**Machine learning and data mining**

**Project**

**Hazem medhat Mohamed**

**Id:211001994**

**Topic of the project is to predict the time of the swimmers in the Olympics through a lot of variables.**

**Video link :** [Meeting with hazem medhat mohamed-20240531\_152658-Meeting Recording.mp4](https://nileuniversity-my.sharepoint.com/:v:/g/personal/h_medhat2194_nu_edu_eg/ESSV0zB_vYBEvkMQTO279V8BGdOyDbpIa1MhIgjrdzavfw?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJTdHJlYW1XZWJBcHAiLCJyZWZlcnJhbFZpZXciOiJTaGFyZURpYWxvZy1MaW5rIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXcifX0%3D&e=wONseK)

**Introduction:** The dataset includes information such as the swimmer's ID, stroke type, time taken to complete the distance, and potentially additional features like the swimmer's age, gender, and country. Analyzing this dataset can offer insights into the evolution of swimming performance over time, the dominance of certain countries or athletes, and the impact of factors like gender and age on swimming performance.

**goal of the project:** Goal of the model is to create a prediction model to predict the time it takes for each swimmer to complete their respective stroke over a set distance by knowing their stroke and overall time, rank, country, distance, year, team and gender.

A screenshot of a computer

Description automatically generated**Task1.1.2:** the provided code conducts a comprehensive examination of a Data Frame, primarily focusing on three key aspects. Initially, it scrutinizes the Data Frame for missing values, employing the **df.isnull().sum()** method to determine the total count of missing values in each column. Subsequently, it investigates the presence of duplicate rows within the Data Frame using the **df.duplicated()** function and showcases any identified duplicates. Lastly, the code shifts its attention to outlier detection, concentrating on the 'Results' column. It converts the data in this column to numeric format and utilizes the Interquartile Range (IQR) method to identify outliers.

**Task1.3:** then I used a code to generates a correlation heatmap using Matplotlib and Seaborn libraries. It first selects numeric columns from a Data Frame, creates a figure with specified dimensions, and then plots the correlation matrix of the numeric data.

A screenshot of a graph

Description automatically generated.

* This code produces the histogram with a KDE overlay of the Numerical Column data shows a bimodal distribution with peaks around values 25 and 60. Results are more frequent near these peaks, with a peak frequency exceeding 80 around 60. A dip in frequency occurs between 30 and 50, indicating fewer occurrences in this range. The KDE line illustrates the bimodal pattern smoothly, emphasizing two distinct groups in the dataset. This analysis provides insights into the distribution characteristics, prevalent value ranges, and frequency patterns within the data.

**A screenshot of a graph

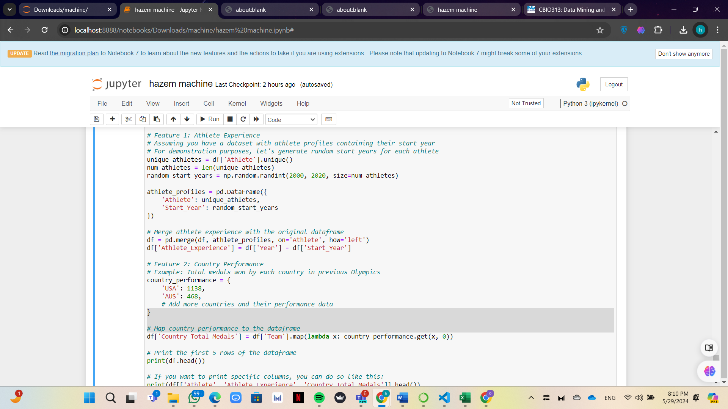
Description automatically generated**

**A graph of a number of columns

Description automatically generated**

**Task2:** This code snippet enhances an Olympic swimming dataset by introducing two significant features. Firstly, it computes the "Athlete Experience" by associating each athlete with a randomly generated start year, allowing for the calculation of their experience level in the year of the swimming event. Secondly, it integrates "Country\_Total\_Medals" data, providing insight into the historical performance of each athlete's country in previous Olympics. By merging these features into the original dataset, it enriches the analytical capabilities, offering valuable context for understanding athlete performance and country dynamics in Olympic swimming events.

A screenshot of a computer

Description automatically generated

* **Task2.2:** I used code snippet performs a regression task using a RandomForestRegressor model t prepares the data by handling missing values, encoding categorical variables, and converting a column to numeric format. After splitting the data into training and testing sets, the model is trained on the training data and evaluated using Mean Squared Error (MSE) and Mean Absolute Error (MAE). The output shows MSE of 5.57 and MAE of 1.70, indicating the model's performance in predicting the target variable ('Results'). because we have to check on the results column because its the one the model depending on

A screenshot of a computer

Description automatically generated

* **Task3**This code snippet performs classification using Logistic Regression in scikit-learn. It preprocesses data by dropping missing values, converting columns, and encoding categorical variables. After splitting data into training and testing sets, it constructs a classification model pipeline incorporating preprocessing and Logistic Regression. The model achieves an accuracy of approximately 0.61 on the test set, with a classification report indicating precision, recall, and F1-score for each rank category. Finally, the code demonstrates making predictions on

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**A screenshot of a computer

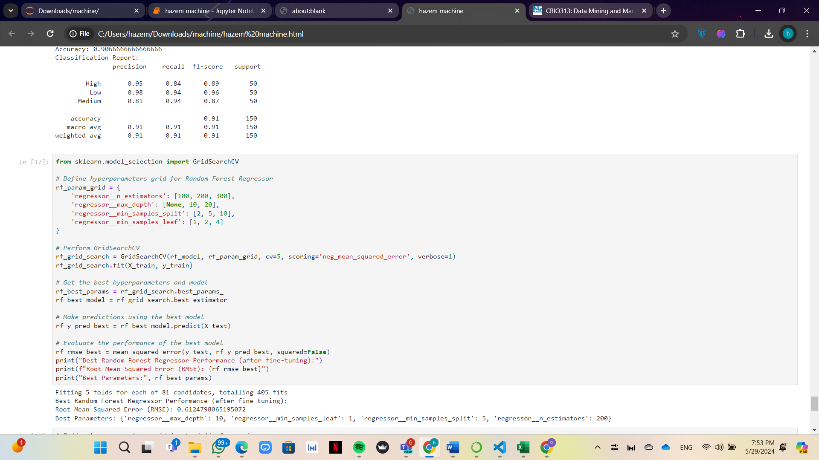
Description automatically generated**

**logistic regression accuracy: 90%**

**Random Forest Regressor Performance:**

**Mean Absolute Error (MAE): 0.4805908479041084**

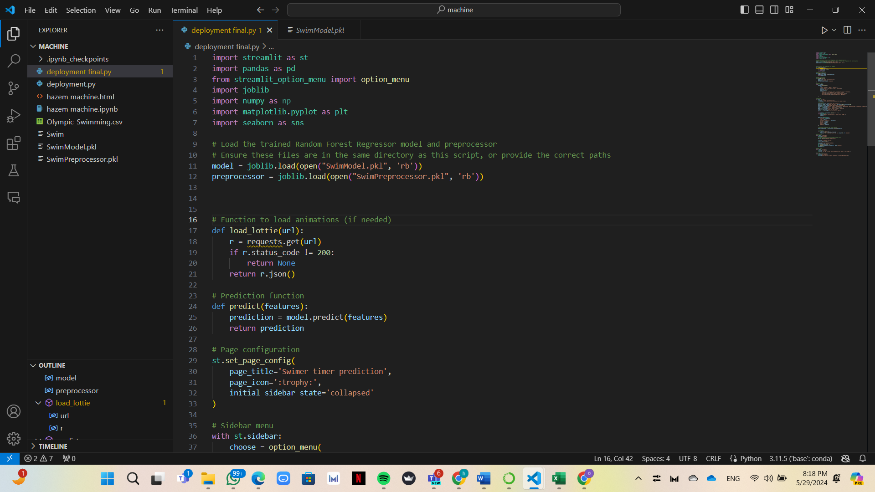
**Mean Squared Error (MSE): 0.3759350255332017**

* **task4**:I used GridSearchCV to fine-tune a Random Forest Regressor model's hyperparameters. It defines a parameter grid containing various values for parameters like the number of estimators and maximum depth. GridSearchCV then exhaustively searches through this grid, using cross-validation and negative mean squared error as the scoring metric. After finding the best combination of hyperparameters, it retrieves the best parameters and best model. Finally, it evaluates the best model's performance on the test set in terms of root mean squared error and prints the results along with the best parameters.

**Task 5:** I started by loading a pre-trained Random Forest Regressor model and a preprocessor to handle data transformation. The user interface consists of multiple pages accessible through a sidebar menu: "Home," "Graphs," "About," and "Contact." The "Home" page allows users to input parameters like location, year, stroke type, and athlete details, generating predictions for swim times using the loaded model. On the "Graphs" page, users can explore visualizations such as an overview of the dataset and a correlation heatmap, aiding in data analysis and understanding. The "About" page offers a concise description of the application's purpose, while the "Contact" page provides contact information for further inquiries.

A screenshot of a computer screen

Description automatically generated



A screenshot of a computer program

Description automatically generated

**Streamlit predction app**

**A screenshot of a computer

Description automatically generated**

