# Data Science and IA

Mini project – individual (50 %)

**GitHub Repository :**
Link : GitHub Repository

**Name** : Martin JOUBERT DE LA MOTTE
**Student number** : 73154

# Predicting Titanic Passenger Survival with Logistic Regression

## Introduction

The goal of this project is to develop a predictive model for passenger survival on the Titanic using machine learning techniques. The dataset comprises information about passengers, including features such as age, sex, class, and embarkation port. In this report, we detail the algorithm used, steps taken to improve model performance, and present the final model's training and evaluation results.

## Algorithm Used

We employed a Logistic Regression algorithm for this survival prediction task. Logistic Regression is suitable for binary classification problems, making it an appropriate choice for predicting survival (1) or death (0) on the Titanic. It models the relationship between the dependent variable and one or more independent variables using the logistic function, mapping the predicted values to the range [0, 1] and facilitating interpretation as probabilities.

## Data Preprocessing

### Duplicate Removal

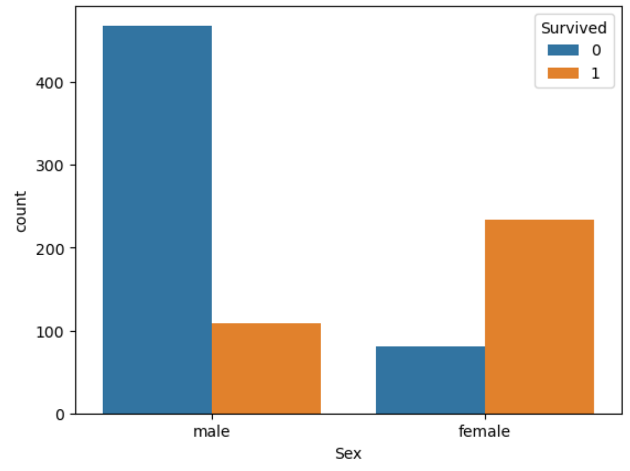
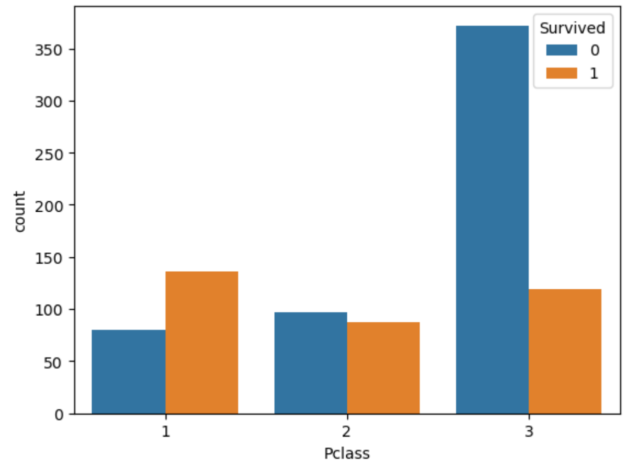
We started by identifying and removing duplicate records in both the training and test datasets to ensure data integrity. In this database, fortunately, there were not any.

### Handling Null Values

Dealing with missing values is crucial for model performance. We replaced missing age values with the mean, embarked values with the mode, and dropped the 'Cabin' column due to a high number of nulls.

## Exploratory Data Analysis (EDA)

Visualizations were utilized to gain insights into the relationships between various features and survival. Key visualizations included:

| The impact of the passenger sex on survival | Survival rates across different passenger classes |
|---|---|
|  |  |

As we can see, both sex and Pclass are most likely correlated with survival.

## Model Training and Hyperparameter Tuning

Data was already split between test and train data so the Logistic Regression were directly applied to the training set. To optimize model performance, a hyperparameter grid search was conducted using GridSearchCV. This iterative process helped identify the best hyperparameters for the Logistic Regression model. It is a quite simple function that test every parameter given and uses the one with the best score. In this program is tested five parameters which are:

- Penalty: Specify the norm of the penalty. **Testing l1 and l2**
- C: Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization. **Testing 0.001, 0.01, 0.1, 1, 10, 100, 1000**
- fit_intercept: Specifies if a constant (a.k.a. bias or intercept) should be added to the decision function. **Testing True, False**
- solver: Algorithm to use in the optimization problem. **Testing newton-cg, lbfgs, liblinear, sag, saga**
- max_iter: Maximum number of iterations taken for the solvers to converge. **Testing 50, 100, 200, 500**

## Model Evaluation

### Metrics Used

We evaluated model performance using accuracy scores, confusion matrices, and classification reports. These metrics provided a comprehensive understanding of the model's predictive capabilities.

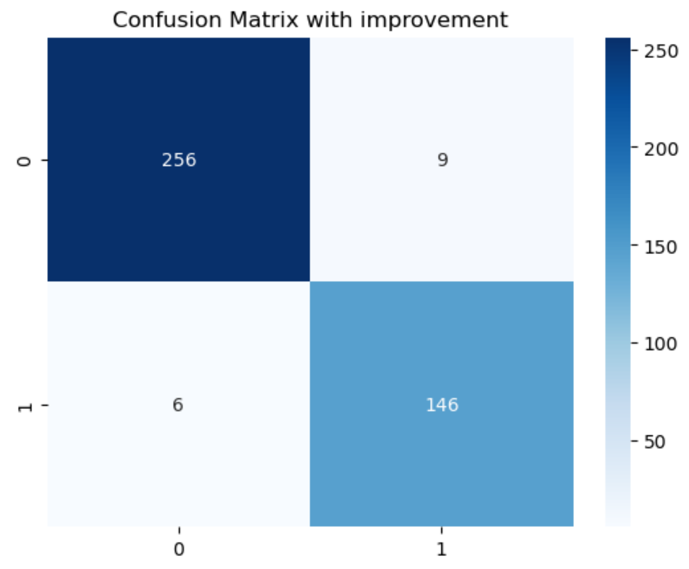
### Model Comparison

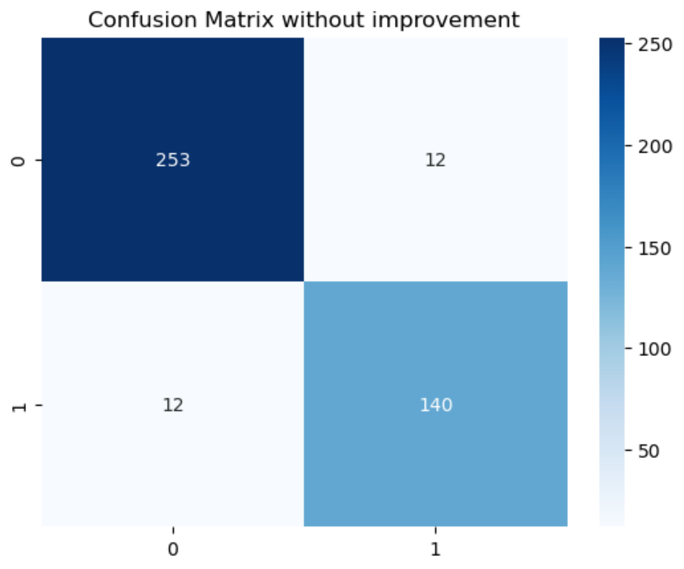
We compared the performance of our tuned Logistic Regression model with a baseline model to demonstrate the impact of hyperparameter tuning. The final model showed improved accuracy and robustness.

**accuracy score with model's hyper-parameter improvement: 0.9640287769784173**

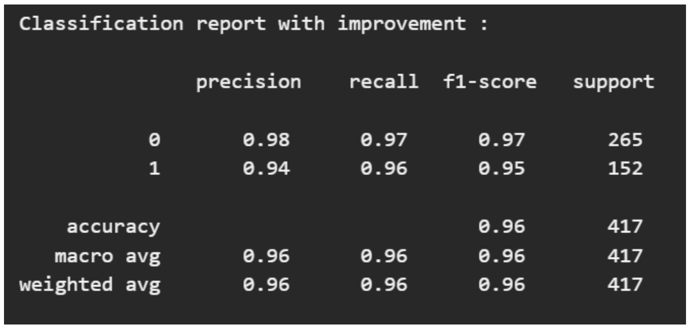
**accuracy score without model's hyper-parameter improvement: 0.9424460431654677**

We visualized the confusion matrices for both the model with hyperparameter tuning and the baseline model. The visualizations provided a clear understanding of true positive, true negative, false positive, and false negative predictions.





Classification reports were generated to provide a detailed breakdown of precision, recall, and F1-score for both models. These reports offer a nuanced perspective on model performance.

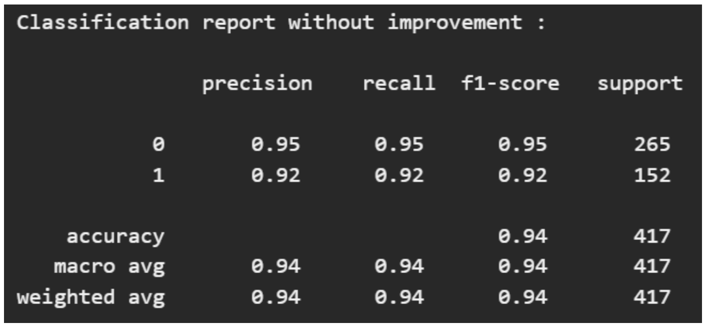


```
Classification report with improvement :

              precision    recall  f1-score   support

           0       0.98      0.97      0.97       265
           1       0.94      0.96      0.95       152

    accuracy                           0.96       417
   macro avg       0.96      0.96      0.96       417
weighted avg       0.96      0.96      0.96       417
```



```
Classification report without improvement :

              precision    recall  f1-score   support

           0       0.95      0.95      0.95       265
           1       0.92      0.92      0.92       152

    accuracy                           0.94       417
   macro avg       0.94      0.94      0.94       417
weighted avg       0.94      0.94      0.94       417
```

As we can see, hyper-parameter modification in the logistic regression made quite a significant improvement in the prediction. The accuracy score improved by 2%. The final accuracy score is not perfect, with 15 mistakes out of the 417 samples but it is still quite high in my opinion.

## Conclusion

In conclusion, the developed Logistic Regression model with optimized hyperparameters demonstrated improved accuracy and reliability compared to the baseline model. The careful preprocessing of data, exploratory data analysis, and iterative hyperparameter tuning were crucial steps in achieving this outcome.

The final model, along with its hyperparameters, has been saved for future use. The complete code and related files can be accessed on our GitHub repository.

# Iris Flower Classification: SVM Model Development and Evaluation

## Introduction

This project focuses on classifying iris flowers using Support Vector Machines (SVM). SVM is a powerful supervised learning algorithm that is effective for both classification and regression tasks. The report details the steps involved in building and optimizing an SVM model, as well as the evaluation metrics used to assess its performance.

## Algorithm Used

Support Vector Machines (SVM) were employed for the iris flower classification task. SVM is particularly useful for multiclass classification, making it suitable for predicting the species of iris flowers based on features like sepal length, sepal width, petal length, and petal width.

## Data Preprocessing

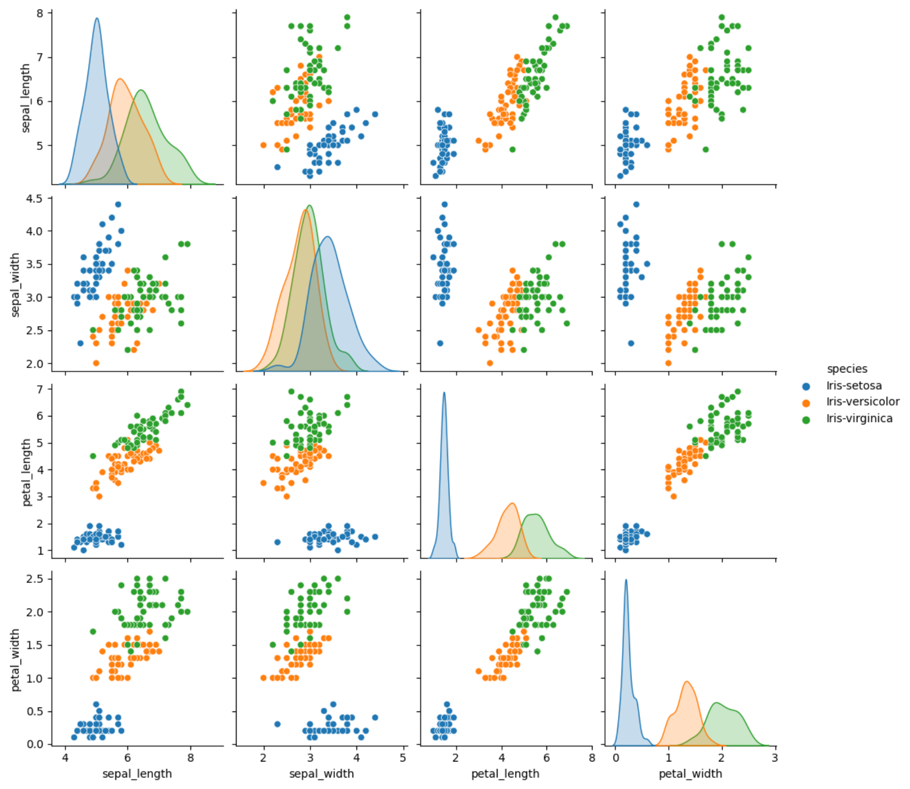
### Data Loading

The dataset, obtained from the "IRIS Flower Dataset," was loaded into a Pandas DataFrame for analysis and model training.

### Exploratory Data Analysis (EDA)

Visualizations, such as pair plots, were utilized to explore the relationships between different features and the distribution of iris flower species. EDA provided insights into the characteristics of the dataset.

From this pairplot, we can understand that firstly, iris setosa are very well recognizable because they are well separated from the other two in every category. Moreover, we can deduce that iris virginica is the longest and the iris setosa the shortest



## Model Training and Hyperparameter Tuning

The dataset was split into training and testing sets, and an SVM model was trained using the training data. A grid search with cross-validation was employed to optimize hyperparameters such as the choice of kernel, regularization parameter (C), and gamma value. The best hyperparameters were determined using `GridSearchCV.

## Model Evaluation

## Metrics Used

Model performance was evaluated using accuracy scores, confusion matrices, and classification reports. These metrics offered a comprehensive understanding of the model's ability to correctly classify iris flower species.

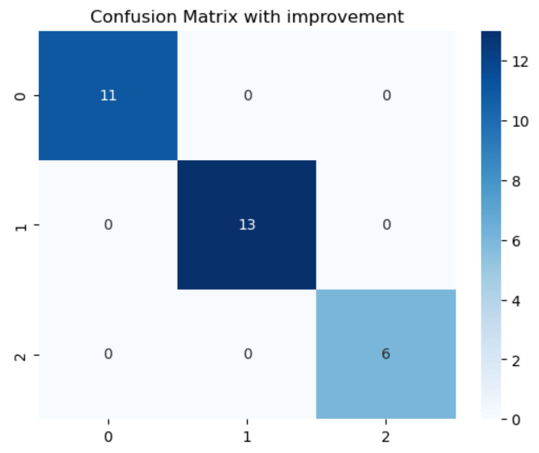
## Model Comparison

The performance of the tuned SVM model was compared with a baseline SVM model to highlight the impact of hyperparameter tuning. Unfortunately, there was no sense comparing them because the accuracy in every case was 100%. However, the algorithm was kept to show an example of what could be achieved to improve the model

## Graphical Representation

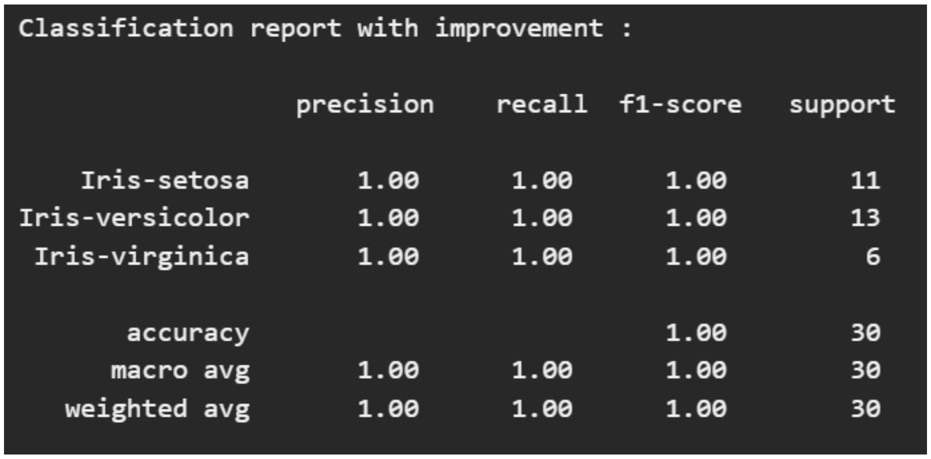
### Confusion Matrices

Visualizations of confusion matrices were generated for both the model with hyperparameter tuning and the baseline model. These matrices provided insights into the true positive, true negative, false positive, and false negative predictions.



## Classification Reports

Classification reports were presented to showcase precision, recall, and F1-score for each class in the iris flower dataset. These reports offered a detailed breakdown of the model's classification performance.



```
Classification report with improvement :

                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      1.00      1.00        13
 Iris-virginica       1.00      1.00      1.00         6

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```

## Conclusion

In conclusion, the SVM model was proved to be worthy in this case but because the database was a bit limited. We couldn’t really reach the limit of this code and we were so unable to improve it.