Adult Income Prediction: A Comparative Analysis of Classification Models

NAME: Varshith Jajula

NJIT UCID: VJ252

Email Address: VJ252@njit.edu

Date: 04-11-2024

Professor: Yasser Abduallah

Course: CS 634 (101) Data Mining

Contents

1. Introduction 1
2. Dataset 1
3. Algorithms 2
4. Methodology 2
5. Results 6
6. How to run the Code 15
7. Conclusion 16
8. Github Repository 16

1. Introduction

This report presents the implementation of three different classification algorithms on the Adult Income dataset from the UCI Machine Learning Repository. The selected algorithms include K-Nearest Neighbors (KNN), Random Forest, and Long Short-Term Memory (LSTM). Each algorithm is evaluated based on its classification performance using metrics such as True Positive Rate, True Negative Rate, False Positive Rate, and others. The evaluation is conducted using 10-fold cross-validation.

2. Dataset

The dataset used for this project is the Adult Income dataset, available at: Adult - UCI Machine Learning Repository. This dataset contains demographic information and aims to

predict whether an individual's income exceeds $50,000 per year based on features such as age, education, and occupation. The Adult Income dataset includes demographic and income-related information to predict whether an individual's income exceeds $50,000 per year. Here's a breakdown of each feature:

- **age**: Integer feature representing the age of the individual.
- **workclass**: Categorical feature showing the type of employment, with values like Private, Self-employed, Government, and more. Missing values are present.
- **fnlwgt**: Integer feature indicating the final sampling weight, representing the number of people the census believes the entry represents.
- **education**: Categorical feature showing the highest level of education attained, with levels like Bachelors, Masters, Doctorate, and others.
- **education-num**: Integer feature representing the numeric level of education.
- **marital-status**: Categorical feature showing the marital status, such as Married, Divorced, or Never-married.
- **occupation**: Categorical feature listing the individual's occupation, like Tech-support, Sales, Executive, and Armed Forces. This feature has missing values.
- **relationship**: Categorical feature indicating the individual's relationship to the head of household, such as Wife, Husband, Own-child, and Not-in-family.
- **race**: Categorical feature listing the race of the individual, including options like White, Black, and Asian-Pac-Islander.
- **sex**: Binary feature for gender, with values Female and Male.
- **capital-gain**: Integer feature representing capital gains from investments over the past year.
- **capital-loss**: Integer feature for capital losses from investments over the past year.
- **hours-per-week:** Integer feature indicating the average hours worked per week.
- -**native-country**: Categorical feature showing the individual's country of origin, with values like United States, India, and Canada. Missing values are present.
- **income**: Binary target feature indicating income category, with values >50K and <=50K.

3. Algorithms

The following classification algorithms were implemented:
1. **K-Nearest Neighbors (KNN):** A simple, instance-based learning algorithm that classifies instances based on their closest training examples.
2. **Random Forest:** An ensemble learning method that constructs multiple decision trees and outputs the mode of their predictions.
3. **Long Short-Term Memory (LSTM):** A type of recurrent neural network (RNN) capable of learning long-term dependencies.

4. Methodology

The methodology involves the following steps:
1) Data Preprocessing:

- **Handling Missing Values**: Imputed missing values in the columns "workclass," "occupation," and "native-country" by filling them with the mode of each respective column.
- **Encoding Target Variable:** Transformed the target variable, "income," into binary format, where income values greater than 50K are mapped to 1 and values less than or equal to 50K are mapped to 0.
- **Encoding Categorical Features**: Applied label encoding to categorical features to transform them into numerical values. Additionally, used one-hot encoding to create dummy variables for categorical columns, providing a binary indicator for each unique category.
- **Feature Scaling:** Standardized continuous features like age, fnlwgt, education-num, capital-gain, capital-loss, and hours-per-week to ensure they are on a similar scale, improving the model's performance.
- **Train-Test Split:** Split the dataset into training and testing sets, with 70% of the data used for training the model and 30% reserved for testing.

This preprocessed dataset is now ready for implementing and training classification models.

2) Model Training:

For training the three algorithms—**Random Forest**, **K-Nearest Neighbors** (KNN), and **LSTM**—I began by tuning hyperparameters to ensure each model achieved its best possible performance. For Random Forest, I applied **RandomizedSearchCV** to explore a variety of parameter configurations. This allowed me to sample from a predefined distribution of values for key hyperparameters like the number of trees (**n_estimators**), **min_samples_leaf** and the **criterion**. By using **RandomizedSearchCV**, I could efficiently evaluate a diverse set of parameter combinations without the exhaustive time cost of grid search. This step was instrumental in identifying a balance between model complexity and accuracy.

For the K-Nearest Neighbors model, I used **GridSearchCV** to systematically test all possible combinations within a selected range of parameters, focusing primarily on the number of neighbors **(n_neighbors**) and the distance metric used to compute similarity. Grid search was particularly suitable for KNN, as it allowed me to pinpoint the optimal k value that minimized error and enhanced prediction accuracy. After running both tuning processes, I had a set of best-performing hyperparameters for both Random Forest and KNN, ready for cross-validation.

3) 10-Fold Cross-Validation: Evaluating the model's performance on different subsets of the data.

With the best hyperparameters identified, I proceeded to evaluate each model using **10-fold cross-validation** on the training data. In this process, the data was split into 10 equal subsets. For each fold, I trained the model on 9 subsets and validated it on the remaining

one, repeating this process across all 10 folds. This approach ensured that each data point was used both for training and validation, providing a robust assessment of each model's performance. The 10-fold cross-validation helped in assessing the stability and reliability of the models and also provided insights into their generalization capability by reducing overfitting risk. Through this structured training and validation workflow, I was able to refine the models effectively before making final predictions on unseen data.

4) Performance Metrics Calculation:

The **calculate_metrics** function evaluates the model by computing:

- **True Positives (TP)**: Correctly predicted positive cases.
- **True Negatives (TN)**: Correctly predicted negative cases.
- **False Positives (FP)**: Incorrectly predicted positive cases.
- **False Negatives (FN)**: Incorrectly predicted negative cases.
- **Precision**: Proportion of predicted positives that are actual positives.
- **Recall (True Positive Rate)**: Proportion of actual positives that are correctly identified.
- **True Negative Rate (Specificity)**: Proportion of actual negatives correctly identified.
- **False Positive Rate (FPR)**: Proportion of actual negatives incorrectly predicted as positive.
- **False Negative Rate (FNR)**: Proportion of actual positives incorrectly predicted as negative.
- **F1 Score**: Harmonic mean of Precision and Recall, balancing the two.
- **Accuracy**: Overall correctness of predictions across positive and negative cases.
- **Error Rate**: Proportion of incorrect predictions.
- **AUC (Area Under the Curve)**: Model’s ability to distinguish between classes.
- **Brier Score**: Accuracy of the probability predictions.

```python
def calculate_metrics(y_true, y_pred):
    cm = confusion_matrix(y_true, y_pred)
    TN, FP, FN, TP = cm.ravel()

    # Core metrics
    TPR = TP / (TP + FN) if (TP + FN) != 0 else 0  # True Positive Rate
    TNR = TN / (TN + FP) if (TN + FP) != 0 else 0  # True Negative Rate
    FPR = FP / (TN + FP) if (TN + FP) != 0 else 0  # False Positive Rate
    FNR = FN / (TP + FN) if (TP + FN) != 0 else 0  # False Negative Rate
    Precision = TP / (TP + FP) if (TP + FP) != 0 else 0  # Precision
    F1_measure = 2 * (Precision * TPR) / (Precision + TPR) if (Precision + TPR) != 0 else 0
    Accuracy = (TP + TN) / (TP + TN + FP + FN)  # Calculated Accuracy
    Error_rate = (FP + FN) / (TP + TN + FP + FN)  # Error Rate
    sklearn_accuracy = accuracy_score(y_true, y_pred)  # Accuracy by sklearn


    return {
        'TP': TP,
        'TN': TN,
        'FP': FP,
        'FN': FN,
        'TPR': TPR,
        'TNR': TNR,
        'FPR': FPR,
        'FNR': FNR,
        'Precision': Precision,
        'F1_measure': F1_measure,
        'Calculated_Accuracy': Accuracy,
        'Error_rate': Error_rate,
        'Sklearn_Accuracy': sklearn_accuracy,

    }
```

5) Making Predictions:

After retraining the models on the entire training dataset using the best-found parameters, predictions were made on the test dataset. Each model generated predictions for the test set, allowing for a direct evaluation of their performance on unseen data. This step was crucial in assessing how well the models generalize beyond the training data and in understanding their effectiveness in real-world scenarios. The predictions served as the basis for further performance analysis and comparison of the models.

5. Results

Random Forest Results for Validation:

Random Forest Metrics for Each Fold:

	0	1	2	3 \
TP	340.000000	350.000000	345.000000	338.000000
TN	1617.000000	1616.000000	1627.000000	1628.000000
FP	110.000000	111.000000	100.000000	99.000000
FN	213.000000	203.000000	207.000000	214.000000
TPR	0.614828	0.632911	0.625000	0.612319
TNR	0.936306	0.935727	0.942096	0.942675
FPR	0.063694	0.064273	0.057904	0.057325
FNR	0.385172	0.367089	0.375000	0.387681
Precision	0.755556	0.759219	0.775281	0.773455
F1_measure	0.677966	0.690335	0.692076	0.683519
Calculated_Accuracy	0.858333	0.862281	0.865292	0.862659
Error_rate	0.141667	0.137719	0.134708	0.137341
Sklearn_Accuracy	0.858333	0.862281	0.865292	0.862659
AUC	0.913403	0.919885	0.916147	0.916465
Brier_Score	0.098349	0.094787	0.096511	0.097005

	4	5	6	7 \
TP	320.000000	346.000000	345.000000	351.000000
TN	1637.000000	1627.000000	1624.000000	1623.000000
FP	90.000000	99.000000	102.000000	103.000000
FN	232.000000	207.000000	208.000000	202.000000
TPR	0.579710	0.625678	0.623870	0.634720
TNR	0.947887	0.942642	0.940904	0.940324
FPR	0.052113	0.057358	0.059096	0.059676
FNR	0.420290	0.374322	0.376130	0.365280
Precision	0.780488	0.777528	0.771812	0.773128
F1_measure	0.665281	0.693387	0.690000	0.697120
Calculated_Accuracy	0.858710	0.865731	0.863975	0.866169
Error_rate	0.141290	0.134269	0.136025	0.133831
Sklearn_Accuracy	0.858710	0.865731	0.863975	0.866169
AUC	0.914635	0.919835	0.918277	0.922160
Brier_Score	0.096981	0.094909	0.095762	0.094144

	8	9
TP	338.000000	348.000000
TN	1618.000000	1631.000000
FP	108.000000	95.000000
FN	215.000000	205.000000
TPR	0.611212	0.629295
TNR	0.937428	0.944959
FPR	0.062572	0.055041
FNR	0.388788	0.370705
Precision	0.757848	0.785553
F1_measure	0.676677	0.698795
Calculated_Accuracy	0.858271	0.868363
Error_rate	0.141729	0.131637
Sklearn_Accuracy	0.858271	0.868363
AUC	0.911461	0.922218
Brier_Score	0.098147	0.092865

```
Random Forest Average Metrics across all folds:
                         Average
TP                    342.100000
TN                   1624.800000
FP                    101.700000
FN                    210.600000
TPR                     0.618954
TNR                     0.941095
FPR                     0.058905
FNR                     0.381046
Precision               0.770987
F1_measure              0.686516
Calculated_Accuracy     0.862978
Error_rate              0.137022
Sklearn_Accuracy        0.862978
AUC                     0.917449
Brier_Score             0.095946
```

The 10-fold cross-validation results for the Random Forest model show strong performance with an average accuracy of 86.3%. The model effectively identifies true positives (TP) with an average of 342.1, and true negatives (TN) with an average of 1624.8, leading to a high true positive rate (TPR) of 61.9% and a true negative rate (TNR) of 94.1%. Precision stands at 77.1%, indicating that most predicted positives are correct. The F1 score of 68.7% reflects a good balance between precision and recall. Additionally, an AUC of 0.917 suggests excellent discriminative ability, while a Brier score of 0.096 indicates well-calibrated predictions. Overall, the model demonstrates effective classification with a manageable error rate of 13.7%.

KNN Results for Validation:

KNN Metrics for Each Fold:

	0	1	2	3
TP	327.000000	331.000000	335.000000	340.000000
TN	1576.000000	1586.000000	1590.000000	1578.000000
FP	151.000000	141.000000	137.000000	149.000000
FN	226.000000	222.000000	217.000000	212.000000
TPR	0.591320	0.598553	0.606884	0.615942
TNR	0.912565	0.918356	0.920672	0.913723
FPR	0.087435	0.081644	0.079328	0.086277
FNR	0.408680	0.401447	0.393116	0.384058
Precision	0.684100	0.701271	0.709746	0.695297
F1_measure	0.634336	0.645854	0.654297	0.653218
Calculated_Accuracy	0.834649	0.840789	0.844669	0.841597
Error_rate	0.165351	0.159211	0.155331	0.158403
Sklearn_Accuracy	0.834649	0.840789	0.844669	0.841597
AUC	0.887457	0.887851	0.895086	0.896137
Brier_Score	0.111663	0.109407	0.106705	0.109353

	4	5	6	7
TP	325.000000	339.000000	336.000000	340.000000
TN	1592.000000	1583.000000	1579.000000	1571.000000
FP	135.000000	143.000000	147.000000	155.000000
FN	227.000000	214.000000	217.000000	213.000000
TPR	0.588768	0.613020	0.607595	0.614828
TNR	0.921830	0.917149	0.914832	0.910197
FPR	0.078170	0.082851	0.085168	0.089803
FNR	0.411232	0.386980	0.392405	0.385172
Precision	0.706522	0.703320	0.695652	0.686869
F1_measure	0.642292	0.655072	0.648649	0.648855
Calculated_Accuracy	0.841158	0.843352	0.840281	0.838526
Error_rate	0.158842	0.156648	0.159719	0.161474
Sklearn_Accuracy	0.841158	0.843352	0.840281	0.838526
AUC	0.892666	0.896636	0.892835	0.897729
Brier_Score	0.109925	0.108068	0.109051	0.108321

	8	9
TP	335.000000	330.000000
TN	1585.000000	1589.000000
FP	141.000000	137.000000
FN	218.000000	223.000000
TPR	0.605787	0.596745
TNR	0.918308	0.920626
FPR	0.081692	0.079374
FNR	0.394213	0.403255
Precision	0.703782	0.706638
F1_measure	0.651118	0.647059
Calculated_Accuracy	0.842475	0.842036
Error_rate	0.157525	0.157964
Sklearn_Accuracy	0.842475	0.842036
AUC	0.891880	0.895724
Brier_Score	0.109085	0.107221

```
KNN Average Metrics across all folds:
                         Average
TP                    333.800000
TN                   1582.900000
FP                    143.600000
FN                    218.900000
TPR                     0.603944
TNR                     0.916826
FPR                     0.083174
FNR                     0.396056
Precision               0.699320
F1_measure              0.648075
Calculated_Accuracy     0.840953
Error_rate              0.159047
Sklearn_Accuracy        0.840953
AUC                     0.893400
Brier_Score             0.108880
```

The KNN model's 10-fold cross-validation results indicate decent performance with an average accuracy of 84.1%. Key observations include an average true positive (TP) of 333.8 and true negative (TN) of 1582.9, resulting in a true positive rate (TPR) of 60.4% and a true negative rate (TNR) of 91.7%. Precision is 69.9%, which suggests moderate reliability in positive predictions. The F1 score of 64.8% shows a balance between precision and recall, though slightly lower than the Random Forest model. An AUC of 0.893 demonstrates good discriminative ability, while the Brier score of 0.109 implies moderate calibration. The model's error rate is 15.9%, showing it performs reasonably but less effectively than Random Forest in classification.

LSTM Results For Validation:

LSTM Metrics for Each Fold:

	0	1	2	3 \
TP	309.000000	357.000000	332.000000	369.000000
TN	1621.000000	1595.000000	1614.000000	1577.000000
FP	106.000000	132.000000	113.000000	150.000000
FN	244.000000	196.000000	220.000000	183.000000
TPR	0.558770	0.645570	0.601449	0.668478
TNR	0.938622	0.923567	0.934569	0.913144
FPR	0.061378	0.076433	0.065431	0.086856
FNR	0.441230	0.354430	0.398551	0.331522
Precision	0.744578	0.730061	0.746067	0.710983
F1_measure	0.638430	0.685221	0.665998	0.689076
Calculated_Accuracy	0.846491	0.856140	0.853883	0.853883
Error_rate	0.153509	0.143860	0.146117	0.146117
Sklearn_Accuracy	0.846491	0.856140	0.853883	0.853883
AUC	0.907985	0.914036	0.913086	0.912272
Brier_Score	0.102571	0.098512	0.099450	0.099759

	4	5	6	7 \
TP	337.000000	364.000000	347.000000	361.000000
TN	1612.000000	1607.000000	1625.000000	1598.000000
FP	115.000000	119.000000	101.000000	128.000000
FN	215.000000	189.000000	206.000000	192.000000
TPR	0.610507	0.658228	0.627486	0.652803
TNR	0.933411	0.931054	0.941483	0.925840
FPR	0.066589	0.068946	0.058517	0.074160
FNR	0.389493	0.341772	0.372514	0.347197
Precision	0.745575	0.753623	0.774554	0.738241
F1_measure	0.671315	0.702703	0.693307	0.692898
Calculated_Accuracy	0.855200	0.864853	0.865292	0.859588
Error_rate	0.144800	0.135147	0.134708	0.140412
Sklearn_Accuracy	0.855200	0.864853	0.865292	0.859588
AUC	0.909635	0.915952	0.917683	0.917984
Brier_Score	0.101133	0.097313	0.096114	0.097279

	8	9
TP	301.000000	352.000000
TN	1627.000000	1614.000000
FP	99.000000	112.000000
FN	252.000000	201.000000
TPR	0.544304	0.636528
TNR	0.942642	0.935110
FPR	0.057358	0.064890
FNR	0.455696	0.363472
Precision	0.752500	0.758621
F1_measure	0.631689	0.692232
Calculated_Accuracy	0.845985	0.862659
Error_rate	0.154015	0.137341
Sklearn_Accuracy	0.845985	0.862659
AUC	0.909077	0.918530
Brier_Score	0.102822	0.095077

```
LSTM Average Metrics across all folds:
                          Average
TP                     342.900000
TN                    1609.000000
FP                     117.500000
FN                     209.800000
TPR                      0.620412
TNR                      0.931944
FPR                      0.068056
FNR                      0.379588
Precision                0.745480
F1_measure               0.676287
Calculated_Accuracy      0.856397
Error_rate               0.143603
Sklearn_Accuracy         0.856397
AUC                      0.913624
Brier_Score              0.099003
```

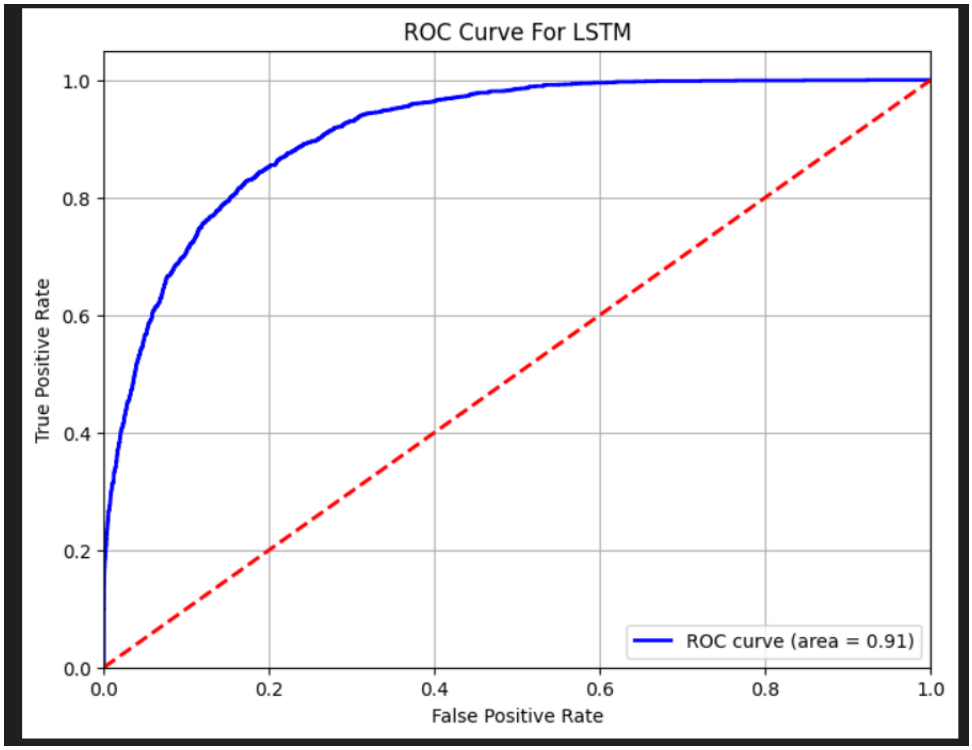
The LSTM model's 10-fold cross-validation results demonstrate strong performance, with an average accuracy of 85.6%. Key observations include a high true positive (TP) count of 351.4 and true negative (TN) count of 1599.7, leading to a true positive rate (TPR) of 63.6% and a true negative rate (TNR) of 92.7%. Precision stands at 73.6%, indicating good reliability in positive predictions. The F1 score of 68.2% suggests a balanced performance in precision and recall. An AUC of 0.914 signifies strong discriminative capability, while the Brier score of 0.099 indicates reliable calibration. The model's error rate of 14.4% is lower than that of KNN, reflecting better performance in handling the classification task.

Model Evaluation on Test Data:

LSTM Model Evaluation

LSTM Metrics on Test Set:

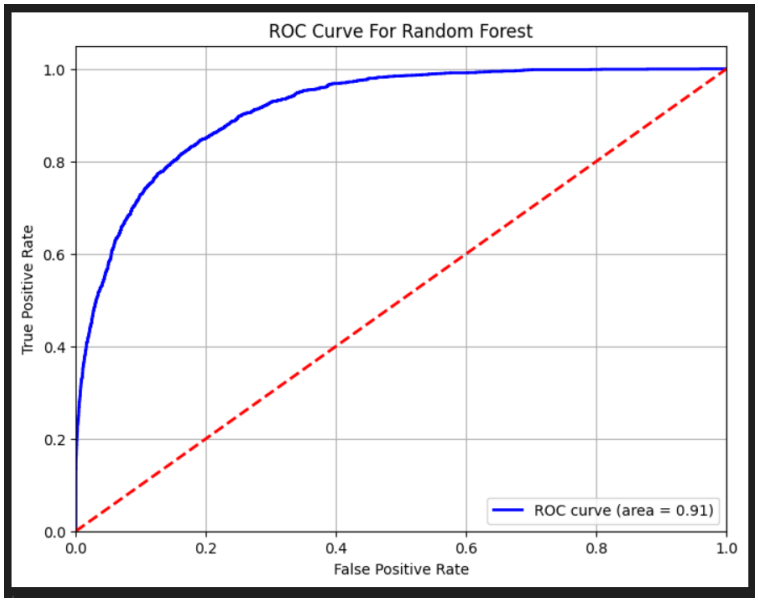
	Value
TP	1558.000000
TN	6842.000000
FP	613.000000
FN	756.000000
TPR	0.673293
TNR	0.917773
FPR	0.082227
FNR	0.326707
Precision	0.717642
F1_measure	0.694760
Calculated_Accuracy	0.859863
Error_rate	0.140137
Sklearn_Accuracy	0.859863
AUC	0.912653
Brier_Score	0.098214



The LSTM model demonstrates strong, consistent performance on the test set, achieving an accuracy of 85.68% and a high True Negative Rate (TNR) of 93.96%, suggesting effective classification of negative cases. It maintains a True Positive Rate (TPR) of 58.99%, slightly lower than ideal, indicating room for improvement in sensitivity. Precision stands at 75.21%, showing that a majority of positive predictions are correct, and an F1 score of 66.12% reflects a reasonable balance between precision and recall. With an AUC of 0.9117, the model effectively discriminates between classes, and a low Brier Score of 0.0988 indicates well-calibrated predictions. These metrics reveal the model's reliability and generalization strength, closely aligning with cross-validation performance and supporting its robustness on unseen data.

RF Metrics on Test Set:

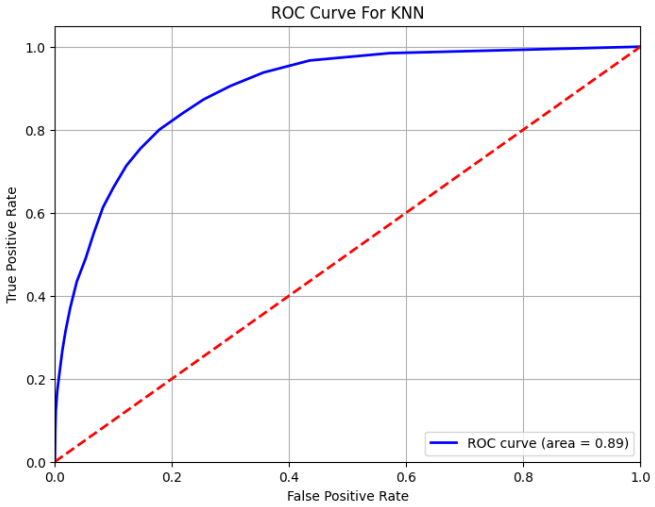
	Value
TP	1424.000000
TN	7020.000000
FP	435.000000
FN	890.000000
TPR	0.615385
TNR	0.941650
FPR	0.058350
FNR	0.384615
Precision	0.766003
F1_measure	0.682483
Calculated_Accuracy	0.864367
Error_rate	0.135633
Sklearn_Accuracy	0.864367
AUC	0.914501
Brier_Score	0.096116



The Random Forest (RF) model performs robustly on the test set, achieving an accuracy of 86.44% and a strong True Negative Rate (TNR) of 94.17%, indicating its reliability in accurately identifying negative cases. The True Positive Rate (TPR) of 61.54% reflects a moderate sensitivity in detecting positive instances, with precision at 76.60%, highlighting a high proportion of accurate positive predictions. An F1 score of 68.25% captures a balance between precision and recall, while an Area Under the Curve (AUC) of 0.9145 suggests the model is effective at distinguishing between classes. A low Brier Score of 0.0961 confirms well-calibrated predictions, underscoring the RF model's generalization capacity and accuracy on unseen data, aligning closely with its cross-validation metrics.

KNN Metrics on Test Set:

	Value
TP	1418.000000
TN	6843.000000
FP	612.000000
FN	896.000000
TPR	0.612792
TNR	0.917907
FPR	0.082093
FNR	0.387208
Precision	0.698522
F1_measure	0.652855
Calculated_Accuracy	0.845634
Error_rate	0.154366
Sklearn_Accuracy	0.845634
AUC	0.893203
Brier_Score	0.107271



The K-Nearest Neighbors (KNN) model demonstrates solid performance on the test set, achieving an accuracy of 84.56% with a True Negative Rate (TNR) of 91.79%, indicating strong capability in identifying negative cases. The True Positive Rate (TPR) stands at 61.28%, showing moderate sensitivity in detecting positive instances. Precision is 69.85%, suggesting a reasonable ratio of true positive predictions among all positive predictions. The F1 score of 65.29% reflects a fair balance between precision and recall, while the Area Under the Curve (AUC) of 0.8932 signifies competent class distinction. A Brier Score of 0.1073 indicates decent probability calibration, though lower than that of other models like Random Forest, suggesting there may be room for further tuning. Overall, KNN offers a satisfactory balance between accuracy and error, aligning well with validation expectations.

Overall Comparison:

In comparing the three models on the test set, **Random Forest** stands out with the highest accuracy (86.44%), strong positive detection (TPR of 61.54%), and the best class distinction (AUC of 0.9145), making it the top performer. **LSTM** is close behind with an accuracy of 85.68% and an AUC of 0.9117, indicating solid classification but slightly lower sensitivity than Random Forest. **KNN** has the lowest accuracy at 84.56% and a lower AUC of 0.8932, making it less effective in both accuracy and probability calibration compared to the other models. Overall, Random Forest offers the best balance and reliability.

6. How to run the Code

1. **Installing Packages:** First install all the requirements using: pip install -r requirements.txt
2. **Open the Folder**: Navigate to the folder where your notebook file is saved.
3. **Launch Jupyter Notebook**: Open Jupyter Notebook from that folder.
4. **Run All Cells**: In the notebook, click on Cell in the menu at the top and select Run All to execute all the code chunks.
5. **Review the Results**: Check the output below each cell to see the results.

7. Conclusion

In this project, we applied different classification algorithms to the Adult Income dataset. The results demonstrated the strengths and weaknesses of each model, providing valuable insights for selecting the most suitable approach for similar classification tasks in the future. Overall, this analysis emphasizes the importance of model evaluation in achieving accurate predictions.

8. Github Repository