Machine Learning Classification Report

# Introduction

The objective of this assignment is to apply various machine learning models and techniques to classify numerical and categorical data. This task is crucial in the field of machine learning as it demonstrates the application of different algorithms and preprocessing techniques to achieve optimal performance.

# Dataset Overview and Preprocessing

The datasets used for this task include both numerical and categorical data. Preprocessing steps involved handling missing values, converting categorical features to numerical features, and splitting the data into training and testing sets.

# Implementation of Models

The team has implemented the following models:  
(a) **Logistic Regression**, ***Decision Tree***, and ***Random Forest*** for the classification problem and the choice for two models to compare the result between each model.  
Also use **Grid search** for find the optimal value for hyper parameter to tune and train the model

# Data Preprocessing

The data preprocessing involved handling missing values, converting categorical features to numerical features, and splitting the data into training and testing sets. The specific steps included:

- Loading the data

- Handling missing values

- Converting categorical features to numerical features using techniques like one-hot encoding

- Splitting the data into training and testing sets

### Loading the data

Upon initial examination of the dataset, we can observe the following:

* The target column appears to contain binary values (0 and 1).
* This suggests that we are dealing with a binary classification problem.
* The target variable likely represents a yes/no or true/false outcome.

Dataset Summary

Target Classes:

Class 0: 466 instances

Class 1: 459 instances

The dataset is well-balanced with nearly equal representation of both classes, which helps in building unbiased machine learning models.

A screenshot of a computer program

Description automatically generated

### Handling missing values

The team has performed data preprocessing tasks such as loading the data, handling missing values, and converting categorical features to numerical features.  
the main issue in the data can be summarized as following:

The missing a lot of values in **var4** (total row are **952** and missing **600** rows)  
A screenshot of a computer screen

Description automatically generated

And the disruption of the values for var4 as following.

A graph of a distribution of data

Description automatically generated with medium confidence

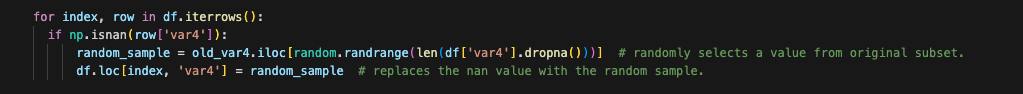
The histogram shows a roughly normal distribution for var4, with a slight positive skew. It's unimodal, peaking around 90-100. The data ranges from about 60 to 180, with no obvious outliers. The distribution is approximately bell-shaped, with gradually tapering tails and the highest frequency reaching about 60 occurrences. Overall, it represents a typical, mildly right-skewed distribution centered in the 90-100 range.

### Solution

The solution that team took after testing many solutions such as

* Filling the null values with mean
* Using interpolates
* KNNImputer
* IterativeImputer.

All those solutions led to having many outliers with affected the training model.

The last solution, team choose is to implement random sampling imputation for missing values in the 'var4' column of a Data Frame. It iterates through the Data Frame, identifies Nan values in 'var4', and replaces each with a randomly selected non-null value from the original dataset. This method preserves the overall distribution of 'var4' while filling in gaps, though it doesn't consider relationships between variables.  
  


# Converting categorical features to numerical features

The team selected one-hot encoding to convert categorical features into a numerical format, ensuring that the machine learning algorithms could process the data without implying any ordinal relationships. Var6 is a kind of {“yes” , “no”} category so it was converted to numerical feature.  
A black screen with blue and white text

Description automatically generated

# Dropping inconvenient features

Some features are not important in training the model live var3 and var7 and team chose to drop them. Var3 is country name and var7 is date and from the exploration of the data there are not important to the training.

A black screen with white text

Description automatically generated

# Rescaling for the Features

Rescaling numeric columns using MinMaxScaler ensures all features contribute equally to analysis and improves model performance by bringing features to a common scale. This step prevents numerical instabilities and enhances interpretability, making comparisons between features straightforward

A screen shot of a computer code

Description automatically generated

# Splitting the data into training and testing sets

Split the dataset into training and testing sets to evaluate model performance. Using 70% of the data for training and 30% for testing

# A screenshot of a computer program Description automatically generated

# Data Visualization

Data visualization was performed using appropriate plots to understand the features and the target variable. This included:

- Histogram plots

- Scatter plots

- Box plots

# Train the Model

Perform grid search to find optimal hyperparameters for a decision tree classifier

Train and evaluate the model using the best parameters, and calculate its accuracy

A screen shot of a computer program

Description automatically generated

Here's a breakdown of the process:

1. **Define the Parameter Grid**: The param\_grid dictionary specifies the range of hyperparameters to be tested during the grid search. These include max\_depth, min\_samples\_split, and min\_samples\_leaf.
2. **Create the Classifier**: A DecisionTreeClassifier object is instantiated, which will be used as the base model for training.
3. **Grid Search Initialization**: A GridSearchCV object is created, which takes the decision tree classifier and the parameter grid as inputs. Grid search systematically tests all possible combinations of the specified hyperparameters to find the optimal set.
4. **Fit the Grid Search**: The grid\_search.fit(X\_train, y\_train) method trains the decision tree model using the training data and performs cross-validation to evaluate each combination of hyperparameters.
5. **Retrieve the Best Parameters**: The best\_params attribute of the grid search object contains the combination of hyperparameters that resulted in the best cross-validation performance.
6. **Train the Optimized Model**: A new DecisionTreeClassifier is created using the optimal hyperparameters obtained from the grid search, and this model is trained on the entire training dataset.
7. **Evaluate the Model**: The optimized decision tree model is used to predict the target variable for the test dataset. The predictions are then compared to the actual target values to calculate the model's accuracy.
8. **Print Results**: The best hyperparameters and the accuracy of the optimized decision tree model on the test data are printed.

# Model Fine-tuning

The team has tuned the hyperparameters of the models to improve performance using techniques such as GridSearchCV as following.

A dictionary called param\_grid is created, specifying the hyperparameters to tune and the range of values for each. In this case, the hyperparameters include:

* **max\_depth**: The maximum depth of the tree, tested with values from 1 to 4.
* **min\_samples\_split**: The minimum number of samples required to split an internal node, tested with values [2, 3, 4, 5, 10].
* **min\_samples\_leaf**: The minimum number of samples required to be at a leaf node, tested with values [1, 2, 3, 4, 5, 6].

# Print the Decision tree here is a detailed summary of the decision tree: Total Levels: The tree has 5 levels, including the root node and the leaf nodes. Total Nodes: The tree contains 31 nodes in total, comprising both decision nodes and leaf nodes. Root Node: The root node splits based on the feature var1. Intermediate Nodes: The tree has multiple intermediate decision nodes that split based on various features (var1, var2, var5), refining the classification at each level. Leaf Nodes: The tree has 16 leaf nodes (terminal nodes) where the final class prediction is made. In summary, the tree has 5 levels with a total of 31 nodes, including 16 leaf nodes, and was generated after training a decision tree model to achieve optimal classification results.A diagram of a company structure Description automatically generated

# Performance Evaluation

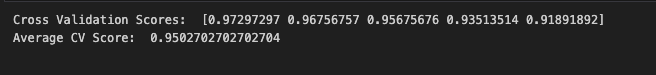
The models' performance was evaluated using metrics such as accuracy, confusion matrix, and cross-validation scores.

## Decision Tree

### Confusion Matrix

A screen shot of a black screen

Description automatically generated

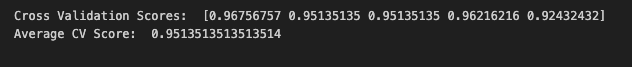
Cross-Validation Scores  


## Random Forest

### Confusion Matrix

A black screen with white text

Description automatically generated

Cross-Validation Scores  


# Summary

1. Exploring the Data:
   * Initial examination of the dataset revealed that the target column contains binary values (0 and 1), indicating a binary classification problem.
   * The dataset is well-balanced with 466 instances of Class 0 and 459 instances of Class 1.
2. Preprocessing:
   * The data preprocessing involved several steps to prepare the dataset for model training.
3. Handling Missing Values:
   * A significant number of missing values were found in the var4 column (600 out of 952 rows).
   * Various methods were tested to handle missing values, including filling with the mean, using interpolation, KNNImputer, and IterativeImputer.
   * The chosen solution was random sampling imputation for the var4 column to preserve the overall distribution without introducing outliers.
4. Converting Categorical Features:
   * Categorical features were converted to numerical format using one-hot encoding to ensure compatibility with machine learning algorithms.
   * The var6 column, which contained {“yes”, “no”} categories, was converted into numerical features.
5. Dropping Inconvenient Features:
   * Features that were not important for model training, such as var3 (country name) and var7 (date), were dropped from the dataset.
6. Rescaling Features:
   * Numeric columns were rescaled using MinMaxScaler to ensure all features contributed equally to the analysis and to improve model performance by bringing features to a common scale.
7. Splitting the Data:
   * The dataset was split into training and testing sets using a 70/30 split to evaluate model performance.
8. Data Visualization:
   * Various plots, such as histograms, scatter plots, and box plots, were used to understand the features and the target variable.
9. Model Training and Tuning:
   * Multiple models were implemented, including Logistic Regression, Decision Tree, and Random Forest.
   * Hyperparameters for the Decision Tree model were tuned using GridSearchCV to find the optimal values.
   * The parameter grid for the grid search included ranges for max\_depth, min\_samples\_split, and min\_samples\_leaf.
10. Training the Optimized Model:
    * The best hyperparameters obtained from the grid search were used to train a new DecisionTreeClassifier.
    * The model was then trained on the entire training dataset.
11. Model Evaluation:
    * The optimized Decision Tree model was evaluated using the test dataset.
    * Predictions were compared to actual target values to calculate the model's accuracy.
    * Performance metrics such as accuracy, confusion matrix, and cross-validation scores were used to assess the models' effectiveness.

The overall process demonstrates a comprehensive approach to data preprocessing, model training, and evaluation, leading to the development of effective machine learning models for binary classification.