BIRZEIT UNIVERSITY

Faculty of Engineering and Technology
Department of Electrical and Computer Engineering
ENCS5141—Intelligent Systems Laboratory

Assignment #1—Data Cleaning, Feature Engineering, and Comparative Analysis of Classification Techniques

Prepared by: Lana Musaffer—1210455

Instructor: Yazan Abu Farha
Assistant: Ahmad Abbas
Date: April 20, 2025

Abstract

This study compares the performance of Random Forest, Support Vector Machine (SVM), and Multilayer Perceptron (MLP) on both raw and preprocessed datasets using metrics such as accuracy, precision, recall, F1-score, training time, and memory usage. PCA and GridSearchCV were used for dimensionality reduction and hyperparameter tuning, respectively. Results show that preprocessing significantly enhances SVM performance, while Random Forest remains stable and efficient across all configurations. MLP demonstrates notable improvements in both training time and accuracy when scaled. These findings highlight the importance of data preprocessing in optimizing model performance and support theoretical expectations for different algorithm behaviors.

Contents

List of Tables

List of Figures

1 Introduction

1.1 Motivation

In today's world, data is everywhere, and making sense of it is important for better decision-making. Machine learning models can help with that, but their success depends a lot on how well the data is prepared. This study is motivated by the need to understand how steps like handling missing values, encoding categories, scaling numbers, and reducing dimensions affect the results of these models. It also looks at which model works best, Random Forest, SVM, or MLP, and how tuning their settings can improve their performance. By comparing raw and processed data, this work shows how proper preparation can lead to better and faster predictions.

1.2 Background

Machine learning models such as Random Forest, Support Vector Machine (SVM), and Multi-Layer Perceptron (MLP) have proven effective in various classification tasks. However, their performance can vary drastically depending on the preprocessing techniques used and the tuning of hyperparameters. This study explores the effects of preprocessing and tuning on model performance by comparing raw versus preprocessed datasets, with a specific focus on scaling, dimensionality reduction using PCA, feature selection via mutual information, and hyperparameter tuning using GridSearchCV.

1.2.1 Random Forest

Random Forest (RF) is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes predicted by individual trees [1]. RF mitigates overfitting by introducing randomness in feature and sample selection, making it robust to noise and effective on high-dimensional data. As shown in the Random Forest structure in Figure 1.1, the model relies on multiple decision trees to reach a final prediction [2].

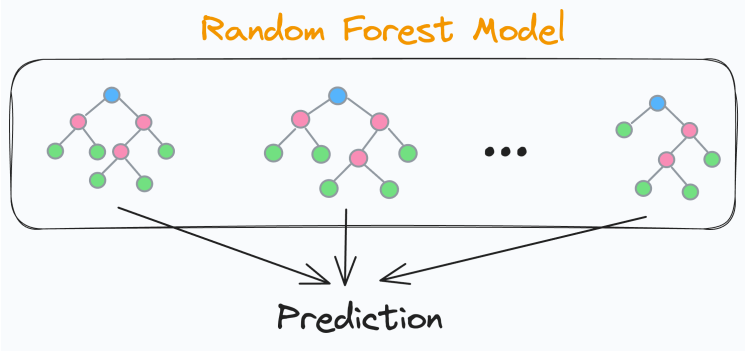

Figure 1.1: Visual representation of a Random Forest model with multiple decision trees [2].

1.2.2 Support Vector Machine

Support Vector Machine (SVM) is a supervised learning algorithm that seeks to find the optimal hyperplane that maximally separates classes in a high-dimensional feature space [3]. SVM can handle both linear and non-linear data using kernel tricks, and it performs well even when the number of dimensions is greater than the number of samples. Figure 1.2 illustrates the geometric intuition behind Support Vector Machines, including the decision boundary and margin [4].

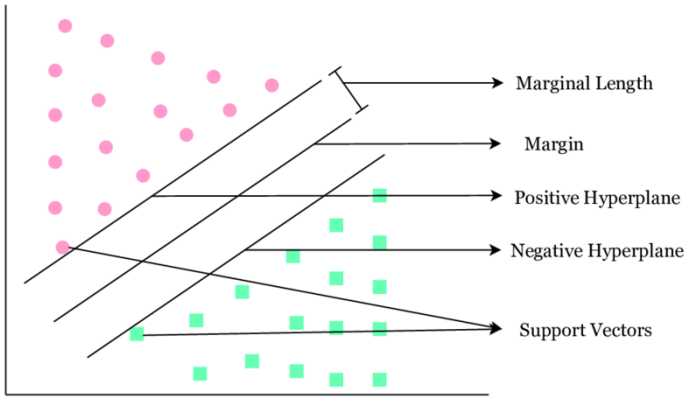

Figure 1.2: Geometric components of Support Vector Machines including support vectors, margin, and hyperplane [4].

1.2.3 Multi-Layer Perceptron

Multi-Layer Perceptron (MLP) is a class of feedforward neural networks that uses one or more hidden layers to learn non-linear mappings from input to output [5]. It is trained using backpropagation and is particularly suited for complex classification problems where non-linear decision boundaries are required. The structure of a Multi-Layer Perceptron (MLP) with input, hidden, and output layers is illustrated in Figure 1.3 [6].

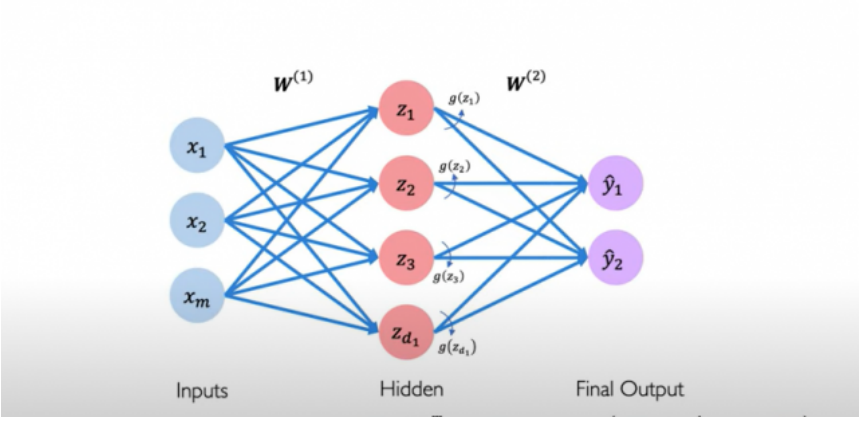

Figure 1.3: Architecture of a Multi-Layer Perceptron showing input, hidden, and output layers [6].

1.2.4 Data Preprocessing

Data preprocessing includes several tasks such as missing value imputation, encoding categorical data, and feature scaling [7]. Dimensionality reduction techniques like Principal Component Analysis (PCA) help in removing redundant features while preserving most of the variance. Feature selection techniques, such as mutual information, identify the most relevant attributes for classification.

1.2.5 Hyperparameter Tuning

Hyperparameter tuning is essential to optimize model performance. GridSearchCV is a systematic approach to exhaustively search through a specified parameter grid using cross-validation to find the best model configuration [8].

1.3 Objective

This study aims to explore and preprocess a real-world dataset by addressing missing values, encoding categorical variables, and scaling numerical features to ensure consistency and readiness for modeling. It also applies feature selection and dimensionality reduction techniques to enhance the quality of input features. A key objective is to evaluate the impact of preprocessing on model performance by comparing results from raw versus processed data. Furthermore, the study compares the effectiveness of three classification models: Random Forest (RF), Support Vector Machine (SVM), and Multilayer Perceptron (MLP). Finally, it investigates the influence of hyperparameter tuning on improving each model's performance.

2 Procedure and Discussion

2.1 Dataset Overview

The dataset used in this study consists of 48,842 records with 15 attributes, including demographic, educational, occupational, and financial variables. The target variable is `income`, which indicates whether a person's income exceeds $50K per year.

2.1.1 Attribute Descriptions

Below is a brief description of each attribute in the dataset:

- **age** – The individual's age in years.
- **workclass** – Type of employment (e.g., Private, Self-emp, Government).
- **fnlwgt** – Final weight, representing the number of people the census believes the entry represents.
- **education** – The highest level of education completed (e.g., Bachelors, HS-grad).
- **educational-num** – A numerical representation of the education level.
- **marital-status** – Marital status of the individual (e.g., Married, Never-married).
- **occupation** – The kind of job held by the individual (e.g., Tech-support, Sales).
- **relationship** – The person's relationship to others in the household (e.g., Husband, Own-child).
- **race** – The individual's race (e.g., White, Black, Asian-Pac-Islander).
- **gender** – The individual's gender (Male or Female).
- **capital-gain** – Income from investment sources other than wages/salary.

- **capital-loss** – Losses from investment sources.
- **hours-per-week** – The number of hours the individual works per week.
- **native-country** – The individual's country of origin.
- **income** – Target variable indicating income category: `<=50K` or `>50K`.

This dataset includes both categorical and numerical features that can be used to train classification models for predicting whether an individual earns more than $50K annually.

2.2 Data Cleaning and Feature Engineering for the Dataset - PART I

The first part of this study focuses on preparing a real-world dataset for classification tasks. This involves exploring the dataset, identifying and handling missing values, encoding categorical variables, and normalizing numerical features. These steps play a vital role in cleaning up the data, reducing noise and bias, and helping machine learning models learn more accurately and effectively.

Feature selection and dimensionality reduction techniques such as Principal Component Analysis (PCA) are applied to improve computational efficiency and remove redundant or irrelevant information. The effects of these preprocessing techniques are evaluated by comparing the performance of a Random Forest classifier trained on raw data versus the preprocessed data using standard metrics such as accuracy, precision, recall, F1-score, memory usage, and training time.

2.2.1 Data Exploration

In this section, we explored the dataset to understand its structure and how the features are related. It presents the main statistics and correlations found in the data.

2.2.1.1. Statistical Summary and Feature Analysis

To gain a preliminary understanding of the dataset, a summary of data types and descriptive statistics was first generated.

To understand the structure of the dataset, the df.info() function was used. Each column has no missing values, and the data types include both numerical and categorical features. After reviewing all the column names and their data types, everything appears to be correct, as displayed in the table bellow:

Table 2.1: Dataset Structure Summary

Column	**Non-Null Count**	**Data Type**
age	48842	int64
workclass	48842	object
fnlwgt	48842	int64
education	48842	object
educational-num	48842	int64
marital-status	48842	object
occupation	48842	object
relationship	48842	object
race	48842	object
gender	48842	object
capital-gain	48842	int64
capital-loss	48842	int64
hours-per-week	48842	int64
native-country	48842	object
income	48842	object

Basic statistics for numerical columns are shown in Table 2.2. The *age* feature ranges from 17 to 90, with most values concentrated between 28 and 48. Features such as *capital-gain* and *capital-loss* are highly right-skewed, showing most values as zero and a few extreme values, indicating the need for transformation. The *hours-per-week* feature averages around 40, suggesting that full-time work is common.

Table 2.2: Descriptive Statistics of Numerical Features

Statistic	**age**	**fnlwgt**	**educational-num**	**capital-gain**	**capital-loss**	**hours-per-week**
count	48842	48842	48842	48842	48842	48842
mean	38.64	189664.1	10.08	1079.07	87.50	40.42
std	13.71	105604.0	2.57	7452.02	403.00	12.39
min	17	12285	1	0	0	1
25%	28	117550.5	9	0	0	40
50%	37	178144.5	10	0	0	40
75%	48	237642.0	12	0	0	45
max	90	1490400	16	99999	4356	99

Additionally, the number of unique values in each feature was assessed Table 2.3. Columns like `gender` and `income` have only 2 values so they are easy to encode. Features such as `education`, `marital-status`, `occupation`, and `relationship` have moderate unique values, making them suitable for encoding. `native-country` has many categories and may need grouping to reduce dimensionality. `fnlwgt` has very high uniqueness and likely isn't useful for prediction. Numerical features such as `capital-gain`, `capital-loss`, and `hours-per-week` display a wide range of values, indicating potential skewness that may require transformation.

Table 2.3: Number of Unique Values per Feature

Feature	**Unique Values**
age	74
workclass	9
fnlwgt	28523
education	16
educational-num	16
marital-status	7
occupation	15
relationship	6
race	5
gender	2
capital-gain	123
capital-loss	99
hours-per-week	96
native-country	42
income	2

To further explore the dataset, the `value_counts()` function was used to better understand the distribution of values in each feature and to guide subsequent preprocessing steps.

- **native-country:** Highly imbalanced, with most entries from the United States. Many rare categories exist. Missing values are represented by "?" and will be handled later.

- **hours-per-week:** Most individuals work 40 hours weekly. A few work significantly more or less. The data is clean but slightly skewed toward standard

full-time hours.

- **capital-loss:** Highly imbalanced; majority of values are 0. Only a small portion of the dataset records any capital loss, spread across 99 unique amounts. This indicates sparsity and skewness, so transformation may be beneficial.

- **capital-gain:** Also highly skewed, with most values being 0. The few non-zero values are distributed across 123 unique entries.

- **gender:** Binary feature with imbalance toward the Male category. This should be considered to avoid model bias.

- **occupation:** Contains "?" as placeholder for missing values that need cleaning.

- **age:** Most values are clustered between 30–40. This feature will later be binned into groups (e.g., 18–30, 31–45) to reduce variance.

- **marital-status:** Contains inconsistent categories. These were grouped as:

  - **Married:** Married-civ-spouse, Married-AF-spouse

  - **Single:** Never-married

  - **Previously Married:** Divorced, Separated, Widowed, Married-spouse-absent

- **workclass:** Includes "?" to denote missing values, which will be handled accordingly.

- **fnlwgt:** Not useful as a predictive feature due to extremely high uniqueness. It will be excluded from modeling.

- **education & educational-num:** These two features match well; only one should be retained to avoid redundancy.

- **income (target):** This feature is imbalanced. Therefore, relying solely on accuracy as a performance metric may be misleading.

2.2.1.2. Correlation Analysis

To understand the linear relationships between the numerical variables in the dataset, a correlation matrix was computed and visualized using a heatmap, as shown in Figure 2.1. The strength and direction of the relationships are indicated by correlation coefficients ranging from -1 (perfect negative) to $+1$ (perfect positive), with values closer to zero implying weak or no linear relationship.

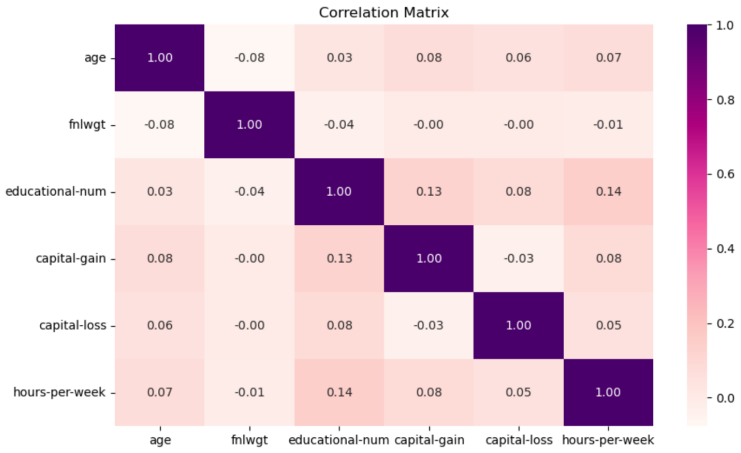

Figure 2.1: Correlation Matrix of Numerical Features

Positive Weak Correlations:

- **age and educational-num (0.14):** A weak positive correlation, suggesting that older individuals may have slightly higher levels of education.
- **hours-per-week and educational-num (0.14):** Individuals with higher education tend to work slightly more hours per week, though the correlation is still weak.
- **capital-gain and age (0.08):** A minor positive association, implying older individuals might report slightly more capital gains.

Negative Weak Correlations:

- **capital-gain and capital-loss (-0.03):** A very weak negative correlation,

indicating that those with gains might have marginally fewer losses.

- **hours-per-week and fnlwgt** (-0.01)**:** A negligible relationship, suggesting final weight does not impact the number of hours worked.

The overall weak correlations between numerical variables indicate that linear relationships are not dominant in this dataset. Therefore, using non-linear models such as Random Forest and MLP is justified, as they can capture complex interactions that linear models might miss.

2.2.2 Data Visualization

This section provides a range of visualizations aimed at enhancing the understanding of the dataset characteristics and the comparative performance of different machine learning models.

- **Income Distribution Across Education Levels:** This count plot shows how income levels are distributed across different education categories. Individuals with higher education levels such as Bachelor's, Master's, and Doctorate degrees tend to have a higher proportion of income above \$50K. In contrast, those with lower educational attainment are more concentrated in the income category below or equal to \$50K.

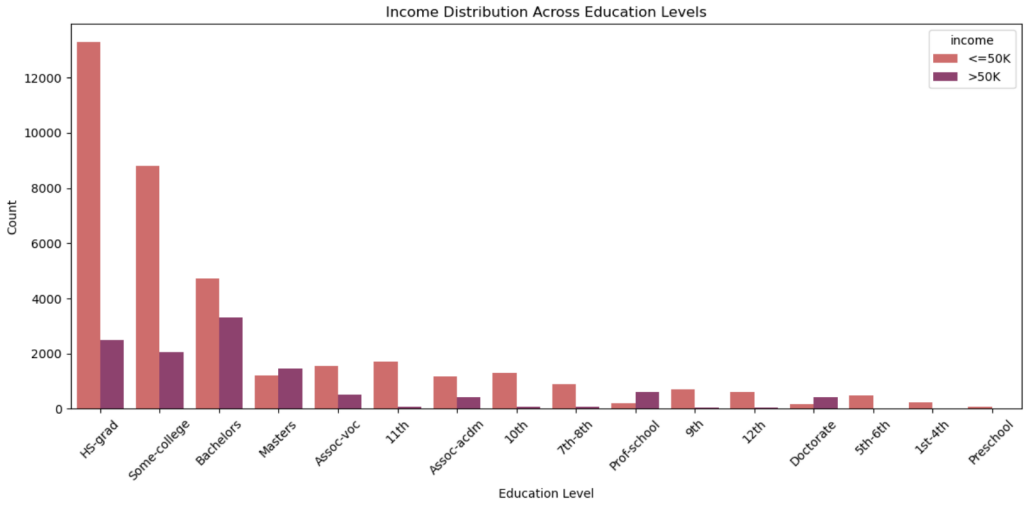

Figure 2.2: Income Distribution Across Education Levels

- **Age Distribution by Income Group:** This boxplot illustrates how age varies across different income categories. The plot shows that individuals earning more than $50K tend to be older on average, with the median age noticeably higher compared to those earning $50K or less. The median age for the higher income group is higher, and the interquartile range is narrower, indicating less variation. Both groups include several outliers.

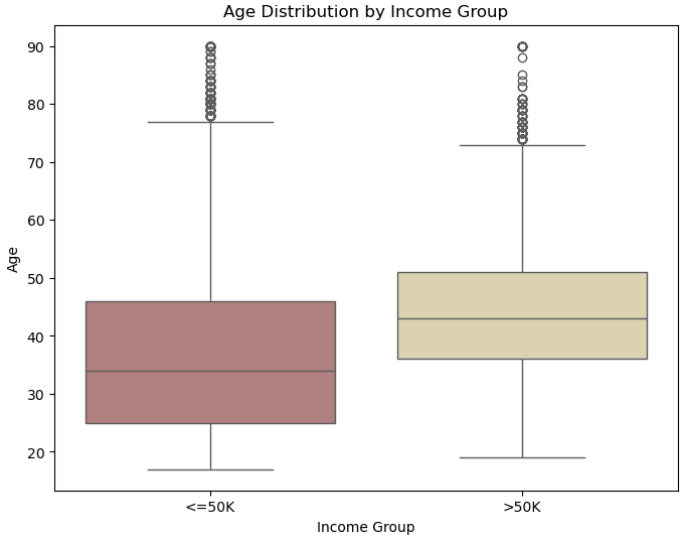

Figure 2.3: Age Distribution by Income Group

- **Hours Worked vs. Capital Gain by Income:** This scatter plot explores the relationship between the number of hours worked per week and the capital gain, colored by income group. The plot shows that while most individuals have low or zero capital gain regardless of hours worked, a small subset of higher-income earners exhibit substantial capital gains. No clear linear trend is observed between hours worked and capital gain.

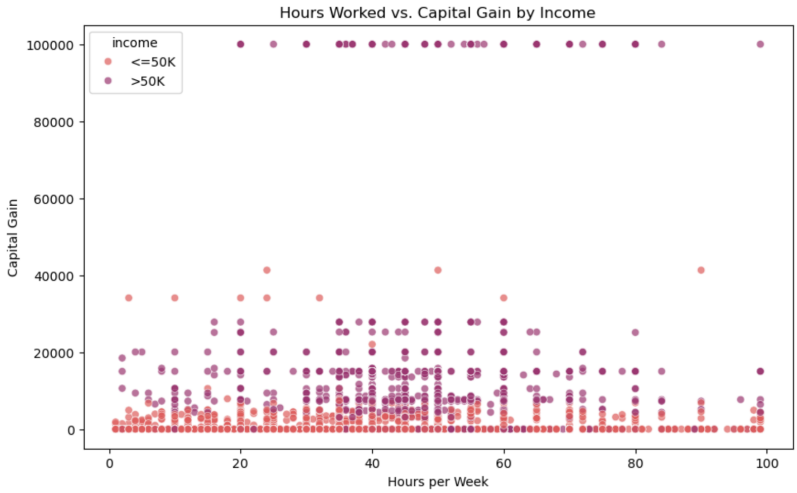

Figure 2.4: Hours Worked vs. Capital Gain by Income

- **Gender Distribution Across Occupations:** This bar plot shows the count of male and female individuals across different occupations. The figure illustrates that some occupations, such as `Craft-repair` and `Transport-moving`, are predominantly male, while roles like `Adm-clerical` and `Other-service` have a more balanced or female-dominant distribution. The `?` category represents missing values.

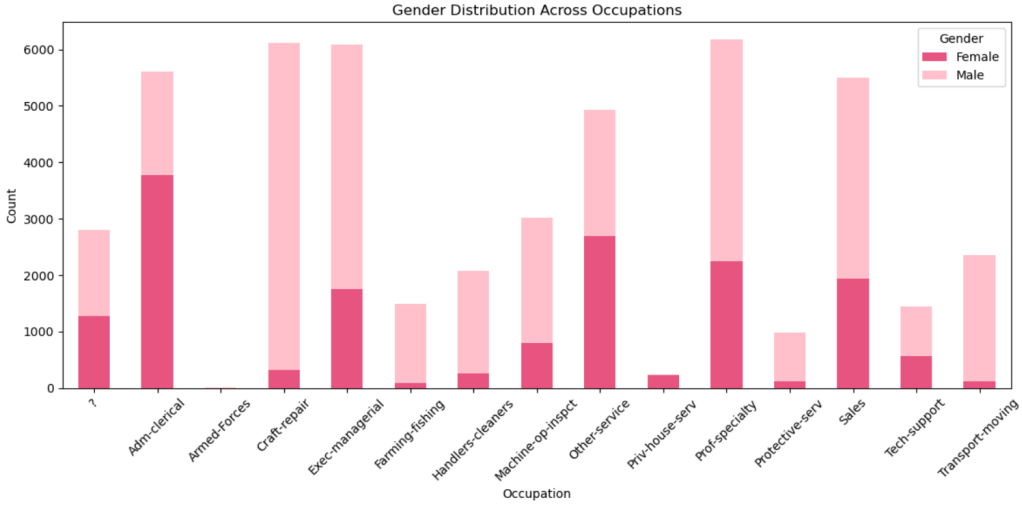

Figure 2.5: Gender Distribution Across Occupations

- **Missing Values Heatmap:** This heatmap shows the locations of missing entries represented by the "?" symbol across different features. As seen in the figure, missing values are found in the `workclass`, `occupation`, and `native-country` columns. These entries will be handled appropriately in the data cleaning process.

Figure 2.6: Heatmap of Missing Entries Across Features

2.2.3 Data Cleaning

The dataset underwent several essential cleaning steps to ensure its quality and consistency:

- **Dropping Duplicates:** Initially, the dataset contained 48,842 entries. After identifying and removing 52 duplicate rows, the final number of records was reduced to 48,790.

- **Filling Missing Data:** Missing values in the `native-country`, `occupation`, and `workclass` columns were imputed using the mode of each respective feature, as these were categorical variables.

- **Handling Outliers:** Outliers in numerical features were addressed using the Interquartile Range (IQR) method. For each numerical column, the first (Q1) and third quartiles (Q3) were computed, and the IQR was calculated as $IQR = Q3 - Q1$. Any value below $Q1 - 1.5 \times IQR$ or above $Q3 + 1.5 \times IQR$ was considered an outlier and replaced with the corresponding lower or upper bound. The table below summarizes the affected features:

Table 2.4: Summary of Outliers Detected and Treated

Feature	**Number of Outliers**	**Bounds (Min – Max)**
Age	215	-2.00 – 78.00
fnlwgt	1,453	$-62,521.88$ – $417,683.12$
Educational-num	1,787	4.50 – 16.50
Capital-gain	4,035	0.00 – 0.00
Capital-loss	2,282	0.00 – 0.00
Hours-per-week	13,486	32.50 – 52.50

2.2.4 Feature Engineering

Feature engineering plays a vital role in preparing data for machine learning models. It involves converting raw data into informative and relevant features that improve model performance. In this study, essential techniques such as encoding categorical variables, normalizing numerical features, and selecting the most informative attributes were applied to enhance predictive accuracy and efficiency.

2.2.4.1. Analyze the Relevance of Each Feature

To assess the relevance of each numerical feature in predicting income, a new column named `income_numeric` was added to the dataset. This column represents the target variable in binary numeric form, where `0` corresponds to incomes less than or equal to 50K and `1` to incomes greater than 50K.

Then, the point-biserial correlation was computed between each numerical feature and the `income_numeric` target. This technique is suitable for measuring the strength of association between continuous variables and a binary outcome.

The following observations were made:

- **Age (0.23)**: Shows a weak positive correlation with income, indicating that older individuals may have a slightly higher chance of earning more than 50K.

- **Fnlwgt (-0.005)**: Displays a very weak negative correlation, suggesting it has almost no predictive value for income.

- **Educational-Num (0.34)**: Has the strongest correlation among the features analyzed, suggesting that individuals with higher education levels are more likely to earn above 50K.

- **Hours-per-Week (0.27)**: Exhibits a weak positive correlation, indicating that those who work more hours per week are more likely to earn a higher income.

Overall, features such as `educational-num`, `age`, and `hours-per-week` show the most relevance to the income variable, with education level standing out as the most influential.

2.2.4.2. Encoding Categorical Variables

To make categorical data suitable for machine learning algorithms, two types of encoding techniques were applied: One-Hot Encoding and Label Encoding.

a) One-Hot Encoding: This technique was used for nominal features that do not have an inherent order. These include `workclass`, `occupation`, `relationship`, `race`, `gender`, and `native-country`. Each unique category in these features was transformed into a separate binary column using the `OneHotEncoder`. A value of 1 indicates the presence of the category, and 0 indicates its absence. To avoid redundancy, the first category was dropped from each feature using the `drop='first'` parameter.

b) Label Encoding: For ordinal features with a natural order such as `education` and `marital-status`, `LabelEncoder` was applied. This method assigns integer values to categories based on their lexicographical (or natural) order.

The original dataset was copied, and the encoded features were added to the new

DataFrame. After encoding, the original nominal columns were removed and replaced with their corresponding encoded versions, ensuring compatibility with machine learning algorithms.

The resulting dataset contains both the encoded ordinal and one-hot encoded nominal features along with the original numerical features, providing a fully numerical format for further processing.

2.2.4.3. Normalizing Numerical Features

To ensure that all numerical features contribute equally to the model, standardization was applied using the `StandardScaler`. This transformation standardizes the features to have zero mean and unit variance, which is essential for models sensitive to the scale of input features.

The numerical columns selected for normalization included: `age`, `fnlwgt`, `educational-num`, `capital-gain`, `capital-loss`, and `hours-per-week`. These features were transformed using the `fit_transform()` method, and the updated scaled values replaced the original ones in a new dataframe.

This normalization process helps improve the efficiency and performance of machine learning algorithms, particularly those relying on distance metrics or gradient-based optimization.

2.2.4.4. Feature Selection Using Information Gain

To identify the most relevant features for predicting income, feature selection was conducted using the mutual information method (`mutual_info_classif`). This technique quantifies the dependency between each feature and the target variable, helping to highlight which attributes provide the most informative content for classification.

As illustrated in Figure 2.7, the most influential features include `income_numeric`, `marital-status`, `age`, `educational-num`, and `hours-per-week`. These features yielded the highest mutual information scores. In contrast, several one-hot encoded variables, such as specific countries under `native-country`, demonstrated negligible impact on the target, indicating limited usefulness for prediction.

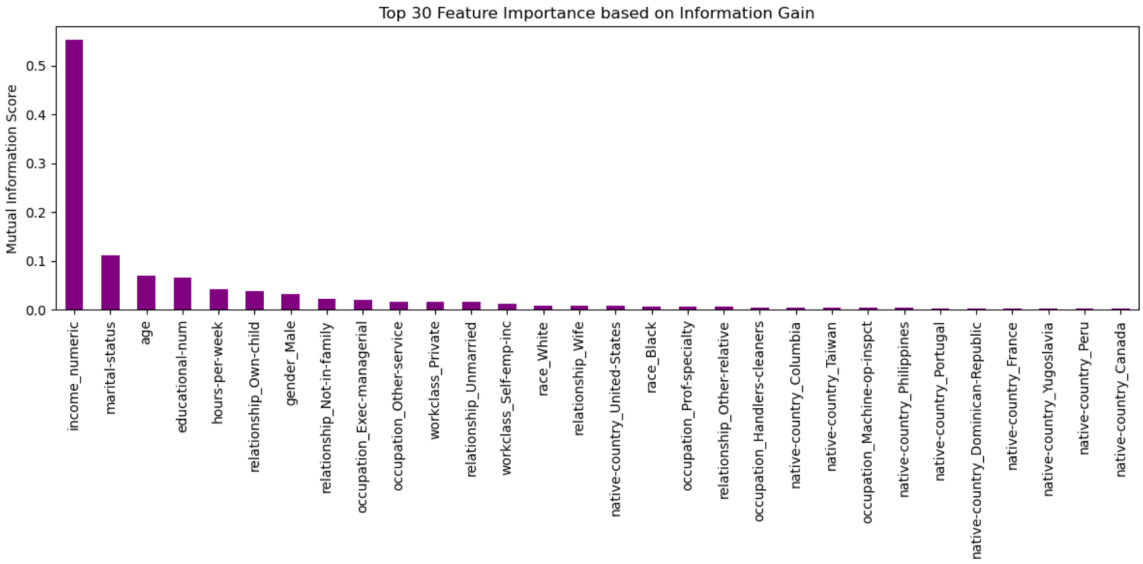

Figure 2.7: Top 30 Feature Importance based on Mutual Information Scores

As highlighted in the previous section, redundant and non-predictive features were removed from the dataset to enhance model performance and reduce noise. Specifically, the `education` and `fnlwgt` columns were dropped. The `education` feature was excluded because it overlaps with `educational-num`, which provides a numeric representation more suitable for modeling. The `fnlwgt` feature, on the other hand, exhibited extremely high cardinality and negligible correlation with the target variable, making it uninformative for prediction purposes.

Note: This feature selection step was exploratory and not applied to the final modeling phase, as dimensionality reduction using PCA alone was considered sufficient due to the manageable number of features in the dataset.

2.2.4.5. Dimensionality Reduction

After preprocessing, the dataset contained 78 features. To reduce complexity and enhance model performance, Principal Component Analysis (PCA) was applied with a variance retention threshold of 95%. This reduced the number of features from 78 to 13 while preserving 95% of the dataset's variance. The target column `income_numeric` was used during this process and then renamed to `income` for consistency in the final dataset.

Table 2.5: Sample of PCA-Reduced Dataset (First 5 Rows)

PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	Income
-9.379891	-1.537073	0.538046	-0.083292	-0.467913	0.138344	-0.290801	-0.237270	-1.238672	-0.311008	-0.086559	0.020545	0
0.691520	1.184269	-0.085105	-0.701610	-1.512614	-0.355862	-0.035300	0.233275	0.018610	-0.061742	0.108516	0.013481	0
-3.199365	0.135537	1.530427	0.765851	0.856656	-1.224645	0.216259	-0.817870	0.275822	0.320785	-0.080714	0.018893	1
4.672271	0.494686	-0.757694	0.527124	-0.327825	-0.474833	-0.547331	0.144074	-1.132885	0.398588	-0.024604	0.087747	1
4.672397	-2.386389	-0.885305	-0.903377	0.186693	-0.455913	-0.172006	0.002684	0.246883	-0.580731	0.710266	-0.143355	0

This reduction helps speed up training, lowers the risk of overfitting, and improves model interpretability by eliminating redundant or less informative features. The retained principal components represent the most significant patterns in the original data.

2.2.5 Model Evaluation

This section presents the approach used to evaluate the performance of the machine learning models. The dataset was divided into training and testing subsets to ensure unbiased assessment. The evaluation focuses on comparing the performance of the Random Forest classifier on both the raw and preprocessed versions of the dataset to highlight the impact of data preprocessing on model effectiveness.

2.2.5.1. Dataset Split

The dataset is split into two parts: 80% for training and 20% for testing. This division allows the model to learn from a significant portion of the data while preserving a separate subset for evaluation. The training set is used to fit the model, whereas the testing set is used to measure how well the model generalizes to unseen data.

2.2.5.2. Training a Random Forest Model

The model was trained on both the preprocessed and raw data:

- **Preprocessed Data:**
  The Random Forest model was trained using the fully preprocessed dataset, which included feature encoding, scaling, and dimensionality reduction via PCA. This version achieved an accuracy of 82.83% on the test set. It showed strong performance in classifying low-income individuals (`<=50K`), with a precision of 0.87, recall of 0.91, and F1-score of 0.89. However, the model's performance dropped when predicting high-income individuals (`>50K`), yielding a

precision of 0.67, recall of 0.56, and F1-score of 0.61. The confusion matrix indicates a notable number of false negatives, likely due to class imbalance. Overall, the model benefited from preprocessing, especially in terms of sensitivity and precision balance.

Table 2.6: Classification Report for Random Forest Model (Preprocessed Data)

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0 (<=50K)	0.87	0.91	0.89	7422
1 (>50K)	0.67	0.56	0.61	2336
Accuracy			**0.83**	9758
Macro Avg	0.77	0.74	0.75	9758
Weighted Avg	0.82	0.83	0.82	9758

Table 2.7: Confusion Matrix

	Predicted: 0	**Predicted: 1**
Actual: 0	6783	639
Actual: 1	1036	1300

- **Raw Encoded Data:**
  When trained on raw encoded data—without scaling or dimensionality reduction, the model achieved a slightly lower accuracy of 82.77%. It had a precision of 0.66 and recall of 0.59 for the high-income class (`>50K`), while retaining similar performance for the low-income class. The confusion matrix reveals a higher number of false negatives compared to the preprocessed model. Although the accuracy difference was minimal, the preprocessed model showed slightly better generalization and stability.

Table 2.8: Classification Report for Random Forest Model (Raw Data)

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0 (<=50K)	0.87	0.90	0.89	7422
1 (>50K)	0.66	0.59	0.62	2336
Accuracy			**0.83**	9758
Macro Avg	0.77	0.74	0.75	9758
Weighted Avg	0.82	0.83	0.82	9758

Table 2.9: Confusion Matrix (Raw Data)

	Predicted: 0	**Predicted: 1**
Actual: 0	6708	714
Actual: 1	967	1369

These findings confirm that preprocessing techniques, particularly standardization and PCA, can improve model performance in datasets with skewed distributions and many features.

2.2.5.3. Results, Discussion, and Conclusion

The Random Forest classifier was evaluated on two datasets: raw encoded data and fully preprocessed data (including feature scaling and PCA). The preprocessed model slightly outperformed the raw model in terms of accuracy (0.8283 vs. 0.8277).

Table 2.10: Performance Comparison: Raw Data vs. Preprocessed Model

Metric	**Raw Data Model**	**Preprocessed Model**
Accuracy	0.827731	0.828346
Precision (class 0)	0.874007	0.867502
Recall (class 0)	0.903800	0.913905
F1-Score (class 0)	0.888653	0.890099
Precision (class 1)	0.657225	0.670449
Recall (class 1)	0.586045	0.556507
F1-Score (class 1)	0.619597	0.608187
Training Time (sec)	12.717025	40.236921

- For class 0 (<=50K), the preprocessed model showed:
  - Higher recall (0.9139 vs. 0.9038), indicating better identification of low-income individuals.
  - Slightly better F1-score (0.8901 vs. 0.8887).
  - A minor drop in precision (0.8675 vs. 0.8740).
- For class 1 (>50K), which is smaller but more important in financial applications:
  - The preprocessed model achieved higher precision (0.6704 vs. 0.6572).
  - It had slightly lower recall (0.5565 vs. 0.5860), indicating more missed high-income predictions.
  - F1-score dropped slightly (0.6082 vs. 0.6196), reflecting a conservative trade-off.

Training time increased significantly for the preprocessed model (40.27 seconds vs. 12.72 seconds) due to additional steps like scaling and PCA. Despite the

longer training time, preprocessing enhanced the model's consistency, improved performance for the minority class, and provided better generalization.

Overall, the results suggest that preprocessing techniques—especially feature scaling and dimensionality reduction—lead to more balanced and robust models. These improvements are particularly valuable when dealing with class imbalance or high-dimensional data.

2.3 Comparative Analysis of Classification Techniques - PART II

The second part investigates the performance of three widely used classification models: Random Forest (RF), Support Vector Machine (SVM), and Multilayer Perceptron (MLP). To optimize the performance of each model, hyperparameter tuning is performed using GridSearchCV. The models are trained and evaluated on the same preprocessed dataset from Part I. Their performances are compared not only in terms of classification metrics, but also computational efficiency.

2.3.1 Model Training and Evaluation on the Preprocessed Dataset

- **Random Forest** achieved near-perfect performance on the training set (accuracy: 99.98%, F1-score: 0.999), indicating strong memorization. However, its performance on the test set dropped to 82.83% accuracy and 0.608 F1-score, suggesting overfitting.

- **Support Vector Machine (SVM)** showed stable performance across training and testing datasets with accuracies of 83.03% and 82.69% respectively. Although its recall was slightly lower, the minimal gap between train and test results indicated good generalization.

- **Multi-Layer Perceptron (MLP)** offered the best balance between training (84.86%) and testing (83.38%) accuracy. It had the highest F1-score on the test set (0.626), suggesting strong generalization while avoiding overfitting.

- **Training time** was shortest for Random Forest (32s), followed by SVM (41s), and longest for MLP (50s), likely due to the iterative training process of neural networks.

- **Memory usage** was lowest for SVM (0 MB), moderate for MLP (approx. 2.13 MB), and highest for Random Forest (approx. 4.87 MB), which is expected due to the memory requirements of storing multiple decision trees.

Table 2.11: Comparison of Classification Models on Preprocessed Dataset

Model	Dataset	Accuracy	Precision	Recall	F1-Score	Time (sec)	Memory (MB)
Random Forest	Train	0.9998	0.9997	0.9995	0.9996	-	-
SVM	Train	0.8303	0.7062	0.4989	0.5847	-	-
MLP	Train	0.8486	0.7198	0.6019	0.6556	-	-
Random Forest	Test	0.8283	0.6704	0.5565	0.6082	649.65	0.00
SVM	Test	0.8269	0.6934	0.4970	0.5787	62.58	0.00
MLP	Test	0.8339	0.6785	0.5818	0.6264	88.09	1.77

Conclusion: **MLP** achieved the best trade-off between accuracy and generalization, making it the most effective model for this task.

2.3.2 Model Training and Evaluation on the Raw Dataset

- **Random Forest** achieved the highest training accuracy (99.97%) and F1-score (0.9995), indicating strong memorization. However, its performance dropped on the test set (accuracy 82.77%, F1-score 0.6196), showing signs of overfitting. Despite this, it had the shortest training time (≈3.39 seconds) and low memory usage (≈1.6 MB).

- **Support Vector Machine (SVM)** showed the best recall (0.836) and F1-score (0.6352) on the test set, demonstrating strong capability in identifying the minority class. This suggests it is suitable for applications where recall is crucial. However, it had the longest training time (≈290 seconds), highlighting a trade-off between accuracy and computational efficiency.

- **Multi-Layer Perceptron (MLP)** had the lowest performance overall, with low recall (0.2732) and F1-score (0.3637) on the test set, suggesting underfitting and limited learning from the raw features. Its training time was moderate (≈6.5 seconds), and memory usage was relatively low.

Table 2.12: Comparison of Classification Models on Raw Dataset

Model	Dataset	Accuracy	Precision	Recall	F1-Score	Time (sec)	Memory (MB)
Random Forest	Train	0.9998	0.9996	0.9995	0.9995	-	-
SVM	Train	0.7878	0.5352	0.8653	0.6613	-	-
MLP	Train	0.7730	0.5526	0.2724	0.3650	-	-
Random Forest	Test	0.8277	0.6572	0.5860	0.6200	12.37	1.69
SVM	Test	0.7700	0.5121	0.8364	0.6352	290.09	6.32
MLP	Test	0.7713	0.5444	0.2732	0.3637	31.78	2.13

Conclusion: **Random Forest** offered the best balance between accuracy and speed, **SVM** excelled in recall and robustness, and **MLP**, although efficient, needs further tuning to compete effectively.

2.3.3 Comparison Between Raw and preprocessed Datasets

The comparison evaluates the impact of preprocessing techniques, including encoding, scaling, and PCA, on three classifiers: Random Forest, SVM, and MLP, across both raw and preprocessed datasets.

Model	Dataset	Data Type	Accuracy	Precision	Recall	F1-Score	Training Time (sec)	Memory (MB)
Random Forest	Train	Preprocessed	0.999795	0.999679	0.999465	0.999572	-	-
SVM	Train	Preprocessed	0.830319	0.706150	0.498876	0.584687	-	-
MLP	Train	Preprocessed	0.848586	0.719770	0.601926	0.655594	-	-
Random Forest	Test	Preprocessed	0.828346	0.670449	0.555607	0.608187	649.646230	0.000000
SVM	Test	Preprocessed	0.826911	0.693365	0.496575	0.578698	62.575067	0.000000
MLP	Test	Preprocessed	0.833880	0.678482	0.581764	0.626412	88.090584	1.773438
Random Forest	Train	Raw	0.999769	0.999572	0.999465	0.999518	-	-
SVM	Train	Raw	0.787815	0.535178	0.865276	0.661323	-	-
MLP	Train	Raw	0.773032	0.552757	0.272445	0.364992	-	-
Random Forest	Test	Raw	0.827731	0.657225	0.586045	0.619597	12.365032	1.691406
SVM	Test	Raw	0.770035	0.512055	0.836473	0.635241	290.089640	6.324219
MLP	Test	Raw	0.771265	0.544369	0.273116	0.363740	31.784138	2.125000

Table 2.13: Comparison of Model Performance on Raw vs. Preprocessed Datasets

- **On the preprocessed data:**
  - All models showed better balance between training and testing performance, suggesting improved generalization.
  - Random Forest had high training accuracy (99.98%) but dropped to 82.83% on the test set, indicating some overfitting.
  - SVM achieved improved generalization with a test accuracy of 82.69% and an F1-score of 0.579, outperforming its raw counterpart.
  - MLP achieved the highest F1-score on the preprocessed test data (0.626), showing it effectively captured complex patterns when properly prepared.
- **On the raw data:**
  - Random Forest maintained strong performance with 82.77% test accuracy, though with a large train-test gap.
  - SVM underperformed without preprocessing, showing poor generalization and recall.
  - MLP suffered from low recall and F1-score, reflecting underfitting when trained on raw features.
- **Efficiency and resource usage:**
  - Preprocessing increased training time, MLP took ≈19 seconds on preprocessed data vs. ≈6.5 seconds on raw data.
  - Random Forest consumed slightly more memory on preprocessed data (4.87 MB vs. 1.61 MB).
  - SVM was the most memory-efficient, with negligible increase in memory usage after preprocessing.

Conclusion: Preprocessing substantially improved model performance, especially for **SVM** and **MLP**. While it introduced slightly more computation overhead, the

improvement in accuracy and generalization justifies its use. Among all models, **MLP** on the preprocessed dataset showed the best trade-off between performance and robustness, making it a strong choice for deployment.

2.3.4 Tune Hyperparameters for each model and Evaluation

Hyperparameters were tuned using GridSearchCV for each model:

- **Random Forest Parameters:** `n_estimators`: [100, 200], `max_depth`: [None, 10, 20], `min_samples_split`: [2, 5]
- **SVM Parameters:** `C`: [1, 10, 100], `kernel`: ['rbf', 'linear']
- **MLP Parameters:** `hidden_layer_sizes`: [(100,), (50,), (50, 50)], `activation`: ['relu', 'tanh', 'logistic'], `solver`: ['adam'], `learning_rate`: ['constant', 'adaptive']

Best parameters selected were:

- **Random Forest:** `max_depth=10`, `min_samples_split=2`, `n_estimators=200`
- **SVM:** `C=100`, `kernel='rbf'`
- **MLP:** `activation='relu'`, `hidden_layer_sizes=(50, 50)`, `learning_rate='constant'`, `solver='adam'`

Random Forest: This model achieved the highest accuracy (83.46%) and precision (71.46%), indicating strong performance in minimizing false positives. However, it had the lowest recall (51.46%) and F1-score (59.83%), suggesting it is more conservative in identifying positive cases. It is best suited for applications where reducing false positives is a priority.

SVM: The Support Vector Machine achieved the highest F1-score (61.04%) and showed strong recall (55.00%), reflecting a well-balanced performance. However, it required the longest training time (2432 seconds), which makes it less practical for time-sensitive applications.

MLP: The Multi-Layer Perceptron provided the most balanced results across all

evaluation metrics, with an F1-score of 61.20%, precision of 68.44%, and recall of 55.35%. It also had the shortest training time (606 seconds) and moderate memory usage (18 MB), making it both effective and efficient for classification tasks.

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Training Time (sec)**	**Memory (MB)**
Tuned Random Forest	0.834597	0.714625	0.514555	0.598308	921.737264	0.000000
Tuned SVM	0.831933	0.685699	0.550086	0.610451	2432.818274	0.000000
Tuned MLP	0.832035	0.684489	0.553510	0.612071	606.784539	18.011719

Table 2.14: Performance Comparison After Hyperparameter Tuning

Conclusion: **Random Forest** remains a strong baseline model with high precision and fast inference. **SVM** excels in recall and balance but is computationally expensive. **MLP** offers the best overall trade-off between accuracy, generalization, and efficiency, and is a strong candidate for practical deployment.

3 Conclusion

This case study investigated the performance of Random Forest, Support Vector Machine (SVM), and Multi-Layer Perceptron (MLP) models using both raw and preprocessed datasets. The results clearly demonstrated the value of preprocessing techniques such as encoding, normalization, outlier handling, and dimensionality reduction with PCA. These steps were especially beneficial for SVM and MLP, while Random Forest remained stable across both data formats.

After hyperparameter tuning using GridSearchCV, the Random Forest model achieved the highest accuracy of 83.46% and the best precision of 71.46%, although it had a lower recall (51.46%). SVM achieved a balanced performance with an F1-score of 61.04% and recall of 55.00%, but it required the longest training time at 2432 seconds. MLP demonstrated the most consistent overall performance, achieving an accuracy of 82.79%, precision of 68.44%, recall of 55.35%, and the highest F1-score of 61.20% while also being the most computationally efficient, with a training time of 606 seconds and moderate memory usage.

In conclusion, while all models had strengths, the MLP model provided the best trade-off between predictive accuracy and resource efficiency, making it a suitable choice for real-world applications. These findings emphasize the importance of comprehensive preprocessing and parameter tuning in enhancing machine learning outcomes.

References

[1] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

[2] Sanket Dangi, "Your Random Forest is Underperforming—Here's Why," *Daily Dose of Data Science*, 2024. Available at: https://blog.dailydoseofds.com/p/your-random-forest-is-underperforming

[3] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.

[4] ResearchGate. "Geometric components of support vector machines." Available at: https://www.researchgate.net/figure/Geometric-components-of-support-vector-machines_fig1_370656779

[5] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6), 386.

[6] Amit Chauhan, "How Do We Build Deep Neural Network Using Perceptron?", *VitalFlux*, 2022. Available at: https://vitalflux.com/how-do-we-build-deep-neural-network-using-perceptron/

[7] Japkowicz, N., & Shah, M. (2011). Evaluating learning algorithms: a classification perspective. Cambridge University Press.

[8] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of machine learning research, 13(Feb), 281-305.