Comparative Analysis of Machine Learning Algorithms for Binary Classification

CS 634 Data Mining

Final Term Project

Dr. Yasser Abduallah

Department of Computer Science

New Jersey Institute of Technology

Rahul CH

rc738@njit.edu

14th April 2024

Table of content

Github Link:

https://github.com/rahul7381/CS-634-Data-Mining-Final-Term-Project

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)

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:

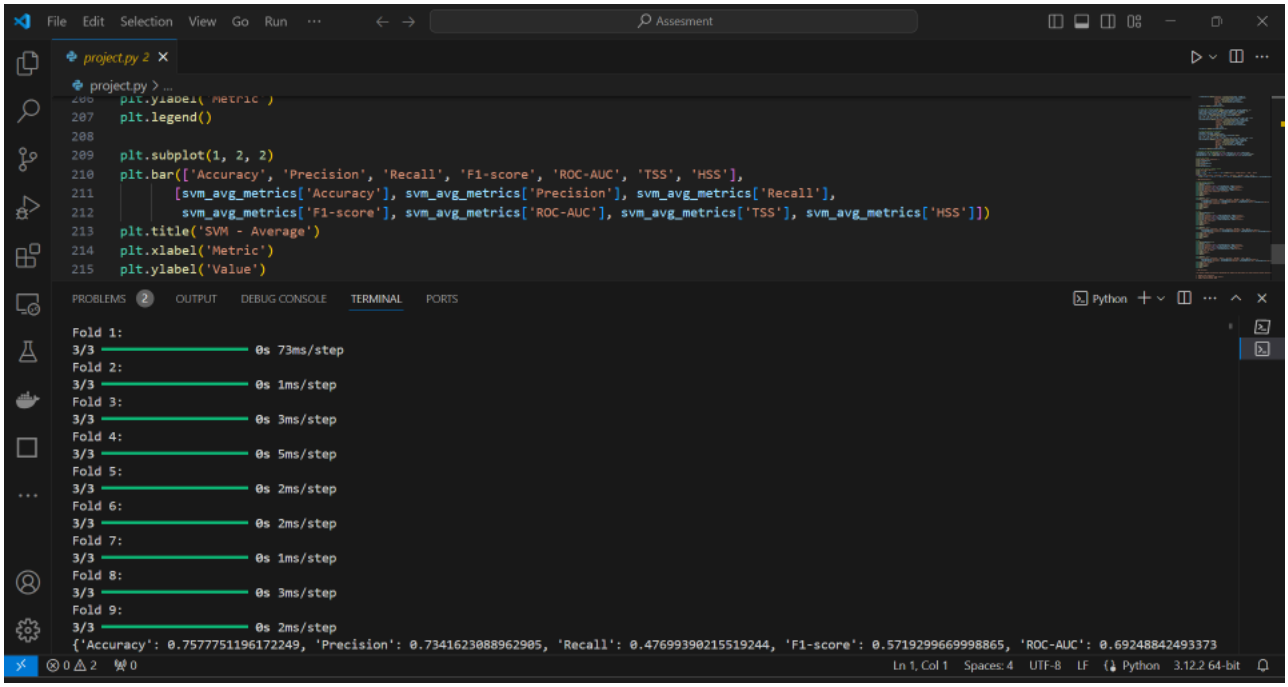
```
pip install numpy pandas matplotlib scikit-learn tensorflow
```

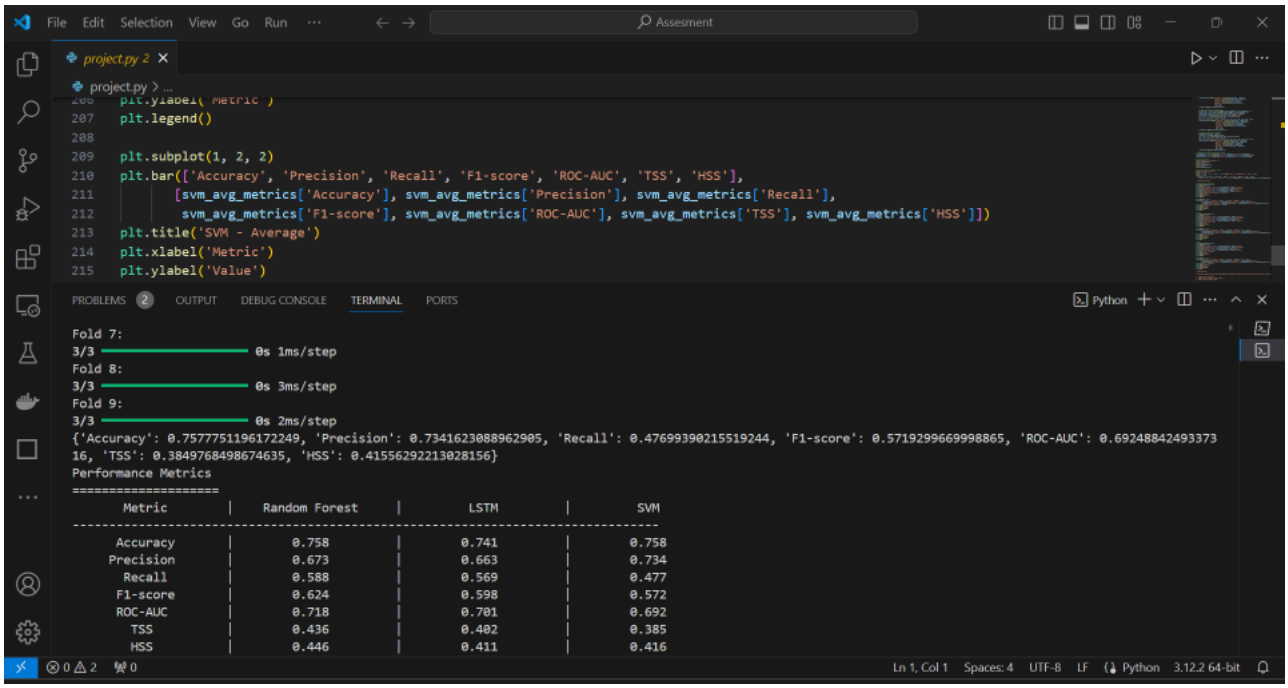
Running the Program using Python file

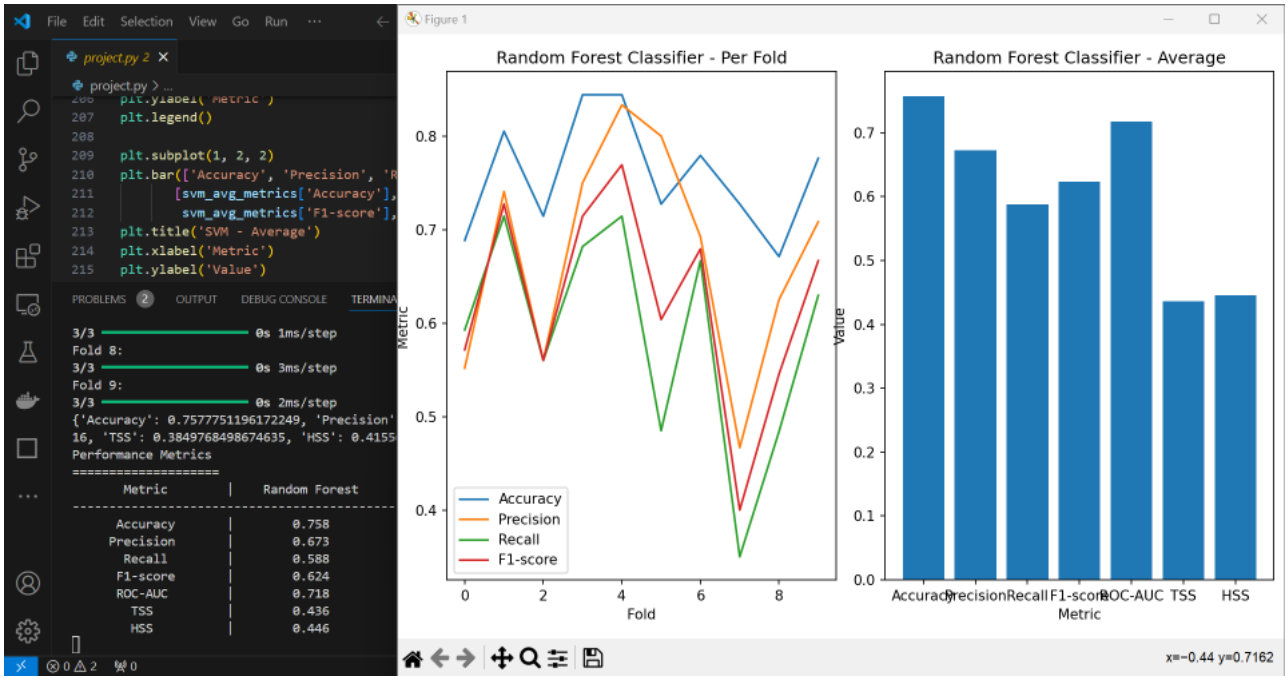
Open the Terminal and navigate to the directory containing the project files and run the following command.

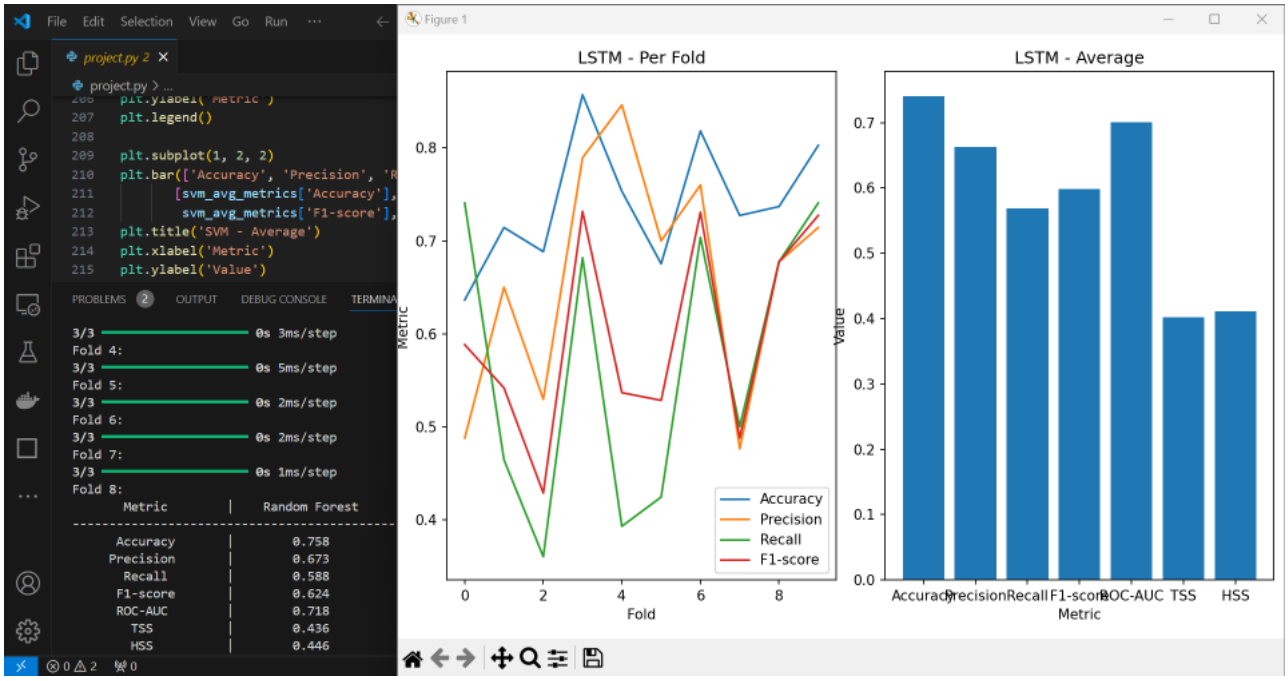
```
python project.py
```

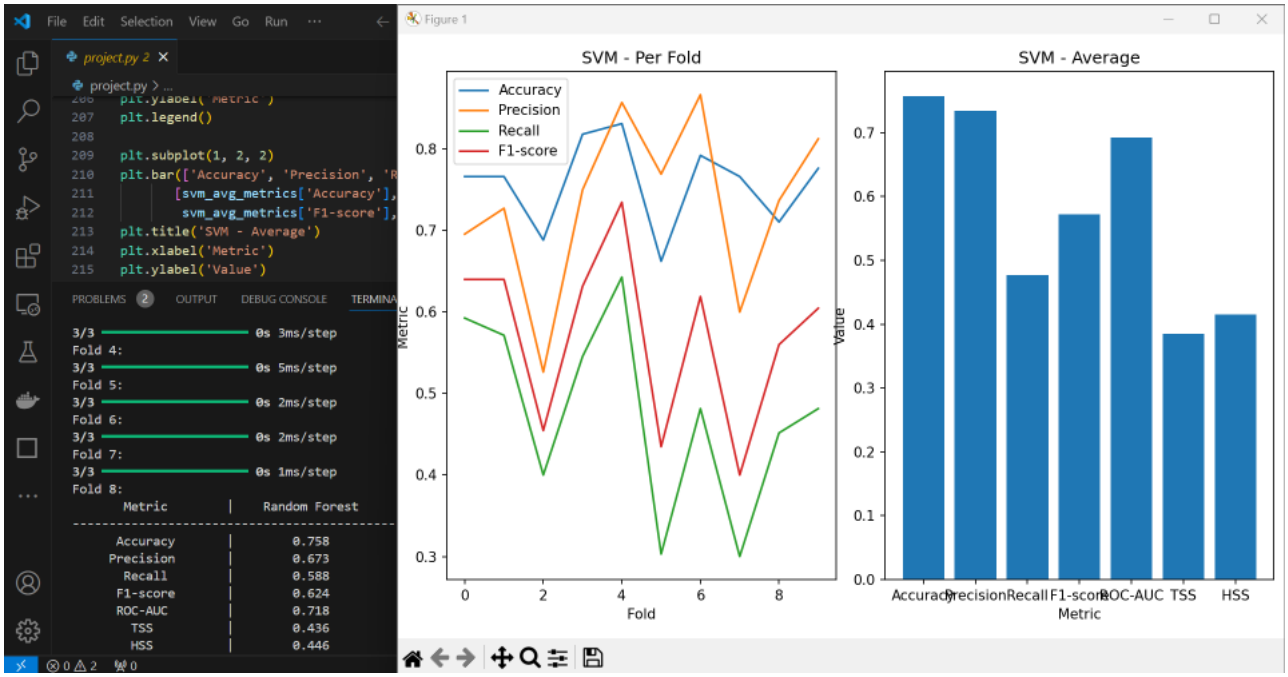
The below screenshot shows running the python file in terminal by selecting Pima Indians Diabetes dataset.


File Edit Selection View Go Run
Assesment
project.py 2
project.py > ...
plt.legend()
plt.subplot(1, 2, 2)
plt.bar(['Accuracy', 'Precision', 'Recall', 'F1-score', 'ROC-AUC', 'TSS', 'HSS'],
[svm_avg_metrics['Accuracy'], svm_avg_metrics['Precision'], svm_avg_metrics['Recall'],
svm_avg_metrics['F1-score'], svm_avg_metrics['ROC-AUC'], svm_avg_metrics['TSS'], svm_avg_metrics['HSS']])
plt.title('SVM - Average')
plt.xlabel('Metric')
plt.ylabel('Value')
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
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': 0.4769939021551924 4, '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
Assesment
project.py 2
project.py > ...
plt.legend()
plt.subplot(1, 2, 2)
plt.bar(['Accuracy', 'Precision', 'Recall', 'F1-score', 'ROC-AUC', 'TSS', 'HSS'],
[svm_avg_metrics['Accuracy'], svm_avg_metrics['Precision'], svm_avg_metrics['Recall'],
svm_avg_metrics['F1-score'], svm_avg_metrics['ROC-AUC'], svm_avg_metrics['TSS'], svm_avg_metrics['HSS']])
plt.title('SVM - Average')
plt.xlabel('Metric')
plt.ylabel('Value')
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
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': 0.4769939021551924 4, 'F1-score': 0.5719299669998865, 'ROC-AUC': 0.69248842493373
16, 'TSS': 0.3849768498674635, 'HSS': 0.4155629221302815 6}
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

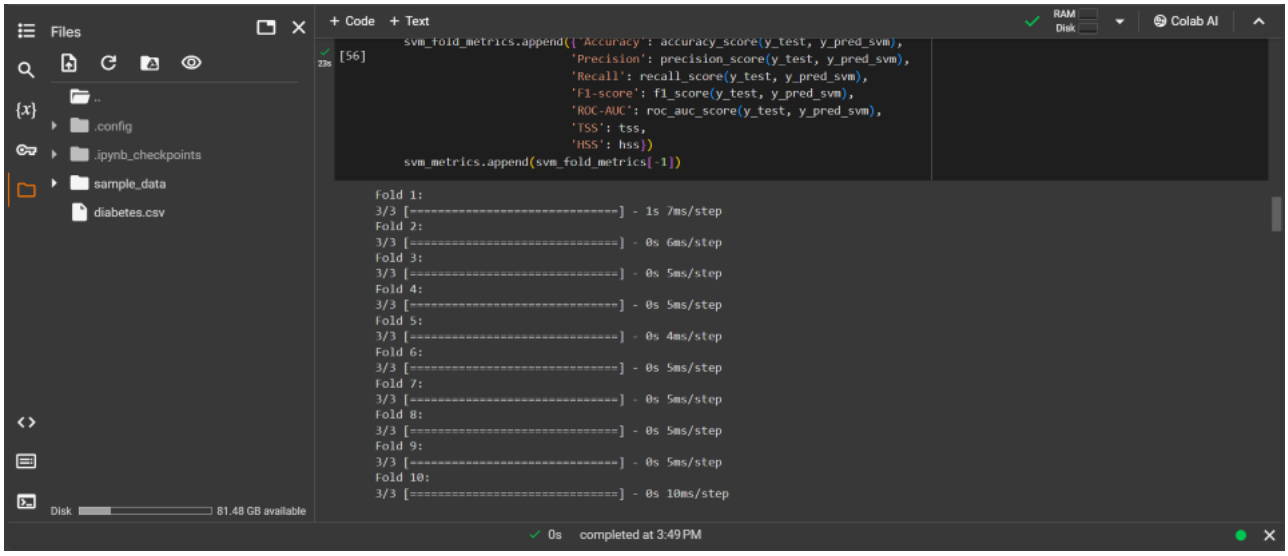

File Edit Selection View Go Run
project.py 2
plt.legend()
plt.subplot(1, 2, 2)
plt.bar(['Accuracy', 'Precision', 'R
[svm_avg_metrics['Accuracy'],
svm_avg_metrics['F1-score'],
plt.title('SVM - Average')
plt.xlabel('Metric')
plt.ylabel('Value')
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINA
3/3 0s 1ms/step
Fold 8:
3/3 0s 3ms/step
Fold 9:
3/3 0s 2ms/step
{'Accuracy': 0.757751196172249, 'Precision'
16, 'TSS': 0.3849768498674635, 'HSS': 0.4155
Performance Metrics
Metric | Random Forest
Accuracy | 0.758
Precision | 0.673
Recall | 0.588
F1-score | 0.624
ROC-AUC | 0.718
TSS | 0.436
HSS | 0.446
Figure 1
Random Forest Classifier - Per Fold
Metric
Fold
Accuracy
Precision
Recall
F1-score
Random Forest Classifier - Average
Value
Metric
Accuracy
Precision
Recall
F1-score
ROC-AUC
TSS
HSS
x=-0.44 y=0.7162

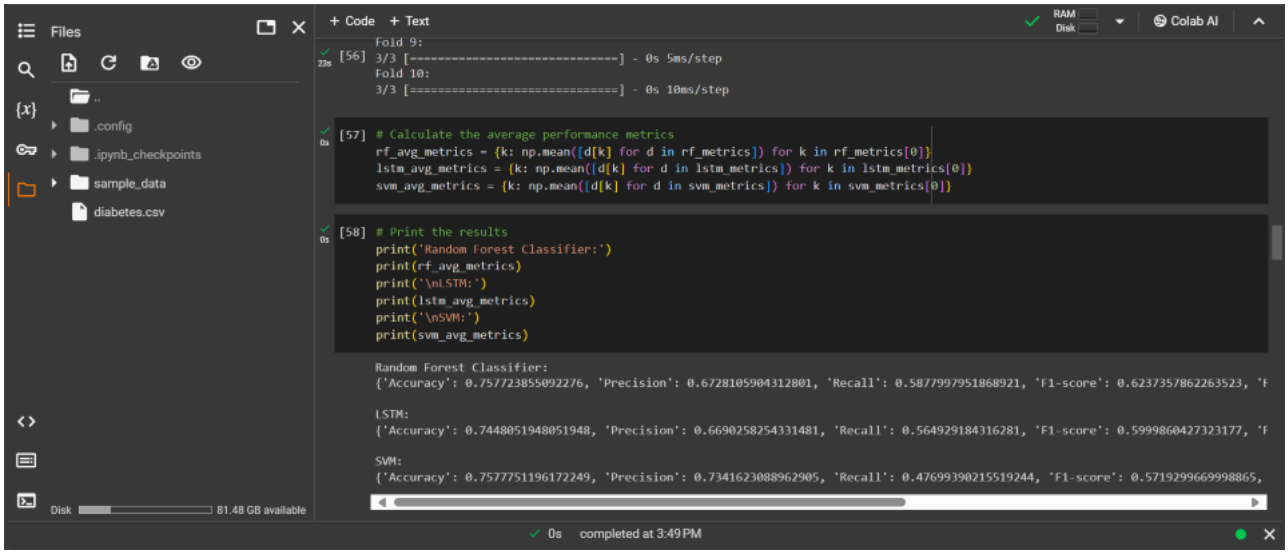

File Edit Selection View Go Run
project.py 2
plt.legend()
plt.subplot(1, 2, 2)
plt.bar(['Accuracy', 'Precision', 'R
[svm_avg_metrics['Accuracy'],
svm_avg_metrics['F1-score'],
plt.title('SVM - Average')
plt.xlabel('Metric')
plt.ylabel('Value')
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINA
3/3 0s 3ms/step
Fold 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:
Metric | Random Forest
Accuracy | 0.758
Precision | 0.673
Recall | 0.588
F1-score | 0.624
ROC-AUC | 0.718
TSS | 0.436
HSS | 0.446
Figure 1
LSTM - Per Fold
Metric
Fold
Accuracy
Precision
Recall
F1-score
LSTM - Average
Value
Metric
Accuracy
Precision
Recall
F1-score
ROC-AUC
TSS
HSS

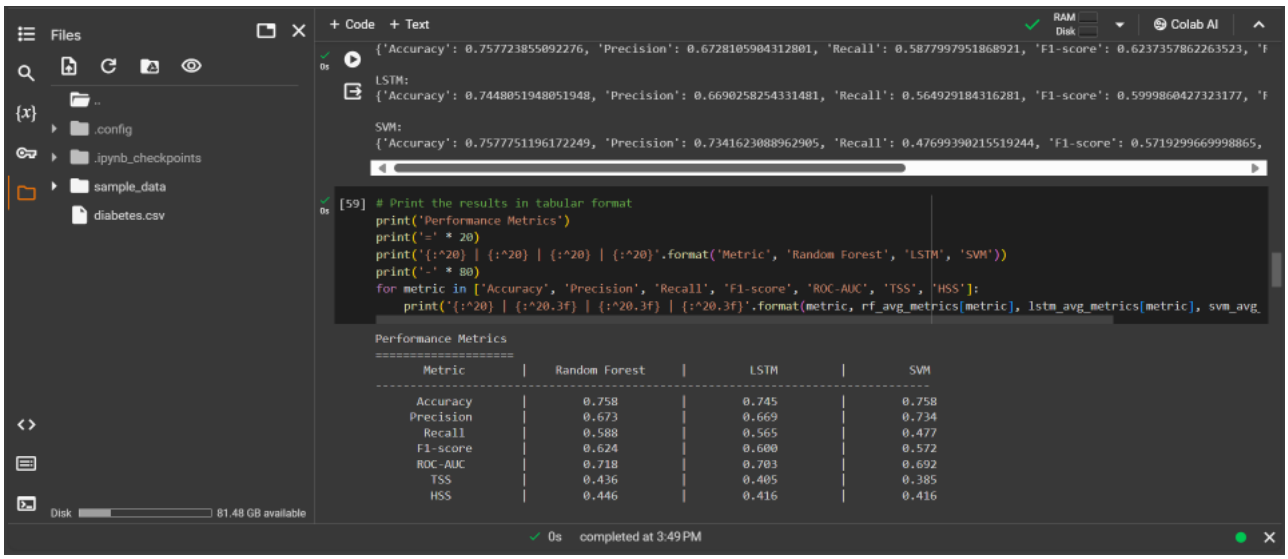


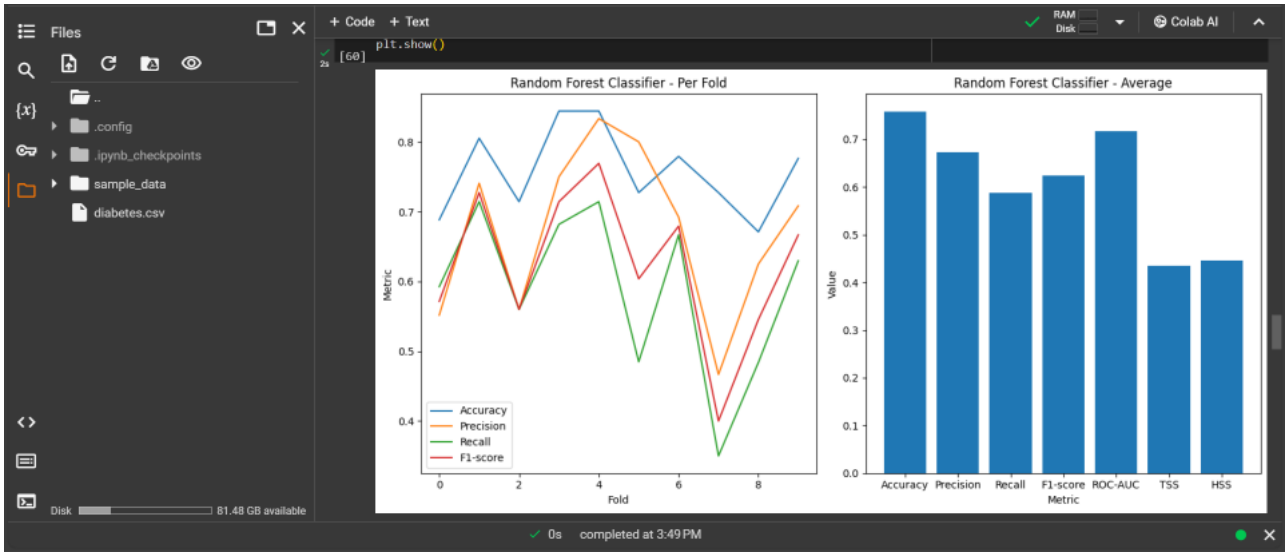
Running the Program using Google Colab

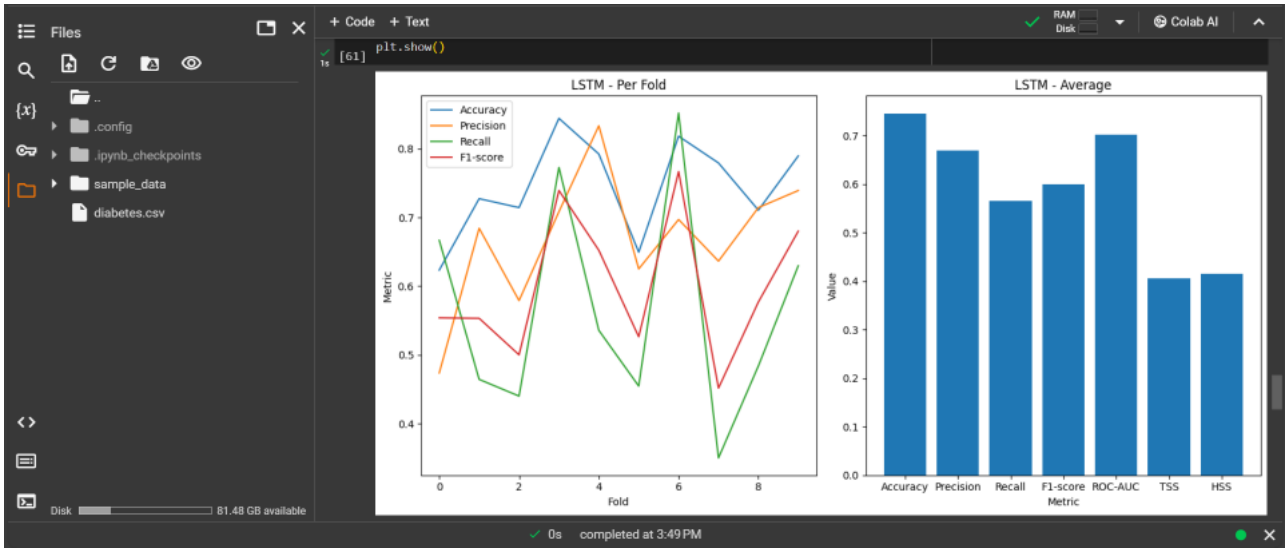
Upload the python notebook on Google Colab and upload the dataset in the project file explorer. Run the cells and explore the output.

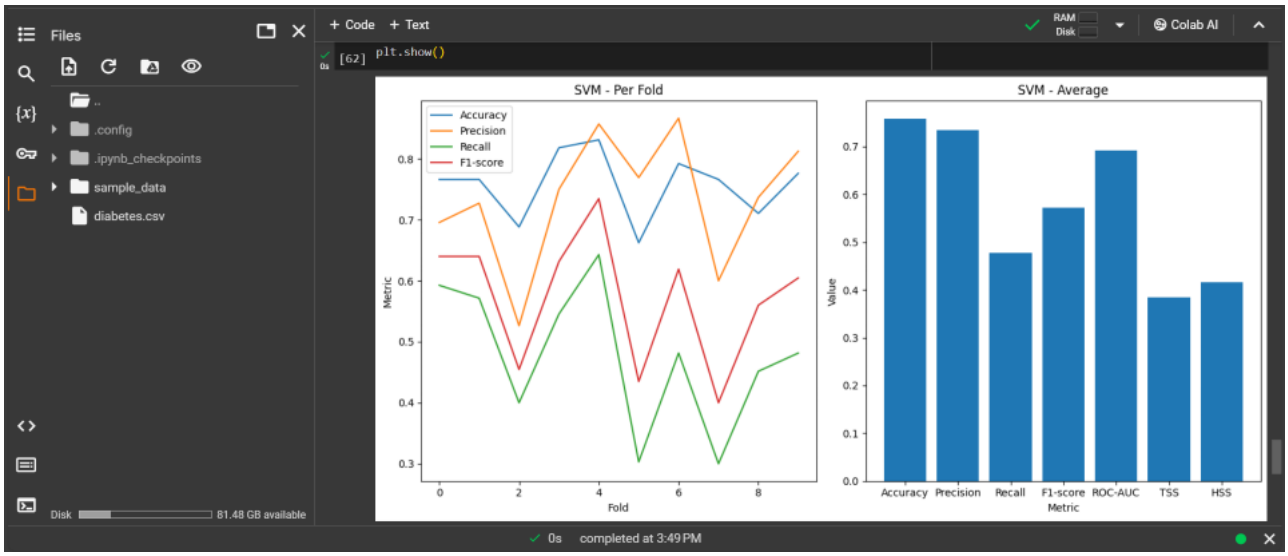

Files
+ Code + Text
RAM
Disk
Colab AI
[56]
svm_fold_metrics.append({'Accuracy': accuracy_score(y_test, y_pred_svm),
'Precision': precision_score(y_test, y_pred_svm),
'Recall': recall_score(y_test, y_pred_svm),
'F1-score': f1_score(y_test, y_pred_svm),
'ROC-AUC': roc_auc_score(y_test, y_pred_svm),
'TSS': tss,
'HSS': hss})
svm_metrics.append(svm_fold_metrics[-1])
Fold 1:
3/3 [==============================] - 1s 7ms/step
Fold 2:
3/3 [==============================] - 0s 6ms/step
Fold 3:
3/3 [==============================] - 0s 5ms/step
Fold 4:
3/3 [==============================] - 0s 5ms/step
Fold 5:
3/3 [==============================] - 0s 4ms/step
Fold 6:
3/3 [==============================] - 0s 5ms/step
Fold 7:
3/3 [==============================] - 0s 5ms/step
Fold 8:
3/3 [==============================] - 0s 5ms/step
Fold 9:
3/3 [==============================] - 0s 5ms/step
Fold 10:
3/3 [==============================] - 0s 10ms/step
.config
.ipynb_checkpoints
sample_data
diabetes.csv
Disk
81.48 GB available
0s completed at 3:49 PM


Files
+ Code + Text
RAM
Disk
Colab AI
Fold 9:
3/3 [==============================] - 0s 5ms/step
Fold 10:
3/3 [==============================] - 0s 10ms/step
[57]
# Calculate the average performance metrics
rf_avg_metrics = {k: np.mean([d[k] for d in rf_metrics]) for k in rf_metrics[0]}
lstm_avg_metrics = {k: np.mean([d[k] for d in lstm_metrics]) for k in lstm_metrics[0]}
svm_avg_metrics = {k: np.mean([d[k] for d in svm_metrics]) for k in svm_metrics[0]}
[58]
# Print the results
print('Random Forest Classifier:')
print(rf_avg_metrics)
print('\nLSTM:')
print(lstm_avg_metrics)
print('\nSVM:')
print(svm_avg_metrics)
Random Forest Classifier:
{'Accuracy': 0.757723855092276, 'Precision': 0.6728105904312801, 'Recall': 0.5877997951868921, 'F1-score': 0.6237357862263523, 'F
LSTM:
{'Accuracy': 0.7448051948051948, 'Precision': 0.6690258254331481, 'Recall': 0.564929184316281, 'F1-score': 0.5999860427323177, 'F
SVM:
{'Accuracy': 0.7577751196172249, 'Precision': 0.7341623088962905, 'Recall': 0.47699390215519244, 'F1-score': 0.5719299669998865,
.config
.ipynb_checkpoints
sample_data
diabetes.csv
Disk
81.48 GB available
0s completed at 3:49 PM


Files
+ Code + Text
RAM
Disk
Colab AI
{'Accuracy': 0.757723855092276, 'Precision': 0.6728105904312801, 'Recall': 0.5877997951868921, 'F1-score': 0.6237357862263523, 'F
LSTM:
{'Accuracy': 0.7448051948051948, 'Precision': 0.6690258254331481, 'Recall': 0.564929184316281, 'F1-score': 0.5999860427323177, 'F
SVM:
{'Accuracy': 0.7577751196172249, 'Precision': 0.7341623088962905, 'Recall': 0.47699390215519244, 'F1-score': 0.5719299669998865,
[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_
Performance Metrics
==========
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
.config
.ipynb_checkpoints
sample_data
diabetes.csv
Disk
81.48 GB available
0s 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.