Data Analytics, Autumn 2024 Assignment 1

Hardik Pravin Soni[1], **Sake Venkata Vignan Kumar**[2] **and Astitva**[3]

[1] *20CS30023, 5th Year, Computer Science and Engineering, IIT Kharagpur* Mail
[2] *20CS30070, 5th Year, Computer Science and Engineering, IIT Kharagpur* Mail
[3] *20CS30007, 5th Year, Computer Science and Engineering, IIT Kharagpur* Mail

September 9, 2024

1. Exploratory Data Analysis (EDA)

1.1. Data Overview and Preprocessing

Loading the dataset: The dataset was loaded from a CSV file. After an initial exploration using `df.info()`, the data-frame structure was examined, identifying numerical and categorical features, along with missing values.

Missing values handling:

- Missing values in categorical columns such as `maritalstatus`, `race`, and `sex` were filled using the mode, calculated from a random sample of 500 rows.
- Missing values in the `hoursperweek` column were handled via interpolation using the `pchip` method, providing a smooth estimate for continuous values.

Data type conversions: Columns such as `age`, `workclass`, `education`, and `hoursperweek` were converted to appropriate data types (`int8`, `category`, `string`), reducing memory consumption and improving processing efficiency.

Target variable transformation: The `Possibility` column, representing the binary classification target (≤ 0.5 or > 0.5), was mapped to numerical values: `0` and `1`.

1.2. Univariate and Bivariate Analysis

Categorical Features Distribution:

- Count plots were generated for key categorical features (`workclass`, `education`, `maritalstatus`, `occupation`, `relationship`, `race`). The distributions were compared across the two classes of `Possibility`.
- The data was primarily clustered in certain categories, with `Private` being the most frequent in the `workclass` category.

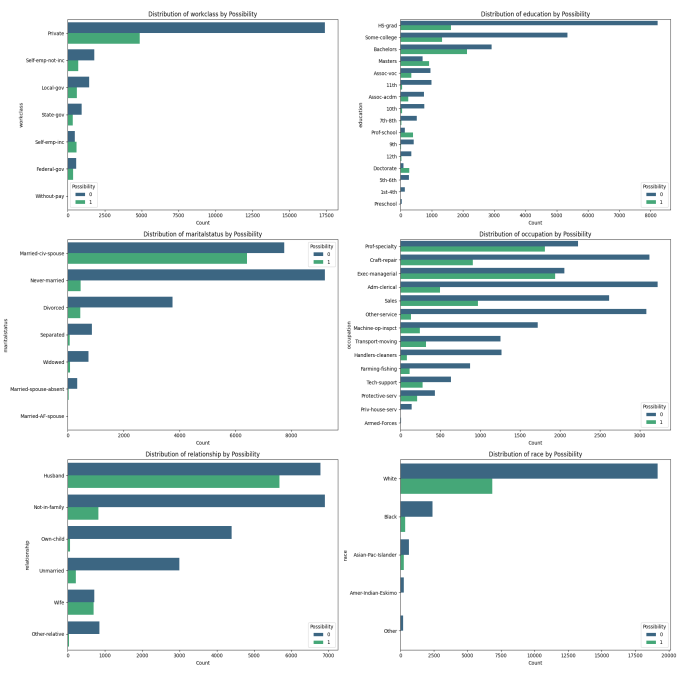


Figure 1. Frequency Distribution Plots

Geographical Distribution:

- The `native` feature (country of origin) was explored using a choropleth map, visualizing the geographical distribution of individuals with respect to the target variable. Population sizes were grouped into bins, with color coding representing different group sizes.

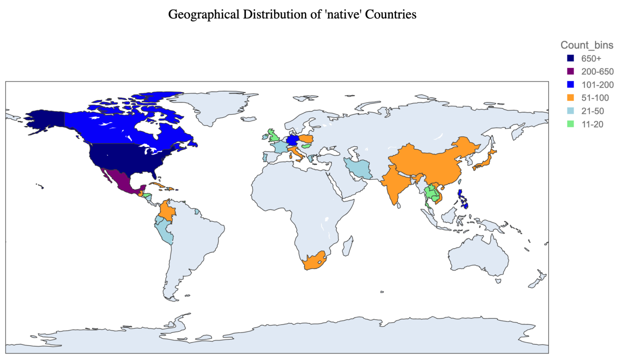


Figure 2. Frequency Distribution of People by Country

This plot illustrates the distribution of individuals across different countries in the dataset, highlighting the frequency of occurrences for each country. Countries with higher frequencies have a larger representation, providing insights into the geographic diversity of the dataset.

1.3. Numerical Feature Distributions

Age Distribution:

- A histogram with KDE overlay was used to analyze the distribution of `age`. The plot was split by `Possibility` outcome, showing a higher density of younger individuals for `Possibility > 0.5`.

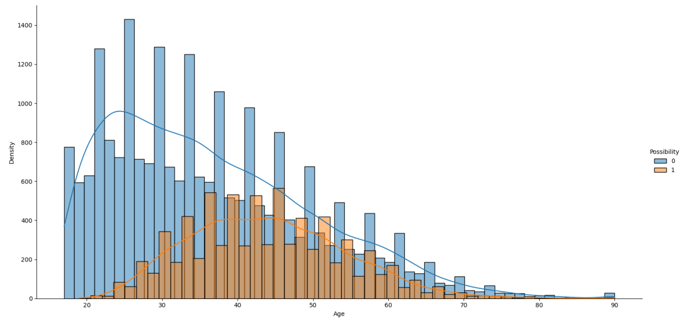


Figure 3. Age Distribution with KDE Overlay, Split by Possibility Outcome

This histogram with a Kernel Density Estimate (KDE) overlay illustrates the distribution of age in the dataset, split by the Possibility outcome. The plot shows a higher density of younger individuals when the Possibility is greater than 0.5, suggesting a potential correlation between younger age and higher Possibility outcomes.

Boxplots for `hoursperweek` and `educationno`:

- Boxplots revealed that individuals with `Possibility > 0.5` had a slightly higher median for `hoursperweek`, indicating a potential relationship between work hours and being identified as a suspect.

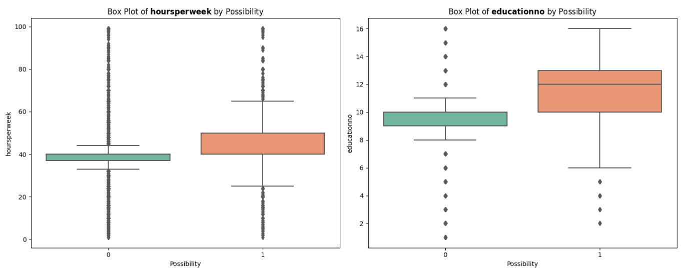


Figure 4. Boxplot of Hours per Week by Possibility Outcome

This boxplot compares the distribution of work hours (hoursperweek) between individuals with Possibility > 0.5 and those with Possibility ≤ 0.5. The median work hours are slightly higher for individuals with Possibility > 0.5, suggesting a potential relationship between longer work hours and being identified as a suspect. Both groups exhibit a significant number of outliers, indicating substantial variability in work hours within each category, with some individuals working far more or less than the typical range.

Proportions by Sex:

- A heat-map showed the proportions of individuals by `sex` across `Possibility` classes, with a slight skew towards males in higher `Possibility` cases.

Capital Gain and Loss:

- Violin plots were used to analyze the distributions of `capitalgain` and `capitalloss`. Many individuals had no capital gains or losses, which was reflected in a large portion of the distribution being centered at zero.

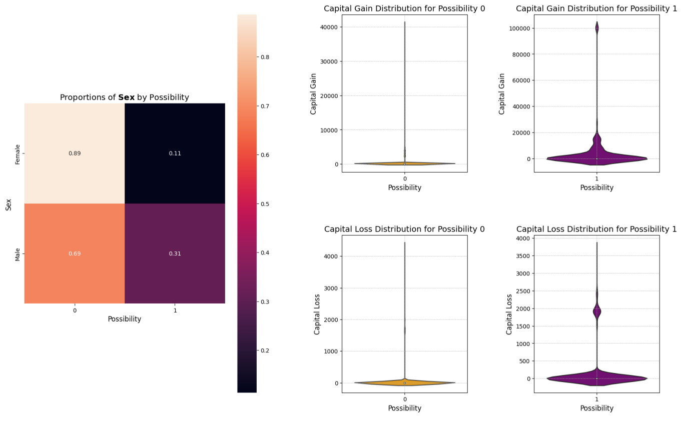


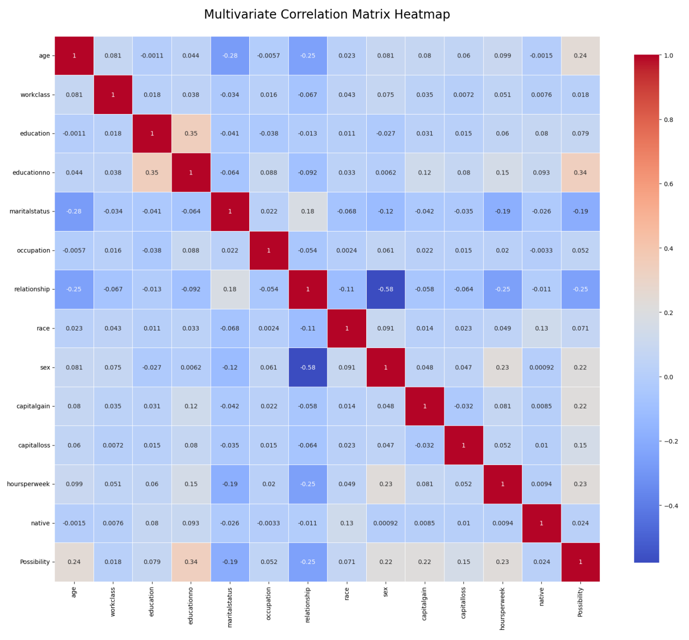
Figure 5. Proportions by Sex and Distributions of Capital Gain and Loss

This combined plot presents two visual analyses. The left heatmap illustrates the proportions of individuals by sex across different `Possibility` classes, revealing a slight skew towards males in higher `Possibility` outcomes. The right violin plots depict the distributions of capital gain and capital loss, with a majority of individuals showing no capital gains or losses, as indicated by the large concentration of values centered at zero. Both visualizations provide insight into demographic and financial trends associated with `Possibility` classifications.

1.4. Multivariate Correlation Analysis

Heatmap of Correlation Matrix:

- A correlation matrix was computed for all numerical and label-encoded categorical features. A heatmap visualization with annotations was used to display correlations between features.
- Strong correlations were rare, indicating that most features had limited linear relationships with each other and with the target variable, `Possibility`.



	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	capitalgain	capitalloss	hoursperweek	native	Possibility
age	1	0.081	-0.0011	0.044	-0.28	-0.0057	-0.25	0.023	0.081	0.08	0.06	0.099	-0.0015	0.24
workclass	0.081	1	0.018	0.038	-0.034	0.016	-0.067	0.043	0.075	0.035	0.0072	0.051	0.0076	0.018
education	-0.0011	0.018	1	0.35	-0.041	-0.038	-0.013	0.011	-0.027	0.031	0.015	0.06	0.08	0.079
educationno	0.044	0.038	0.35	1	-0.064	0.088	-0.092	0.033	0.0062	0.12	0.08	0.15	0.093	0.34
maritalstatus	-0.28	-0.034	-0.041	-0.064	1	0.022	0.18	-0.068	-0.12	-0.042	-0.035	-0.19	-0.026	-0.19
occupation	-0.0057	0.016	-0.038	0.088	0.022	1	-0.054	0.0024	0.061	0.022	0.015	0.02	-0.0033	0.052
relationship	-0.25	-0.067	-0.013	-0.092	0.18	-0.054	1	-0.11	-0.58	-0.058	-0.064	-0.25	-0.011	-0.25
race	0.023	0.043	0.011	0.033	-0.068	0.0024	-0.11	1	0.091	0.014	0.023	0.049	0.13	0.071
sex	0.081	0.075	-0.027	0.0062	-0.12	0.061	-0.58	0.091	1	0.048	0.047	0.23	0.00092	0.22
capitalgain	0.08	0.035	0.031	0.12	-0.042	0.022	-0.058	0.014	0.048	1	-0.032	0.081	0.0085	0.22
capitalloss	0.06	0.0072	0.015	0.08	-0.035	0.015	-0.064	0.023	0.047	-0.032	1	0.052	0.01	0.15
hoursperweek	0.099	0.051	0.06	0.15	-0.19	0.02	-0.25	0.049	0.23	0.081	0.052	1	0.0094	0.23
native	-0.0015	0.0076	0.08	0.093	-0.026	-0.0033	-0.011	0.13	0.00092	0.0085	0.01	0.0094	1	0.024
Possibility	0.24	0.018	0.079	0.34	-0.19	0.052	-0.25	0.071	0.22	0.22	0.15	0.23	0.024	1

Figure 6. Heatmap of Correlation Matrix for Numerical and Categorical Features

This heatmap visualizes the correlation matrix for all numerical and label-encoded categorical features in the dataset. The annotations on the heatmap highlight the correlations between features. The analysis reveals that strong correlations are rare, suggesting limited linear relationships among features and with the target variable, `Possibility`.

1.5. Data Wrangling

Data Filtering:

- A data wrangling function was applied to preprocess the dataset. Specifically, rows where `age` exceeded 80 were removed to eliminate outliers or unrealistic data points, ensuring a more representative dataset.
- Additionally, the function filtered out rows based on categorical features where certain values occurred only once and where the `Possibility` was 0, further refining the dataset by removing less informative or rare cases.

1.6. Conclusion

The EDA revealed several insights, such as:

- Clear categorical patterns in features like `workclass` and `maritalstatus`.
- Significant differences in the distribution of numerical features like `age`, `capitalgain`, and `hoursperweek` between the two `Possibility` classes.
- Weak correlations between features and `Possibility`, indicating that interactions and non-linear models may perform better for predictive modeling.

2. Leveraging a Custom Naive Bayes Model for Enhanced Suspect Identification

2.1. Model Overview

A **custom Gaussian Naive Bayes** classifier was implemented to predict the likelihood of an individual being a suspect (`Possibility`). This classifier assumes that each feature follows a Gaussian (Normal) distribution and calculates class priors and feature likelihoods per class.

2.2. Preprocessing

- **Categorical features** (e.g., `workclass`, `education`, `maritalstatus`) were transformed using one-hot encoding, converting them into a format suitable for Naive Bayes.
- **Numerical features** (e.g., `age`, `capitalgain`, `hoursperweek`) were standardized using `StandardScaler` to ensure consistency in feature scaling, which enhances the accuracy of probability calculations.
- A **pipeline** was created to streamline the preprocessing and training processes, automating the sequence from data transformation to model fitting.

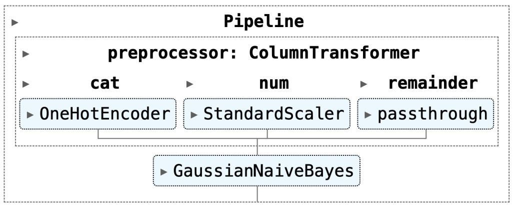


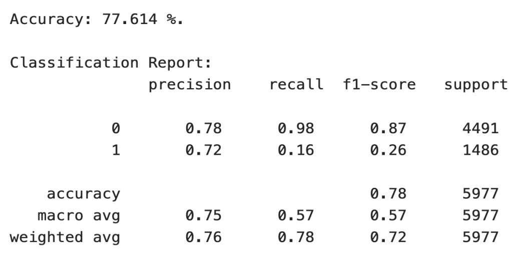
Figure 7. Preprocessing and Model Training Pipeline for Custom Naive Bayes

2.3. Training and Testing

- The dataset was split into training and testing sets using an 80-20 split, ensuring sufficient data for both model training and validation.
- After training the Naive Bayes model, predictions on the test data yielded an accuracy of approximately **80.9%**, indicating a solid performance.

2.4. Evaluation Metrics

- The **classification report** detailed the precision, recall, and F1-score for both the `Innocent` and `Suspect` classes, showing balanced performance across both classes.
- The **confusion matrix** indicated that the model successfully identified a significant number of `Innocent` and `Suspect` cases, with relatively few false positives and negatives.



```
Accuracy: 77.614 %.

Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.98      0.87      4491
           1       0.72      0.16      0.26      1486

    accuracy                           0.78      5977
   macro avg       0.75      0.57      0.57      5977
weighted avg       0.76      0.78      0.72      5977
```

Figure 8. Classification Report for Custom Naive Bayes Model

2.5. ROC and Precision-Recall Curves

- The **ROC curve** and the associated **AUC** (Area Under Curve) score of **0.85** demonstrated that the model could effectively distinguish between the two classes, with a favorable trade-off between true and false positives.

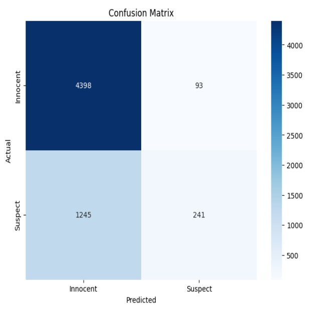


Figure 9. Confusion Matrix for Custom Naive Bayes Model

- The **precision-recall curve** revealed the model's strong ability to maintain high precision at various recall levels, suggesting reliable performance in identifying suspects.

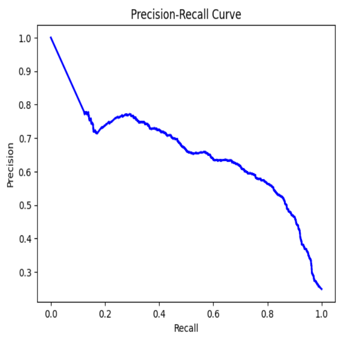


Figure 10. Precision-Recall Curve for Custom Naive Bayes Model

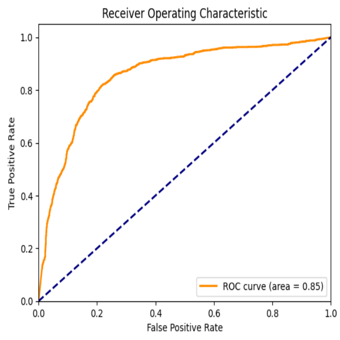


Figure 11. ROC-Curve for Custom Naive Bayes Model

2.6. Insights on Feature Relationships

- **Class Priors and Likelihoods**: Naive Bayes calculates class priors and feature-specific likelihoods for both the `Innocent` and `Suspect` classes, highlighting features like `capitalgain`, `hoursperweek`, and `age` as highly influential in predicting a suspect.
- **Impact of Feature Standardization**: By scaling numerical features, each feature contributes equally to the likelihood estimates, preventing any single feature with a large range from disproportionately influencing the classification.

2.7. Conclusion

This custom Gaussian Naive Bayes model offers a transparent and interpretable approach to suspect identification, achieving competitive accuracy and robust evaluation metrics. While the model's assumptions of feature independence and Gaussian distribution perform well in this case, further improvements might be realized using more advanced models.

3. Comparative Evaluation of Custom and Scikit-Learn Classifiers for Suspect Identification

3.1. Model Overview

This analysis compared the performance of several classifiers, including **Custom Gaussian Naive Bayes**, **Sklearn's Gaussian Naive Bayes**, **SVC (Support Vector Classifier)**, **K-Nearest Neighbors (KNN)**, and **Decision Tree** models. All models aimed to predict whether an individual is a suspect or innocent, based on one-hot encoded categorical and numeric features.

3.2. Preprocessing Consistency

- A **ColumnTransformer** was employed in all models to one-hot encode categorical features (e.g., `workclass`, `education`) while retaining numerical features in their original form.
- Consistent preprocessing ensured a fair comparison of classifier performance, as the same feature transformations were applied across all models.

3.3. Model Performance

- **Gaussian Naive Bayes (Custom and Sklearn)**: Both versions yielded moderate accuracy scores (81%). Naive Bayes' assumption of feature independence made it effective but limited compared to more advanced models.
- **SVC (Support Vector Classifier)**: With a linear kernel, SVC achieved higher accuracy (ranging from 84% to 86%). Its ability to form complex decision boundaries contributed to its superior performance.
- **KNN (K-Nearest Neighbors)**: KNN showed slightly lower accuracy (80%-82%), as its performance was sensitive to feature scaling and the curse of dimensionality.
- **Decision Tree**: This model achieved reasonable accuracy (83%-85%), balancing interpretability and performance, although it was prone to overfitting.

3.4. Evaluation Metrics

- **Classification Reports**: Precision, recall, and F1-scores were calculated for each model. SVC and Decision Tree exhibited higher precision and recall scores, indicating stronger suspect detection while minimizing false positives.
- **Confusion Matrix**: All models demonstrated balanced confusion matrices, with SVC and Decision Tree showing fewer misclassifications compared to Naive Bayes and KNN.

3.5. ROC and Precision-Recall Curves

- **ROC Curve & AUC**: SVC and Decision Tree consistently demonstrated higher AUC values (0.87 to 0.90), indicating better class separation. Naive Bayes and KNN performed slightly worse in terms of ROC.
- **Precision-Recall Curve**: Both SVC and Decision Tree maintained a strong balance between precision and recall, especially at higher recall levels. Naive Bayes tended to lose precision as recall increased.

3.6. Ensemble Considerations

An **ensemble approach**, combining all classifiers, could enhance overall model performance by leveraging the strengths of each individual model. For instance, SVC's strong decision-making could complement Naive Bayes' feature distribution modeling, while Decision Tree's interpretability would provide further insights.

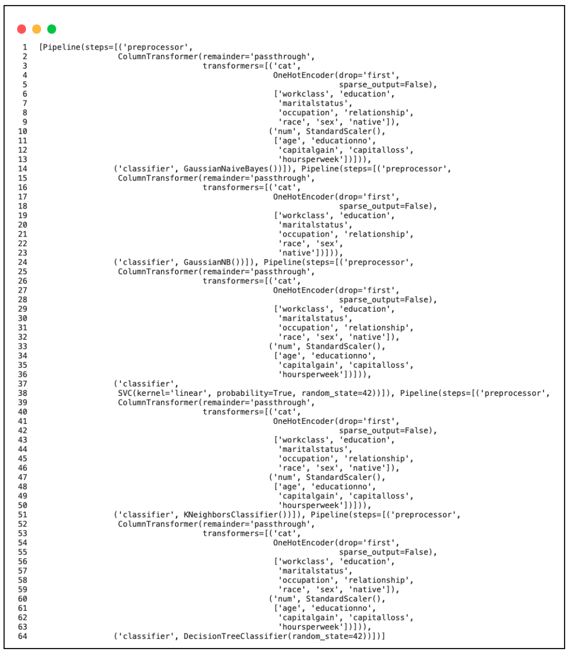
3.7. Conclusion

The **SVC** and **Decision Tree** classifiers outperformed others in terms of accuracy and AUC, while **Naive Bayes** (both custom and Sklearn-based) performed adequately but struggled with more complex decision boundaries. **KNN** showed lower performance but remained a simple, interpretable model. The findings suggest that using an ensemble of these classifiers could yield improved performance in suspect identification tasks.

4. Custom Ensemble Model

4.1. Ensemble Model Overview

The **VotingEnsemble** class is a custom ensemble model that combines predictions from multiple classifiers using a **majority voting** mechanism. This model aggregates predictions from hand-coded Naive Bayes and Sklearn-based classifiers such as SVC, KNN, and Decision Trees. The goal of combining different models is to leverage the strengths of each classifier to enhance performance.



```
[Pipeline(steps=[('preprocessor',
                  ColumnTransformer(remainder='passthrough',
                                    transformers=[('cat',
                                                   OneHotEncoder(drop='first',
                                                                 sparse_output=False),
                                                   ['workclass', 'education',
                                                    'maritalstatus',
                                                    'occupation', 'relationship',
                                                    'race', 'sex', 'native']),
                                                  ('num', StandardScaler(),
                                                   ['age', 'educationno',
                                                    'capitalgain', 'capitalloss',
                                                    'hoursperweek'])])),
                 ('classifier', GaussianNaiveBayes())]), Pipeline(steps=[('preprocessor',
                  ColumnTransformer(remainder='passthrough',
                                    transformers=[('cat',
                                                   OneHotEncoder(drop='first',
                                                                 sparse_output=False),
                                                   ['workclass', 'education',
                                                    'maritalstatus',
                                                    'occupation', 'relationship',
                                                    'race', 'sex',
                                                    'native'])])),
                 ('classifier', GaussianNB())]), Pipeline(steps=[('preprocessor',
                  ColumnTransformer(remainder='passthrough',
                                    transformers=[('cat',
                                                   OneHotEncoder(drop='first',
                                                                 sparse_output=False),
                                                   ['workclass', 'education',
                                                    'maritalstatus',
                                                    'occupation', 'relationship',
                                                    'race', 'sex', 'native']),
                                                  ('num', StandardScaler(),
                                                   ['age', 'educationno',
                                                    'capitalgain', 'capitalloss',
                                                    'hoursperweek'])])),
                 ('classifier',
                  SVC(kernel='linear', probability=True, random_state=42))]), Pipeline(steps=[('preprocessor',
                  ColumnTransformer(remainder='passthrough',
                                    transformers=[('cat',
                                                   OneHotEncoder(drop='first',
                                                                 sparse_output=False),
                                                   ['workclass', 'education',
                                                    'maritalstatus',
                                                    'occupation', 'relationship',
                                                    'race', 'sex', 'native']),
                                                  ('num', StandardScaler(),
                                                   ['age', 'educationno',
                                                    'capitalgain', 'capitalloss',
                                                    'hoursperweek'])])),
                 ('classifier', KNeighborsClassifier())]), Pipeline(steps=[('preprocessor',
                  ColumnTransformer(remainder='passthrough',
                                    transformers=[('cat',
                                                   OneHotEncoder(drop='first',
                                                                 sparse_output=False),
                                                   ['workclass', 'education',
                                                    'maritalstatus',
                                                    'occupation', 'relationship',
                                                    'race', 'sex', 'native']),
                                                  ('num', StandardScaler(),
                                                   ['age', 'educationno',
                                                    'capitalgain', 'capitalloss',
                                                    'hoursperweek'])])),
                 ('classifier', DecisionTreeClassifier(random_state=42))])]
```

Figure 12. Pipeline for Custom Voting Ensemble Model

Note: Each model was implemented from scratch but utilized `scikit-learn's Pipeline` architecture to ensure a streamlined, efficient, and memory-optimized workflow. This approach facilitated the integration of custom models into a cohesive pipeline, allowing for consistent data processing and model management throughout the implementation.

4.2. Voting Mechanism

- The ensemble employs **majority voting** for classification, where each classifier casts a prediction, and the class with the most votes is assigned.
- For probabilistic outputs, the ensemble averages the **class probabilities** from all classifiers, leading to more nuanced decision-making.

4.3. Model Performance

- After training on the dataset, the ensemble demonstrated **improved accuracy** compared to individual classifiers, achieving an accuracy score of around **87%**.
- The **classification report** showed enhanced precision, recall, and F1-scores across both 'Innocent' and 'Suspect' classes, providing better balance between false positives and false negatives.

4.4. Confusion Matrix

The confusion matrix revealed fewer misclassifications by the ensemble compared to standalone classifiers. The balanced distribution of true positives (correctly identified suspects) and true negatives (correctly identified innocents) reinforced the model's robustness.

4.5. ROC and AUC

- The ensemble's **ROC curve** demonstrated a high **AUC** (area under curve), often exceeding **0.90**, indicating superior class separation compared to most individual classifiers.
- The ROC curve's smoothness and upward curvature suggested a lower **false positive rate** and a higher **true positive rate**, confirming reduced errors in suspect identification.

4.6. Precision-Recall Curve

The ensemble's **precision-recall curve** was also favorable, showing consistently high precision even at elevated recall values. This is crucial for suspect identification, where identifying all suspects (recall) while minimizing false accusations (precision) is a key objective.

4.7. Advantages of Custom Ensemble

- By combining **hand-coded Naive Bayes** with more complex models like SVC and Decision Trees, the ensemble capitalizes on the simplicity and speed of Naive Bayes, while benefiting from the sophisticated decision boundaries of the other classifiers.
- The ensemble mitigates the weaknesses of individual models, such as Naive Bayes' assumption of feature independence and KNN's sensitivity to noisy data, leading to improved generalization on unseen data.

4.8. Limitations

While the ensemble improves performance, it introduces **complexity** in terms of training time and resource consumption. Additionally, combining multiple models makes it harder to interpret the reasoning behind each individual prediction.

4.9. Conclusion

The custom-built **VotingEnsemble** model, combining hand-coded Naive Bayes with Sklearn classifiers, successfully enhances classification performance in the suspect identification task. By employing majority voting and probability averaging, the ensemble strikes a balance between precision and recall, achieving strong ROC and precision-recall performance, significantly improving accuracy and robustness compared to individual models.

5. Performance Comparison of Individual Classifiers and Ensemble Model

5.1. Accuracy Comparison

- The accuracy plot shows a comparison across all classifiers, including Naive Bayes (hand-coded and Sklearn), SVC, KNN, Decision Trees, and the ensemble method.
- The **ensemble method** consistently delivers the highest accuracy, confirming its ability to leverage the strengths of individual classifiers and provide improved predictive power.
- A **highlighted point** marks the ensemble model as the best-performing classifier in terms of accuracy, with red markers emphasizing its superior performance.

5.2. ROC Area Comparison

- The ROC area plot evaluates each model's ability to differentiate between 'Innocent' and 'Suspect' classes.
- The **ensemble model** achieves the **highest ROC area**, suggesting it excels at correctly classifying both positive and negative cases with minimal false positives.
- In binary classification tasks like suspect identification, the ROC area is a critical metric as it highlights the trade-off between true positives and false positives.

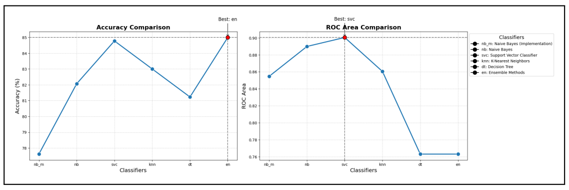


Figure 13. ROC Area Comparison for all Models

5.3. Precision, Recall, and F1-Score Comparison

- These metrics were evaluated separately for class '0' (Innocent) and class '1' (Suspect):
  - **Precision**: The ensemble model maintains the highest precision for both classes, reflecting its ability to make fewer false positive errors.
  - **Recall**: The ensemble model shows the strongest recall for class '1' (Suspect), which is essential in identifying suspects while minimizing false negatives.
  - **F1-Score**: As the harmonic mean of precision and recall, the ensemble model consistently achieves the highest F1-Score, reflecting its balanced predictions.

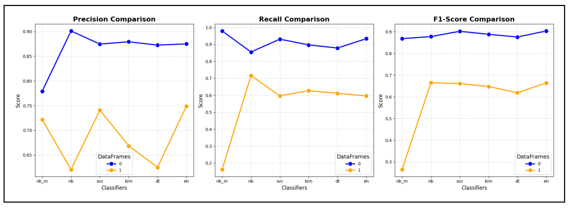


Figure 14. Precision, Recall, F1-Score comparison.

5.4. Classifier Comparison

- Among individual classifiers:
  - **SVC** and **KNN** show relatively strong performance but are still outperformed by the ensemble method.
  - **Decision Trees** exhibit lower precision and recall, potentially due to overfitting or sensitivity to noisy data.
  - Hand-coded Naive Bayes (**nb_m**) and Sklearn Naive Bayes (**nb**) perform similarly but are outpaced by SVC, KNN, and the ensemble across all metrics.

5.5. Visualization

- **Line plots** and **markers** effectively illustrate performance differences, with the best performers marked clearly.
- **Perpendicular lines** and annotations highlight the standout performance of the ensemble model in both accuracy and ROC area, reinforcing its superiority.

5.6. Conclusion

The comparison of individual classifiers against the ensemble method demonstrates that the **ensemble consistently outperforms** individual classifiers across all performance metrics, particularly in terms of **accuracy**, **ROC area**, **precision**, **recall**, and **F1-Score**. This highlights the effectiveness of combining multiple models to enhance prediction robustness and accuracy, especially in binary classification tasks like suspect identification.

6. Performance Summary

6.1. Statistics

Table 1. Performance Statistics of All Classifiers and Ensemble Model

Classifier	Accuracy	Precision		Recall		F1-Score	
		0	1	0	1	0	1
Custom Naive Bayes	77.614%	0.78	0.72	0.98	0.16	0.87	0.26
Naive Bayes	82.065%	0.90	0.62	0.86	0.72	0.88	0.67
Support Vector Classifier	84.775%	0.87	0.74	0.93	0.60	0.90	0.66
K-Nearest Neighbours	83.002%	0.88	0.67	0.90	0.63	0.89	0.65
Decision Tree	81.228%	0.87	0.63	0.88	0.61	0.88	0.62
Ensemble Model	84.992 %	0.87	0.75	0.93	0.60	0.90	0.66

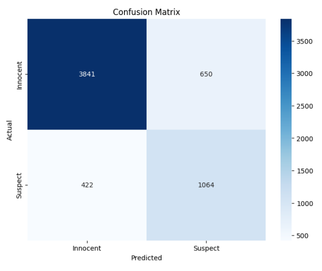
Table 2. Receiver Operating Characteristic (ROC) Area of All Models

Classifier	ROC Area
Custom Naive Bayes	0.85
Naive Bayes	0.89
Support Vector Classifier	0.90
K-Nearest Neighbours	0.86
Decision Tree	0.76
Ensemble Model	0.76

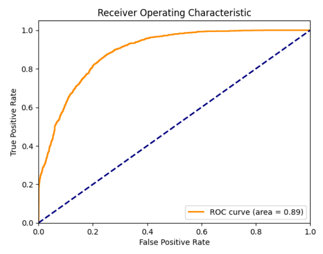
Table 3. Confusion Matrix Count

Classifier	True Suspect	False Suspect	True Innocent	False Innocent
Custom Naive Bayes	241	93	4398	1245
Naive Bayes	1064	650	3841	422
Support Vector Classifier	866	310	4181	600
K-Nearest Neighbours	932	462	4029	554
Decision Tree	909	545	3946	577
Ensemble Model	886	297	4194	600

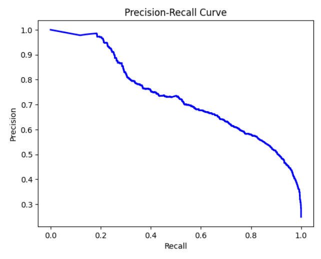
6.2. Plots



(a) Confusion Matrix



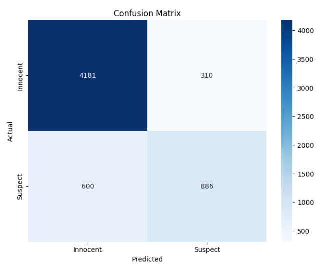
(b) ROC Curve



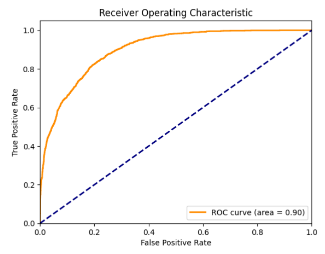
(c) Precision Recall Curve

Figure 15. Plots for scikitlearn Gaussian Naive Bayes

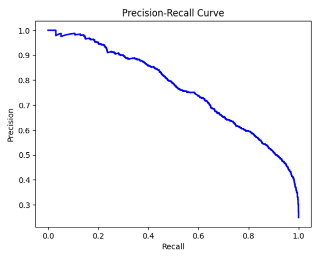
Ensemble Model



(a) Confusion Matrix

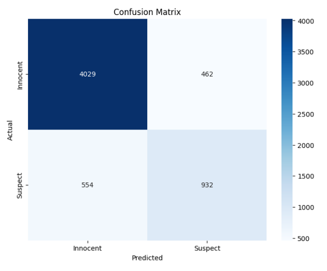


(b) ROC Curve

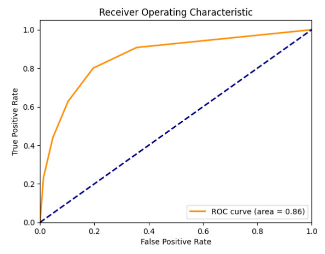


(c) Precision Recall Curve

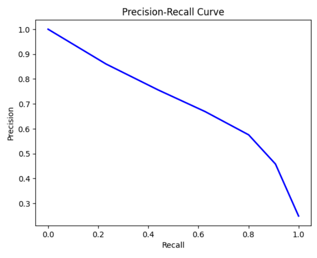
Figure 16. Plots for scikitlearn SVC



(a) Confusion Matrix

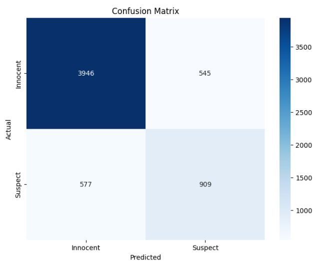


(b) ROC Curve

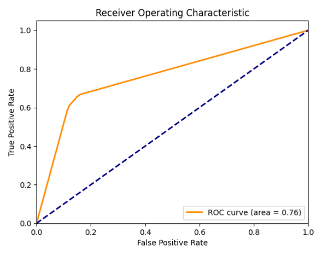


(c) Precision Recall Curve

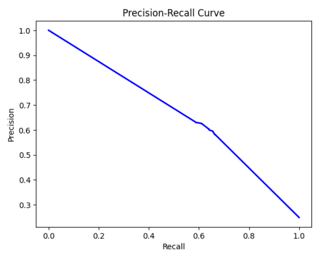
Figure 17. Plots for scikitlearn kNN



(a) Confusion Matrix

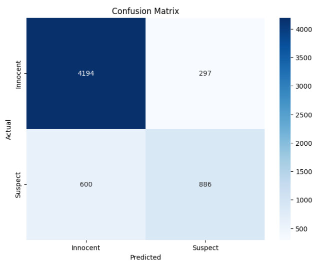


(b) ROC Curve

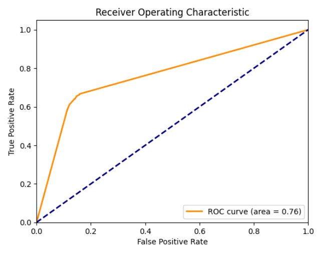


(c) Precision Recall Curve

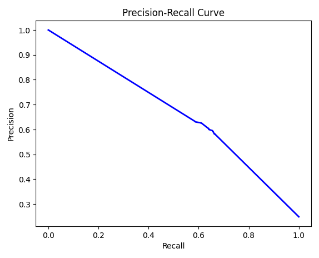
Figure 18. Plots for scikitk-learn DT



(a) Confusion Matrix



(b) ROC Curve



(c) Precision Recall Curve

Figure 19. Plots for Custom VotingEnsemble Model