CISC 5790 DATA MINING COURSE FINAL PROJECT

Demographic Income Prediction

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

OBJECTIVES

Objectives

- Conduct comprehensive analysis of census dataset, including data cleaning and feature selection, to understand variable distributions and relationships affecting income levels.
- Implement various classification models learned in class to predict income classification
- Compare algorithm performance using metrics like accuracy and precision to identify the most effective models for accurate predictions.
- Analyze feature importance to identify key factors impacting income levels significantly and provide predictive insights.

DATA CLEANING AND PREPROCESSING

Missing values

- Identified missing values in features: native-country, work-class, and occupation
- Use the mode to fill missing values for ‘native-country’ and ‘work-class’
- Mode for ‘native-country’ determined as ‘United-States’ due to its high occurrence (8116 out of 16,281 entries)
- Mode for ‘work-class' determined as ‘Private’ due to its high occurrence (6696 out of 16281 entries).
- Replaced missing values represented as ‘?’ with the most common elements.
- Implemented mode-based approach due to the high occurrence of these values.
- Explored alternative techniques such as using the KNN algorithm for predicting missing labels.

Missing values - continued

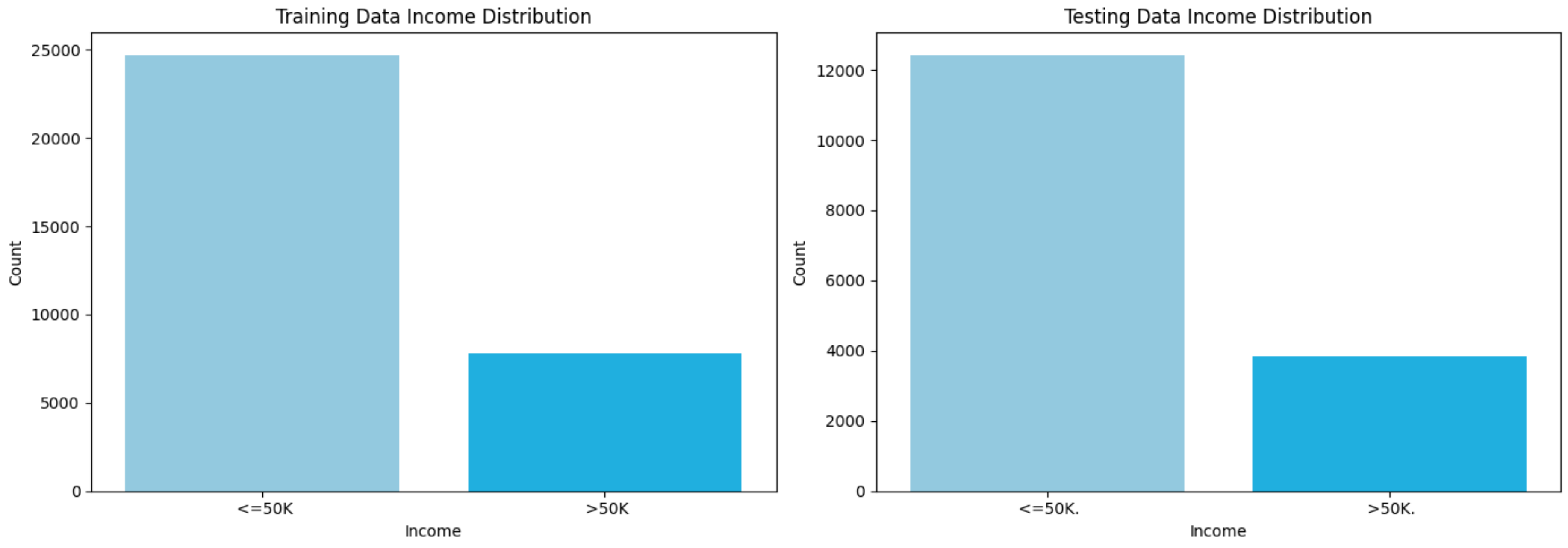
- Determined the mode for the 'occupation' feature as 'Prof-specialty' with an occurrence of 1652
- Employed KNN to predict missing values for the 'occupation' feature, utilizing k=5
- Imported KNN Imputer from sklearn.impute to facilitate the imputation process
- Designated the 'occupation' feature as the target variable and dropped it from both the x train and test data
- Ran KNN to predict missing values, combining the imputed data with the original dataset
- Applied the same methods to fill in missing values for both the train and test data

Features transformation

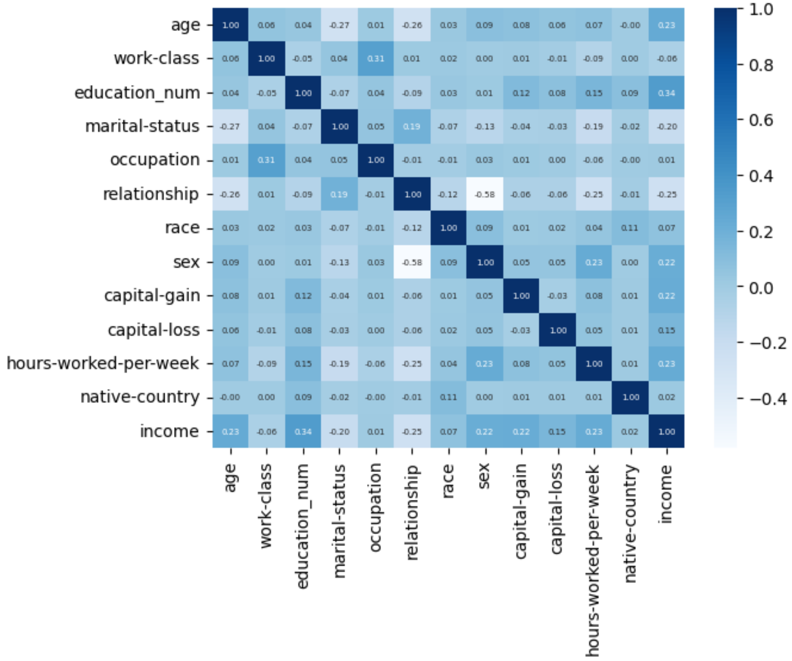
- Utilized LabelEncoder() from the sklearn.preprocessing library to convert all data to numerical format
- Dropped the 'education' feature since 'education-num' represents it
- Designated 'income' as the target variable
- Identified categorical attributes in both training and test datasets, including 'work-class', 'education', 'marital-status', 'occupation', 'relationship', 'race', 'sex', 'native-country', and 'income'
- Implemented label encoding to transform categorical attributes into numerical representations, ensuring compatibility with models like KNN
- Employed One Hot Encoding to effectively manage categorical variables within the Random Forests algorithm
- Normalized the entire dataset to ensure uniform contribution of all features to distance calculations in the KNN model, enhancing robustness and effectiveness of modeling process

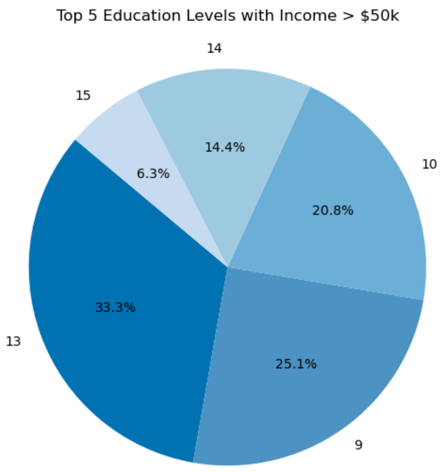
Data standardization

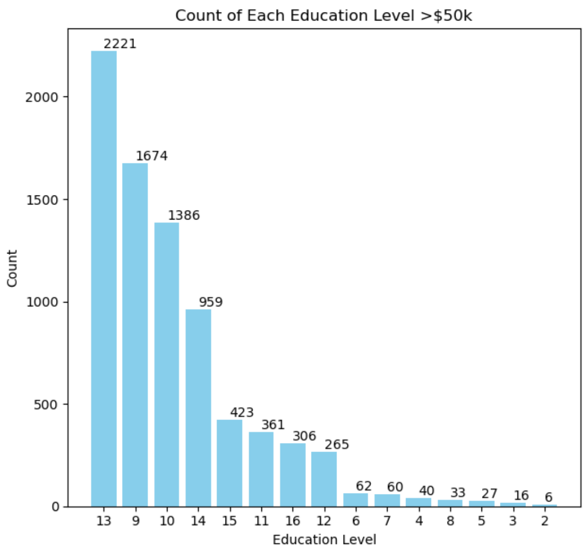
- Significant class imbalance observed in the target variables, with a majority of cases “<=50 k” compared to those earning “>50k”.
- Applied SMOTE to address the imbalance by augmenting the minority class, aiming to reduce the risk of bias towards the majority class and overfitting.
- Considered under sampling but opted against it due to inappropriate dataset size, which could lead to loss of essential data.
- Identified irregularities in the formatting of the target variable, ‘income class’, in both training and test datasets. These inconsistencies were things such as periods, spaces, among others.
- Standardized labels across datasets to ensure uniformity and facilitate seamless model training and evaluation.



DATA VISUALIZATION

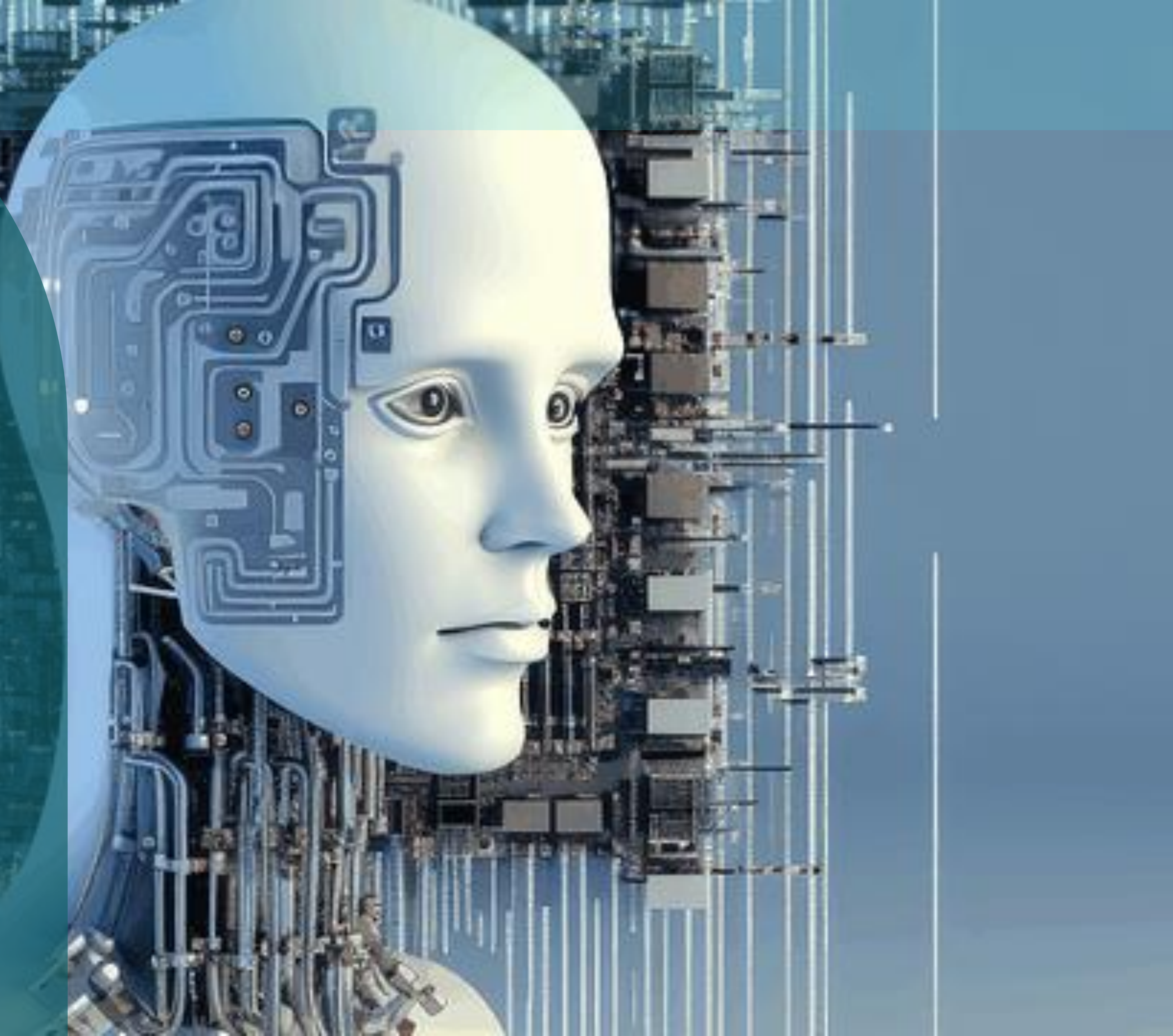






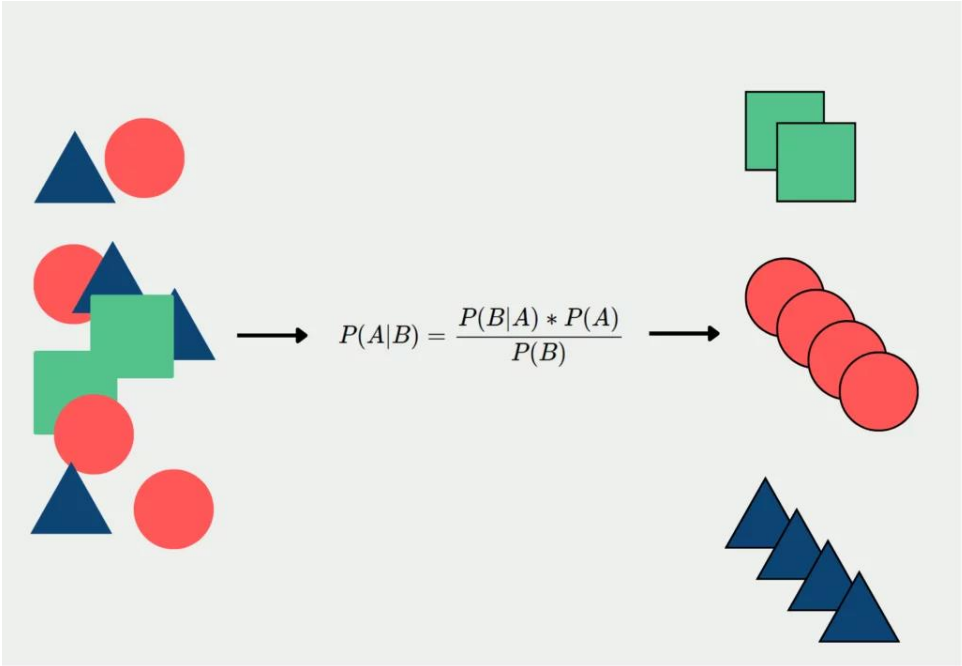
DATA VISUALIZATION

MODEL IMPLEMENTATION & ASSESMENT



Naive Bayes – model implementation

- Implement the model in test and train data
- Used Label Encoder
- Evaluated two variations of the Naive Bayes model to determine the most effective approach for our dataset. (Laplace Smoothing, No Laplace Smoothing)
- We assessed the models based on accuracy, precision, and F1 score.



Naive Bayes - Results

MODEL 1

```
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%
```

```
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%
```

Naive Bayes - Results

MODEL 2 – USING LAPLACE SMOOTHING

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%

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

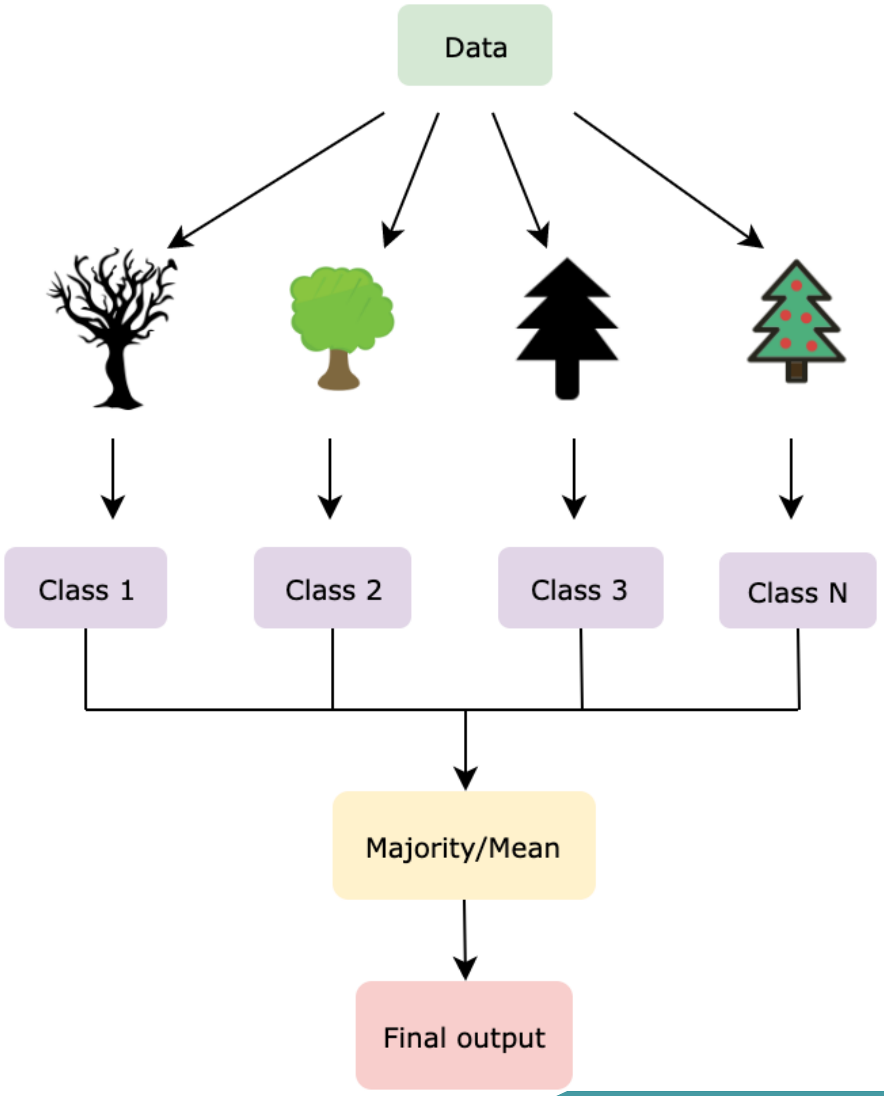
Accuracy: 77.91%

Random Forest model implementation

We utilized the Random Forest method, combining multiple decision trees trained on varied datasets and parameters to achieve optimal results.

We also applied techniques such as One Hot Encoding for categorical values and data balancing through SMOTE.

As a last step, we used Hyperparameter Tuning to choose the best parameter set among max depth, minimum samples per leaf, minimum samples to be considered for a split, and number of trees to be trained (number of estimators).

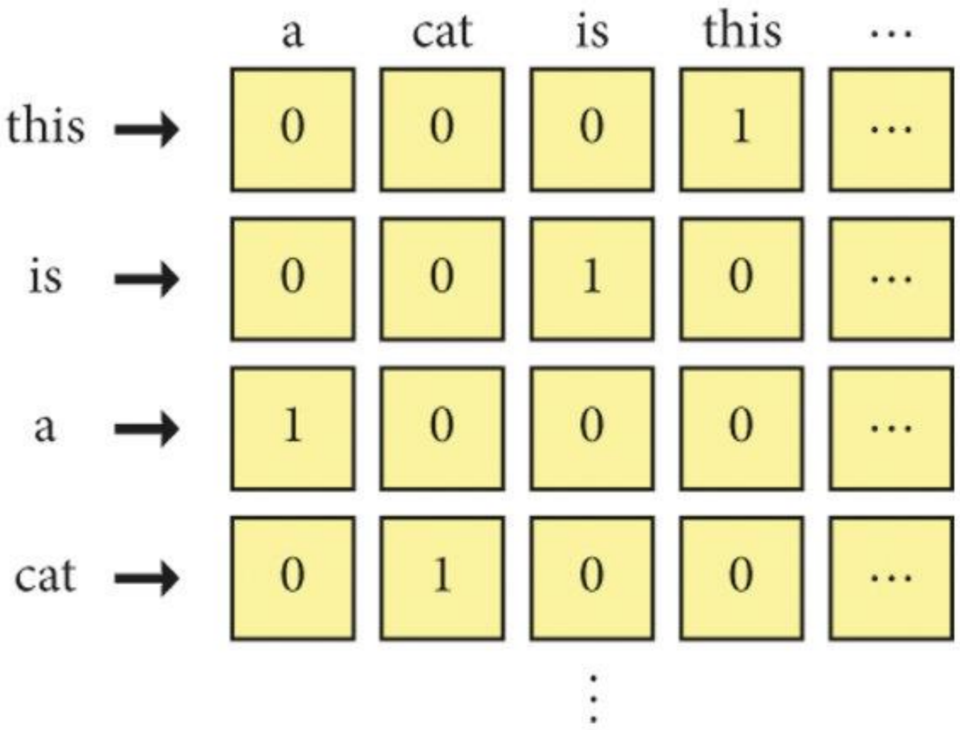


Random Forest - encoding

For the encoding, we evaluated two alternatives, One Hot Encoding and Label Encoding.

As a final decision, we chose One Hot Encoding since Label Encoding introduces bias and applies a relative “order” between categorical values. In our case, our categorical variables are <u>not ordinal,</u> using this technique would have affected the training process.

As a result, our model was trained with 107 features after the one hot encoding process.

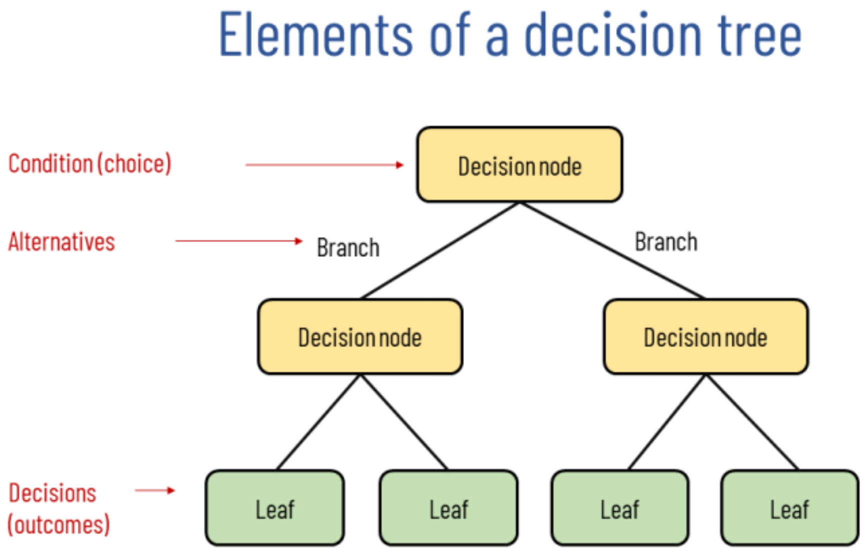


Random Forest – hyperparameter tuning

- After using hyperparameter tuning, we fitted 5 folds, using GridSearchCV, in Sci-kit learn. This implementation checks for all of the possible parameter combinations, evaluates them using 5 fold CV and chooses the best one using an accuracy test.

Decision Tree model implementation

- Performed this model on train and test data
- Imported DecisionTreeClassifier from sklearn.tree
- Target Function: income
- Number of values was 2: ‘<=50k’ : 0 and ‘>50k’ : 1
- For this type of model, normalization is not required
- This is because features are selected based on entropy (measure of randomness)
- Training Data may contain missing values and Errors
- We did handle missing values



Decision Tree model assessment – train data

```
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 is so high because model overfit. It decreases after pruning.
- **Errors Before Pruning**
- False Positives: 0
- False Negatives: 1
- **Errors After Pruning**
- False Positives: 1119
- False Negatives: 4097

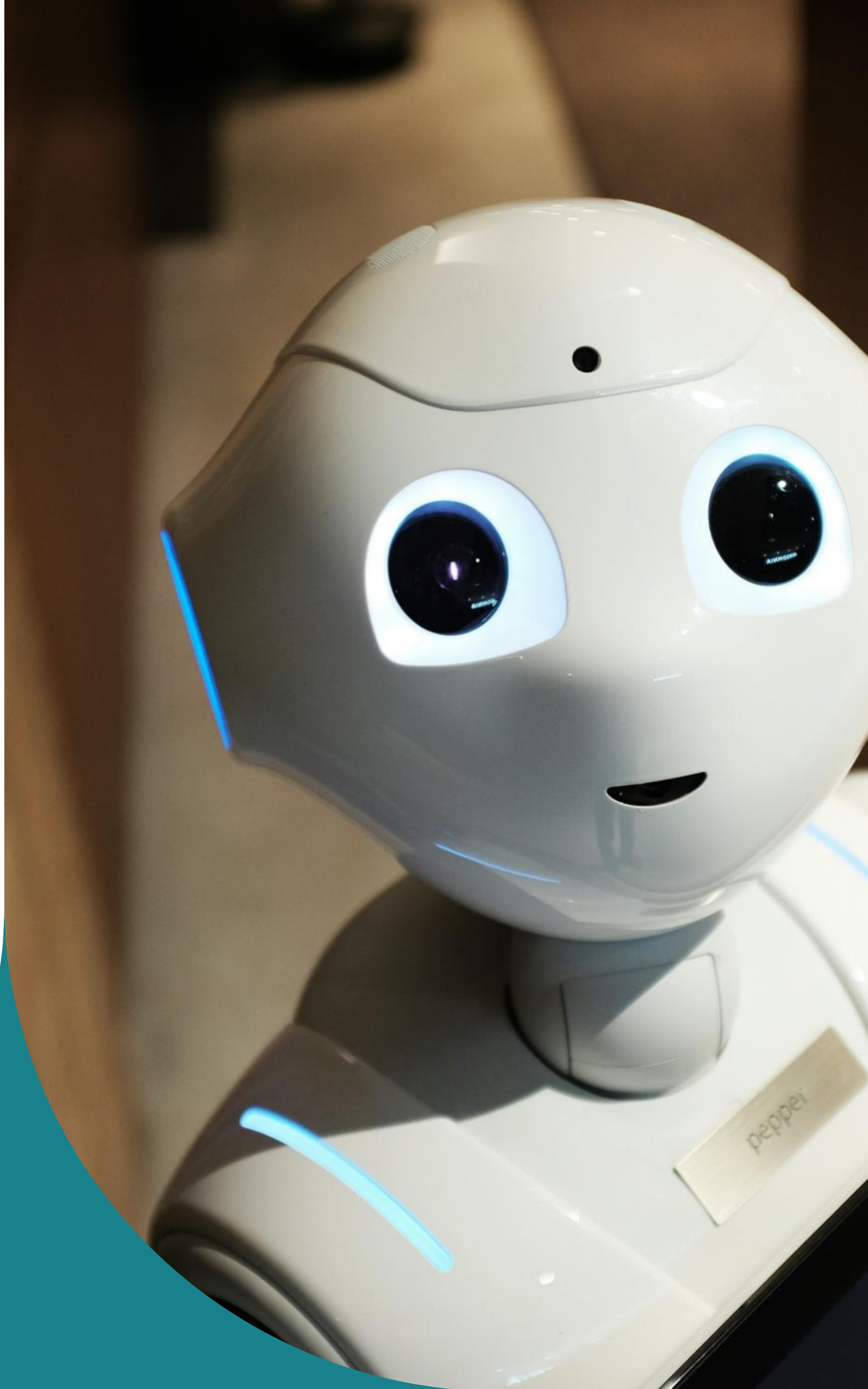
Decision Tree model assessment – test data

```
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
```

- Errors Before Pruning
- False Positives: 1662
- False Negatives: 1461
- Errors After Pruning
- False Positives: 564
- False Negatives: 2054

KNN model implementation

- Implemented a K-nearest neighbor (KNN) algorithm with a normalized data.
- Explored range of k from 1 to 50 without cross – validation to find optimal k for highest accuracy.
- Enhanced the model with 5 –fold cross validation to determine the best k within the range define above.
- Applied Synthetic Minority Over-sampling Technique (SMOTE) to address data imbalance.
- Constructed and compared 4 KNN models.



KNN model assessment

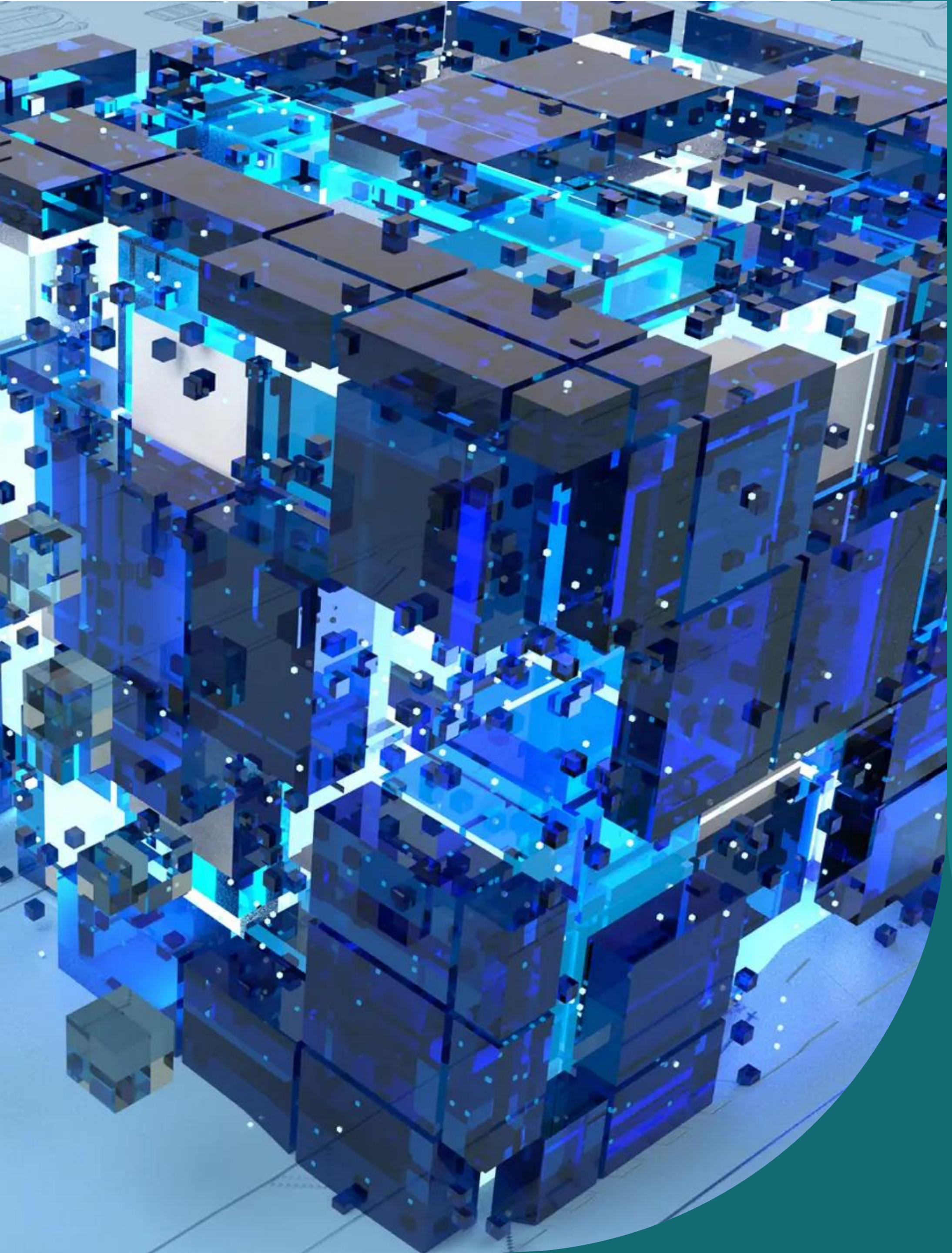
- Evaluated the 4 models using confusion matrices, accuracy, F-1 score, precision, recall.

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%

KNN model selection

- Model 3, initially promising with balanced data and cross-validation, shows overfitting concerns with k = 1. Model 2, despite an imbalanced dataset, performs notably well due to cross-validation. Further optimization can enhance Model 2's deployment suitability, making it the preferred choice among the 4 KNN models.

MODEL SELECTION & FINAL THOUGHTS



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 evaluating KNN as a second option.

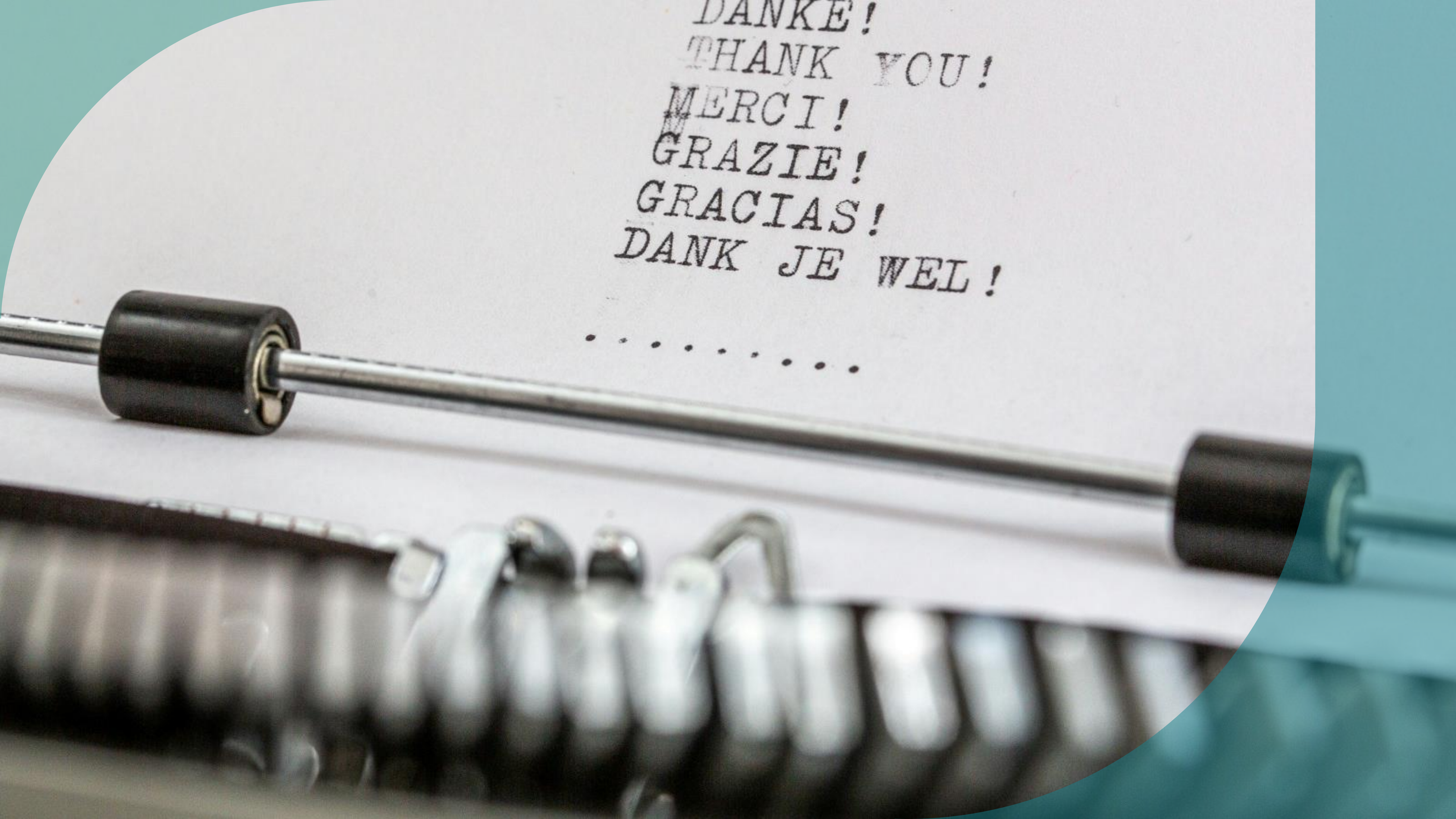
Model 3: Balanced dataset (Smote), O.H.E. and Hyperparameter tuning (focused on accuracy), in the balanced dataset		Model 4: Balanced dataset (Smote), O.H.E. and Hyperparameter tuning (focused on a balance), in the balanced dataset	
Training Data: Accuracy: **93.24%** Low Income: - Precision: 0.95 - 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

Model Selection / Closing thoughts

Model Selection / Closing thoughts

- We checked for a total of 216 combinations across the parameter list in a 5-fold CV, testing 1080 combination cases. As a result, our team got two Hyperparameter sets, from which we chose the balanced approach for the predictions.

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


DANKE!
THANK YOU!
MERCI!
GRAZIE!
GRACIAS!
DANK JE WEL!