CISC-5790 DATA MINING COURSE FINAL PROJECT REPORT

Presented by:
Andres Leonardo De Los Santos
Albert Delgado Baez
Carlos Ferrer
Daniela Johnson

May 6th, 2024

TABLE OF CONTENTS

1. Introduction

In an effort to better understand economic disparities and target socio-economic interventions, our project leverages various data mining techniques to predict individual income levels from census data. The primary objective is to accurately classify individuals into two income categories: those earning above and below the $50K threshold annually.

Our analysis, based on a classification task, utilizes a diverse set of data mining algorithms, including Decision Trees, Random Forest, K-Nearest Neighbors (KNN), and Naive Bayes. Each of these methods brings a unique approach to handling the complexities of the dataset.

This report outlines the methodology used in preparing and analyzing the data, discusses the predictive models developed, and evaluates their performance. The findings aim to provide a deeper understanding of the income dynamics within the population and offer evidence-based recommendations for future initiatives aimed at economic improvement.

2. Objectives

- **Data Exploration and Preparation:** To conduct a thorough analysis of the census dataset, including data cleaning, feature selection, and preliminary data exploration to understand the distribution and relationships of variables that might influence income levels.

- **Model Development:** To implement and train various data mining algorithms—namely Decision Trees, Random Forest, K-Nearest Neighbors (KNN), and Naive Bayes—to create robust predictive models. The objective is to assess each model's capability in accurately classifying individuals based on their income levels.

- **Model Comparison and Evaluation:** To compare the performance of each algorithm using metrics such as accuracy, precision, recall, and F1-score. This will help in identifying the most effective model(s) based on the predictive accuracy and other performance indicators.

- **Feature Importance Analysis:** To analyze and interpret the contribution of each feature in the prediction models, thereby identifying key factors that most significantly impact income levels. This analysis aims to provide insights into which characteristics are most predictive of higher income levels.

3. Data Exploration and Preparation

To properly understand the existing relationships between all of our categories and values, we need to get insights and identify patterns that may enhance our data analysis. This way, our team also was able to identify early on possible anomalies and to improve the data quality.

The initial phase of our project involved an in-depth exploration of the census dataset to understand the structure, quality, and characteristics of the dataset so that we could further prepare our dataset. This section details our findings and the steps taken to prepare the data for further analysis.

The selected dataset consists of approximately 32,561 records and 14 features, covering a wide range of demographic and employment-related variables. The target variable is the income level, classified into two categories: <=50K and >50K.

Listed below are the main points we addressed when preparing our data:

1. Missing values: certain features exhibited missing values. For the features ‘workclass’, ‘occupation’ and ‘native-country’ had entries categorized as ‘?’. We addressed these by applying replacement techniques where appropriate with two approaches: using both the mode and KNN algorithm to predict these missing labels.

2. Data imbalance: during the revision of the target variable, we identified a significant class imbalance, with a major proportion of cases earning “<=50K” compared to individuals earning “>50k”. To address this imbalance, we implemented techniques such as the Synthetic Minority Over-Sampling Technique (SMOTE). By augmenting the minority class, SMOTE aimed to balance the datasets, which might reduce the risk of the models being biased towards the majority class, and overfitting. This approach also held the potential to enhance the performance and the metrics of the models implemented. On the other hand, the undersampling technique was evaluated. However, since our datasets were considered inappropriate size, we opted against undersampling, since this approach might lead to losing of essential data.

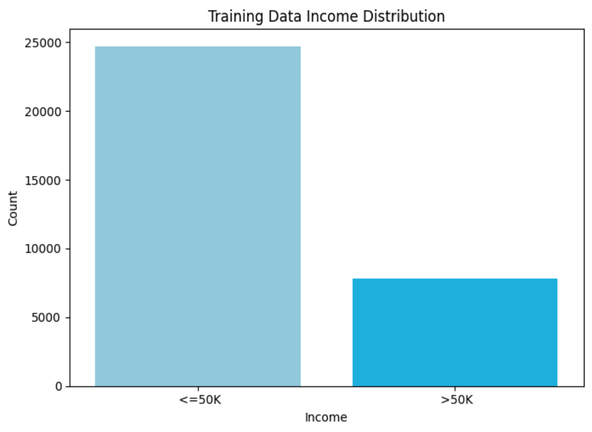
3. Data standardization: irregularities were identified in the formatting of our target variable, the income class, in both training and test datasets. The training sets were “<=50k” and “>50k”, while the test set included periods at the end, as shown “<=50k.” and “>50k”. Additionally, some labels contained spaces. After recognizing these inconsistencies, adjustments were made to standardize labels across datasets, ensuring uniformity and facilitating seamless model training and evaluation.

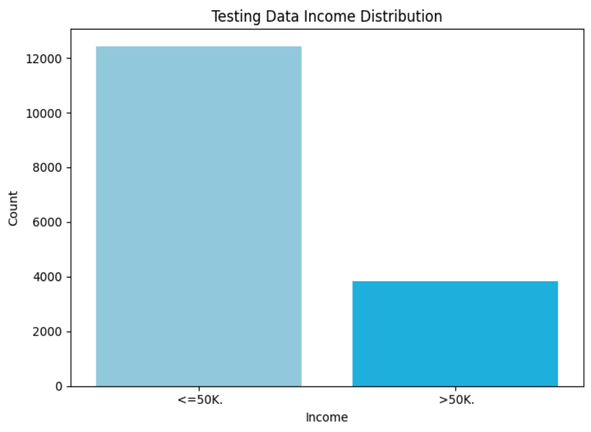
4. Feature transformation: During the evaluation phase, categorical attributes were identified within both the training and test datasets, encompassing 'work-class', 'education', 'marital-status', 'occupation', 'relationship', 'race', 'sex', 'native-country', and 'income'.

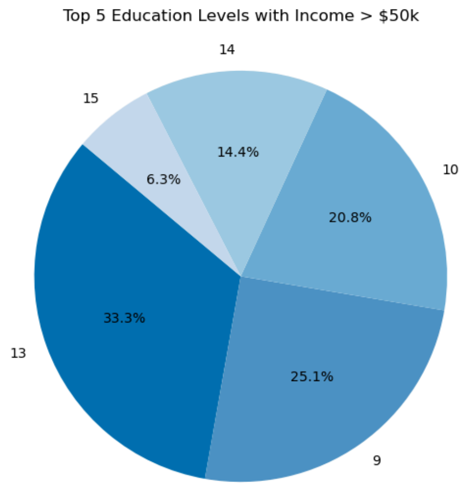
Recognizing the necessity of numerical inputs for models like KNN, a label encoding strategy was adopted to transform these categorical attributes into numerical representations. This conversion facilitated crucial distance computations essential for the operation of the KNN algorithm. Additionally, to effectively manage categorical variables within the Random Forests algorithm, a One Hot Encoding approach was implemented to manage categorical variables within the Random Forests algorithm effectively. These preprocessing techniques ensured the datasets' compatibility with the respective algorithms, enhancing the robustness and effectiveness of the modeling process. Furthermore, the entire dataset was normalized to ensure a uniform contribution of all features to distance calculations in our KNN model.

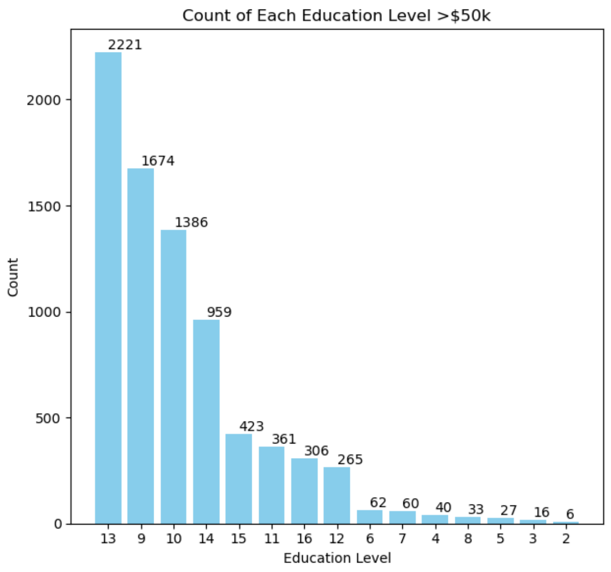
5. Duplicate Rows: during the data cleaning process, we identified 24 duplicate rows within the dataset. These duplicates were exact replicas across all features and the target variable, potentially skewing the analysis and model training with redundant information. To ensure the integrity of our predictive models, we decided to remove these duplicate entries from the dataset.

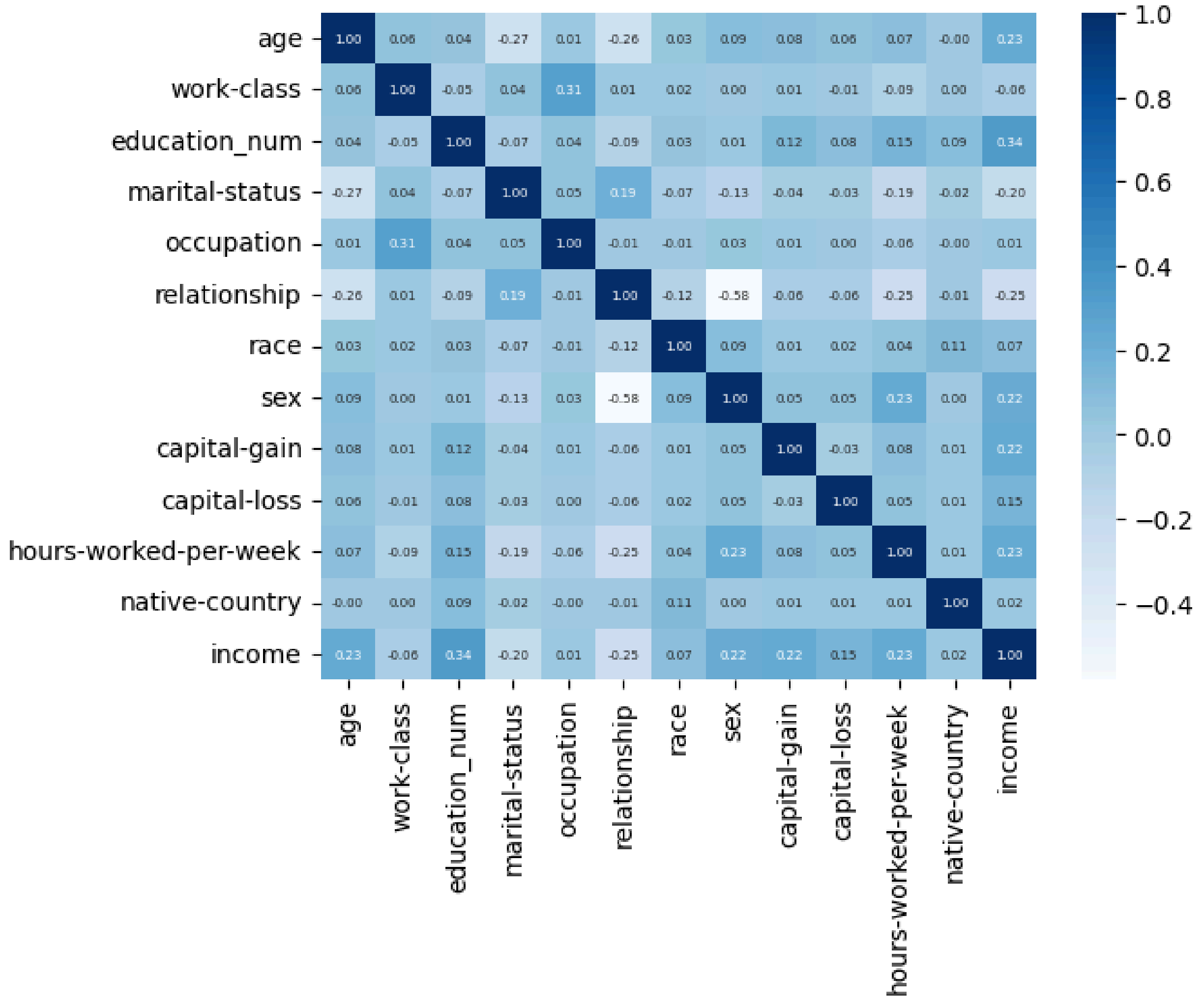
6. Visual Data Analysis: using various visualization tools, we created graphs to visually inspect the distribution and relationship of features. This not only helped in confirming the trends identified in the statistical analysis but also in spotting outliers and anomalies in the data.











4. Model Implementation & Performance Evaluation

This section outlines the implementation and insights gained from implementing various classification techniques learned in class. We tested Naive Bayes, Decision Trees, Random Forest, and K-Nearest Neighbors (K-NN), each offering unique predictive strengths, which we will see summarized below.

4.1 Naive Bayes

After preparing the dataset and handling missing values by replacing them with the mode, Our team implemented the Naive Bayes model and evaluated its accuracy on both the training and testing datasets. After the initial assessment, we introduced Laplace smoothing to the model to observe any changes in accuracy, specifically to see its impact on prediction performance when dealing with categorical variables with zero frequency in the data.

Based on the comparison between Model 1 and Model 2 , Model 1 is the preferable choice for several reasons. Firstly, Model 1 shows slightly higher overall accuracy than Model 2. Model 1 demonstrates superior performance in identifying high-income earners with better precision and F1 scores in the training and testing data. Furthermore, despite the typical use of Laplace smoothing in Model 2 to handle the zero-frequency problem, Model 1 achieves better results without this adjustment, suggesting it effectively manages the dataset as is. Hence, Model 1 is recommended for its overall higher accuracy and better performance in the predictions.

The table below shows the metrics obtained from each model:

Model 1: No Laplace Smoothing		**Model 2: Laplace Smoothing**	
Training Data: Accuracy: 80.08% Low Income: - Precision: 0.82 - Recall: 0.85 - F1-Score: 0.88 - Support: 24,698 High Income: - Precision: 0.68 - Recall: 0.32 - F1-Score: 0.44 - Support: 7,839	Test Data: Accuracy: 80.16% Low Income: - Precision: 0.82 - Recall: 0.95 - F1-Score: 0.88 - Support: 12,435 High Income: - Precision: 0.67 - Recall: 0.32 - F1-Score: 0.43 - Support: 3,846	Training Data: Accuracy: 77.91% Low Income: - Precision: 0.79 Recall: 0.96 - F1-Score: 0.87 - Support: 24,698 High Income: - Precision: 0.62 - Recall: 0.21 - F1-Score: 0.32 - Support: 7,839	Test Data: Accuracy: 78.28% Low Income: - Precision: 0.80 - Recall: 0.96 - F1-Score: 0.87 - Support: 12,435 High Income: - Precision: 0.62 - Recall: 0.21 - F1-Score: 0.32 - Support: 3,846

Note: *the values of precision, recall, and F-1 score shown in the chart above are the weighted average. Please refer to the appendix* ***6.1. "Naive Bayes model metrics"*** *to see a more descriptive analysis of each model performance.*

4.2 Random Forest

Using the Random Forests ensemble method, we used a combination of different decision trees and majority votes to properly classify each new instance of data. Random Forest is an improvement over Decision Trees as it's more resistant to overfitting since each tree is trained with a different subset of features, working well for a wide range of applications.

Our team used different approaches, first by using the RF technique with the default and the KNN completed dataset (with no missing values), implementing One Hot Encoding and Label Encoding. Our categorical columns had to be encoded since the library we were using to train the model relied only on numerical values, but we had to decide which one of the encoding techniques we were going to apply.

After deciding to use the completed dataset with One Hot Encoding (since Label Encoding may have introduced bias into our model by interpreting our not ordinal values as ordinal) we implemented data balancing through the SMOTE technique using the guidelines we learned in class because there was a notorious difference between the number of values in the target function <50K and >50K, as we mentioned in the previous Data Exploration section. After running this encoding technique over the categorical features, and converting the labels in the 'income' column to a numeric value where 0 is <50K and 1 is >50K, we were able to properly train the model with this extended dataset.

Next, we used hyperparameter tuning, to determine the best parameters for this model. The set of parameters that we tuned were: max depth, minimum samples per leaf, minimum samples to be considered for a split, number of trees to be trained (number of estimators) and bootstrapping. Once we determined the best configuration with these parameters, we could then evaluate the results and choose the best model.

We had two hyperparameter configurations: the first one, focused on accuracy that has a higher accuracy in the testing dataset indicating a possible overfit and the second one focused on a balance between training and test set accuracy, which has a lower tendency to overfit.

Model 1: Imbalanced dataset and O.H.E.		**Model 2: Balanced dataset (SMOTE) and O.H.E.**	
Training Data: Accuracy: 99.99% Low Income: - Precision: 1.00 - Recall: 1.00 - F1-Score: 1.00 - Support: 24,720 High Income: - Precision: 1.00 - Recall: 1.00 - F1-Score: 1.00 - Support: 7,841	Test Data: Accuracy: 84.87% Low Income: - Precision: 0.88 - Recall: 0.93 - F1-Score: 0.91 - Support: 12,435 High Income: - Precision: 0.72 - Recall: 0.61 - F1-Score: 0.66 - Support: 3,846	Training Data: Accuracy: 99.99% Low Income: - Precision: 1.00 Recall: 1.00 - F1-Score: 1.00 - Support: 24,720 High Income: - Precision: 1.00 - Recall: 1.00 - F1-Score: 1.00 - Support: 7,841	Test Data: Accuracy: 84.87% Low Income: - Precision: 0.89 - Recall: 0.92 - F1-Score: 0.90 - Support: 12,435 High Income: - Precision: 0.70 - Recall: 0.62 - F1-Score: 0.66 - Support: 3,846
Model 3: Balanced dataset (Smote), O.H.E. and Hyperparameter tuning (focused on accuracy), in the balanced dataset		**Model 4: Balanced dataset (Smote), O.H.E. and Hyperparameter tuning (focused on a balance), in the balanced dataset**	
Training Data: Accuracy: **93.24%** Low Income: - Precision: 0.95 - Recall: 0.96 - F1-Score: 0.96 - Support: 24,720 High Income: - Precision: 0.87 - Recall: 0.85 - F1-Score: 0.86 - Support: 7,841	Test Data: Accuracy: **85.44%** Low Income: - Precision: 0.90 - Recall: 0.91 - F1-Score: 0.91 - Support: 12,435 High Income: - Precision: 0.70 - Recall: 0.67 - F1-Score: 0.69 - Support: 3,846	Training Data: Accuracy: **85.10%** Low Income: - Precision: 0.92 - Recall: 0.88 - F1-Score: 0.90 - Support: 24,720 High Income: - Precision: 0.67 - Recall: 0.75 - F1-Score: 0.71 - Support: 7,841	Test Data: Accuracy: **84.14%** Low Income: - Precision: 0.91 - Recall: 0.88 - F1-Score: 0.89 - Support: 12,435 High Income: - Precision: 0.65 - Recall: 0.72 - F1-Score: 0.68 - Support: 3,846

Note: *the values of precision, recall, and F-1 score shown in the chart above are the weighted average. Please refer to the appendix* ***6.2. "Random Forest model metrics"*** *to see a more descriptive analysis of each model performance.*

Conclusion:

This code needs to be executed only once to identify the optimal parameter set. Evaluating 216 different parameter combinations across five folds yields a total of 1,080 performance metrics. The execution required approximately 3 hours and 16 minutes to complete. After the best hyperparameter set has been determined, all following code executions just need to use the best parameters to run on unseen data.

Accuracy based parameters	Balanced based parameters
● Number of Estimators: 100 ● Maximum Depth: 20 ● Minimum Samples per Leaf: 1 ● Minimum Samples per Split: 10 ● Bootstrap Sampling: False ● Random State (Seed): 42	● Number of Estimators: 100 ● Maximum Depth: 10 ● Minimum Samples per Leaf: 1 ● Minimum Samples per Split: 2 ● Bootstrap Sampling: False ● Random State (Seed): 42

4.3 Decision Trees

After thoroughly preprocessing and exploring the data, we applied a Decision Trees model to our dataset. In this case, normalization of the data was unnecessary for this algorithm, as decisions are made based on feature thresholds rather than absolute values. Furthermore, our approach to handling missing attributes ensured a complete dataset for training.

After training the Decision Trees model, we attained exceptional accuracy on the training data, reaching an impressive 99.9%. However, such high accuracy often signals overfitting, where the model excessively captures noise in the training data rather than the underlying patterns. To address this, we employed pruning techniques to simplify the model's structure, reducing overfitting and enhancing generalization to unseen data. Following pruning, the model's accuracy decreased to 83.98%, indicating a more balanced and realistic representation of the model's performance. This decrease in accuracy post-pruning aligns with expectations, as the model prioritizes simplicity and generalization over training data perfection.

After training the Decision Trees model on the testing data, we obtained an accuracy of 80.82%. Following pruning, the model's accuracy increased to 83.92%. We prefer the model after pruning in both our training and testing data because it performs better on unseen data.

Incorporating the confusion matrix into our analysis provided valuable insights into the model's predictive capabilities. By visually representing true positive, true negative, false positive, and false negative predictions, the confusion matrix offered a distinct understanding of the model's strengths and weaknesses. As expected, our training data before pruning which had such high accuracy, only had 1 false negative and 0 false positives. Screenshots of our confusion matrices and classification reports are provided in section 7.1, which is attached for reference.

Model 1: Before Pruning		**Model 2: After Pruning**	
Training Data Accuracy: 99.99% Precision: 100% Recall: 100% F-1 Score: 100%	Test Data Accuracy: 80.82% Precision: 81% Recall: 81% F-1 Score: 81%	Train Data: Accuracy: 83.98% Precision: 83% Recall: 84% F-1 Score: 83%	Test Data: Accuracy: 83.92% Precision: 83% Recall: 84% F-1 Score: 82%

Note: *the values of precision, recall, and F-1 score shown in the chart above are the weighted average. Please refer to the appendix* ***6.3. "Decision Trees model metrics"*** *to see a more descriptive analysis of each model performance.*

4.4 K-Nearest Neighbors

Following the preprocessing phase and comprehensive data exploration, we implemented a K-Nearest Neighbors (KNN) algorithm on normalized data, varying the number of nearest neighbors (K) from 1 to 50 to discern its behavior. Acknowledging data imbalances, we integrated the Synthetic Minority Over-sampling Technique (SMOTE) to rectify disparities and enhance model robustness. Employing cross-validation techniques on imbalance and balanced datasets, we sought the optimal K value for maximal accuracy, using 5-fold cross-validation. Subsequently, we constructed and rigorously evaluated four KNN models using confusion matrices and critical performance metrics, including accuracy, F-1 score, precision, and recall, to identify models best suited to our research objectives. We evaluated the models using both the training and the test data. The table below shows the metrics obtained from each model:

Model 1: Imbalanced dataset and no CV implemented.		**Model 2: Imbalanced dataset and with CV implemented.**	
Training data Best K: 20 Accuracy: 80.46% Precision: 81% Recall: 80% F-1 Score: 76%	Test data Best K: 20 Accuracy: 80.30% Precision: 81% Recall: 80% F-1 Score: 75%	Training data Best K: 33 Accuracy: 84.60% Precision: 84% Recall: 85% F-1 Score: 84%	Test data: Best K = 33 Accuracy: 83.77% Precision: 83% Recall: 84% F-1 Score: 83%
Model 3: Balanced dataset and CV implemented.		**Model 4: balanced dataset and no CV implemented.**	
Training data Best K: 1 Accuracy: 89.26% Precision: 100% Recall: 100% F-1 Score: 100%	Test data Best K: 1 Accuracy: 89.26% Precision: 80% Recall: 79% F-1 Score: 79%	Training data Best K: 19 Accuracy: 85.15% Precision: 86% Recall: 85% F-1 Score: 85%	Test data: Best K: 19 Accuracy: 83.97% Precision: 84% Recall: 78% F-1 Score: 80%

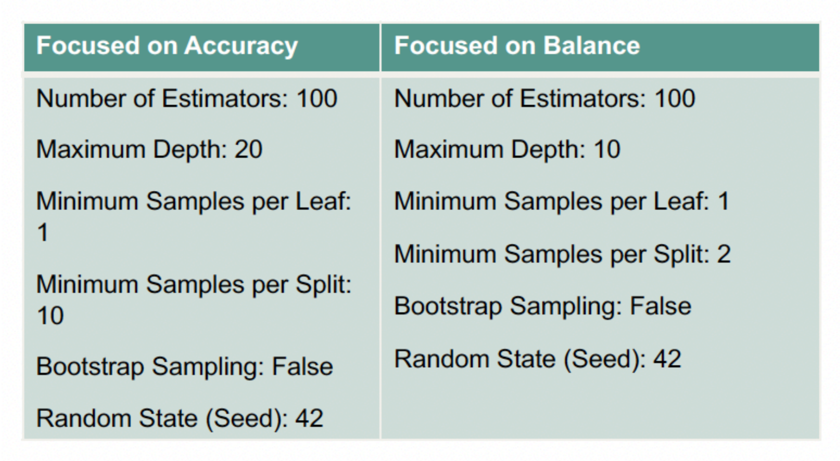
Note: *the values of precision, recall, and F-1 score shown in the chart above are the weighted average. Please refer to the appendix* ***6.4. "KNN model metrics"*** *to see a more descriptive analysis of each model performance.*

The exploration of the KNN models shown above reveals that while Model 3 initially seemed promising with its balanced dataset and cross-validation integration, its reliance on k = 1 raises concerns about overfitting, as evidenced by a slight drop in accuracy on the test set. In contrast, Model 2, despite operating on an imbalanced dataset, exhibits notable performance improvements over Model 1 due to the inclusion of cross-validation. This behavior suggests a pragmatic balance between model performance and dataset characteristics. Further optimization could enhance Model 2's suitability for deployment. Therefore, Model 2 is the preferred choice among the 4 KNN models built for this project.

5. Model selection / Closing thoughts

Our final model selection was Random Forests using the Balanced hyperparameters, since we want to reduce overfitting while using balanced data, after a thorough consideration process of KNN as an alternative.

These hyperparameters, tuned by a process of 3 hours of tuning, gave us two different perspectives on how to approach the predictions. The first one, was a set of parameters that enabled a higher accuracy, while increasing the overfitting, while the second one loses a small percentage of accuracy (1.4%) while it assures that the final overfitting is reduced, hence being selected as our final model.



Focused on Accuracy	Focused on Balance
Number of Estimators: 100 Maximum Depth: 20 Minimum Samples per Leaf: 1 Minimum Samples per Split: 10 Bootstrap Sampling: False Random State (Seed): 42	Number of Estimators: 100 Maximum Depth: 10 Minimum Samples per Leaf: 1 Minimum Samples per Split: 2 Bootstrap Sampling: False Random State (Seed): 42

In summary, Random Forests with Balanced hyperparameters emerged as the optimal choice for our classification problem. This selection guarantees optimal performance on both seen and unseen data, striking a balance between accuracy and overfitting, and ensuring the robustness of our model across various scenarios.

6. Appendix

6.1 Naive Bayes Model Metrics

Model 1: No Laplace Smoothing

```
Accuracy for train data:
              precision    recall  f1-score   support

       <=50K       0.82      0.95      0.88     24698
        >50K       0.68      0.32      0.44      7839

    accuracy                           0.80     32537
   macro avg       0.75      0.64      0.66     32537
weighted avg       0.78      0.80      0.77     32537

Confusion Matrix for train data:
       <=50K  >50K
<=50K  23513  1185
>50K    5297  2542
Accuracy for train data: 80.08%


Accuracy for test data
              precision    recall  f1-score   support

       <=50K       0.82      0.95      0.88     12435
        >50K       0.67      0.32      0.43      3846

    accuracy                           0.80     16281
   macro avg       0.74      0.64      0.66     16281
weighted avg       0.78      0.80      0.77     16281

Confusion Matrix:
       <=50K  >50K
<=50K  11820   615
>50K    2615  1231
Accuracy: 80.16%
```

Model 2: Laplace Smoothing

```
Performance on Train Data W/ Laplace Smoothing:
              precision    recall  f1-score   support

       <=50K       0.79      0.96      0.87     24698
        >50K       0.62      0.21      0.32      7839

    accuracy                           0.78     32537
   macro avg       0.71      0.59      0.59     32537
weighted avg       0.75      0.78      0.74     32537

Confusion Matrix:
       <=50K  >50K
<=50K  23671  1027
>50K    6162  1677
Accuracy: 77.91%


Performance for Test Data W/ Laplace Smoothing
              precision    recall  f1-score   support

       <=50K       0.80      0.96      0.87     12435
        >50K       0.62      0.21      0.32      3846

    accuracy                           0.78     16281
   macro avg       0.71      0.59      0.59     16281
weighted avg       0.75      0.78      0.74     16281

Confusion Matrix:
       <=50K  >50K
<=50K  11931   504
>50K    3032   814
Accuracy: 78.28%
```

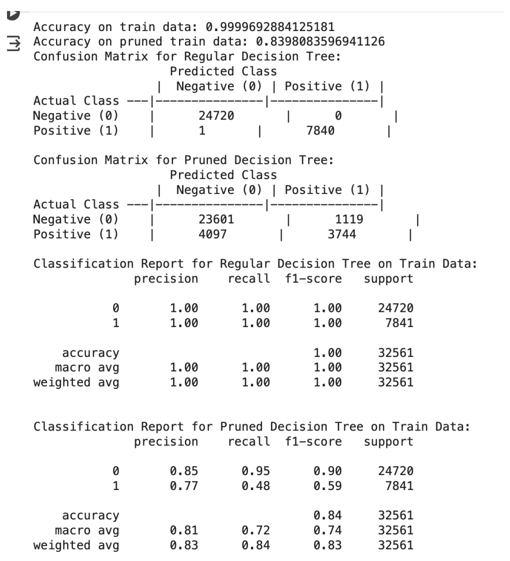
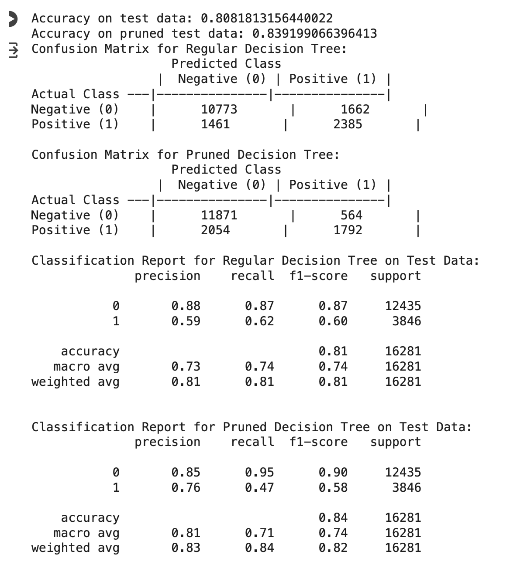
6.2 Random Forest Model Metrics

Model 1: Imbalanced dataset and O.H.E.		**Model 2: Balanced dataset (SMOTE) and O.H.E.**	
Training Data: Accuracy: 99.99% Low Income: - Precision: 1.00 - Recall: 1.00 - F1-Score: 1.00 - Support: 24,720 High Income: - Precision: 1.00 - Recall: 1.00 - F1-Score: 1.00 - Support: 7,841	Test Data: Accuracy: 84.87% Low Income: - Precision: 0.88 - Recall: 0.93 - F1-Score: 0.91 - Support: 12,435 High Income: - Precision: 0.72 - Recall: 0.61 - F1-Score: 0.66 - Support: 3,846	Training Data: Accuracy: 99.99% Low Income: - Precision: 1.00 Recall: 1.00 - F1-Score: 1.00 - Support: 24,720 High Income: - Precision: 1.00 - Recall: 1.00 - F1-Score: 1.00 - Support: 7,841	Test Data: Accuracy: 84.87% Low Income: - Precision: 0.89 - Recall: 0.92 - F1-Score: 0.90 - Support: 12,435 High Income: - Precision: 0.70 - Recall: 0.62 - F1-Score: 0.66 - Support: 3,846

Model 3: Balanced dataset (Smote), O.H.E. and Hyperparameter tuning (focused on accuracy), in the balanced dataset		**Model 4: Balanced dataset (Smote), O.H.E. and Hyperparameter tuning (focused on a balance), in the balanced dataset**	
Training Data: Accuracy: **93.24%** Low Income: - Precision: 0.95 - Recall: 0.96 - F1-Score: 0.96 - Support: 24,720 High Income: - Precision: 0.87 - Recall: 0.85 - F1-Score: 0.86 - Support: 7,841	Test Data: Accuracy: **85.44%** Low Income: - Precision: 0.90 - Recall: 0.91 - F1-Score: 0.91 - Support: 12,435 High Income: - Precision: 0.70 - Recall: 0.67 - F1-Score: 0.69 - Support: 3,846	Training Data: Accuracy: **85.10%** Low Income: - Precision: 0.92 - Recall: 0.88 - F1-Score: 0.90 - Support: 24,720 High Income: - Precision: 0.67 - Recall: 0.75 - F1-Score: 0.71 - Support: 7,841	Test Data: Accuracy: **84.14%** Low Income: - Precision: 0.91 - Recall: 0.88 - F1-Score: 0.89 - Support: 12,435 High Income: - Precision: 0.65 - Recall: 0.72 - F1-Score: 0.68 - Support: 3,846

Note: *the values of precision, recall, and F-1 score shown in the chart above are the weighted average. Please refer to the appendix* ***6.2. "Random Forest model metrics"*** *to see a more descriptive analysis of each model performance.*

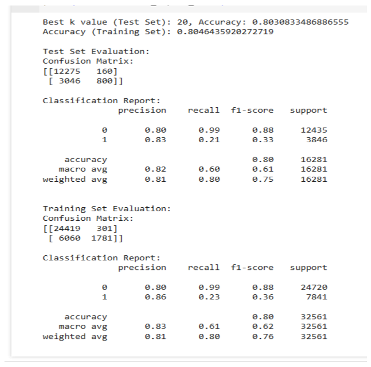
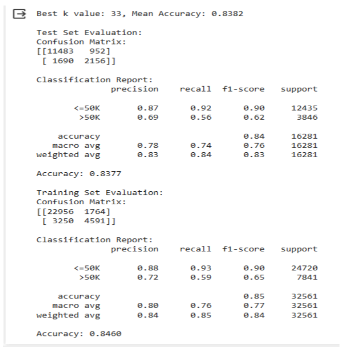
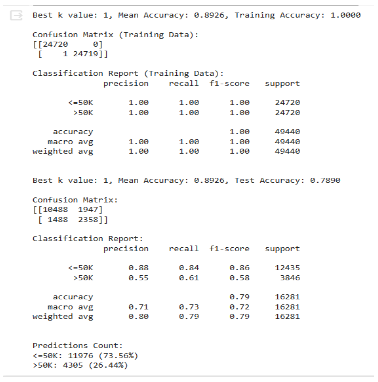
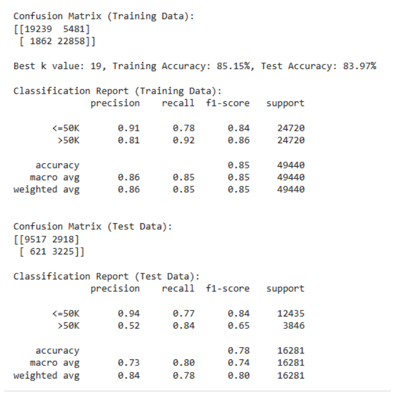
6.3 Decision Tree Model Metrics

Model 1: Train Data Before and After Pruning	**Model 2: Test Data Before and After Pruning**
	

```
Accuracy on train data: 0.9999692884125181
Accuracy on pruned train data: 0.8398083596941126
Confusion Matrix for Regular Decision Tree:
                   Predicted Class
                 |  Negative (0) | Positive (1) |
Actual Class ---|---------------|---------------|
Negative (0)    |      24720      |      0       |
Positive (1)    |      1        |      7840      |

Confusion Matrix for Pruned Decision Tree:
                   Predicted Class
                 |  Negative (0) | Positive (1) |
Actual Class ---|---------------|---------------|
Negative (0)    |      23601      |      1119       |
Positive (1)    |      4097       |      3744       |

Classification Report for Regular Decision Tree on Train Data:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     24720
           1       1.00      1.00      1.00      7841

    accuracy                           1.00     32561
   macro avg       1.00      1.00      1.00     32561
weighted avg       1.00      1.00      1.00     32561


Classification Report for Pruned Decision Tree on Train Data:
              precision    recall  f1-score   support

           0       0.85      0.95      0.90     24720
           1       0.77      0.48      0.59      7841

    accuracy                           0.84     32561
   macro avg       0.81      0.72      0.74     32561
weighted avg       0.83      0.84      0.83     32561
```

```
Accuracy on test data: 0.8081813156440022
Accuracy on pruned test data: 0.839199066396413
Confusion Matrix for Regular Decision Tree:
                   Predicted Class
                 |  Negative (0) | Positive (1) |
Actual Class ---|---------------|---------------|
Negative (0)    |      10773      |      1662       |
Positive (1)    |      1461       |      2385       |

Confusion Matrix for Pruned Decision Tree:
                   Predicted Class
                 |  Negative (0) | Positive (1) |
Actual Class ---|---------------|---------------|
Negative (0)    |      11871      |      564       |
Positive (1)    |      2054       |      1792       |

Classification Report for Regular Decision Tree on Test Data:
              precision    recall  f1-score   support

           0       0.88      0.87      0.87     12435
           1       0.59      0.62      0.60      3846

    accuracy                           0.81     16281
   macro avg       0.73      0.74      0.74     16281
weighted avg       0.81      0.81      0.81     16281


Classification Report for Pruned Decision Tree on Test Data:
              precision    recall  f1-score   support

           0       0.85      0.95      0.90     12435
           1       0.76      0.47      0.58      3846

    accuracy                           0.84     16281
   macro avg       0.81      0.71      0.74     16281
weighted avg       0.83      0.84      0.82     16281
```

6.4 KNN model metrics

Model 1: Imbalanced dataset and no CV implemented.	**Model 2: Imbalanced dataset and with CV implemented.**
	
Model 3: Balanced dataset and CV implemented.	**Model 4: balanced dataset and no CV implemented.**
	

```
Best k value (Test Set): 20, Accuracy: 0.8030833486886555
Accuracy (Training Set): 0.8046435920272719

Test Set Evaluation:
Confusion Matrix:
[[12275   160]
 [ 3046   800]]

Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.99      0.88     12435
           1       0.83      0.21      0.33      3846

    accuracy                           0.80     16281
   macro avg       0.82      0.60      0.61     16281
weighted avg       0.81      0.80      0.75     16281


Training Set Evaluation:
Confusion Matrix:
[[24419   301]
 [ 6060  1781]]

Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.99      0.88     24720
           1       0.86      0.23      0.36      7841

    accuracy                           0.80     32561
   macro avg       0.83      0.61      0.62     32561
weighted avg       0.81      0.80      0.76     32561
```

```
Best k value: 33, Mean Accuracy: 0.8382

Test Set Evaluation:
Confusion Matrix:
[[11483   952]
 [ 1690  2156]]

Classification Report:
              precision    recall  f1-score   support

       <=50K       0.87      0.92      0.90     12435
        >50K       0.69      0.56      0.62      3846

    accuracy                           0.84     16281
   macro avg       0.78      0.74      0.76     16281
weighted avg       0.83      0.84      0.83     16281

Accuracy: 0.8377

Training Set Evaluation:
Confusion Matrix:
[[22956  1764]
 [ 3250  4591]]

Classification Report:
              precision    recall  f1-score   support

       <=50K       0.88      0.93      0.90     24720
        >50K       0.72      0.59      0.65      7841

    accuracy                           0.85     32561
   macro avg       0.80      0.76      0.77     32561
weighted avg       0.84      0.85      0.84     32561

Accuracy: 0.8460
```

```
Best k value: 1, Mean Accuracy: 0.8926, Training Accuracy: 1.0000

Confusion Matrix (Training Data):
[[24720     0]
 [    1 24719]]

Classification Report (Training Data):
              precision    recall  f1-score   support

       <=50K       1.00      1.00      1.00     24720
        >50K       1.00      1.00      1.00     24720

    accuracy                           1.00     49440
   macro avg       1.00      1.00      1.00     49440
weighted avg       1.00      1.00      1.00     49440


Best k value: 1, Mean Accuracy: 0.8926, Test Accuracy: 0.7890

Confusion Matrix:
[[10488  1947]
 [ 1488  2358]]

Classification Report:
              precision    recall  f1-score   support

       <=50K       0.88      0.84      0.86     12435
        >50K       0.55      0.61      0.58      3846

    accuracy                           0.79     16281
   macro avg       0.71      0.73      0.72     16281
weighted avg       0.80      0.79      0.79     16281


Predictions Count:
<=50K: 11976 (73.56%)
>50K: 4305 (26.44%)
```

```
Confusion Matrix (Training Data):
[[19239  5481]
 [ 1862 22858]]

Best k value: 19, Training Accuracy: 85.15%, Test Accuracy: 83.97%

Classification Report (Training Data):
              precision    recall  f1-score   support

       <=50K       0.91      0.78      0.84     24720
        >50K       0.81      0.92      0.86     24720

    accuracy                           0.85     49440
   macro avg       0.86      0.85      0.85     49440
weighted avg       0.86      0.85      0.85     49440


Confusion Matrix (Test Data):
[[9517 2918]
 [ 621 3225]]

Classification Report (Test Data):
              precision    recall  f1-score   support

       <=50K       0.94      0.77      0.84     12435
        >50K       0.52      0.84      0.65      3846

    accuracy                           0.78     16281
   macro avg       0.73      0.80      0.74     16281
weighted avg       0.84      0.78      0.80     16281
```