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

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):** Negative cases incorrectly predicted as positive.
- **False Negatives (FN):** Positive cases incorrectly predicted as negative.
- **Precision:** Measures how many of the predicted positive cases are actually positive.
- **Recall (True Positive Rate):** Measures how well the model identifies actual positive cases.
- **True Negative Rate (Specificity):** Measures how well the model identifies actual negative cases.
- **False Positive Rate (FPR):** Measures the proportion of actual negatives incorrectly predicted as positive.
- **False Negative Rate (FNR)**: Measures the proportion of actual positives incorrectly predicted as negative.
- **Balanced Accuracy (BACC):** Provides a balanced view of accuracy across classes, especially useful for imbalanced datasets.
- **True Skill Statistic (TSS):** Assesses the model's ability to distinguish between classes.
- **Heidke Skill Score (HSS):** Compares model performance to random chance.
- **F1 Score:** Balances Precision and Recall, offering a combined view of the two metrics.
- **Accuracy:** Overall correctness of the model's predictions across both positive and negative cases.
- **Error Rate**: Proportion of incorrect predictions made by the model.
- **Sklearn Accuracy**: Standard accuracy measure calculated using sklearn's function.
- **AUC (Area Under the Curve**): Reflects the model's ability to distinguish between classes, with higher values indicating better performance.

- **Brier Score:** Evaluates the accuracy of probability predictions, with lower values indicating more accurate probability estimates.

```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 (Recall)
    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
    BACC = (TPR + TNR) / 2  # Balanced Accuracy
    TSS = TPR - FPR  # True Skill Statistic
    HSS = 2 * (TP * TN - FP * FN) / ((TP + FN) * (FN + TN) + (TP + FP) * (FP + TN)) if ((TP + FN) * (FN + TN) + (TP + FP) * (FP + TN)) != 0 else 0  # Heidke Skill Score
    Precision = TP / (TP + FP) if (TP + FP) != 0 else 0  # Precision
    F1_measure = 2 * (Precision * TPR) / (Precision + TPR) if (Precision + TPR) != 0 else 0  # F1 Score
    Accuracy = (TP + TN) / (TP + TN + FP + FN) if (TP + TN + FP + FN) != 0 else 0  # Calculated Accuracy
    Error_rate = (FP + FN) / (TP + TN + FP + FN) if (TP + TN + FP + FN) != 0 else 0  # 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,
        'BACC': BACC,
        'TSS': TSS,
        'HSS': HSS,
        '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	355.000000	339.000000	335.000000
TN	1615.000000	1624.000000	1618.000000	1628.000000
FP	112.000000	103.000000	109.000000	99.000000
FN	213.000000	198.000000	213.000000	217.000000
TPR	0.614828	0.641953	0.614130	0.606884
TNR	0.935148	0.940359	0.936885	0.942675
FPR	0.064852	0.059641	0.063115	0.057325
FNR	0.385172	0.358047	0.385870	0.393116
BACC	0.774988	0.791156	0.775508	0.774780
TSS	0.549976	0.582312	0.551015	0.549559
HSS	0.586377	0.618422	0.588750	0.592658
Precision	0.752212	0.775109	0.756696	0.771889
F1_measure	0.676617	0.702275	0.678000	0.679513
Calculated_Accuracy	0.857456	0.867982	0.858710	0.861343
Error_rate	0.142544	0.132018	0.141290	0.138657
Sklearn_Accuracy	0.857456	0.867982	0.858710	0.861343
AUC	0.912724	0.920815	0.914494	0.917812
Brier_Score	0.098682	0.094663	0.097339	0.096441

	4	5	6	7 \
TP	322.000000	349.000000	349.000000	348.000000
TN	1633.000000	1631.000000	1627.000000	1626.000000
FP	94.000000	95.000000	99.000000	100.000000
FN	230.000000	204.000000	204.000000	205.000000
TPR	0.583333	0.631103	0.631103	0.629295
TNR	0.945570	0.944959	0.942642	0.942063
FPR	0.054430	0.055041	0.057358	0.057937
FNR	0.416667	0.368897	0.368897	0.370705
BACC	0.764452	0.788031	0.786873	0.785679
TSS	0.528904	0.576063	0.573745	0.571357
HSS	0.577288	0.617415	0.613316	0.610764
Precision	0.774038	0.786036	0.779018	0.776786
F1_measure	0.665289	0.700100	0.697303	0.695305
Calculated_Accuracy	0.857832	0.868802	0.867047	0.866169
Error_rate	0.142168	0.131198	0.132953	0.133831
Sklearn_Accuracy	0.857832	0.868802	0.867047	0.866169
AUC	0.912488	0.921142	0.919609	0.922568
Brier_Score	0.098325	0.094361	0.094750	0.093966

	8	9
TP	335.000000	351.000000
TN	1625.000000	1637.000000
FP	101.000000	89.000000
FN	218.000000	202.000000
TPR	0.605787	0.634720
TNR	0.941483	0.948436
FPR	0.058517	0.051564
FNR	0.394213	0.365280
BACC	0.773635	0.791578
TSS	0.547270	0.583155

```
Random Forest Average Metrics across all folds:
                          Average
TP                     342.300000
TN                    1626.400000
FP                     100.100000
FN                     210.400000
TPR                      0.619314
TNR                      0.942022
FPR                      0.057978
FNR                      0.380686
BACC                     0.780668
TSS                      0.561336
HSS                      0.602132
Precision                0.773786
F1_measure               0.687880
Calculated_Accuracy      0.863768
Error_rate               0.136232
Sklearn_Accuracy         0.863768
AUC                      0.917564
Brier_Score              0.095974
```

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
BACC                   0.751943     0.758454     0.763778     0.764833
TSS                    0.503885     0.516909     0.527556     0.529665
HSS                    0.528236     0.543992     0.554914     0.551060
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
BACC                   0.755299     0.765085     0.761213     0.762513
TSS                    0.510598     0.530169     0.522427     0.525025
HSS                    0.541288     0.554354     0.545908     0.544428
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
BACC                   0.762047     0.758685
TSS                    0.524095     0.517371
```

```
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
BACC                     0.760385
TSS                      0.520770
HSS                      0.546054
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	353.000000	387.000000	391.000000	361.000000
TN	1589.000000	1576.000000	1563.000000	1588.000000
FP	138.000000	151.000000	164.000000	139.000000
FN	200.000000	166.000000	161.000000	191.000000
TPR	0.638336	0.699819	0.708333	0.653986
TNR	0.920093	0.912565	0.905038	0.919514
FPR	0.079907	0.087435	0.094962	0.080486
FNR	0.361664	0.300181	0.291667	0.346014
BACC	0.779214	0.806192	0.806685	0.786750
TSS	0.558429	0.612384	0.613371	0.573499
HSS	0.580552	0.618083	0.612239	0.592486
Precision	0.718941	0.719331	0.704505	0.722000
F1_measure	0.676245	0.709441	0.706414	0.686312
Calculated_Accuracy	0.851754	0.860965	0.857394	0.855200
Error_rate	0.148246	0.139035	0.142606	0.144800
Sklearn_Accuracy	0.851754	0.860965	0.857394	0.855200
AUC	0.909867	0.914978	0.912755	0.913616
Brier_Score	0.101219	0.098385	0.100304	0.099209

	4	5	6	7 \
TP	323.000000	361.000000	365.000000	350.000000
TN	1612.000000	1610.000000	1588.000000	1596.000000
FP	115.000000	116.000000	138.000000	130.000000
FN	229.000000	192.000000	188.000000	203.000000
TPR	0.585145	0.652803	0.660036	0.632911
TNR	0.933411	0.932793	0.920046	0.924681
FPR	0.066589	0.067207	0.079954	0.075319
FNR	0.414855	0.347197	0.339964	0.367089
BACC	0.759278	0.792798	0.790041	0.778796
TSS	0.518555	0.585595	0.580083	0.557593
HSS	0.557740	0.614282	0.598470	0.583779
Precision	0.737443	0.756813	0.725646	0.729167
F1_measure	0.652525	0.700971	0.691288	0.677638
Calculated_Accuracy	0.849057	0.864853	0.856955	0.853883
Error_rate	0.150943	0.135147	0.143045	0.146117
Sklearn_Accuracy	0.849057	0.864853	0.856955	0.853883
AUC	0.909688	0.915838	0.914539	0.918504
Brier_Score	0.101344	0.097718	0.098070	0.097249

	8	9
TP	375.000000	366.000000
TN	1554.000000	1601.000000
FP	172.000000	125.000000
FN	178.000000	187.000000
TPR	0.678119	0.661844
TNR	0.900348	0.927578
FPR	0.099652	0.072422
FNR	0.321881	0.338156
BACC	0.789233	0.794711
TSS	0.578467	0.589423

LSTM Average Metrics across all folds:

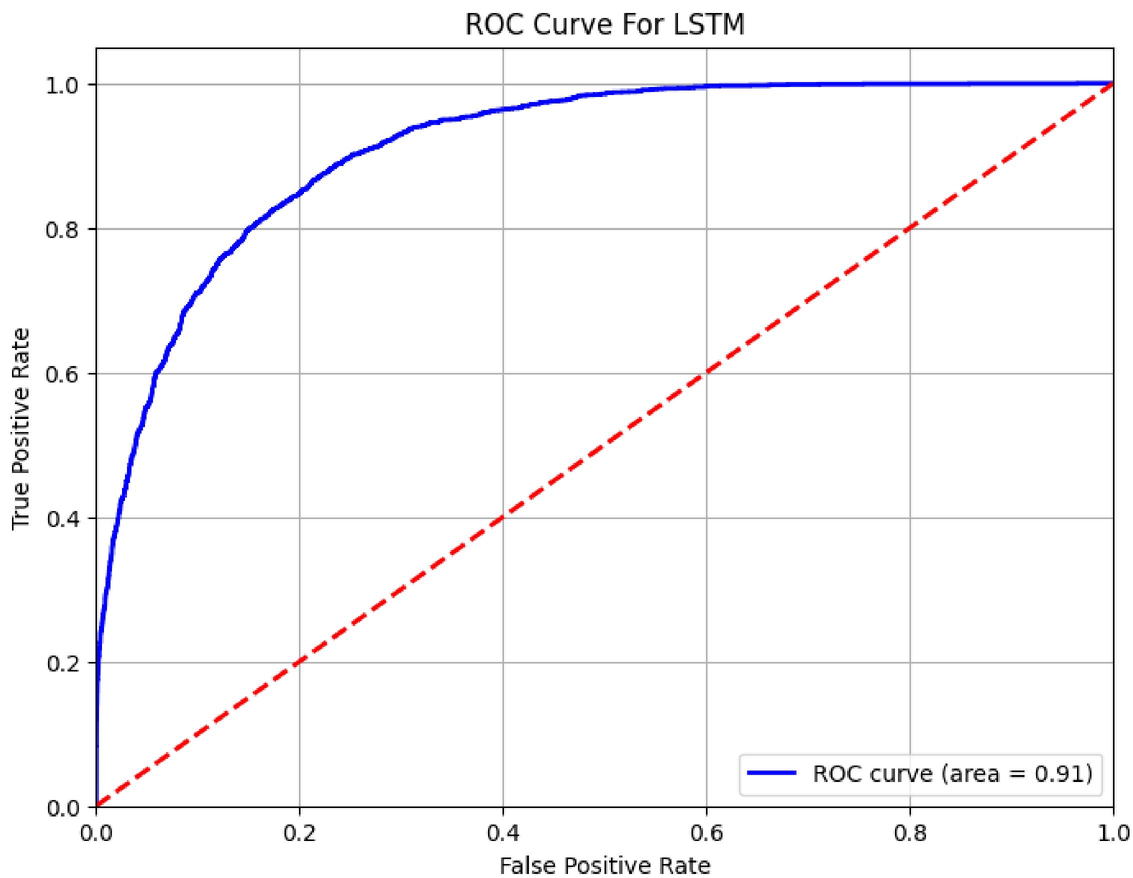
	Average
TP	363.200000
TN	1587.700000
FP	138.800000
FN	189.500000
TPR	0.657133
TNR	0.919607
FPR	0.080393
FNR	0.342867
BACC	0.788370
TSS	0.576740
HSS	0.595101
Precision	0.724482
F1_measure	0.688380
Calculated_Accuracy	0.855958
Error_rate	0.144042
Sklearn_Accuracy	0.855958
AUC	0.913696
Brier_Score	0.099207

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

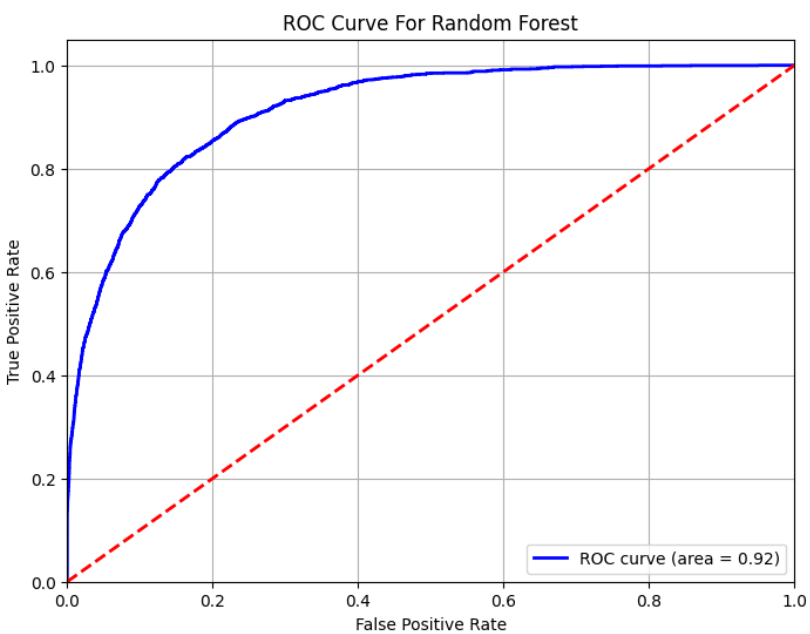
```
LSTM Metrics on Test Set:
                             Value
TP                     1412.000000
TN                     6971.000000
FP                      484.000000
FN                      902.000000
TPR                       0.610199
TNR                       0.935077
FPR                       0.064923
FNR                       0.389801
BACC                      0.772638
TSS                       0.545276
HSS                       0.581494
Precision                 0.744726
F1_measure                0.670784
Calculated_Accuracy       0.858123
Error_rate                0.141877
Sklearn_Accuracy          0.858123
AUC                       0.911497
Brier_Score               0.098472
```



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.

Random Forest on Test Set:

```
 RF Metrics on Test Set:
                            Value
TP                    1427.000000
TN                    7008.000000
FP                     447.000000
FN                     887.000000
TPR                      0.616681
TNR                      0.940040
FPR                      0.059960
FNR                      0.383319
BACC                     0.778361
TSS                      0.556721
HSS                      0.595783
Precision                0.761473
F1_measure               0.681471
Calculated_Accuracy      0.863446
Error_rate               0.136554
Sklearn_Accuracy         0.863446
AUC                      0.915046
Brier_Score              0.095945
```

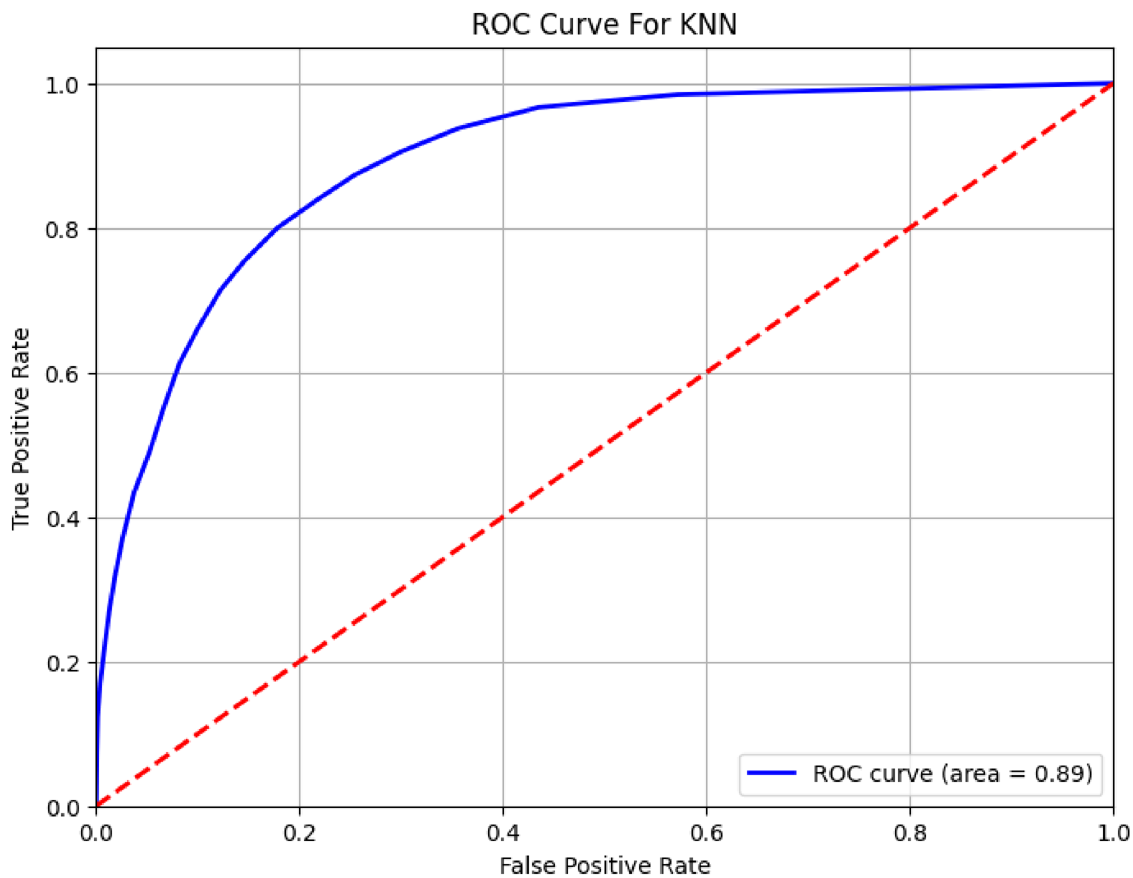


The Random Forest (RF) model performs robustly on the test set, achieving an accuracy of 86.44% and a strong True Negative Rate (TNR) of 94.17%, indicating its reliability in accurately identifying negative cases. The True Positive Rate (TPR) of 61.54% reflects a moderate sensitivity in detecting positive 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 on Test Dataset

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
BACC	0.765350
TSS	0.530699
HSS	0.554150
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

For `.ipynb` (Jupyter Notebook):

1. **Install Packages**: Run `pip install -r requirements.txt` to install all required packages.

2**. Open the Jupyter Notebook** in the project folder.

4. **Run All Cells**: In the Jupyter Notebook, go to the Cell menu and select Run All to execute all code cells.

5. **Review the Results**: Check the output under each cell for the results.

For `.py` (Python Script):

1. **Navigate to Folder**: Open project folder and open CMD from there.

2. **Run Script**: Execute the script by running `python Varshith_Jagula_FinalTermProj.py`.

3. **Review the Results:** Check the terminal output for results and messages.

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

Link : https://github.com/Varshithjajula/Data_Mining_Final