260           0.663129   0.259740 77 53 24
    4 GaussianNB  39  14  10  14 0.416667 0.264151 0.319182 0.307110   0.795918 0.764706  0.688312           0.659591   0.311688 77 53 24
    5 GaussianNB  46  17   6   8 0.260870 0.148148 0.590982 0.577498   0.884615 0.867925  0.818182           0.795491   0.181818 77 54 23
    6 GaussianNB  38  14  15  10 0.517241 0.208333 0.274425 0.286787   0.716981 0.752475  0.675325           0.637213   0.324675 77 48 29
    7 GaussianNB  41  16  11   9 0.407407 0.180000 0.412593 0.420377   0.788462 0.803922  0.740260           0.706296   0.259740 77 50 27
    8 GaussianNB  46  17   6   8 0.260870 0.148148 0.590982 0.577498   0.884615 0.867925  0.818182           0.795491   0.181818 77 54 23
    9 GaussianNB  34  21  17   4 0.447368 0.105263 0.447368 0.475175   0.666667 0.764045  0.723684           0.723684   0.276316 76 38 38
   10 GaussianNB  43  16   9   8 0.360000 0.156863 0.483137 0.488308   0.826923 0.834951  0.776316           0.741569   0.223684 76 51 25
```



```
Metrics for Fold 8:
 Fold    Algorithm  TP  TN  FP  FN      FPR      FNR      TSS      HSS  Precision       F1  Accuracy  Balanced_Accuracy  ErrorRate  T  P  N
    8 RandomForest  48  16   7   6 0.304348 0.111111 0.584541 0.592170   0.872727 0.880734  0.831169           0.792271   0.168831 77 54 23
    8   GaussianNB  46  17   6   8 0.260870 0.148148 0.590982 0.577498   0.884615 0.867925  0.818182           0.795491   0.181818 77 54 23
    8         LSTM  47  16   7   7 0.304348 0.129630 0.566023 0.566023   0.870370 0.870370  0.818182           0.783011   0.181818 77 54 23

Metrics for Fold 9:
 Fold    Algorithm  TP  TN  FP  FN      FPR      FNR      TSS      HSS  Precision       F1  Accuracy  Balanced_Accuracy  ErrorRate  T  P  N
    9 RandomForest  32  17  21   6 0.552632 0.157895 0.289474 0.313932   0.603774 0.703297  0.644737           0.644737   0.355263 76 38 38
    9   GaussianNB  34  21  17   4 0.447368 0.105263 0.447368 0.475175   0.666667 0.764045  0.723684           0.723684   0.276316 76 38 38
    9         LSTM  35  20  18   3 0.473684 0.078947 0.447368 0.485167   0.660377 0.769231  0.723684           0.723684   0.276316 76 38 38

Metrics for Fold 10:
 Fold    Algorithm  TP  TN  FP  FN  FPR      FNR      TSS      HSS  Precision       F1  Accuracy  Balanced_Accuracy  ErrorRate  T  P  N
   10 RandomForest  44  18   7   7 0.28 0.137255 0.582745 0.582745   0.862745 0.862745  0.815789           0.791373   0.184211 76 51 25
   10   GaussianNB  43  16   9   8 0.36 0.156863 0.483137 0.488308   0.826923 0.834951  0.776316           0.741569   0.223684 76 51 25
   10         LSTM  45  19   6   6 0.24 0.117647 0.642353 0.642353   0.882353 0.882353  0.842105           0.821176   0.157895 76 51 25
```

This is the link to the github repo
https://github.com/BenyaminPlaksienkoNJIT/CS-634-Data-Mining-Final-Term-Project