**ROBOTICS & ARTIFICIAL INTELLIGENCE DEPARTMENT**



Data Mining

**Report**

# Submitted To: Dr. Tehreem Qasim

**Student Name:** **Syed Muhammad Mustafa**

# Reg. Number: 22108130

**ROBOTICS & ARTIFICIAL INTELLIGENCE DEPARTMENT**

***Instructions****: Copied or shown assignments will be marked zero. Late submissions are not entertained in any case.*

**Introduction:**

The provided code focuses on predicting the survival of passengers aboard the Titanic using machine learning techniques. The dataset used, "train.csv," contains various attributes such as age, sex, ticket class, and embarkation port, which are utilized to train classification models.

**Data Loading and Library Imports:**

The code begins by importing necessary libraries such as pandas for data manipulation and scikit-learn modules for machine learning tasks. It loads the dataset into a Pandas DataFrame for further processing.

**Exploratory Data Analysis:**

A brief exploratory data analysis is conducted to understand the dataset's structure and quality. This includes displaying column names and checking for null values. Additionally, concise summary information about the DataFrame, including column names, data types, and memory usage, is provided.

**Data Preprocessing:**

Preprocessing is crucial for preparing the data for model training. The code handles tasks such as dropping irrelevant columns ('Name', 'Ticket', 'Cabin'), handling missing values, encoding categorical variables ('Sex', 'Embarked') into numerical format, and splitting the data into training and testing sets.

**Model Training and Evaluation:**

Two classification models, Gaussian Naive Bayes and Decision Tree, are trained on the preprocessed data. Both models are fitted on the training set and evaluated using the testing set. The accuracy of each model is calculated using the accuracy\_score function from scikit-learn.

**Model Comparison and Performance Analysis:**

The performance of the Naive Bayes and Decision Tree classifiers is compared based on their accuracy scores. An analysis of their relative performance is provided, indicating which model performs better or if their performance is comparable.

**Conclusion:**

The code successfully demonstrates a workflow for predicting Titanic survival using machine learning techniques. It covers essential steps such as data loading, preprocessing, model training, evaluation, and performance analysis. Further improvements could involve exploring additional models, hyperparameter tuning, and more in-depth analysis of feature importance.

## Code:

|  |
| --- |
| import pandas as pd  from sklearn.model\_selection import train\_test\_split  from sklearn.naive\_bayes import GaussianNB  from sklearn.tree import DecisionTreeClassifier  from sklearn.metrics import accuracy\_score  df = pd.read\_csv("train.csv") |

This block of code imports necessary libraries for data manipulation and machine learning, then loads the dataset "train.csv" into a Pandas DataFrame.

## Code:

|  |
| --- |
| df.**columns** |

This block of code displays the column names of the DataFrame.

## Code:

|  |
| --- |
| print(f"There are {df.isnull().sum().sum()} null values in the dataset") |

This block calculates and prints the total number of null values in the dataset.

## Output:

|  |
| --- |
|  |

## Code:

|  |
| --- |
| df.info() |

This block displays concise summary information about the DataFrame, including the column names, data types, and memory usage.

## Output:

|  |
| --- |
|  |

## Code:

|  |
| --- |
| df.drop(['Name', 'Ticket', 'Cabin'], *axis*=1, *inplace*=True)  df.dropna(*inplace*=True)  df = pd.get\_dummies(df, *columns*=['Sex', 'Embarked'], *drop\_first*=True)  X = df.drop(['Survived'], *axis*=1)  y = df['Survived']  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, *test\_size*=0.2, *random\_state*=42)  print(f"There are {df.isnull().sum().sum()} null values in the dataset") |

This block preprocesses the data by dropping irrelevant columns, handling missing values, converting categorical variables into dummy/indicator variables, and splitting the data into training and testing sets.

## Output:

|  |
| --- |
|  |

## Code:

|  |
| --- |
| df.info() |

This block displays updated information about the DataFrame after preprocessing.

## Output:

|  |
| --- |
|  |

## Code:

|  |
| --- |
| naive\_bayes = GaussianNB()  naive\_bayes.fit(X\_train, y\_train)  y\_pred\_nb = naive\_bayes.predict(X\_test)  accuracy\_naive\_bayes = accuracy\_score(y\_test, y\_pred\_nb)  accuracy\_naive\_bayes\_percentage = accuracy\_naive\_bayes \* 100  accuracy\_naive\_bayes\_percentage = round(accuracy\_naive\_bayes\_percentage, 2) |

This block trains a Gaussian Naive Bayes classifier, predicts the target variable for the test set, calculates the accuracy, and converts it to a percentage.

## Code:

|  |
| --- |
| decision\_tree = DecisionTreeClassifier()  decision\_tree.fit(X\_train, y\_train)  y\_pred\_dt = decision\_tree.predict(X\_test)  accuracy\_decision\_tree = accuracy\_score(y\_test, y\_pred\_dt)  accuracy\_decision\_tree\_percentage = accuracy\_decision\_tree \* 100  accuracy\_decision\_tree\_percentage = round(accuracy\_decision\_tree\_percentage, 2) |

This block trains a Decision Tree classifier, predicts the target variable for the test set, calculates the accuracy, and converts it to a percentage.

## Code:

|  |
| --- |
| print(f"Naive Bayes Accuracy: {accuracy\_naive\_bayes\_percentage} %")  print(f"Decision Tree Accuracy: {accuracy\_decision\_tree\_percentage} %")  if accuracy\_decision\_tree\_percentage > accuracy\_naive\_bayes\_percentage:      if (accuracy\_decision\_tree\_percentage - accuracy\_naive\_bayes\_percentage) <= 10:          print("Decision Tree performs slighty better than Decision Tree on this dataset ")      else:          print("Decision Tree performs better than Decision Tree on this dataset ")  elif accuracy\_naive\_bayes\_percentage > accuracy\_decision\_tree\_percentage:      if (accuracy\_naive\_bayes\_percentage - accuracy\_decision\_tree\_percentage) <= 10:          print("Naive Bayes performs slighty better than Decision Tree on this dataset ")      else:          print("Naive Bayes performs better than Decision Tree on this dataset ") |

This block compares the accuracies of the Naive Bayes and Decision Tree classifiers and provides an analysis based on their performance. If one model performs significantly better than the other (with a difference of more than 10%), it indicates which model performs better. Otherwise, it states that they perform similarly.

## Output:

|  |
| --- |
|  |