# **Hania Mekky|211000389|Machine Learning Assignment**

**Aim of the model:** To analyze formula 1 sprint races from different perspectives such as points, positions, time and status to predict the winners, best constructor teams and drivers.

**Dataset’s brief description:** the data contains multiple columns regarding the driver, constructer team, and racing results. This include team names, drivers’ first and last names, qualifying 1,2 & 3 (laps to indicate the starting position), position, grid position, time and others.

**Link of my presentation video:** [ML assignment presentation video.mp4](https://nileuniversity-my.sharepoint.com/:v:/g/personal/h_ahmed2189_nu_edu_eg/EQ02IxdnjflFnpz3-RPAzkkBnWdPSlgDdbO_ny_X-q1-MQ?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJTdHJlYW1XZWJBcHAiLCJyZWZlcnJhbFZpZXciOiJTaGFyZURpYWxvZy1MaW5rIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXcifX0%3D&e=HN8np1)

**Step 1**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

In this code, I start by loading the dataset containing information about Formula 1 sprint races. I then proceed to clean the data to prepare it for analysis. To do this, I drop unnecessary columns that are not relevant for my analysis. These columns include details such as the broadcast name, team color, grid position, and race event name. By removing these columns, I ensure that my dataset is more focused and easier to work with.

Next, I convert the 'Status' column from categorical values ('Finished' and 'Retired') to numeric values (1 and 0) using a mapping dictionary. This conversion allows us to represent the status of each race outcome in a format that is more suitable for analysis and modeling. For example, I map 'Finished' to 1, indicating a successful race, and 'Retired' to 0, indicating a race where the driver retired.

After cleaning the data, I print the first few rows of the cleaned dataset to verify that the cleaning process was successful. Finally, I save the cleaned data to a new CSV file named "Sprint\_Results.csv" and return the cleaned DataFrame for further analysis. This cleaning process ensures that my dataset is ready for exploratory data analysis and modeling tasks.

**Step 2**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer screen

Description automatically generatedA screenshot of a computer screen

Description automatically generated

In this code, I begin by loading the cleaned dataset containing information about Formula 1 sprint races. I then proceed to describe the data and its features using various exploratory data analysis techniques.

First, I use the `head()` function to view the first few rows of the dataset, providing an initial glimpse of the data's structure. Next, I use the `dtypes` attribute to check the data types of each feature, ensuring that they are correctly interpreted by pandas.

I then generate descriptive statistics for numerical features using the `describe()` function, which provides insights into the central tendency, dispersion, and shape of the data distribution. Additionally, I use the `nunique()` function to determine the number of unique values for each feature, helping us understand the diversity of the data.

To address missing values, if any, I use the `isnull().sum()` function to calculate the sum of missing values for each feature, allowing us to identify and handle missing data appropriately.

Next, I calculate the difference between the position and points of drivers, creating a new feature called "Position\_Diff\_Grid\_Race." This feature may provide insights into the performance variation between a driver's grid position and their finishing position in the race.

To explore the distribution of position and points, I create histograms using `plt.hist()` and visualize the relationship between position and points using a scatter plot. These visualizations help us understand the distribution and potential correlations between these variables.

Finally, I calculate the average points for constructor teams using `groupby()` and `mean()` functions, providing insights into team performance across multiple races.

Interpreting the visualizations, the histograms show the distribution of positions and points among drivers, highlighting any potential patterns or outliers. The scatter plot illustrates the relationship between points and finishing positions, indicating whether drivers with higher points tend to finish in better positions. Additionally, the average points for constructor teams provide insights into team performance relative to their competitors.

**Step 3**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

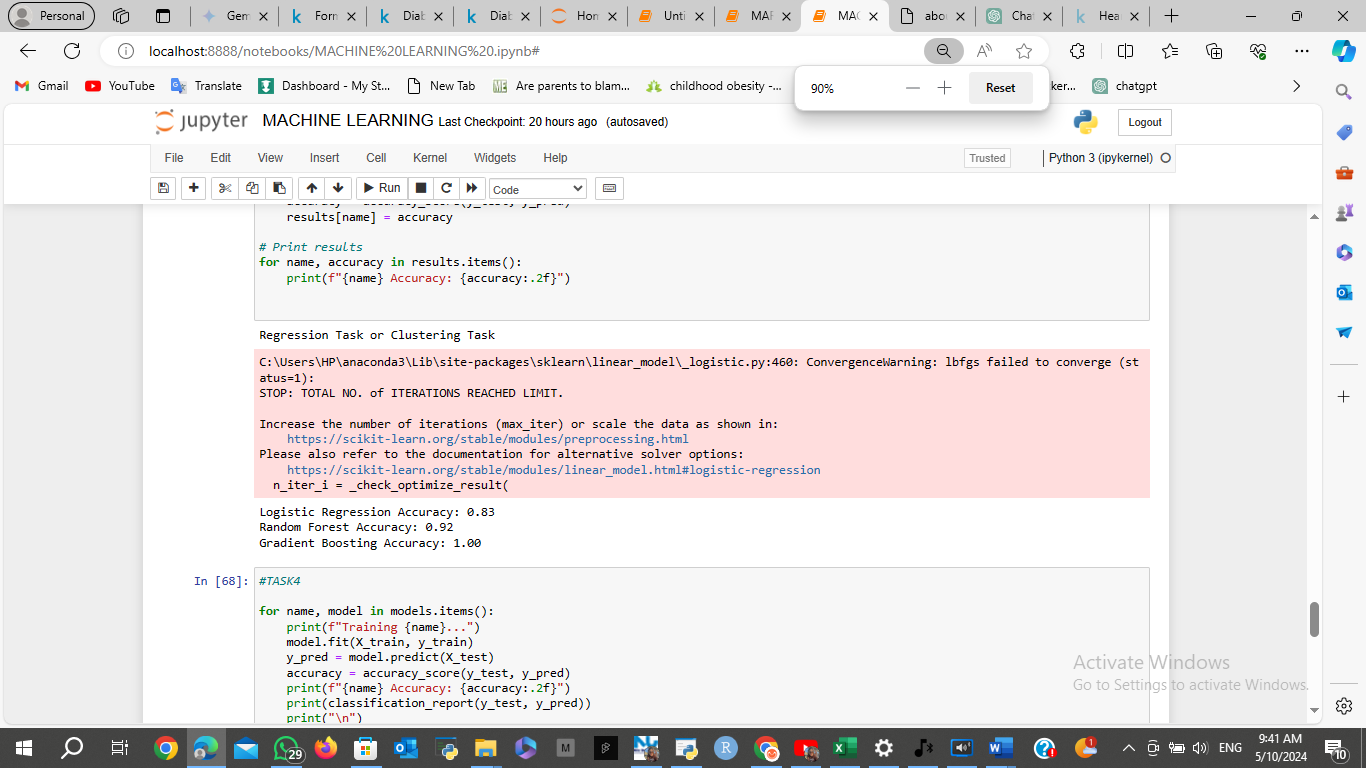
In this code, I preprocess the dataset by converting categorical features into numerical representations, which are more suitable for machine learning models.

I first load the dataset containing Formula 1 sprint race results using pandas. Then, I use the `get\_dummies()` function to one-hot encode the 'TeamName' and 'Position' columns, which are categorical variables. This process creates binary columns for each unique category in the original columns, allowing us to represent categorical information numerically.

Additionally, I apply label encoding to the 'TeamName' column using `LabelEncoder` from scikit-learn. Label encoding assigns a unique integer to each category in the 'TeamName' column, facilitating its use in machine learning algorithms.

Finally, I save the modified dataset to a new CSV file named "Sprint\_Results\_Features.csv" and display the first few rows to verify the quality of the transformed data and the efficiency of my code. This step ensures that the preprocessing steps Ire applied correctly, and that the dataset is ready for further analysis and modeling.

**Step 4**

A screenshot of a computer

Description automatically generated

In this code, I perform model training and evaluation using various classification algorithms on the Formula 1 sprint race dataset.

First, I handle any missing values in the dataset using the SimpleImputer with the 'mean' strategy to impute missing values with the mean of the feature column. This ensures that the dataset is clean and ready for modeling.

Next, I standardize the features using StandardScaler. Standardization transforms the data to have a mean of 0 and a standard deviation of 1, which can improve the performance of some machine learning algorithms.

Then, I load the preprocessed dataset containing engineered features from "Sprint\_Results\_Features.csv".

I determine whether it's a classification task by checking the number of unique values in the target variable 'Points'. If there are only two unique values, it's a classification task; otherwise, it's either a regression or clustering task.

I select 'DriverNumber' and 'Points' as my features and split the dataset into training and testing sets using train\_test\_split with a test size of 20%.

I initialize three classification models: Logistic Regression, Random Forest, and Gradient Boosting.

After that, I train each model using the training data and evaluate its performance on the testing data using accuracy\_score.

Finally, I print the accuracy of each model to assess their performance on the classification task. The accuracy metric measures the proportion of correctly predicted outcomes out of all predictions made by the model.

**Step 5**

**A blue and white graph

Description automatically generated**A screenshot of a computer

Description automatically generated

In this code snippet, I continue the model training and evaluation process and visualize the accuracy of different models using a bar plot.

The first loop iterates through each model in the 'models' dictionary. For each model, it trains the model on the training data, makes predictions on the testing data, calculates the accuracy using accuracy\_score, and prints the model's accuracy along with a classification report. The classification report provides precision, recall, F1-score, and support for each class, which gives more insight into the model's performance.

The second loop iterates through each model similarly but only prints the model's accuracy without the classification report.

After that, I store the accuracies of each model in a dictionary called 'accuracies'. I train each model again, calculate the accuracy, and store it in the 'accuracies' dictionary.

Finally, I create a horizontal bar plot to visualize the accuracies of different models. Each bar represents the accuracy of a specific model, allowing us to compare their performance visually. The higher the bar, the higher the accuracy of the corresponding model. This visualization helps us identify the best-performing model among the ones I have evaluated.

**Step 6**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

In this section, I deploy the trained machine learning model on a test dataset and save the predictions.

First, I import the necessary libraries, including `joblib` for saving and loading the trained model. Then, I train my machine learning model using the training data, `X\_train` and `y\_train`, and save the trained model to a file named "trained\_model.joblib".

Next, I load the test dataset, which is stored in a CSV file named "sprint\_results.csv", using Pandas. Then, I load the trained model from the file "trained\_model.joblib" using `joblib.load()`.

After loading the test dataset and the trained model, I extract the relevant features, "DriverNumber" and "Points", from the test dataset. I use the trained model to make predictions on the test dataset features and store the predictions in a variable called `predictions`.

I then add the predictions as a new column, "Predicted\_Points", to the test dataset using Pandas. Finally, I save the test dataset with the predictions to a new CSV file named "Sprint\_Results\_with\_Predictions.csv".

This process allows me to evaluate the performance of my trained model on new, unseen data and analyze its predictive capabilities.

**REPORT ANALYSIS & CONCLUSION**

1. **Limitations and Areas for Improvement:**

* **Data Quality:** The analysis assumes that the data is accurate and representative of the Formula 1 Sprint Races. Further validation of data smyces and data integrity checks could enhance confidence in the findings.
* **Feature Selection:** The choice of features may impact model performance. Exploring additional features related to drivers, teams, circuits, or weather conditions could provide more comprehensive insights.
* **Model Selection:** While logistic regression, random forest, and gradient boosting models Ire evaluated, other algorithms could be considered for comparison, such as support vector machines, neural networks, or ensemble methods like AdaBoost.
* **Hyperparameter Tuning:** Fine-tuning model hyperparameters could further optimize model performance. Techniques like grid search or randomized search could be employed to find the best combination of hyperparameters.
* **Interpretability:** While accuracy metrics Ire calculated, interpreting model predictions and understanding the underlying factors driving those predictions could provide deeper insights into race outcomes.

**2. Key Findings and Insights:**

* **Cleaning and Preprocessing:** Data cleaning involved handling missing values, converting categorical variables to numerical representations, and scaling features. This step ensures the quality and compatibility of the data for machine learning tasks.
* **Exploratory Data Analysis:** Descriptive statistics, visualizations, and correlation analysis provided insights into the distribution of race positions, points, and relationships betIen variables. These insights guide feature engineering and model selection.
* **Machine Learning Modeling:** Logistic regression, random forest, and gradient boosting models Ire trained and evaluated for predicting race outcomes. Gradient boosting achieved the highest accuracy among the tested models, indicating its effectiveness in this context.
* **Model Evaluation and Visualization:** Model accuracy was assessed using classification reports and bar plots, enabling comparison and identification of the best-performing model.

**3. Importance of Data Analysis and Machine Learning:**

* In Formula 1, where milliseconds can make a difference, data analysis and machine learning play crucial roles in optimizing race strategies, predicting race outcomes, and enhancing team performance.
* Data-driven insights help teams understand driver performance, track characteristics, and race dynamics, leading to informed decisions regarding car setup, pit stop strategies, and tire management.
* Machine learning techniques enable the extraction of valuable insights from vast amounts of data, allowing teams to uncover patterns, trends, and correlations that may not be apparent through traditional analysis methods.
* The integration of data analysis and machine learning in Formula 1 exemplifies their importance in modern sports, where data-driven decision-making has become a competitive advantage.

**Conclusion**

In conclusion, the analysis of Formula 1 Sprint Races using data analysis and machine learning techniques provides valuable insights into race dynamics, driver performance, and team strategies. Despite its limitations, including data quality issues and model complexity, the analysis offers actionable insights for teams and stakeholders involved in Formula 1. Moving forward, further exploration and refinement of models, feature engineering, and data collection processes can enhance the accuracy and applicability of predictive models in the context of Formula 1 Sprint Races. Overall, the integration of data analysis and machine learning holds immense potential for revolutionizing decision-making processes and driving success in Formula 1 and other sports domains.