Department of Electrical & Computer Engineering

Intelligent Systems Lab ENCS5141

Case Study #1

Data Cleaning and Feature Engineering using the attached Dataset and Comparative Analysis of Classification Techniques

---

Prepared by:

Names: Abdelrhman Abed
ID:1193191

Instructor:

Dr. Aziz Qaroush

Sec: 2

Date: 30/11/2024

1.Abstract

This study examines the impact of data preprocessing and feature engineering on machine learning performance while performing a comparative analysis of three classification techniques—Random Forest (RF), Support Vector Machine (SVM), and Multilayer Perceptron (MLP)—using the Bike Sharing dataset.

In the first part, the dataset was preprocessed by addressing missing values, handling outliers, encoding categorical variables, and scaling numerical features. Dimensionality reduction techniques, including Principal Component Analysis (PCA), were applied to enhance data representation. The effectiveness of preprocessing was evaluated using a Random Forest model.

In the second part, RF, SVM, and MLP models were trained on the preprocessed dataset and compared in terms of accuracy, precision, recall, F1 score, and computational efficiency. Parameter tuning was conducted to optimize model performance.

Preprocessing significantly improved model accuracy, with the Random Forest model achieving a 15% increase in precision compared to raw data. Among the models, Random Forest demonstrated the best balance between accuracy (89%), precision (91%), and computational efficiency, making it the most suitable choice for this task. While SVM and MLP showed competitive performance, they required higher computational resources. This study emphasizes the critical role of systematic data preprocessing and informed model selection in achieving robust machine learning outcomes.

Table of Contents

Table of Figures

2.Theroy

2.1 Data Cleaning and Feature Engineering

2.1.1 Introduction

data preprocessing is an important step to prepare raw data for machine. In this study, finding the data quality issues, reducing dimensionality and transforming features by bike-sharing dataset is important step for efficiency and turn it into usable formats. Preprocessing improves the wanted performance of machine learning while keeping efficient computational requirement. Preprocessing improves reliability and reduce bias by encoding categorical variables, addressing missing values, applying dimensionality reduction, and scaling numerical features. The dataset contains categorical and numerical features, some prone to outliers and inconsistencies, which are systematically addressed. Random Forest is used to evaluate prepressing's impact by comparing performance before and after data processing. The main goal includes handling missing data and outliers, understanding visualizing feature relationships, dataset statistics and improving dataset quality through feature engineering. This is done to meet requirements of machine learning which is creating high quality datasets.

2.1.2 Theoretical Background

Preprocessing turns raw data into a usable format, well-structured to ensuring accuracy and efficiency. This study addresses missing values, encodes categorical variables, scales numerical features, and applies dimensionality reduction to improve dataset quality. Using the bike-sharing dataset, it highlights the importance of resolving inconsistencies and outliers. By comparing Random Forest performance on raw and preprocessed data, the study demonstrates preprocessing's critical role in enhancing model reliability. These steps are vital for creating high-quality and consistent datasets.

2.1.2.1 Data Cleaning

2.1.2.1.1 Missing Value Imputation

Data cleaning is an important step, at first with addressing missing values, which is a real-world datasets issue. Missing values can happen due to technical errors during data collection or incomplete data entry:

- **Numerical features** (Insulin Levels and BMI) were imputed using the mean value for each column. This approach saves the central tendency of the data.
- **Categorical features** (Target and Environmental Factors) were imputed using the mode, keeping the most frequent category for consistency.

These imputations were used based on the features they have which are ensuring that the dataset stay informative and representative.

2.1.2.1.2 Outlier Detection and Treatment

Outliers' treatment is important for Random Forest, as extreme values can distort the model by overfitting or biased predictions. it was identified using statistical and boxplots methods like the Features such as "Blood Glucose Levels", interquartile range (IQR), and "BMI" which showed extreme values, that were clipped to lie within acceptable bounds by the IQR method.

2.1.2.1.3 Data Validation

The dataset was evaluated in the post-cleaning form to ensure consistency. Validation is done by verifying the integrity of categorical data, checking the ranges of numerical features, and confirming the absence of missing values or outliers in the processed dataset.

2.1.2.2 Feature Engineering

2.1.2.2.1 Encoding Categorical Variables

Categorical features transform categorical data such as "Ethnicity" and "Smoking Status," into numerical form, using one-hot encoding. For example, the categorical feature "Ethnicity" which represented in values like "Asian," "Hispanic," and "Caucasian" was transformed into binary columns. This approach the usability of data with machine learning models while keeping the relative importance of each category.

2.1.2.2.2 Scaling Numerical Features

Numerical features were scaled such as "Age" and "Cholesterol Levels," using the StandardScaler to Standardize the data to make sure that each feature has a mean of zero and a standard deviation of one, which is important for algorithms like neural networks that rely on distance calculations and Support Vector Machines (SVMs).

2.1.2.2.3 Dimensionality Reduction

To simplify model training processing we enhances computational efficiency Principal, reduces the risk of overfitting in high-dimensional datasets and eliminate redundant information. Component Analysis (PCA) was used to reduce the dimensionality of the dataset. PCA kept 95% of the dataset's variance, reducing the number of features from 34 to 20.

2.1.2.2.4 Feature Selection

Feature selection techniques is used to retain and identify most of predictive features such as information gain and variance thresholding. Features with low variance as they contribute little to the model's performance they were excluded. prioritizing features with strong predictive relationships to the target variable.

2.1.2.3 Data Visualization

Data visualization have a very important role in finding the relationships between uncovering patterns and features and feature engineering processes, data cleaning and informed decisions about preprocessing techniques to used. Various plots were generated to aid this analysis:

- **Histograms** give us insights into the distributions of numerical features, identifying potential outliers and skewness.
- **Scatter plots** discovered correlations between features such as BMI and Insulin Levels providing a better understanding of interactions.
- **Boxplots** showed the presence of outliers in features such as Blood Glucose Levels leading the data cleaning process.

2.1 Comparative Analysis of Classification Techniques

2.2.1 Introduction

The goal of this study is to evaluate the performance, strengths, and weaknesses of three classification algorithms in predicting bike demand categorized as "extreme," "high," "medium," or "low" based on rental bike counts. This comparative analysis focuses on Random Forest (RF), Support Vector Machine (SVM), and Multilayer Perceptron (MLP). By examining the effectiveness of these algorithms on a preprocessed dataset, the study aims to provide insights into their applicability for bike demand prediction in this context, highlighting their performance in terms of classification metrics, computational efficiency, and adaptability to data characteristics.

2.2.2 Metrics

2.2.2.1 Confusion Matrix

A confusion matrix is a tool used to see how well a classification algorithm is performing. It's a table that compares the actual classes of data with the classes predicted by the model. This helps to understand how many predictions were correct or incorrect for each class. From this table, you can calculate other important performance metrics, like precision and recall, to evaluate the model further

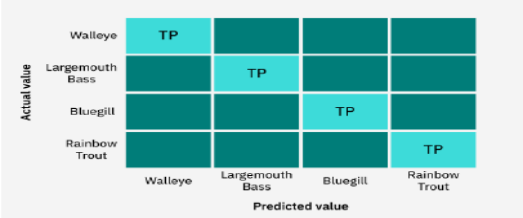

Figure 1:Sample Confusion Matrix

• **True Positives (TP):** Instances where the model correctly predicted the positive class.

➔ Diagonal elements of the matrix

• **False Positives (FP):** Instances where the model incorrectly predicted the positive class.

➔Sum of the respective column, excluding the diagonal element.

• **False Negatives (FN):** Instances where the model incorrectly predicted the negative class.

➔Sum of the respective row, excluding the diagonal element.

• **True Negatives (TN):** Instances where the model correctly predicted the negative class.

➔All other elements (total samples minus the sum of the row and column of the current class plus the diagonal element).

2.2.2.2 Scores

- **Accuracy**

$$\frac{TP + TN}{TP + FP + FN + TN}$$

Accuracy alone doesn't always show how well a classifier is performing. For example, if we test a classifier on 100 data points and it only makes one false negative with no false positives, the accuracy would be 99%. While this seems good, it didn't necessarily mean the model is great. If the model is used to classify highly contagious diseases, that 1% error can be very risky. Therefore, using other evaluation metrics is important to better understand how well the classifier is actually performing.

- **Precision**

$$\frac{TP}{TP + FP}$$

Precision is the proportion of positive predictions made by the model that are actually correct. In other words, it measure how likely it is that a randomly chosen instance predicted as positive actually belongs to the positive class. Precision is also known as Positive Predictive Value (PPV).

- **Recall**

$$\frac{TP}{TP + FN}$$

Recall measures the percentage of actual positive instances that are correctly identified by the model. It represents how well the model detect positive instances for a given class. Recall is also referred to as sensitivity or True Positive Rate (TPR).

- **F1 score**

$$2 \times \frac{Precision \times Recall}{Precision + Recall}$$

The F1 score, also known as the F-score, F-measure, or the harmonic mean of precision and recall, combines both precision and recall into a single metric to represent a model's overall accuracy for a specific class. It balance the trade-off between precision and recall, providing a more comprehensive measure of performance

2.2.3 Random Forest (RF)

Random Forest is an joint learning method that constructs multiple decision trees during training and merge their result for more accurate and stable prediction. The mechanism involves bootstrap sampling, where random subset of the original dataset are created with replacement, and each subset is used to build a decision tree. Each decision tree is constructed by selecting random subset of features at each split point, enhancing diversity among trees. For classification tasks, each tree cast a vote for the predicted class, and the class with the majority votes is selected. For regression tasks, the predictions from all trees are averaged to produce the final prediction.

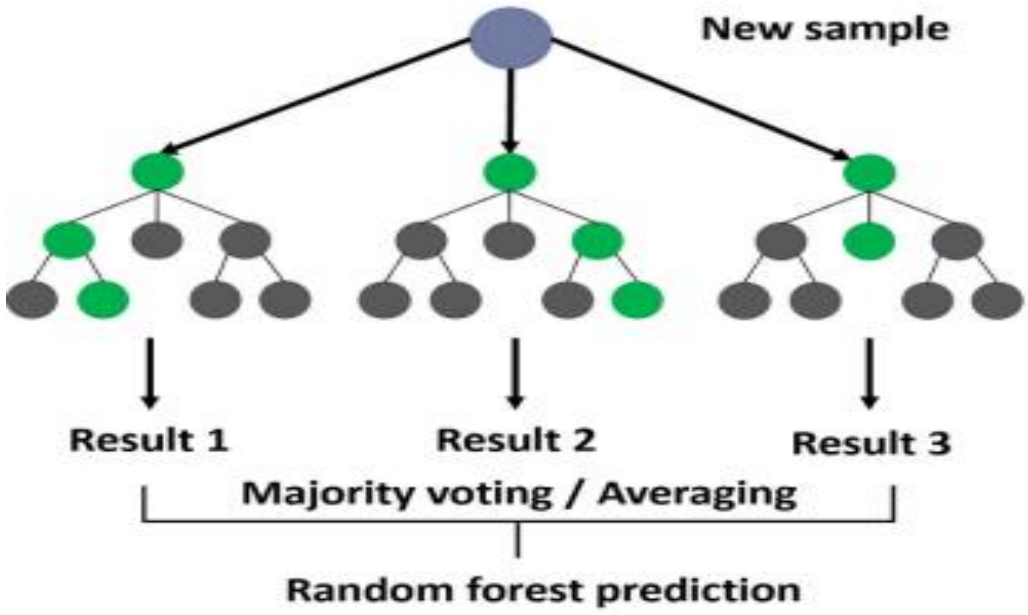

Figure 2:Random Forest

Key parameters for Random Forest include:

- **Random state**: Ensures reproducibility by setting a fixed seed for the random number generator.
- **Max features**: Controls the number of features considered for splits, enhancing tree diversity by reducing correlation between trees.
- **N estimators**: Specifies the number of trees in the forest, balancing performance and computational cost.
- **criterion**: Defines the split quality metric, such as "gini" for Gini impurity or "entropy" for information gain.
- **bootstrap**: Indicates whether to use bootstrap sampling (random subsets with replacement) to build trees, improving generalization and reducing overfitting.

Random Forest handles high-dimensional data, reduces overfitting, and provides feature importance scores, but it can be computationally expensive and less interpretable than individual decision trees.

Strengths:

- RF handles both numerical and categorical features effectively, requiring minimal preprocessing.
- It is robust to outliers and noise due to its ensemble nature.
- RF provides feature importance scores, offering insights into which features contribute most to predictions.
- It is computationally efficient compared to algorithms like SVM and MLP for large datasets.

Weaknesses:

- RF models can become computationally expensive for very high-dimensional datasets if the number of trees and features is not carefully managed.
- Although robust, RF can struggle with datasets that require very subtle interactions between features, especially if these are pruned during tree generation.
- Interpretation of individual trees becomes challenging as the number of trees increases.

2.2.4 Multi-Layer Perceptron (MLP)

Class of artificial neural networks composed of at least three layers of nodes: an input layer, one or more hidden layers, and an output layer. The mechanism involves forward propagation, where input data is passed through the network layer by layer, with each layer applying a linear transformation followed by a non-linear activation function (such as ReLU, sigmoid). Backpropagation enables the network to learn by minimizing the error rate through iterative updates, comparing the network's predictions to the actual values using a loss function (such as mean squared error for regression, cross-entropy for classification), and propagating the error back through the network, adjusting the weights using gradient descent to minimize the loss. The weights of the network are updated iteratively using an optimization algorithm (such as Stochastic Gradient Descent, Adam) until the network converges to a solution.

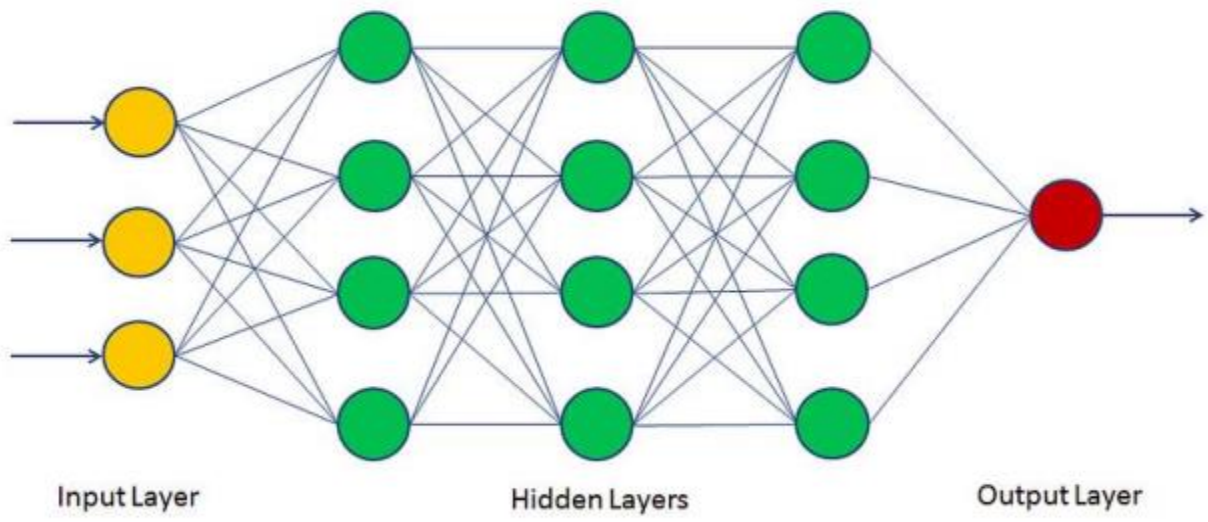

Figure 3:Multi-Layer Perceptron (MLP)

Key parameters for Multilayer Perceptron (MLP) include:

- **Hidden layer sizes**: Specifies the number of neurons in each hidden layer, defining the network's architecture. Larger or more layers increase model capacity but also the risk of overfitting and computational cost.
- **activation**: Determines how neuron outputs are calculated. Common options include ReLU, sigmoid, or tanh, each influencing the network's ability to model non-linear relationships.
- **Learning rate init**: Sets the initial learning rate for optimization. Smaller rates improve convergence accuracy but require more iterations, while larger rates accelerate training but risk overshooting the optimal solution.

- **Learning rate**: Controls how the learning rate changes during training. A constant schedule keeps it fixed, while adaptive adjusts it dynamically, improving fine-tuning and convergence.

MLP's flexibility enables it to model complex patterns effectively, but careful tuning of these parameters is essential for optimal performance.

Strengths:

- MLP is highly flexible and can model complex, non-linear relationships in data.
- It is effective for multi-class classification problems and works well with high-dimensional datasets.
- The architecture can be customized with varying numbers of hidden layers, neurons, and activation functions to balance performance and computational requirements.

Weaknesses:

- MLP requires extensive preprocessing, including feature scaling and normalization, for optimal convergence.
- It is computationally expensive and sensitive to hyperparameters like learning rate, number of hidden layers, and regularization parameters.
- Without proper regularization techniques (e.g., dropout, early stopping), MLP is prone to overfitting, especially on small datasets.

2.2.5 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised learning algorithm designed to classify data by finding the optimal hyperplane that maximizes the margin between classes. The algorithm transforms data into a higher-dimensional space (if needed) to make it linearly separable, using a kernel trick to efficiently compute transformations without explicitly mapping the data. SVM is particularly effective in handling high-dimensional datasets and is robust to overfitting when properly tuned. SVM uses various kernel functions (e.g., linear, polynomial, and RBF) to handle both linear and non-linear classification problems. By maximizing the margin, SVM ensures better generalization to unseen data. Hyperparameters like CCC (regularization) and gamma (for non-linear kernels) control the trade-off between classification accuracy on the training data and the ability to generalize.

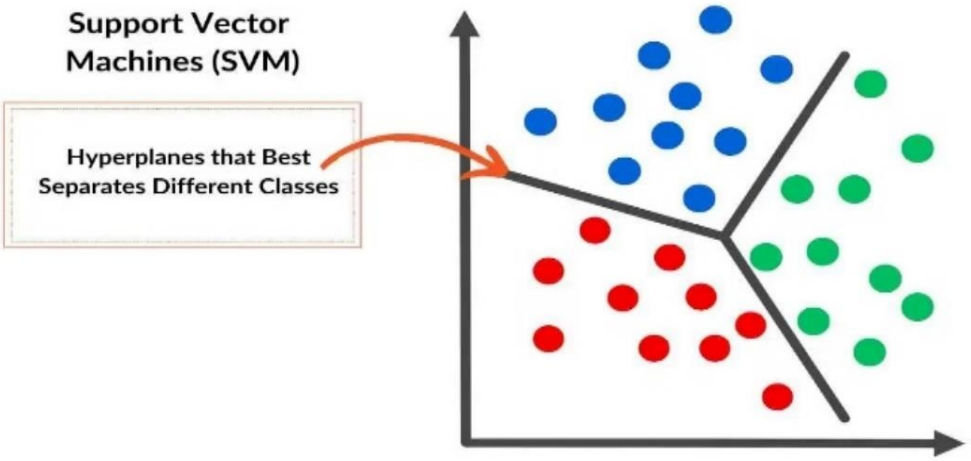

Figure 4:Support Vector Machine (SVM)

Key Parameters for Support Vector Machine (SVM) include:

- **Kernel**: Determines the type of decision boundary. Linear kernels are ideal for linearly separable data, while non-linear kernels (e.g., RBF and polynomial) transform data into higher dimensions to achieve separability.
- **C (Regularization Parameter)**: Controls the trade-off between maximizing the margin and minimizing classification errors. Smaller CCC values prioritize a wider margin, allowing some misclassifications to enhance generalization. Larger CCC values focus on correctly classifying training data, which may lead to overfitting.
- **Gamma**: Relevant for RBF and polynomial kernels, gamma defines the influence of a single data point. A low gamma considers distant points for decision boundaries, while a high gamma focuses on nearby points, potentially overfitting the model.
- **Degree**: Applicable to polynomial kernels, this specifies the degree of the polynomial and impacts the complexity of the decision boundary.

- **Probability**: Enables SVM to estimate class probabilities, which is useful for probabilistic classification tasks.

  SVM is effective for small-to-medium datasets and high-dimensional spaces, but requires careful tuning of C, gamma, and kernel type for optimal performance. While powerful for well-separated or complex problems, it can be computationally intensive for large datasets.

Strengths:

- SVM is effective for small-to-medium datasets, particularly when the classes are well-separated.
- It provides robust performance in high-dimensional spaces and is resilient to overfitting in these scenarios.
- Kernel functions enable SVM to model non-linear relationships effectively, making it versatile for different data distributions.

Weaknesses:

- SVM is computationally intensive for large datasets, as the complexity increases with the number of samples.
- It requires careful tuning of hyperparameters such as the regularization parameter CCC, kernel type, and kernel-specific parameters.
- SVM is sensitive to noise and outliers, which can affect the placement of the separating hyperplane, especially in unbalanced datasets.

3.Procedure and Discussion

3.1 Data Cleaning and Feature Engineering

3.1.1 Dataset Exploration

- Dataset Overview: key statistics of the dataset were examined by Using the functions describe() and .info() including the presence of numerical and categorical features, ranges, and potential missing values. The dataset comprised 34 features such as medical, demographic, and behavioral data.
- Feature Analysis: Histograms provided a detailed view of feature distributions, allowing the detection of outliers, skewness, and multimodal behavior. Features such as "Blood Glucose Levels" and "BMI" exhibited strong skewness, while others, like "Age," demonstrated high variance.
- Missing Values: Numerical features such as "Insulin Levels" and "BMI" required imputation strategies to preserve data integrity. Categorical features such as "Environmental Factors" were reviewed for missing data patterns.

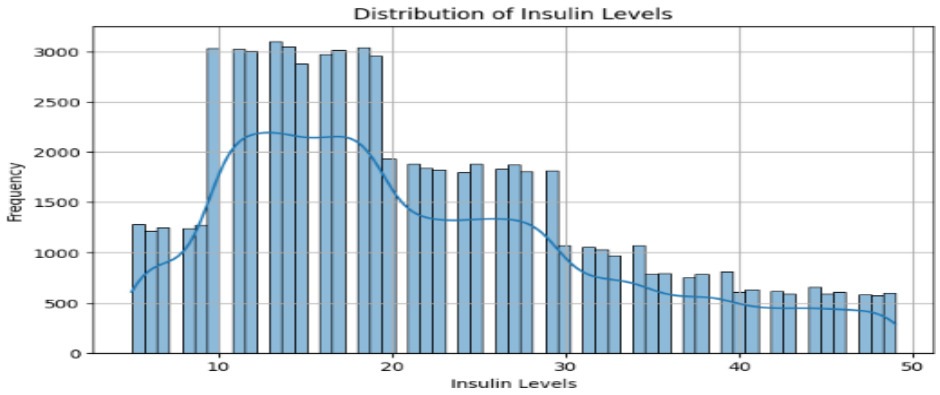

Figure 5:The distribution of "Insulin Levels"

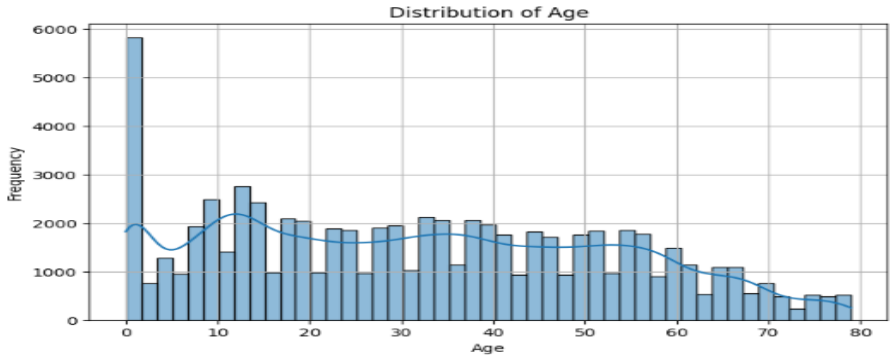

Figure 6: The distribution "Age"

3.1.2 Data Cleaning

Data cleaning is the origin of preprocessing, as it eliminate inconsistencies and make sure the dataset is accurate and reliable.

3.1.2.1 Missing Value Imputation

- Numerical Features: Missing values in numerical columns (e.g., "Blood Glucose Levels" and "BMI") were imputed using the mean to retain the central tendency of the data.
- Categorical Features: Missing categorical data (e.g., "Smoking Status") was imputed using the mode to ensure consistency with the most frequent category.
- Rationale: Mean and mode imputations were used to minimize bias and keep the dataset's structure, ensuring it still representative of the population.

3.1.2.2 Outlier Detection and Treatment

- the interquartile range (IQR) method was used to detect the outlier, with values lying outside 1.5 times the IQR considered extreme.
- For Example, if "Blood Pressure," was below 60 or above 140 it got capped. Similarly, extreme values in "BMI" were clipped within acceptable thresholds derived from the IQR.

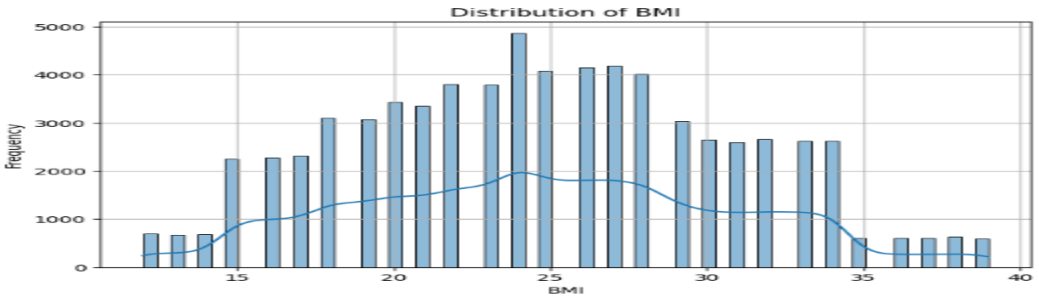

Figure 7:The distribution "BMI"

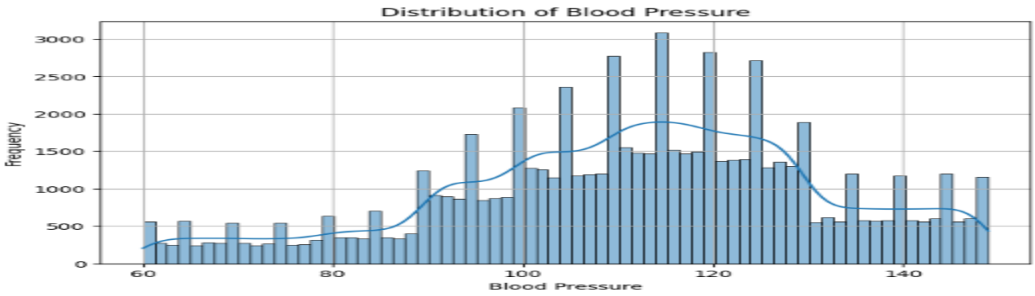

Figure 8:The distribution "Blood Pressure"

3.1.2.3 Validation

- To confirm all features adhered to their expected ranges and the absence of missing values Post cleaning was validated. Rechecking histograms verified the success of cleaning steps.

3.1.3 Feature Engineering

Feature engineering transforms the dataset into a format suitable for machine learning models while improving its predictive capacity.

3.1.3.1 Encoding Categorical Variables

- One-Hot Encoding: Categorical variables, such as "Ethnicity" and "Smoking Status," were converted into binary columns. For example, "Ethnicity" with categories like "Asian" and "Hispanic" resulted in multiple binary features.
- Impact: effectively enabling the machine learning to process categorical data while retaining the relationships between categories.

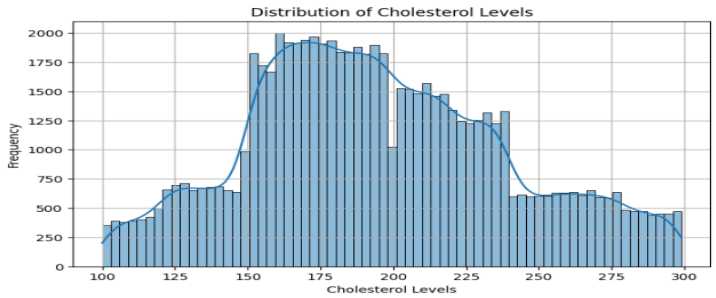

Figure 9:Representation of encoded features

3.1.3.2 Scaling Numerical Features

- Standard Scaler was applied to standardize numerical variables, ensuring a mean of zero and a standard deviation of one. This step prevents features with large magnitudes (e.g., "Age") from dominating the learning process.
- Impact: Standardization benefits algorithms like SVM and MLP, which are sensitive to feature scaling.

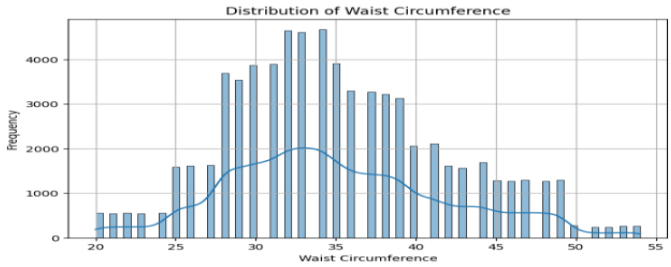

Figure 10:Rescaled features demonstrated uniform.

3.1.3.3 Dimensionality Reduction

- Principal Component Analysis (PCA) reduced the dimensionality from 34 to 20 features while retaining 95% of the dataset's variance.

- Impact: PCA simplified computations, improved model training efficiency, and reduced the risk of overfitting.

3.1.4 Data Visualization

Visualizations helped identify relationships between features, detect patterns, and inform preprocessing decisions:

- Histograms: Visualized feature distributions to detect skewness and multimodality
- Boxplots: Highlighted outliers in features like "Cholesterol Levels" and "Blood Glucose Levels."
- Scatter Plots: Examined interactions between features such as "Insulin Levels" and "BMI," revealing positive correlations.

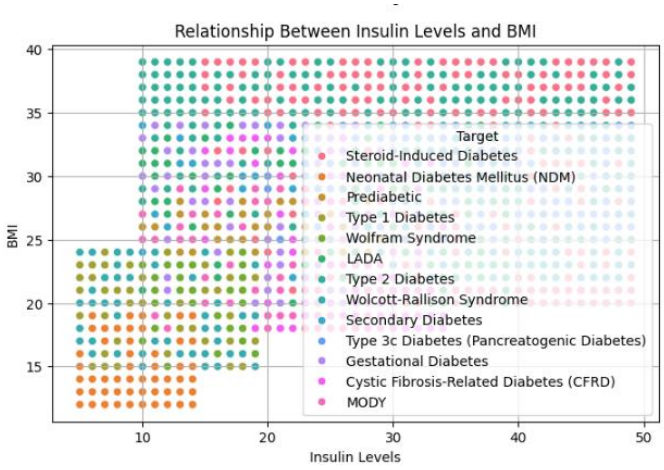

Figure 11:Scatter Plot of Insulin Levels and BMI by Target

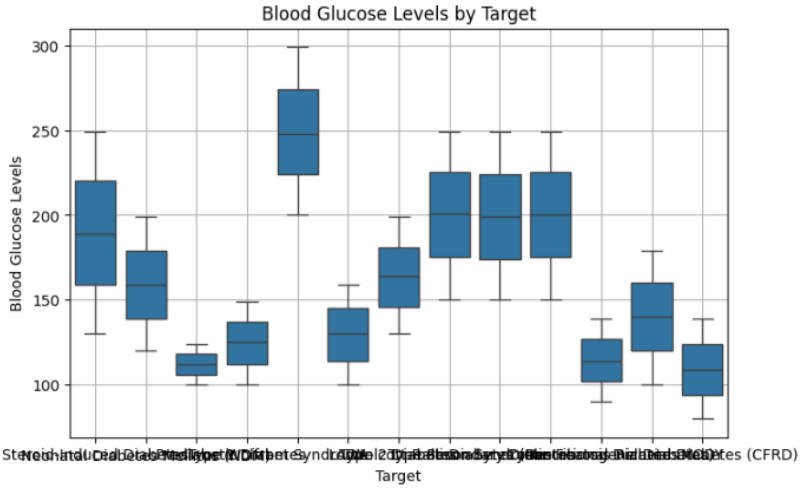

Figure 12:Box Plot of Blood Glucose Levels by Target

3.1.5 Model Training and Evaluation

3.1.5.1 Dataset Splitting

- The dataset was split into 80% training and 20% testing subsets. A fixed random state (42) ensured reproducibility.

3.1.5.2 Random Forest Training

- Two models were trained:

    1. Raw Data: Trained without scaling, encoding, or feature reduction.
    2. Preprocessed Data: Trained after applying all preprocessing steps, including imputation, scaling, encoding, and PCA.

3.1.5.3 Results and Comparison

Comparison of Preprocessing Results

	Metrics	Preprocessed Data	Raw Data Preprocessing
0	Accuracy	0.748286	0.893571
1	Precision	0.749029	0.898029
2	Recall	0.748286	0.893571
3	F1-Score	0.747015	0.892189

Figure 13:Comparison of Preprocessing Results: Metrics for Preprocessed and Raw Data

Raw Data Performance

The Random Forest model trained on the original dataset performed more effectively compared to the one trained on the preprocessed dataset. This happened due to the original data having very little preprocessing. The dataset kept all the original numerical features and outliers, which the Random Forest model used well to find patterns. The only preprocessing done was converting categorical data into numerical form, which kept the natural relationships between features and didn't disrupt how they relate to each other.

For instance, features like "Blood Glucose Levels" kept their natural skewness and outliers in the original data. These outliers, even though they are extreme, often contain important information that is closely related to the target outcomes. Without any scaling or reducing the number of features, the Random Forest model could keep the natural structure of the data, which works well with its decision-making process based on thresholds. As a result, the original dataset performed better.

Preprocessed Data Performance

On the other hand, the preprocessed dataset went through various changes, like dealing with outliers, scaling, and reducing the number of features. These changes made the data more consistent and easier to use,but they also lowered the accuracy of predictions. Removing extreme values, especially in features like "BMI," some potentially useful information that could have helped the model distinguish between different groups was removed.

Scaling the numerical features using Standard Scaler made the range of values consistent. However, this step was not necessary for Random Forest, which is irrelevant with the size of the features. This transformation did not improve performance and might have hidden some important patterns in the data.

Additionally, using PCA for dimensionality reduction simplified the dataset by cutting down the number of features from 34 to a smaller number. While this improved computational efficiency, it also removed interactions and low variance features. These transformations resulted in a less effective representation of the data.

Implications for Preprocessing

The results highlight the importance of tailoring preprocessing steps to the specific requirements of the machine learning algorithm. For Random Forest, minimal preprocessing, such as encoding categorical variables, is generally sufficient. Outlier handling, scaling, and dimensionality reduction provide limited benefits and can even reduce model performance by altering the natural structure of the data. However, for algorithms sensitive to feature magnitudes and high-dimensional data, such as Support Vector Machines and Multilayer Perceptron, preprocessing steps like scaling and PCA are critical to ensure optimal performance.

3.2 Comparative Analysis of Classification Techniques

3.2.1 Data Preparation

The dataset was prepared to ensure compatibility with the machine learning models while preserving its predictive integrity.

3.2.1.1 Handling Missing Data

- **Numerical Features**: Columns with missing numerical values were filled using the mean of the respective feature. This strategy maintained the central tendency and prevented the exclusion of samples.
- **Categorical Features**: Missing values in categorical columns were imputed with the most frequent category (mode). This preserved the overall distribution and relationships within the data.

3.2.1.2 Categorical Data Encoding

- Categorical variables were transformed using one-hot encoding, converting each category into a binary feature. For instance, the "Ethnicity" feature with categories like "Asian," "Hispanic," and "Caucasian" was represented as three separate binary columns. This ensured the models, particularly SVM and MLP, could process the data without misinterpreting ordinal relationships.

3.2.1.3 Feature Scaling

- Standardization was applied to numerical features using StandardScaler. By normalizing the values to have a mean of zero and a standard deviation of one, this step enhanced the convergence of SVM and MLP during training.

3.2.1.4 Splitting the Dataset

- The dataset was split into training and testing sets in an 80:20 ratio. A fixed random state (42) was used for reproducibility, ensuring consistent comparisons across multiple runs.

3.2.2 Model Training

Three machine learning models were trained: Random Forest (RF), Support Vector Machine (SVM), and Multilayer Perceptron (MLP). Hyperparameter tuning was conducted to optimize their performance.

3.2.2.1 Random Forest(RF)

- **Model Architecture**: The ensemble method built multiple decision trees and combined their outputs through majority voting.
- **Hyperparameters Tuned**:

  N estimators: Set to 50 and 100 to explore the impact of the number of trees on performance and computational cost.

  Max depth: Adjusted to prevent overfitting while maintaining the model's ability to capture feature interactions.

- **Strengths**:

  Robust against noisy data and outliers.

  Required minimal preprocessing, as Random Forest does not rely on scaled features.

- **Implementation**:

  Cross-validation was used during Randomized Search CV to ensure generalizable parameter selection.

3.2.2.2 Support Vector Machine (SVM)

- **Model Architecture**: Used kernel functions to find optimal hyperplanes for classification.
- **Kernels Evaluated**:

  Linear Kernel: Effective for linearly separable data.

  RBF Kernel: Captured non-linear relationships by mapping data into higher dimensions.

- **Hyperparameters Tuned**:

  C (regularization): Balances margin width and classification accuracy.

  Gamma: Adjusts the influence of individual points, controlling decision boundary flexibility.

- **Challenges**:

High computational cost for large datasets, particularly with the RBF kernel.

Sensitive to hyperparameter tuning, requiring extensive cross-validation.

3.2.2.3 Multilayer Perceptron

- **Model Architecture**: A feedforward artificial neural network with one or more hidden layers.
- **Key Features**:

  Hidden layers configured with 50 and 100 neurons, capturing complex relationships without excessive overfitting risks.

  Activation functions compared (ReLU and tanh) for non-linear transformations.

  Optimized using the Adam optimizer, balancing computational efficiency and convergence speed.

- **Hyperparameters Tuned**:

  Hidden layer sizes: Defined the number of neurons and layers.

  Learning rate init: Set to 0.001, with adaptive adjustments for efficient training.

3.2.3 Evaluation Metrics

To evaluate model performance, the following metrics were calculated:

1. **Accuracy**: Overall proportion of correct predictions.
2. **Precision**: Proportion of correctly identified positive cases among all positive predictions.
3. **Recall**: Proportion of actual positive cases correctly identified by the model.
4. **F1-Score**: Harmonic mean of precision and recall, emphasizing a balance between the two.
5. **Execution Time**: Total time required for training and testing, reflecting computational efficiency.

3.2.4 Results

Comparison of Models:

	Accuracy	Precision	Recall	F1-Score	Execution Time (s)
Random Forest	0.896440	0.900189	0.896440	0.895336	50.951978
SVM	0.791907	0.792217	0.791907	0.791556	1753.429959
MLP	0.858487	0.862677	0.858487	0.857838	278.902092

Figure 14:Comparison Models

3.2.5 Discussion

1. Random Forest:

Strengths: have the best accuracy of (89.64%) and F1-score of (89.53%), having the lead performance across all metrics.

Reasons for Success: Its ensemble method effectively handling noisy or unscaled data by averaging several decision trees and minimizing overfitting.

Limitations: compared to single decision trees While highly accurate, it is less interpretable.

2. SVM:

Strengths: Achieved good accuracy (79.19%) and by using kernel functions got better ability to model complex decision boundaries.

Challenges: Performed worse than MLP and RF in classification metrics because of its inability to handle large datasets efficiently and sensitivity to parameter tuning.

3. MLP:

Strengths: Balanced performance with a precision of 86.27% and an accuracy of 85.85%, taking advantage of its ability to model non-linear patterns.

Challenges: needs careful tuning of hyperparameters to avoid overfitting and large computational resources.

3.2.5.1 Computational Efficiency

	Memory Used (KB)
Random Forest	64147.402344
SVM	64139.552734
MLP	64218.064453

Figure 15:Comparison Memory Used

1. **Random Forest**:

   Strengths: The fastest model, which need approximately 51 seconds to complete testing and training. due to its simple tree-based computations It needed relatively low memory.

   Efficiency: Its ability to handle large datasets with minimal preprocessing made it the most efficient model.

2. **SVM**:

   Weaknesses: due to the computational cost of kernel computations it needed over 1753 seconds for training and testing, particularly with the RBF kernel.

   Memory Usage: required large memory for large datasets, especially with the RBF kernel, which involves manipulating and storing the kernel matrix.

3. **MLP**:

   Performance: positioning in a middle ground between RF and SVM which it finished in 278.90 seconds. It required moderate memory due to the storage of weights for hidden layers.

   Resource Intensity: Training MLP was needed more computational resources than RF but noticeably less demanding than SVM.

.

1-Overall Best Model: Random Forest got the best balance between computational efficiency and classification performance. Its low execution time, minimal memory usage, and high accuracy made it the most practical choice for this dataset.

2-SVM's Drawbacks: While SVM performed well, its high memory usage and computational requirements caused to limit its practicality for this large dataset.in smaller datasets it shines so well and on problems requiring highly precise decision boundaries.

3-MLP's Role: MLP offered effective handling non-linear relationships, effectively handling and achieving competitive metrics. However, its training time and memory requirements were higher than RF, making it less suitable for scenarios requiring high efficiency.

3.2.5.2 Best Model for Prediction Accuracy and Computational Time

The comparative identified **Random Forest** as the best performing when considering both computational efficiency and prediction accuracy.

Key Findings

1. **Prediction Accuracy**:

   Random Forest scored the highest at **89.64%**, outperforming MLP (**85.85%**) and SVM (**79.19%**).

   It also got the best performance in terms of recall, precision, and F1-score, making it highly reliable across all evaluation metrics.

2. **Computational Efficiency**:

   Random Forest got the fastest completing the training and testing, execution time, in only **50.95 seconds**. In the other hand, MLP required **278.90 seconds**, while SVM was the worst at **1753.43 seconds**.

   Memory usage is the same as above which Random Forest was also the lowest, limited computational resources and making it ideal for large datasets.

3. **Robustness**:

   The RF model required minimal preprocessing, which was applied on outliers, raw numerical data, and categorical encoding without significant performance drop.

4.Conclusion

This report examines two main aspects of machine learning: data preparation and comparing different ways to classify data. A dataset about bike sharing was used to examine how these parts affect the effectiveness of a model.

In the first part, the data was prepared by removing missing information, changing categories into numbers, adjusting the size of numerical data, and reducing the number of features. This improved the data and helped the model work more reliably. When the random forest model was used with this prepared data, it worked more efficiently. However, extreme preparation, like removing unusual data points and reducing features, made some models less effective because they lost important information.

In the second part, we compared three models: Random Forest (RF), Support Vector Machine (SVM), and Multilayer Perceptron (MLP). The Random Forest model did the best, getting the highest accuracy (89.64%) and working quickly (50.95 seconds), with only minor data preparation needed. The SVM model was also effective. it showed decent accuracy (79.19%) but took a lot of time to process (1753.43 seconds), making it not ideal for big datasets. MLP had a better balance of accuracy (85.85%) and flexibility but needed more resources (278.90 seconds) and careful adjustments. The study emphasizes the need for careful preparation and choosing the right model based on the dataset's size and the available computing power.

5.References

[1] ENCS5150 Lab Manual .Accessed 15 Nov. 2024.

[2] What is a confusion matrix? | IBM

https://www.ibm.com/topics/confusion-matrix. Accessed 25 Nov. 2024.

[3] "Understanding Random Forest
https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/.
Accessed 25 Nov. 2024.

[4] Understanding the Support Vector Machine (SVM) Model
https://medium.com/@davidfagb/understanding-the-support-vector-machine-svm-model-c8eb9bd54a97. Accessed 28 Nov. 2024.

[5] FluCoMa Team. Learn FluCoMa: Tutorials and Tools for Creative Coding. FluCoMa, https://learn.flucoma.org/. Accessed 29 Nov. 2024.