Course: Fundamentals of Data Analysis and Machine Learning

Report Title: Analysis of the *Formula 1 World Championship (1950 - 2024)* Dataset

Students:

Marat Zhanel, Borambaev Duman

Main part

1. Goal

The goal of this study is to analyze the *Formula 1 World Championship (1950 - 2024)* dataset in order to predict drivers' points and race outcomes (classification: whether a driver finishes the race or not) using different machine learning models.

2. Methods

- **Regression:** Linear Regression, Linear Regression with Momentum
- **Classification:** Logistic Regression, Decision Tree Classifier
- **Data preprocessing:**
    - Selection of relevant features: grid, laps, fastestLapSpeed
    - Handling missing values (\N replaced by NaN)
    - Numeric conversion and normalization of features
- **Train/test split:** 70% train, 30% test for classification tasks
- **Evaluation metrics:**
    - Regression: MSE, RMSE, R^2
    - Classification: Accuracy, Confusion Matrix, ROC AUC, Classification Report

3. Architectures

- **Linear Regression:** Standard single-layer linear model: $y=Xw+by = Xw + by=Xw+b$
- **Linear Regression with Momentum:** Gradient descent with momentum for faster convergence
- **Logistic Regression:** Weighted gradient descent for binary classification
- **Decision Tree:** Maximum depth = 5, balanced class weights

4. Hyperparameters

- Learning rate (lr): tested values [0.0001, 0.001, 0.01, 0.1]
- Number of epochs (epochs): 100
- Decision Tree: max_depth=5, class_weight='balanced'

My steps

Linear Regression

Initially, we tried to create a linear regression model to predict race time based on the starting grid position. However, the results were not satisfactory because the relationship between grid position and race time is not linear. Small differences in race time measured in milliseconds caused large prediction errors. To improve the model, we added additional features, including the number of laps and fastest lap speed. Despite this, the predictions remained poor, so we decided to focus on predicting points instead of race time.

For predicting points, we used the features grid, laps, and fastestLapSpeed. We first tried optimizing the model using the Adam optimizer, which adapts the learning rate for faster convergence. Unfortunately, it did not significantly improve the results. Next, we applied momentum in gradient descent, which slightly improved the model performance. The results were:

- Coefficients: [-3.25, 0.99, 0.32], Intercept: 3.67, MSE: 27.36, RMSE: 5.23, R^2: 0.36
- Coefficients: [-3.73, 1.01, 0.25], Intercept: 4.22, MSE: 26.81, RMSE: 5.18, R^2: 0.37

The results are still limited because predicting points is influenced by many factors not present in the dataset, and linear regression cannot capture complex nonlinear interactions between features.

Logistic Regression and Decision Tree

For the classification task, we aimed to predict whether a driver finished the race or not (Finished vs Not Finished). At the beginning, the dataset had an imbalance between classes, with one class dominating the data. This caused the model to always predict the majority class correctly while failing to recognize the minority class. To address this, we applied class weighting to balance the influence of both classes.

We implemented logistic regression from scratch by realizing the core functions: the sigmoid function, prediction using logistic regression, log-loss, and training with gradient descent. Specifically, we:

- Used the sigmoid function to map the linear combination of features to a probability between 0 and 1:

```
def sigmoid(z):
   return 1 / (1 + np.exp(-z))
```

- Predicted probabilities with:

```
def predict_logistic(X, w, b):
   return sigmoid(X @ w + b)
```

- Calculated weighted log-loss to handle class imbalance.
- Trained the model using gradient descent, updating weights and bias iteratively while applying class weights to the gradients:

```
def train_logistic_regression_balanced(X, y, lr, epochs):
   ...
```

This approach represents one method in classification: logistic regression for binary classification.

To improve performance, we also tested a second model, a Decision Tree, which is capable of capturing nonlinear relationships between features and the target, handling interactions between variables automatically, and being less sensitive to feature scaling and noise. This additional model performed better than logistic regression, achieving higher accuracy and ROC AUC.

We divided the classification problem into two classes (binary classification: Finished vs Not Finished). Logistic regression served as the required model, and the Decision Tree was chosen as an additional model to explore nonlinear patterns in the data.

By balancing classes and using both models, we ensured more reliable classification results, with the Decision Tree ultimately performing better due to its ability to capture complex patterns in the dataset.

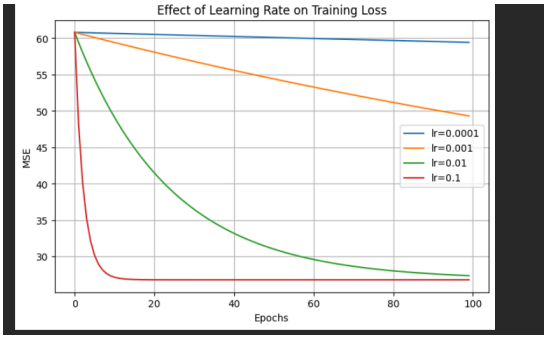
Results and Tests – Linear Regression

We tested Linear Regression with different learning rates and epochs to see how these hyperparameters affect training.

- **LR = 0.01, Epochs = 100**: Coefficients [-3.25, 0.99, 0.32], Intercept 3.67, MSE 27.36, RMSE 5.23, R^2 0.36

- **LR = 0.05, Epochs = 240**: Coefficients [-3.73, 1.01, 0.25], Intercept 4.24, MSE 26.81, RMSE 5.18, R^2 0.37

- **LR = 0.01, Epochs = 500**: Coefficients [-3.73, 1.01, 0.25], Intercept 4.23, MSE 26.81, RMSE 5.18, R^2 0.37



Coefficients show how each feature affects points: negative for grid (starting further back reduces points), positive for laps and fastestLapSpeed. **Intercept** is the baseline prediction when features are zero.

Observations: Increasing learning rate and epochs slightly improved R^2 and reduced MSE. Metrics stabilized after enough epochs, and coefficients remained consistent. The model explains ~37% of variance, showing linear regression's limitation for this dataset.

Results and Tests – Logistic Regression

We tested Logistic Regression for predicting whether a driver finished the race (0) or not (1) using grid, laps, and fastestLapSpeed. Different learning rates and epochs were evaluated to optimize performance.

- **LR = 0.01, Epochs = 100**: Coefficients [0.21, -0.13, -0.07], Intercept -0.00026, Accuracy 0.74, ROC AUC 0.817

- **LR = 0.04, Epochs = 240**: Coefficients [0.92, -0.60, -0.31], Intercept 0.012, Accuracy 0.75, ROC AUC 0.817

- **LR = 0.10, Epochs = 500**: Coefficients [1.12, -0.85, -0.44], Intercept 0.052, Accuracy 0.74, ROC AUC 0.817

- **LR = 0.01, Epochs = 500**: Coefficients [0.49, -0.30, -0.16], Intercept -0.00048, Accuracy 0.74, ROC AUC 0.817

Among these tests, the model with **learning rate 0.04 and 240 epochs** performed the best, achieving the highest accuracy of **0.75** while maintaining ROC AUC of **0.817**.

Explanation:

- Coefficients show the effect of each feature on the log-odds of not finishing the race. Positive values increase the probability of not finishing, negative decrease it.

- Intercept represents the baseline log-odds when all features are zero.

Observations and Conclusions:

- Increasing learning rate from 0.01 to 0.04 improved accuracy slightly.

- Increasing epochs helped the model converge, but beyond 240 epochs, performance did not improve significantly.

- The best model (LR = 0.04, Epochs = 240) shows that proper tuning of learning rate and epochs can optimize logistic regression performance while keeping coefficients reasonable.

- Overall, Logistic Regression performed steadily, with class weighting ensuring balanced predictions between drivers who finished and did not finish.