**Report of Capstone Project**

**Purpose:**

To show how five different machine learning algorithms applied on F1 dataset, making predictions for Points and how the performance will be at the end. The target variable for prediction is the 'points\_x' column.

I have applied five different models: LinearRegression (), DecisionTreeRegressor (), RandomForestRegressor (), KNeighborsRegressor () and CNN LSTM.

The performance of each model is evaluated based on RMSE, R^2 score, and adjusted R^2.

**Dataset:**

The datasets I have used are related to Formula 1 racing. I am working with data related to Formula 1 race results, driver details, race details, constructor details, and driver standings.

Then I have merged these data and made a new dataset, later I have found out the relevant columns: 'grid', 'position\_x', 'laps', 'fastestLap', 'rank', 'circuitId', 'points\_x'.

* **grid**: This column shows the starting position of the driver in a race, indicating their initial position on the grid.
* **position\_x**: This column represents the finishing position of the driver in a race, indicating the rank or order in which they completed the race.
* **laps:** This column indicates the total number of laps completed by the driver in a race, reflecting the distance covered during the race.
* **fastestLap:** This column captures the lap time of the driver’s fastest lap during the race, providing insights into their speed and performance.
* **rank:** This column represents the driver’s overall position or rank in the championship standings, indicating their performance relative to other drivers throughout the season.
* **circuitId:** This column contains a unique identifier for the circuit where the race took place, helping identify the specific racetrack associated with the data.
* **points\_x:** This column reflects the number of points scored by the driver in a race, with higher positions earning more points based on their finishing position.

**Data Pre-Processing:**

Before training the models, I performed several data preprocessing steps on the F1 Formula dataset. I checked missing values and the duplicate values.

A black and white text

Description automatically generated



Then I did exploratory data analysis. scatter plot visualizes the relationship between the ‘grid’ and ‘points\_x’ variables, showing how the points scored (‘points\_x’) vary based on the starting position (‘grid’) in the data. It helps identify if there is any correlation or pattern

#between the grid position and the points scored.

A graph of points vs grid

Description automatically generated

The purpose of calculating the average points by circuit and creating a bar plot is to understand the variations in point scoring across different circuits. It helps in identifying circuits where drivers tend to score higher or lower points on average. This analysis can provide insights into circuit-specific factors that may impact driver performance or influence point distributions.

It calculated the average points scored by drivers for each circuit in the dataset using the groupby () function. It groups the data by the ‘circuitId’ column and then calculates the mean of the ‘points\_x’ column, representing the average points earned by drivers for each circuit.

A graph of blue lines

Description automatically generated

I have followed these steps to prepare the data for model training, including selecting relevant features, handling missing values, ensuring feature scaling, and splitting the data into training and testing sets for evaluation purposes.

**a) Feature Selection:**

In the feature selection step, I examined the initial set of columns in the dataset and identified relevant features that are likely to have an impact on predicting the points scored by the drivers. Based on domain knowledge and feature importance analysis, I selected specific columns such as ‘grid’, ‘position\_x’, ‘laps’, ‘fastestLap’, and ‘rank’ as potential predictors. The target variable for prediction was identified as ‘points\_x’, representing the points scored by the drivers.

**b) Handling Missing Values:**

To handle missing values in the dataset, I first replaced any missing values encountered with NaN (Not a Number). By doing so, I marked the missing values for identification and further processing. Afterward, I proceeded to drop the rows that contained missing values. This step ensured that the dataset used for model training did not contain any incomplete or missing information.

**c) Feature Scaling:**

To ensure that all selected features are on the same scale, I applied the Min-Max scaling technique. Min-Max scaling rescales the values of the features to a uniform range, typically between 0 and 1. By performing this scaling, I prevented any particular feature from dominating the model training process due to differences in the original value ranges. This step promoted fair treatment and equal influence from all selected features during the model training phase.

**d) Train-Test Split:**

To evaluate the performance of the trained model, I divided the dataset into training and testing sets using an 80:20 split ratio. This means that 80% of the data was allocated for training the model, while the remaining 20% was reserved for testing or evaluating the trained model’s performance. The dataset was partitioned into X\_train, X\_test, y\_train, and y\_test sets, where X represents the input features and y represents the target variable.

**Training model:**

The focus of this part is on training four different, making predictions on the test data, and evaluating the model’s performance using the mean squared error (MSE). This section compares the performance of multiple regression algorithms, including the Random Forest Regressor, on both the training and testing sets. The purpose is to assess and compare the overall performance and suitability of various regression algorithms for the task.

A screenshot of a computer program

Description automatically generated

**CNN (Convolutional Neural Network):**

Then dataset is reshaped to fit the CNNLSTM model architecture. The model consists of a 1D convolutional layer (Conv1D), a long short-term memory layer (LSTM), and a dense layer. It is compiled with the Adam optimizer and mean squared error loss. The model is trained on the training data for 10 epochs. After training, the model is used to make predictions on both the training and testing sets. Evaluation metrics such as mean squared error and R-squared are calculated to assess the model’s performance.

A close-up of a computer error

Description automatically generated

**Graphical Presentation:**

Showing two graphical view here:

**A graph showing the value of a certain value

Description automatically generatedA graph showing a line of values

Description automatically generated**

**Comparison:**

Based on the evaluation metrics (R-squared), I can compare the algorithms and determined the higher and lower accuracies:

**Best Performance:**

Algorithm: DecisionTreeRegressor

Mean Squared Error (Testing Set): 0.001370417872299979

This algorithm achieved the smallest MSE among all the algorithms, indicating better predictive accuracy in capturing the variance between the predicted and actual values. It demonstrates the highest performance in terms of minimizing the prediction error.

**Worst Performance:**

Algorithm: CNNLSTM Model

Mean Squared Error (Testing Set): 1.009076688849869

The CNNLSTM Model shows the highest MSE among all the algorithms, suggesting a higher degree of prediction error. It has the lowest performance in terms of capturing the variance between the predicted and actual values.

**Conclusion:**

In this project, I have applied five different machine learning algorithms, including Linear Regression, Decision Tree Regressor, Random Forest Regressor, K-Nearest Neighbors Regressor, and a CNN LSTM model, to predict the points scored by Formula 1 drivers based on various race-related features. Decision Tree Regressor and Random Forest Regressor exhibited the best performance among all the algorithms, with extremely low Mean Squared Error (MSE) values and high R-squared scores. The CNN LSTM model performed the poorest among all the algorithms, as evident from the significantly higher MSE value and negative R-squared score. Linear Regression showed moderate performance, achieving relatively higher MSE values and lower R-squared scores compared to the tree-based models. While it provided reasonable predictions, it didn’t capture the variability in the target variable as effectively as the decision tree-based algorithms. K-Nearest Neighbors Regressor delivered competitive results with a relatively low MSE and a high R-squared score.