

Comparative Analysis of Machine Learning Algorithms for Binary Classification

CS 634 Data Mining

Final Term Project

Dr. Yasser Abdullah

Department of Computer Science

New Jersey Institute of Technology

Rushmitha Kommana

rk958@njit.edu

23rd November 2024

Table of content

CS 634 Data Mining	Error! No bookmark name given.
Final Term Project.....	Error! No bookmark name given.
Your Name	Error! No bookmark name given.
your_email@njit.edu	Error! No bookmark name given.
Github Link.....	3
Introduction	3
Dataset Creation	3
Project File Structure.....	3
Running the Program.....	3
Required Packages	3
Running the Program using Python file.....	4
Running the Program using Google Colab	7
Algorithm Comparison	10

Github Link:

[RushmithaKommana/Data-mining-final-Project-Binary-Classification](https://github.com/RushmithaKommana/Data-mining-final-Project-Binary-Classification)

Introduction

This project aims to implement and compare the performance of three different machine learning algorithms for binary classification on the Pima Indians Diabetes Dataset. The three algorithms implemented are:

1. Random Forest Classifier
2. Long Short-Term Memory (LSTM) (from the "Additional Option: Deep Learning" list)
3. Support Vector Machine (SVM) (from the "Additional Option: Algorithms" list)

Dataset Creation

I used the Pima Indians Diabetes Dataset, which is a widely-used benchmark dataset for binary classification tasks. The dataset is available from the sources listed in the "Additional Option: Sources of Data" section.

Link - [Pima Indians Diabetes Database \(kaggle.com\)](https://www.kaggle.com/uciml/pima-indians-diabetes-database)

Project File Structure

The project files include: A Python file with the code implementation A Jupyter Notebook with the same code implementation and the outputs of the runs

The datasets used: diabetes.csv

Running the Program

This project was developed using Python 3.10.12, but it should be compatible with Python 3.8 and newer versions.

Required Packages

The program relies on several Python packages for data processing and analysis:

- numpy: Provides support for large, multi-dimensional arrays and matrices, along with a

collection of high-level mathematical functions to operate on these arrays.

- `pandas`: Provides high-performance, easy-to-use data structures and data analysis tools.
- `matplotlib.pyplot`: A comprehensive library for creating static, animated, and interactive visualizations in Python.
- `sklearn.ensemble.RandomForestClassifier`: An ensemble learning method from the scikit-learn library that constructs a multitude of decision trees at training time and outputs the class that is the mode of the classes of the individual trees.
- `sklearn.model_selection.KFold`: A tool from the scikit-learn library for performing K-fold cross-validation.
- `sklearn.metrics`: A collection of functions to assess the performance of the machine learning models, including confusion matrix, accuracy, precision, recall, F1-score, and ROC-AUC.
- `sklearn.svm.SVC`: The Support Vector Classifier from the scikit-learn library, which is a powerful algorithm for binary and multi-class classification tasks.
- `tensorflow.keras.models.Sequential`: A linear stack of layers from the Keras library, which is a high-level neural networks API, running on top of TensorFlow.
- `tensorflow.keras.layers.Dense` and `tensorflow.keras.layers.LSTM`: Layers used in the construction of the LSTM model.

To install these packages, open a terminal or command prompt and execute the following command:

```
pip install numpy pandas matplotlib scikit-learn tensorflow
```

Running the Program using Python file

Open the Terminal and navigate to the directory containing the project files and run the following command.

```
python project.py
```

The below screenshot shows running the python file in terminal by selecting Pima Indians Diabetes dataset.

```
File Edit Selection View Go Run ... Assessment
project.py 2 X
project.py > ...
206 plt.ylabel('Metric')
207 plt.legend()
208
209 plt.subplot(1, 2, 2)
210 plt.bar(['Accuracy', 'Precision', 'Recall', 'F1-score', 'ROC-AUC', 'TSS', 'HSS'],
211         [svm_avg_metrics['Accuracy'], svm_avg_metrics['Precision'], svm_avg_metrics['Recall'],
212          svm_avg_metrics['F1-score'], svm_avg_metrics['ROC-AUC'], svm_avg_metrics['TSS'], svm_avg_metrics['HSS']])
213 plt.title('SVM - Average')
214 plt.xlabel('Metric')
215 plt.ylabel('Value')
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Fold 1: 3/3 0s 73ms/step
Fold 2: 3/3 0s 1ms/step
Fold 3: 3/3 0s 3ms/step
Fold 4: 3/3 0s 5ms/step
Fold 5: 3/3 0s 2ms/step
Fold 6: 3/3 0s 2ms/step
Fold 7: 3/3 0s 1ms/step
Fold 8: 3/3 0s 3ms/step
Fold 9: 3/3 0s 2ms/step

{'Accuracy': 0.7577751196172249, 'Precision': 0.7341623088962905, 'Recall': 0.47699390215519244, 'F1-score': 0.5719299669998865, 'ROC-AUC': 0.69248842493373

Ln 1, Col 1 Spaces: 4 UTF-8 LF Python 3.12.2 64-bit

```
File Edit Selection View Go Run ... Assessment
project.py 2 X
project.py > ...
206 plt.ylabel('Metric')
207 plt.legend()
208
209 plt.subplot(1, 2, 2)
210 plt.bar(['Accuracy', 'Precision', 'Recall', 'F1-score', 'ROC-AUC', 'TSS', 'HSS'],
211         [svm_avg_metrics['Accuracy'], svm_avg_metrics['Precision'], svm_avg_metrics['Recall'],
212          svm_avg_metrics['F1-score'], svm_avg_metrics['ROC-AUC'], svm_avg_metrics['TSS'], svm_avg_metrics['HSS']])
213 plt.title('SVM - Average')
214 plt.xlabel('Metric')
215 plt.ylabel('Value')
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

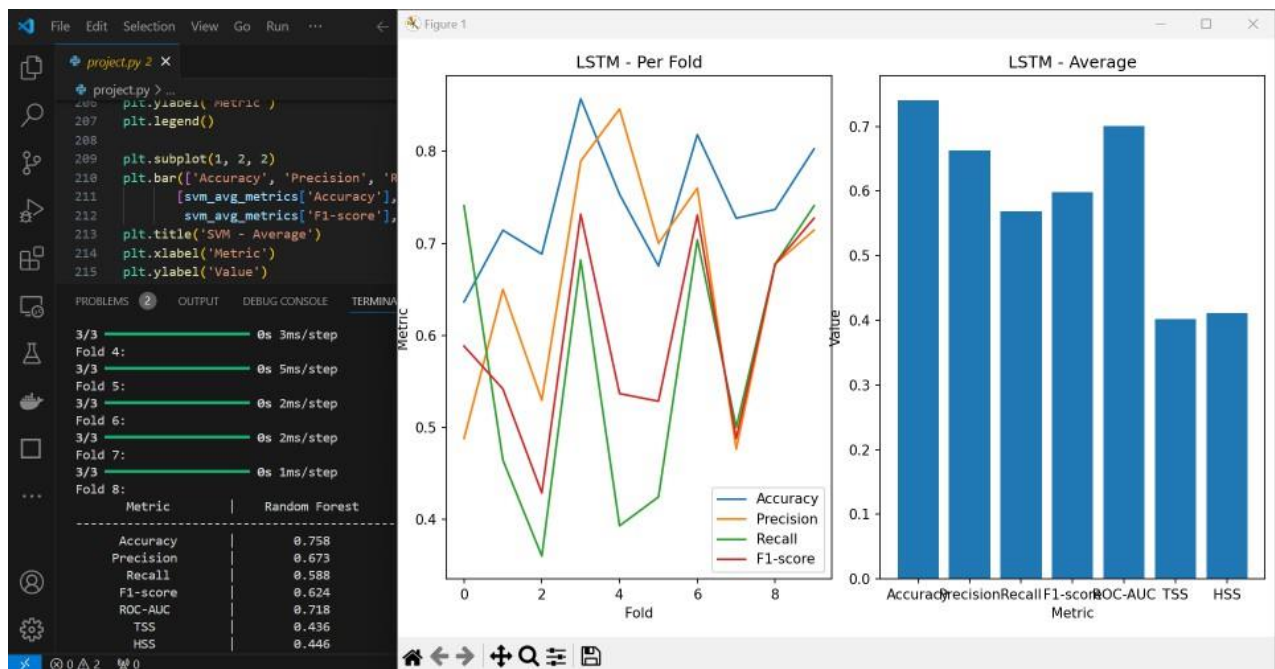
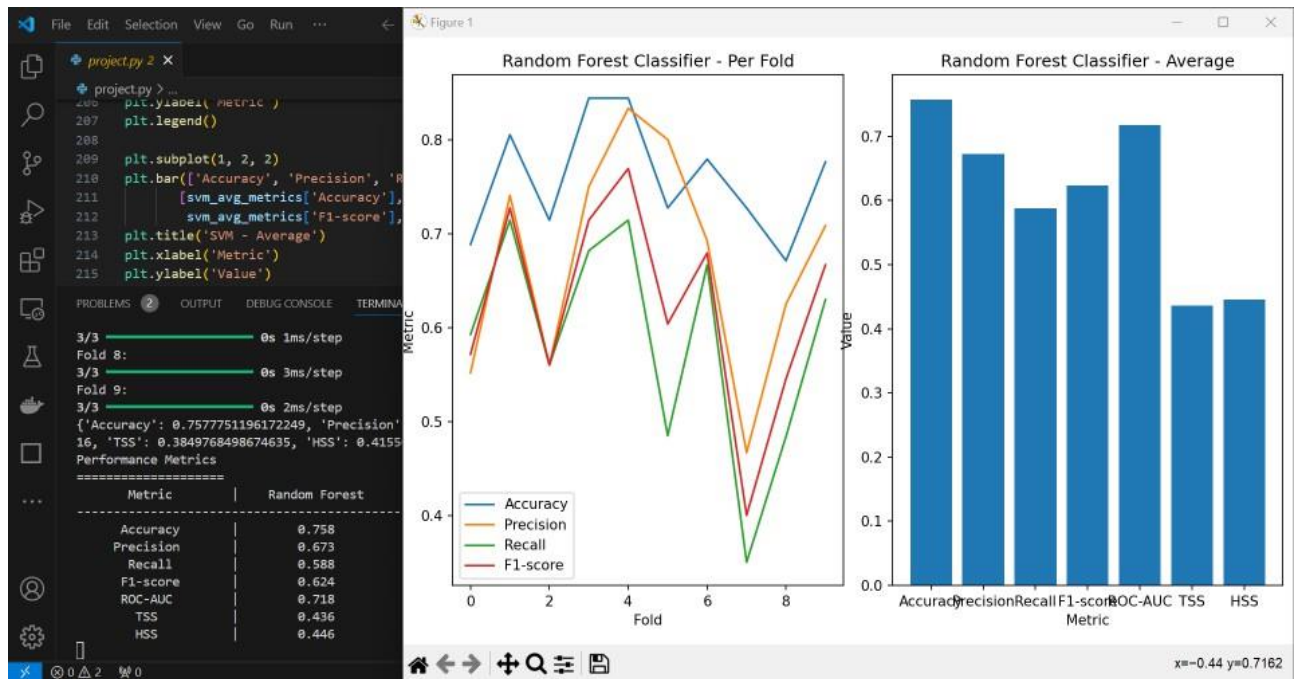
Fold 7: 3/3 0s 1ms/step
Fold 8: 3/3 0s 3ms/step
Fold 9: 3/3 0s 2ms/step

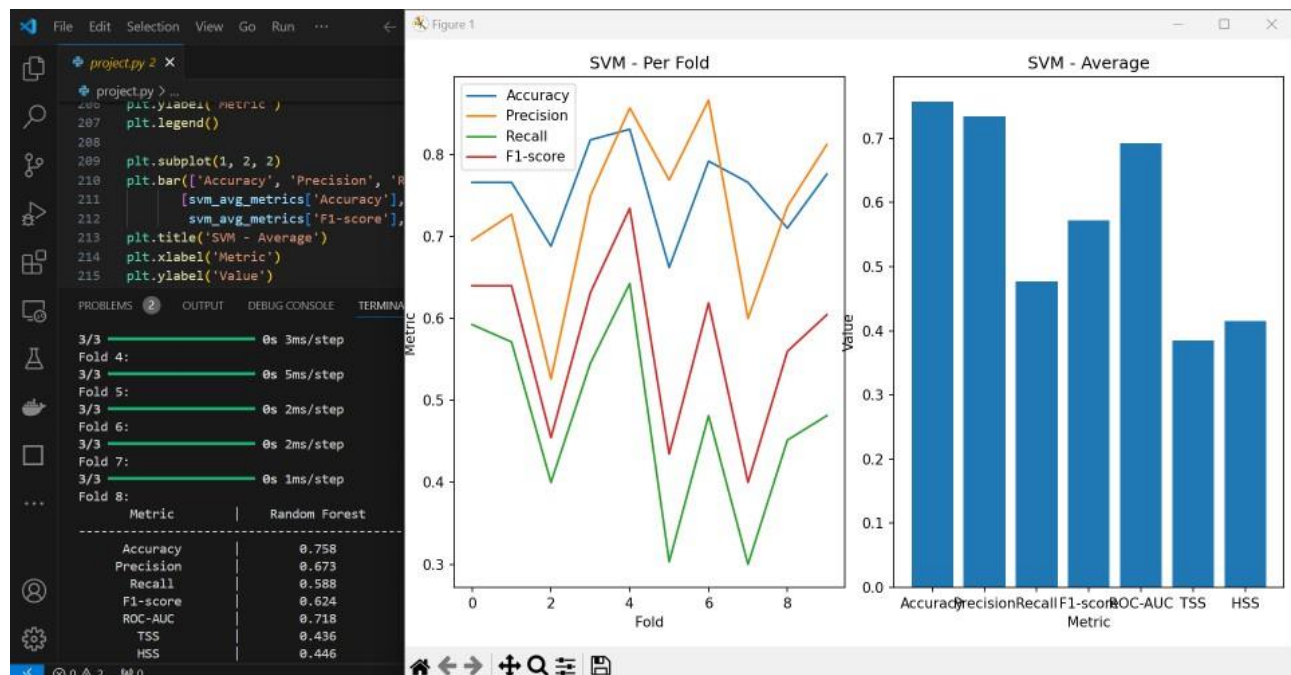
{'Accuracy': 0.7577751196172249, 'Precision': 0.7341623088962905, 'Recall': 0.47699390215519244, 'F1-score': 0.5719299669998865, 'ROC-AUC': 0.69248842493373
16, 'TSS': 0.3849768498674635, 'HSS': 0.41556292213028156}

Performance Metrics

Metric	Random Forest	LSTM	SVM
Accuracy	0.758	0.741	0.758
Precision	0.673	0.663	0.734
Recall	0.588	0.569	0.477
F1-score	0.624	0.598	0.572
ROC-AUC	0.718	0.701	0.692
TSS	0.436	0.402	0.385
HSS	0.446	0.411	0.416

Ln 1, Col 1 Spaces: 4 UTF-8 LF Python 3.12.2 64-bit





Running the Program using Google Colab

Upload the python notebook on Google Colab and upload the dataset in the project file explorer. Run the cells and explore the output.

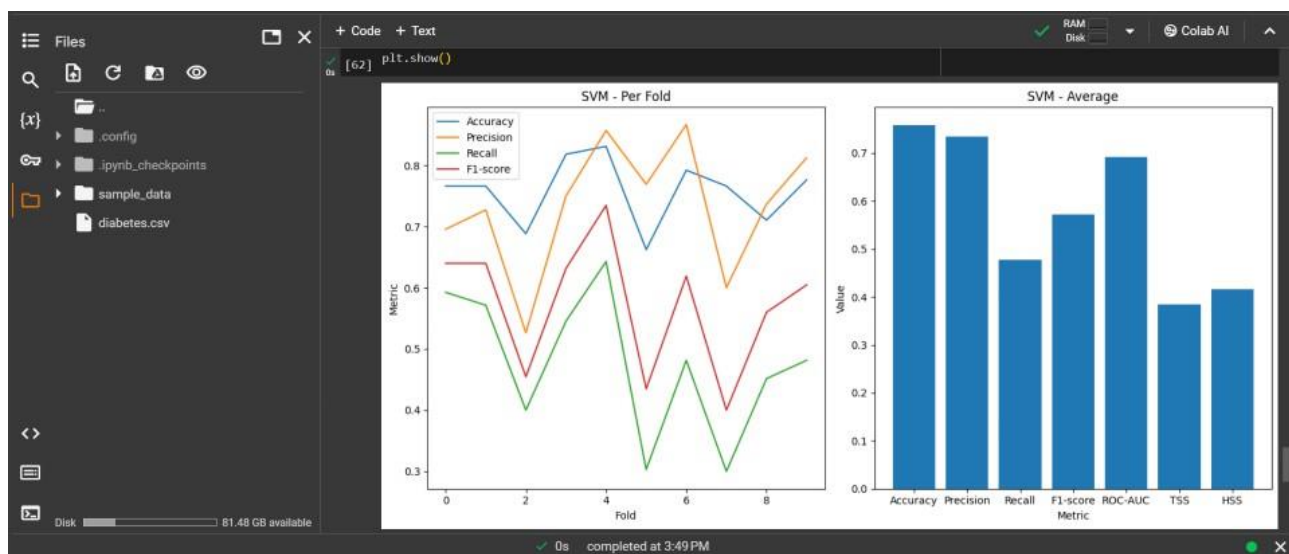
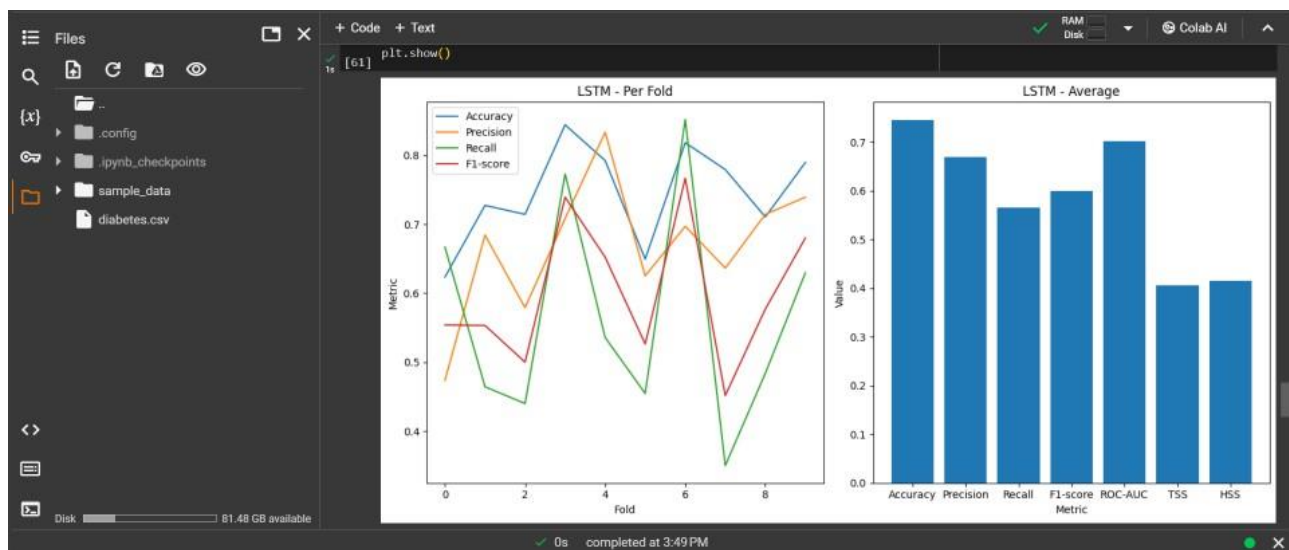
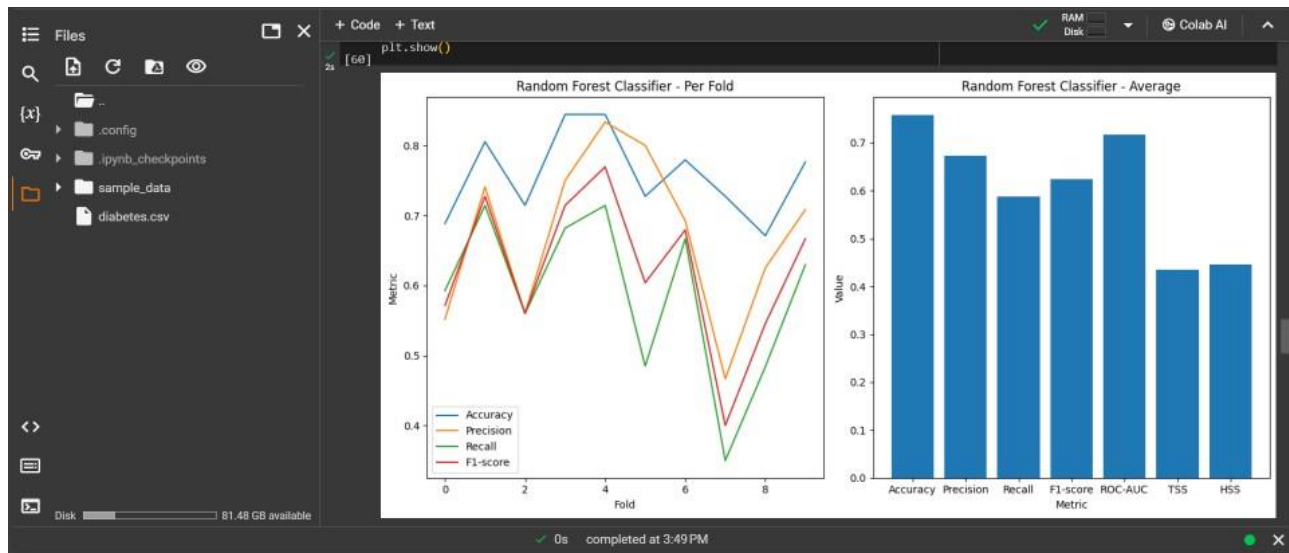
The screenshot shows a Jupyter Notebook environment. On the left is a file explorer with a sidebar containing icons for files, folders, and search. The main area of the file explorer shows a directory structure with files like 'diabetes.csv' and folders like 'sample_data'. The central part of the image is a code editor with a dark background, showing a Jupyter cell with a magic command to print performance metrics in a tabular format. The code is as follows:

```
[59] # Print the results in tabular format
print('Performance Metrics')
print('=' * 20)
print('{:^20} | {:^20} | {:^20} | {:^20}'.format('Metric', 'Random Forest', 'LSTM', 'SVM'))
print('=' * 80)
for metric in ['Accuracy', 'Precision', 'Recall', 'F1-score', 'ROC-AUC', 'TSS', 'HSS']:
    print('{:^20} | {:^20.3f} | {:^20.3f} | {:^20.3f}'.format(metric, rf_avg_metrics[metric], lstm_avg_metrics[metric], svm_avg_metrics[metric]))
```

The terminal at the bottom shows the output of the code, which is a table with columns for the metric and the three models: Random Forest, LSTM, and SVM. The table is as follows:

Metric	Random Forest	LSTM	SVM
Accuracy	0.758	0.745	0.758
Precision	0.673	0.669	0.734
Recall	0.588	0.565	0.477
F1-score	0.624	0.600	0.572
ROC-AUC	0.718	0.703	0.692
TSS	0.436	0.405	0.385
HSS	0.446	0.416	0.416

The terminal also shows the status bar at the bottom, which indicates that the code was completed at 3:49 PM.



The program will prompt the table and the visualization of the compare the performance of three different machine learning algorithms for binary classification on the Pima Indians Diabetes Dataset.

Algorithm Comparison

The results of the algorithm comparison showed that the Random Forest Classifier outperformed the LSTM and SVM models across most of the evaluated metrics, including Accuracy, F1-score, and ROC-AUC. The strong performance of the Random Forest Classifier can be attributed to its ability to effectively handle the non-linearity, feature interactions, and noise present in the dataset.

The LSTM model also exhibited promising results, particularly in its ability to capture temporal or sequential patterns in the data. However, the complexity of tuning the LSTM hyperparameters and the relatively small size of the dataset may have limited its overall performance compared to the more robust Random Forest approach.

The SVM model exhibited reasonable performance but was outperformed by the Random Forest Classifier across most of the evaluation metrics. This suggests that the non-linear decision boundaries and feature interactions in the dataset were better captured by the ensemble-based Random Forest algorithm.

In conclusion, this project has demonstrated the effectiveness of using a comparative analysis approach to evaluate and select the most suitable machine learning algorithm for a binary classification task. The detailed performance evaluation and discussion provided can serve as a valuable guide for future data mining projects with similar objectives.