**pathania\_vandana\_final\_project July 10, 2024**

**--------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**Project Title:** Using KNN, RF and LSTM to predict whether a student will go to college based on various factors.

**Goal:** The goal of this project is to train and evaluate three different machine learning models K-Nearest Neighbors, Random Forest, and LSTM on a dataset to predict whether a student will go to college based on various factors. The dataset includes features such as school type, school accreditation, gender, interest, residence, parents' age, parents' salary, house area, average grades, and whether a parent attended college. The performance of each model is assessed using various metrics, and the results are compared to determine the best performing model.

**Github Link:** [**https://github.com/vp758/pathania\_vandana\_final\_term\_project.git**](https://github.com/vp758/pathania_vandana_final_term_project.git)

**Dataset Link:** [**High school student data related to the decision to go to college**](https://www.kaggle.com/datasets/saddamazyazy/go-to-college-dataset)

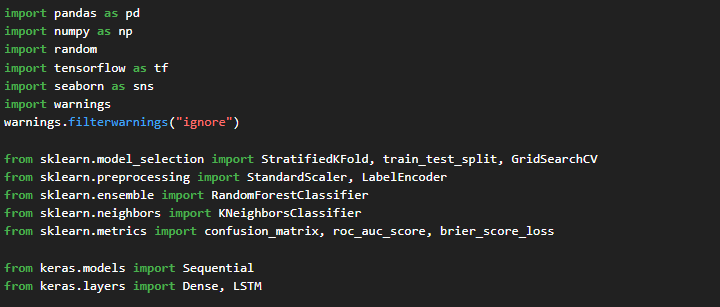
**Software and Tools:**

The following software libraries and tools were used in this project:

* **Python:**
  + Download and install Python from the official website: <https://www.python.org/>
* **Pandas:**
  + Used for data manipulation and analysis.
  + Official documentation: <https://pandas.pydata.org/>
* **NumPy:**
  + Used for numerical computations.
  + Official website: <https://numpy.org/>
* **Matplotlib:**
  + Used for creating static, animated, and interactive visualizations.
  + Official documentation: <https://matplotlib.org/>
* **Seaborn:**
  + Used for statistical data visualization.
  + Official documentation: <https://seaborn.pydata.org/>
* **Scikit-learn:**
  + Used for implementing machine learning algorithms.
  + Official website: <https://scikit-learn.org/>
* **Keras:**
  + Used for building and training deep learning models.
  + Official website: <https://keras.io/>
* **TensorFlow:**
* Used as the backend for Keras and for setting random seeds for reproducibility.
* Official website: https://www.tensorflow.org/
* **Random:**
* Used for generating random numbers and setting random seeds for reproducibility.
* Documentation: <https://docs.python.org/3/library/random.html>
* **Warnings:**
* Used to manage and filter warnings in the code.
* Documentation: https://docs.python.org/3/library/warnings.html

**1. Jupyter full workflow with the code divided according to the specified steps:**

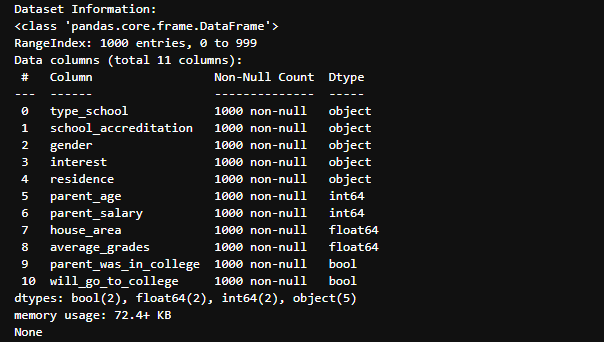
**1.1. Import necessary libraries**



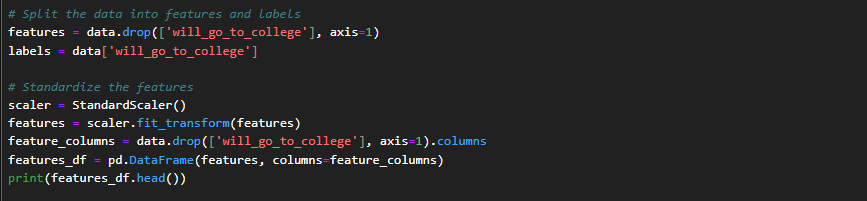
**1.2. Load the dataset, handle missing values, and encode categorical variables.**



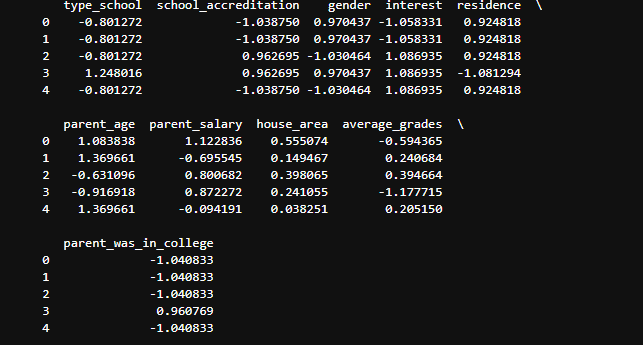
**Print(data.info())**



**1.3. Standardize the features to have a mean of 0 and a standard deviation of 1**



**Print(features\_df.head())**



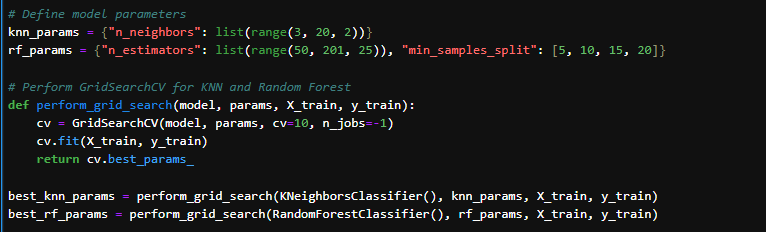
**1.4. Split the data into training and testing sets.**



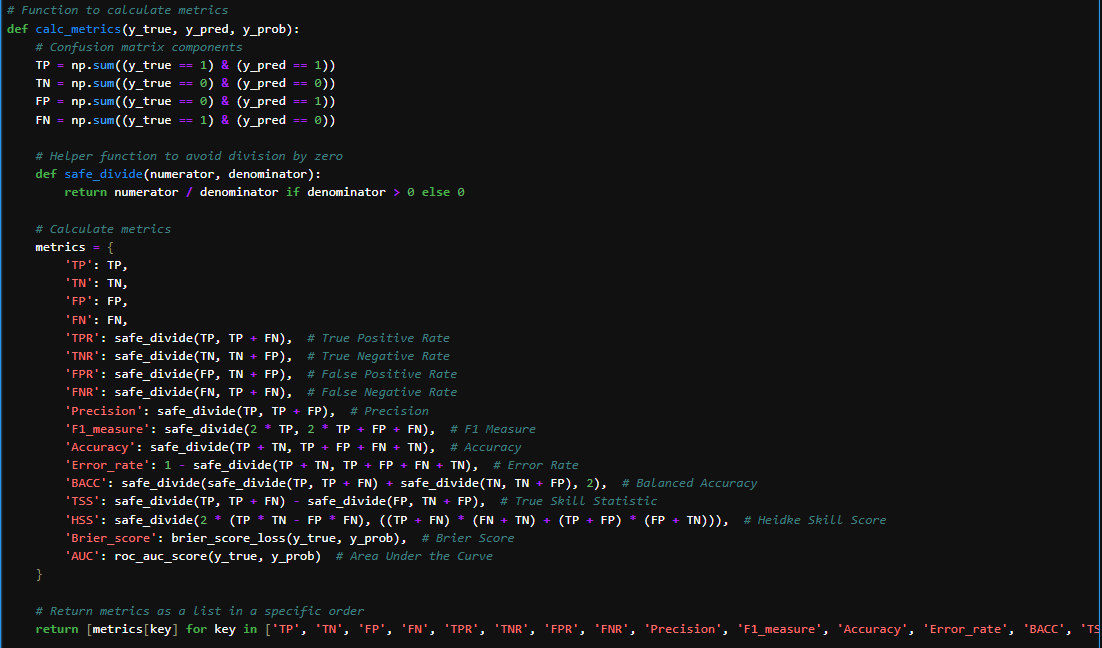
**1.5. Define a Stratified K-Fold cross-validator**



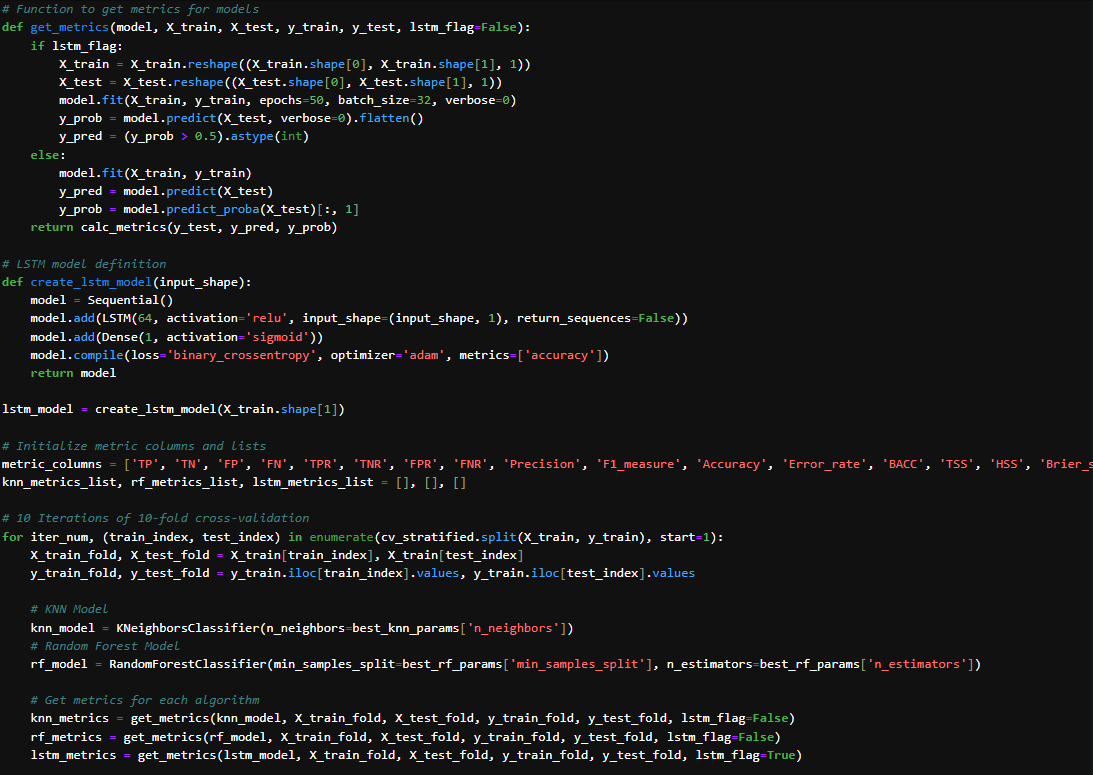
**1.6. Perform grid search to find the best hyperparameters for KNN and Random Forest**

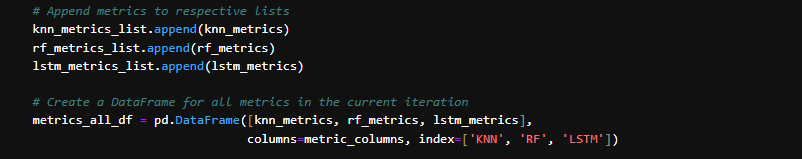


**1.7. Define a calc\_metrics function to calculate various performance metrics**

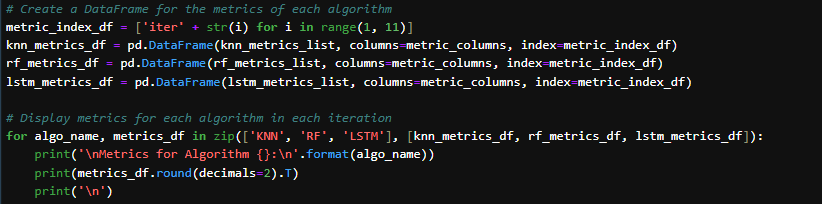


**1.8. Train and evaluate KNN, Random Forest, and LSTM models using 10-fold cross-validation**

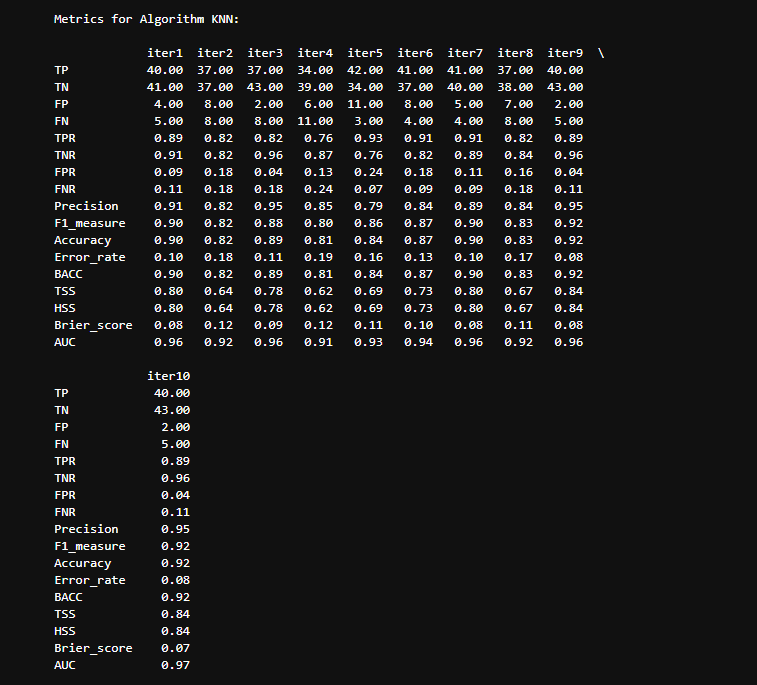




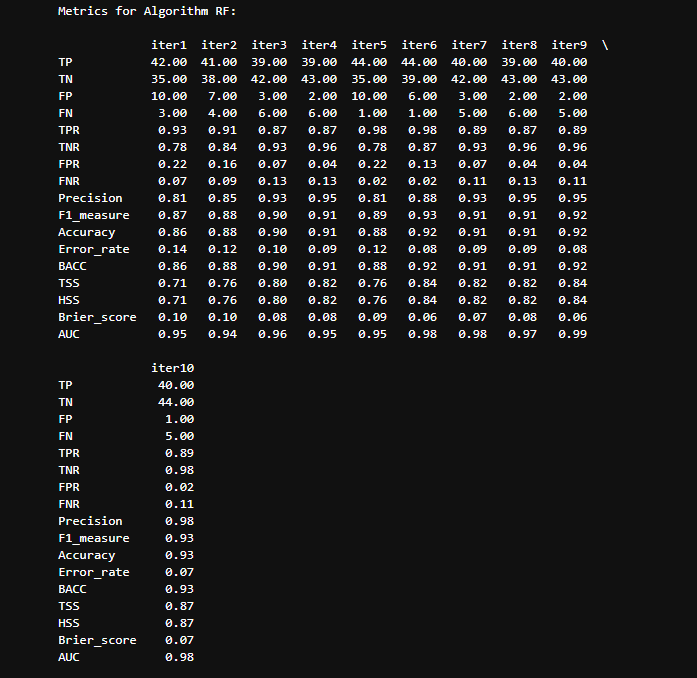
**1.9. Create DataFrames to store the metrics for each fold and algorithm, and then display the results**



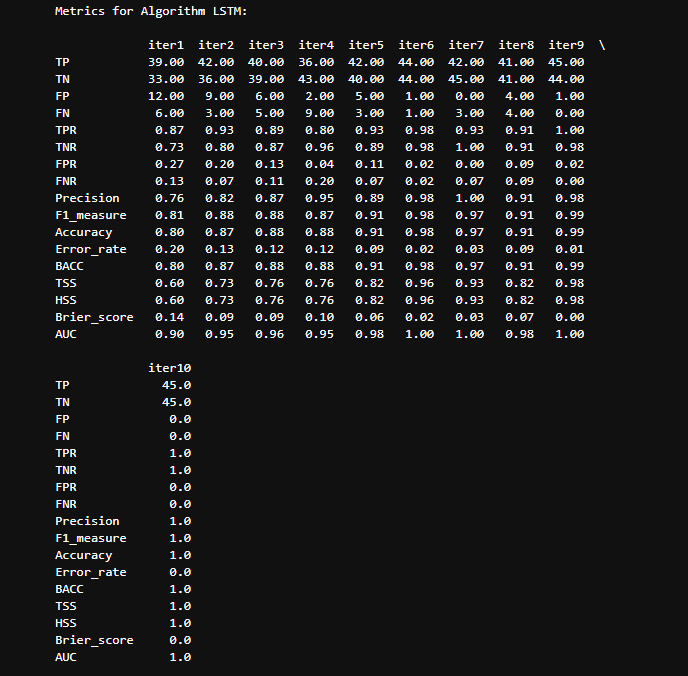
**Metrics for Algorithm KNN:**



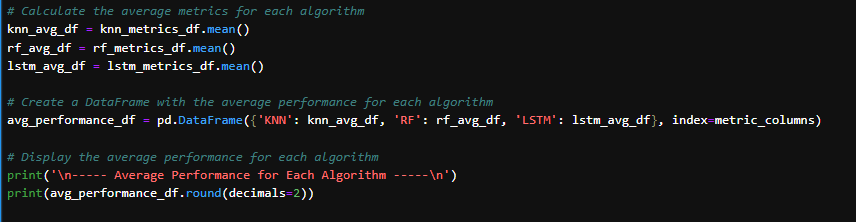
**Metrics for Algorithm RF:**

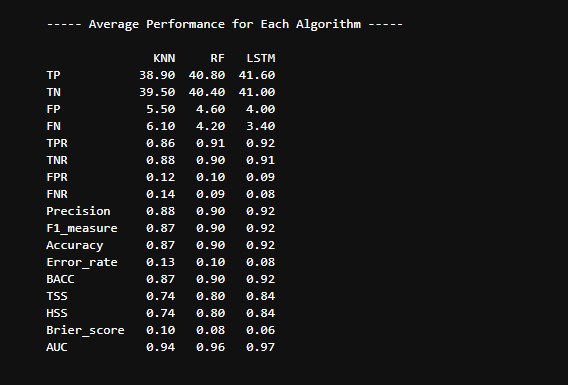


**Metrics for Algorithm LSTM:**



**1.10. Calculate the average performance metrics for each algorithm and display the results**





**2. Python full workflow with the code divided according to the specified steps:**

**2.1. Install Python**

Make sure you have Python installed on your system. You can download it from the official Python website. Follow the installation instructions for your operating system.

**2.2. Open Command Prompt (CMD)**

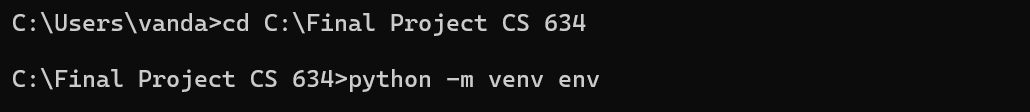
Windows: Press Win + R, type cmd, and press Enter.

**2.3. Create a Virtual Environment**

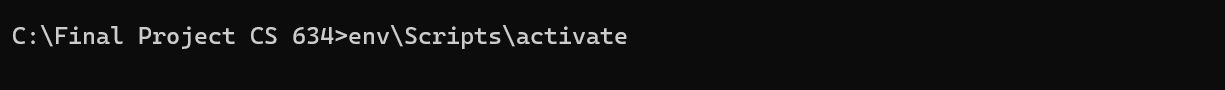
Navigate to the directory where you want to create your virtual environment.



Create the virtual environment using the venv module:



**2.4. Activate the Virtual Environment**



After activation, your command prompt should show the name of the virtual environment, indicating that it is active.

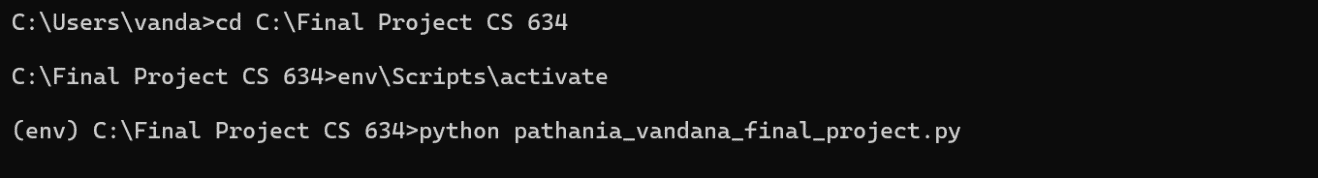
**2.5. Install Third-Party Tools Using pip**

With the virtual environment activated, you can install any third-party Python package using pip.

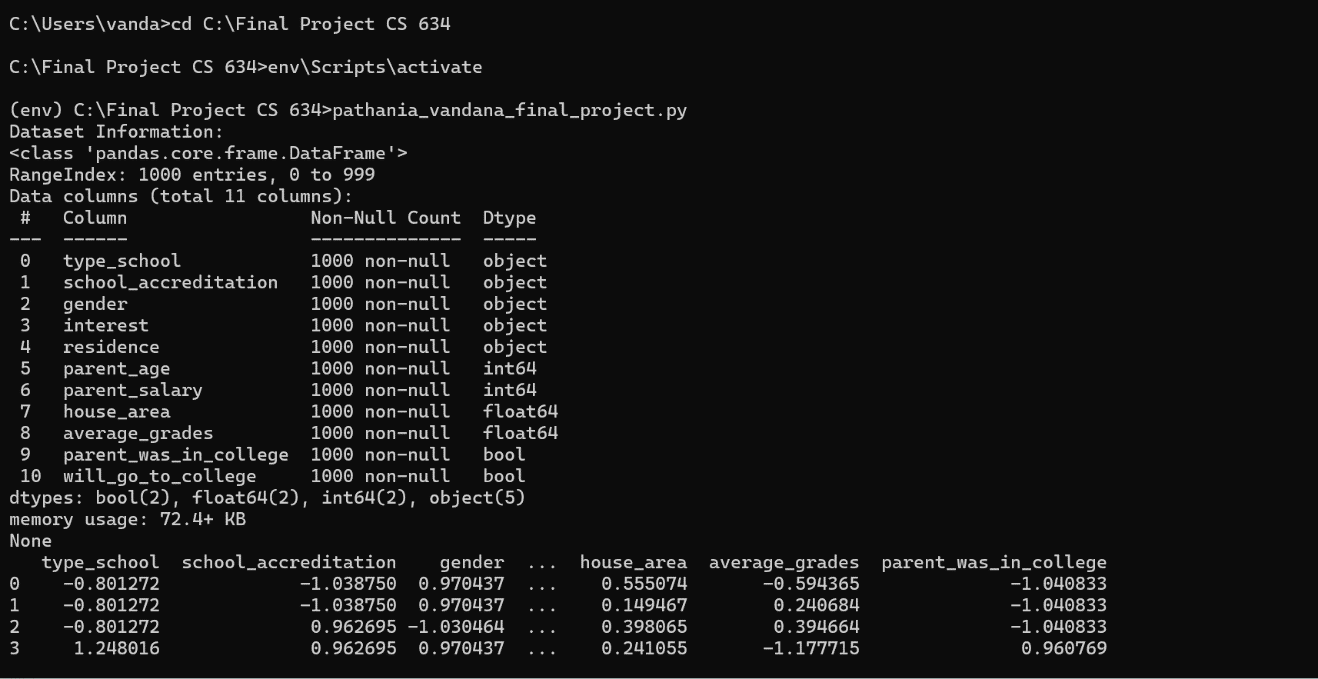


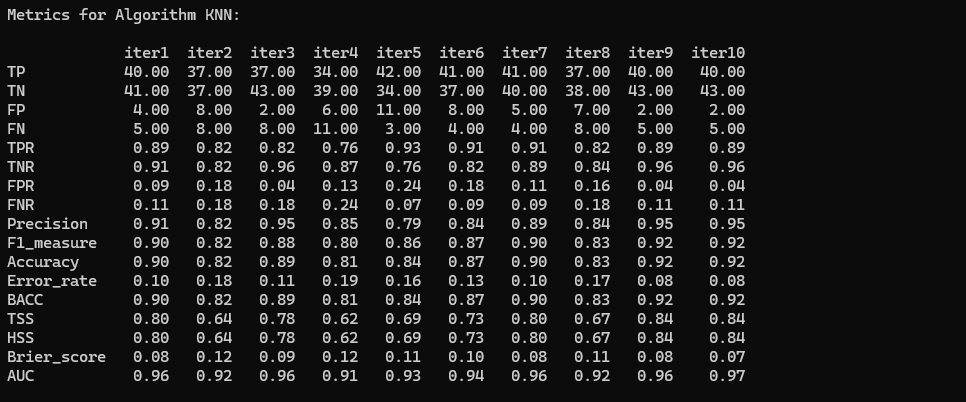
**2.6.** **Open Python Interpreter**:

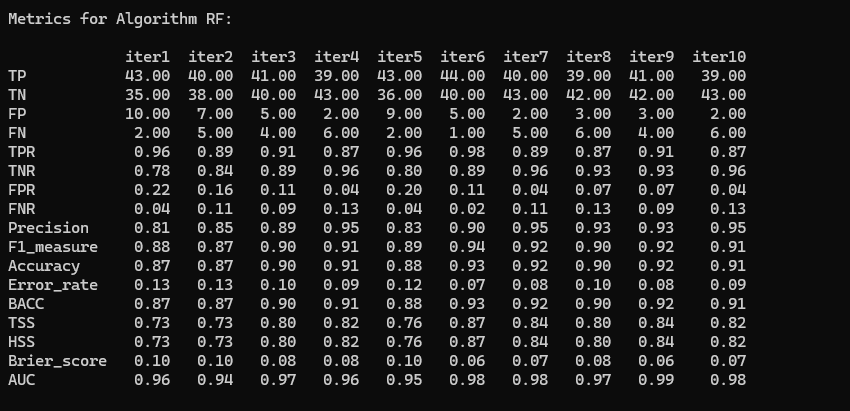
To open the Python interpreter within the virtual environment, simply type python and name of file and execute it.

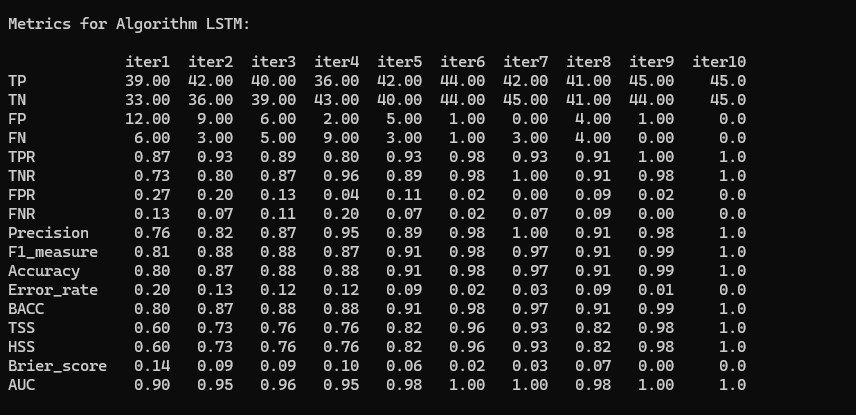


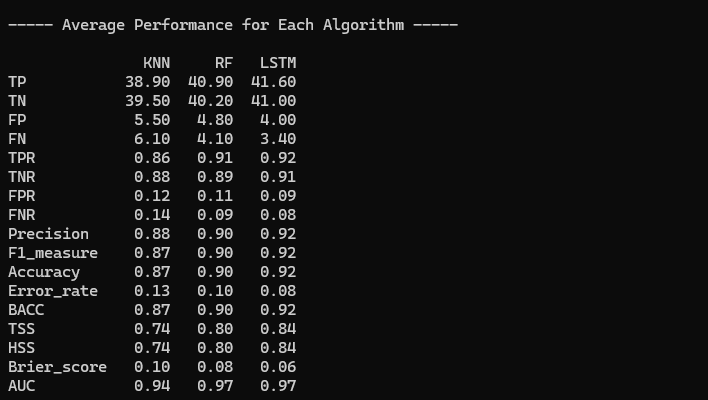
**2.7. Output:**



****

****

****

****

**Which Algorithm Performs Better and Why?**

Long Short-Term Memory (LSTM) performs better than KNN and Random Forest across the following key performance indicators:

* **Higher Accuracy and Lower Error Rate:**
* Accuracy: LSTM achieves the highest accuracy (0.92) compared to RF (0.90) and KNN (0.87).
* Error Rate: LSTM has the lowest error rate (0.08), indicating fewer misclassifications.
* **Better Precision and Recall:**
* Precision: LSTM has the highest precision (0.92), meaning it has the best performance in correctly identifying positive cases.
* Recall (TPR): LSTM also has the highest true positive rate (0.92), indicating it is more effective at identifying actual positive cases.
* **Lower False Positive and False Negative Rates:**
* FPR (False Positive Rate): LSTM has the lowest FPR (0.09), which means it has the fewest incorrect positive predictions.
* FNR (False Negative Rate): LSTM has the lowest FNR (0.08), indicating fewer missed positive cases.
* **Superior Balance and Discrimination:**
* Balanced Accuracy (BACC): LSTM achieves the highest BACC (0.92), showing a good balance between sensitivity and specificity.
* AUC (Area Under the Curve): LSTM has the highest AUC (0.97), indicating excellent discrimination between positive and negative classes.
* **Better Calibration of Predicted Probabilities:**
* Brier Score: LSTM has the lowest Brier score (0.06), indicating the best calibration and confidence in its predictions.
* **Robust Skill Scores:**
* True Skill Statistic (TSS): LSTM has the highest TSS (0.84), reflecting its ability to distinguish between positive and negative cases better.
* Heidke Skill Score (HSS): LSTM also has the highest HSS (0.84), further indicating its overall superior predictive performance.

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------