Statistical Learning in Formula One: Predicting Lap Times and Podium Finishes

Team (146) Member: Shehzad Anwar

Midterm Report

Introduction

This project applies data mining and statistical learning techniques to a comprehensive Formula One (F1) dataset. F1 is a heavily data-driven sport where race outcomes are dependent on a complex interplay of driver skill, car engineering and team strategy.

The motivation for this project comes from a desire to move beyond the qualitative, expert-driven analysis common in F1 broadcasting. While descriptive statistics are used, there is a clear opportunity to apply robust predictive models to answer complex questions about performance. This project will look to determine which factors are most predictive of race outcomes and to build models that can handle the unique challenges of F1 data.

The primary data mining challenge identified in the dataset is severe multi-collinearity among predictor variables (e.g. grid position, qualifying position and various lap times are all highly correlated). The project strategy is specifically designed to address this challenge using the appropriate methods.

This report outlines the data source, full methodology proposed to answer the research questions, and the preliminary results from the Exploratory Data Analysis. It concludes with a clear plan for the modeling and analysis to be completed for the final report.

Problem Statement and Data Source

The data for this analysis was obtained from the "Formula 1 World Championship (1950-2024)" dataset, a public-domain collection on Kaggle containing data from the last 74 years. The dataset consists of 14 individual *.csv* files, including *races.csv, results.csv, qualifying.csv, lapt_times.csv*, and more.

These files were loaded, cleaned and merged into a single master dataframe. The cleaned dataset used for the analysis has 8252 data points, each row representing a single driver's result in one race. This number reflects the result of merging multiple tables and then dropping rows with missing target variables, like *fastestLapTime_ms*. In other words, the models were trained on drivers who finished the race and were able to set a time. This could be considered a limitation.

Table 1: Sample of Processed Data This table shows a sample of the final, merged data, showcasing how the key predictors and targets are organized.

fastestLapTime_ms	positionOrder	podium	year	grid	qualifying_position	avg_lap_time_ms	driverRef
87452.00	1	1	2008	1	1.00	98114.07	hamilton
87739.00	2	1	2008	5	5.00	98208.52	heidfeld
88090.00	3	1	2008	7	7.00	98254.81	rosberg
88603.00	4	0	2008	11	12.00	98410.29	alonso
87418.00	5	0	2008	3	3.00	98424.66	kovalainen

Table 2: Key Variable Definitions The following variables (subset of the whole data set) have been identified for analysis.

Variable	Role	Description
`fastestLapTime_ms`	Y_1 (Response)	**Target for Q1:** The driver's fastest lap of the race, in milliseconds.
`positionOrder`	Y_2 (Response)	**Target for Q2:** The driver's final finishing position (e.g., 1, 5, 18).
`podium`	Y_3 (Response)	**Target for Q3:** A binary flag (1 if `positionOrder` ≤ 3, 0 otherwise).
`grid`	X (Predictor)	The driver's starting grid position.
`qualifying_position`	X (Predictor)	The driver's final position in the qualifying session.
`avg_lap_time_ms`	X (Predictor)	**Engineered Feature:** The driver's mean lap time over the entire race.
`std_lap_time_ms`	X (Predictor)	**Engineered Feature:** The standard deviation of the driver's lap times.
`driver_age_at_race`	X (Predictor)	**Engineered Feature:** The driver's age on race day.

`q1_ms`, `q2_ms`, `q3_ms`	X (Predictor)	Lap times in milliseconds for qualifying sessions 1, 2, and 3.
`year`, `circuitId`	X (Predictor)	Contextual variables for the race.

The initial key predictors were selected based on the team's domain knowledge of F1, with factors like qualifying performance and lap times being widely understood as being critical. A goal for the final report is to algorithmically validate this initial selection using the outputs of LASSO and Random Forest. LASSO will perform the variable selection while Random Forest will rank the feature importance, ultimately proving which variables are the most paramount predictors statistically.

Proposed Methodology

This project will answer three distinct research questions using the appropriate methods with justification. For each model, 10-fold cross-validation will be used to tune the parameters, one such being the number of components in PCR (*k*). All predictor variables (*X*) will be standardized before fitting.

Question 1: Can we accurately predict a driver's fastest lap time?

- **Methods:** LASSO and Ridge Regression
- **Justification:** The preliminary analysis confirms high multicollinearity. These shrinkage methods are specifically designed to handle this, preventing unstable coefficients of standard linear regression. LASSO will perform automated variable selection, allowing for interpretation of which factors are most influential to lap time.

Question 2: What underlying factors best explain a driver's final race position?

- **Methods:** Principal Component Regression (PCR) vs. Partial Least Squares (PLS)
- **Justification:** This question is mainly about dimension reduction. By comparing PCR, an unsupervised method that finds components which explain the variance in predictors (*X*), against PLS, a supervised method which finds components that best predict the response variable (*Y, positionOrder*). This will demonstrate which approach is most accurate.

Question 3: Can we classify whether a driver will achieve a podium (top 3) finish?

- **Methods:** Logistic Regression, Linear Discriminant Analysis (LDA), and Ensemble Methods (e.g. Random Forest, etc.)
- **Justification:** Logistic Regression and LDA will serve as strong, interpretable baselines. Their predictive performances will be compared against a non-linear ensemble method such as Random Forest. This allows for a direct comparison in the trade-off between the model interpretability (linear models) and the raw predictive accuracy (ensembles).

For the final report, we will explore creating interaction and polynomial features (e.g. *driver_age^2* or *grid * circuitId*) in order to improve the model performance, as suggested from feedback.

Analysis and Results (Preliminary)

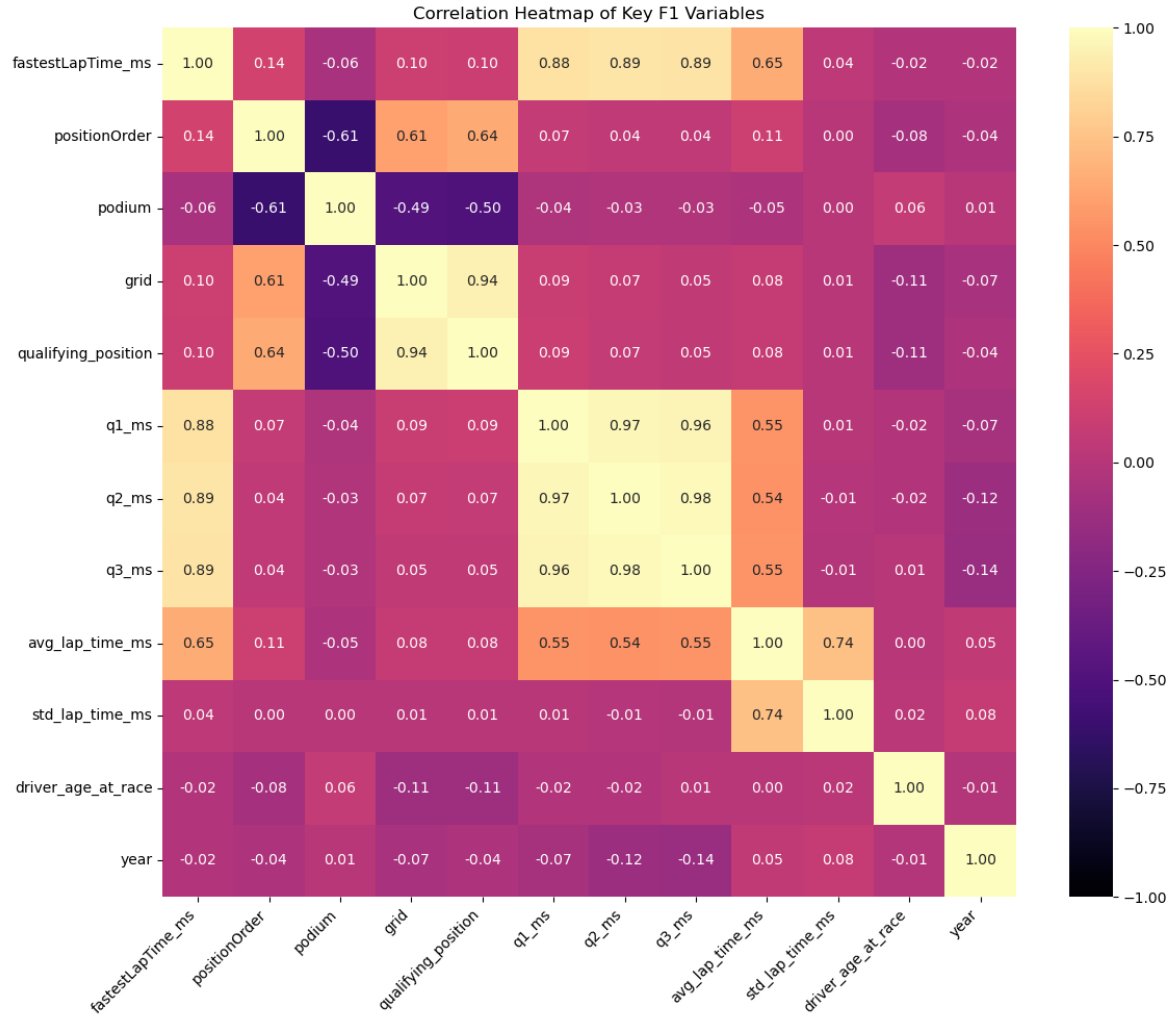
As this is a midterm report, the main focus of this section will be the Exploratory Data Analysis and data preparation status.

Table 3: Descriptive Statistics of Key Variables A summary of key numeric variables gives an insight into their distribution.

	fastestLapTime_ms	positionOrder	grid	qualifying_position	avg_lap_time_ms	driver_age_at_race
count	8252.00	8252.00	8252.00	8222.00	8249.00	8252.00
mean	90827.52	10.66	10.83	10.95	99251.22	28.21
std	12338.93	5.96	6.19	6.14	20791.10	5.11
min	55404.00	1.00	0.00	1.00	62932.34	17.45
25%	80848.25	6.00	5.00	6.00	85764.07	24.22
50%	90267.00	11.00	11.00	11.00	97321.96	27.29
75%	99476.00	16.00	16.00	16.00	107296.40	31.52
max	202300.00	24.00	24.00	24.00	507859.22	43.89

It should be noted that the average/mean of the finishing position (*positionOrder*) is 10.66. The variable *grid* has a minimum of 0, which indicate a pit lane start. This will be handled during modeling.

Correlation Analysis A correlation heatmap highlighting the strength of the relationships between the numeric variables.



The heatmap confirms the heavy multi-collinearity. The correlation between *grid* and *qualifying_position* is extremely high at 0.94. Furthermore, the correlations among the qualifying time predictors are also exceptionally high (i.e. *q1_ms* vs *q2_ms* is 0.97; *q2_ms* vs *q3_ms* is 0.98; *q1_ms* vs *q3_ms* is 0.96). This confirms that these variables are highly redundant and that including them in a standard linear model would only cause unstable coefficients. This finding is the central justification for the proposed methodology, since LASSO and Ridge are specifically designed to handle this exact problem.

Conclusion

This report is confirmation that the project is on track. The data has been successfully acquired, cleaned, and merged. The preliminary EDA has provided insights, but also, more importantly, has validated the choices we made for the statistical methods. The data has been fully preprocessed and ready for the modeling phase. No final conclusions have been drawn about F1 performance as of this Midterm Report.

Unfinished and Future Work The following tasks will be the focus for the final report:

1. **Model Training:** The models proposed in the proposed methodology must be trained on the prepared training data (70% of the data)
2. **Model Tuning**: Tune all the parameters using 10-fold cross-validation systematically.
3. **Model Evaluation:** Evaluation of the final models on the hold-out test set using R^2/MSE (for regression) and AUC-ROC (for classification).
4. **Threshold Selection:** For the classification models, we'll move beyond AUC-ROC and select a final prediction threshold c^* based on validation data to optimize for a misclassification rate.
5. **Interpretation:** We will analyze the coefficients, components, and feature importance plots from Random Forest to provide comprehensive answers to the research questions and discuss the impact.

Appendix

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import glob
import os
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler

# ============================================================================
# Step 1: Load All CSV Files
# ============================================================================
file_paths = glob.glob('f1datasets/*.csv')

f1_dfs = {} # Dictionary to hold DataFrames

print(f"Loading {len(file_paths)} CSV files...")
for file_path in file_paths:
    basename = os.path.basename(file_path)
    df_name = basename.split('.')[0] # Get the file name without extension
(results.csv runs into issues with 'results' keyword)
    try:
        f1_dfs[df_name] = pd.read_csv(file_path)
    except Exception as e:
        print(f"Error loading {file_path}: {e}")



# ============================================================================
# Step 2: Cleaning and Preprocessing the Data
# ============================================================================
for df_name in f1_dfs:
    f1_dfs[df_name].replace(r'\\N', np.nan, inplace=True)

def time_to_ms(time_str): # Convert time strings like '1:23.456' to milliseconds
    if pd.isna(time_str):
        return np.nan

    parts = str(time_str).split(':')

    try: # Handle different time formats
```

```
        if len(parts) == 2:
            minutes = int(parts[0])
            seconds = float(parts[1])
            return int((minutes * 60 + seconds) * 1000)
        elif len(parts) == 3:
            hours = int(parts[0])
            minutes = int(parts[1])
            seconds = float(parts[2])
            return int((hours * 3600 + minutes * 60 + seconds) * 1000)
        else:
            return np.nan
    except ValueError: # In case of conversion error
        return np.nan
print("Cleaning data...\nConverting time strings to milliseconds...")

# Data cleaning for specific DataFrames
# Results
df_results = f1_dfs['results'].copy()
df_results['fastestLapTime_ms'] = df_results['fastestLapTime'].apply(time_to_ms)
df_results['positionOrder'] = pd.to_numeric(df_results['positionOrder'],
errors='coerce')
df_results['podium'] = df_results['positionOrder'].apply(lambda x: 1 if x <= 3
else 0)
df_results['grid'] = pd.to_numeric(df_results['grid'], errors='coerce')

# Qualifying
df_quali = f1_dfs['qualifying'].copy()
df_quali['q1_ms'] = df_quali['q1'].apply(time_to_ms)
df_quali['q2_ms'] = df_quali['q2'].apply(time_to_ms)
df_quali['q3_ms'] = df_quali['q3'].apply(time_to_ms)
df_quali_short = df_quali[['raceId', 'driverId', 'constructorId', 'position',
'q1_ms', 'q2_ms', 'q3_ms']]
df_quali_short = df_quali_short.rename(columns={'position':
'qualifying_position'})

# Races
df_races = f1_dfs['races'].copy()
df_races['date'] = pd.to_datetime(df_races['date'])
df_races_short = df_races[['raceId', 'year', 'round', 'circuitId', 'date']]

# Drivers
df_drivers = f1_dfs['drivers'].copy()
df_drivers['dob'] = pd.to_datetime(df_drivers['dob'])
```

```
df_drivers_short = df_drivers[['driverId', 'driverRef', 'nationality', 'dob']]

# Constructors
df_constructors = f1_dfs['constructors'].copy()
df_constructors_slim = df_constructors[['constructorId', 'name', 'nationality']]
df_constructors_slim = df_constructors_slim.rename(columns={'name':
'constructor_name'})



# ============================================================================
# Step 3: Feature Engineering
# ============================================================================
print("Engineering features from 'lap_times'...")
df_lap_times = f1_dfs['lap_times'].copy()
df_lap_times['milliseconds'] = pd.to_numeric(df_lap_times['milliseconds'],
errors='coerce')

# Computing average and std of lap times
lap_stats = df_lap_times.groupby(['raceId',
'driverId'])['milliseconds'].agg(['mean', 'std']).reset_index()
lap_stats = lap_stats.rename(columns={'mean': 'avg_lap_time_ms', 'std':
'std_lap_time_ms'})



# ============================================================================
# Step 4: Merging DataFrames
# ============================================================================
print("Merging DataFrames...")
master_f1 = pd.merge(df_results, df_races_short, on='raceId', how='left')
master_f1 = pd.merge(master_f1, df_quali_short, on=['raceId', 'driverId',
'constructorId'], how='left')
master_f1 = pd.merge(master_f1, df_drivers_short, on='driverId', how='left')
master_f1 = pd.merge(master_f1, df_constructors_slim, on='constructorId',
how='left')
master_f1 = pd.merge(master_f1, lap_stats, on=['raceId', 'driverId'], how='left')

# Computing drivers' age at the time of race
master_f1['driver_age_at_race'] = (master_f1['date'] - master_f1['dob']).dt.days
/ 365.25
print("Data preparation complete. Master DataFrame ready for analysis.")
```

```
# ============================================================================
# Step 5: Creating Final Analysis DataFrame
# ============================================================================
print("Creating final DataFrame for modeling...")

key_cols = [
    # Response Variables (Y)
    'fastestLapTime_ms', 'positionOrder', 'podium',
    # Predictor Variables (X)
    'year', 'circuitId', 'grid', 'qualifying_position', 'q1_ms', 'q2_ms',
'q3_ms',
    'avg_lap_time_ms', 'std_lap_time_ms', 'driver_age_at_race',
    # ID Columns
    'driverRef', 'constructor_name'
]

final_df = master_f1[key_cols].copy()
final_df.dropna(subset=['fastestLapTime_ms'], inplace=True) # Drop rows with
missing target variable

print(f"Final DataFrame shape: {final_df.shape[0]} rows and {final_df.shape[1]}
columns.")



# ============================================================================
# Step 6: Graphical Analysis
# ============================================================================
print("Generating output visualizations...")

# Data Sample for Table 1
ds1 = final_df.head()
print("\nData Sample for Table 1:")
print(ds1[['fastestLapTime_ms', 'positionOrder', 'podium', 'year', 'grid',
'qualifying_position', 'avg_lap_time_ms', 'driverRef']].to_markdown(index=False))

# Descriptive Statistics
desc_stats = final_df[['fastestLapTime_ms', 'positionOrder', 'grid',
'qualifying_position', 'avg_lap_time_ms', 'driver_age_at_race']].describe()
print("\nDescriptive Statistics:")
print(desc_stats.to_markdown(floatfmt=".2f"))

# Correlation Heatmap
print("Generating correlation heatmap...")
```

```
num_cols_for_corr = [
    'fastestLapTime_ms', 'positionOrder', 'podium', 'grid',
    'qualifying_position', 'q1_ms', 'q2_ms', 'q3_ms',
    'avg_lap_time_ms', 'std_lap_time_ms',
    'driver_age_at_race', 'year'
]
corr_matrix = final_df[num_cols_for_corr].corr()

plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='magma', vmin=-1, vmax=1)
plt.title('Correlation Heatmap of Key F1 Variables')
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig('f1_correlation_heatmap.png')

print("\n'f1_correlation_heatmap.png' is saved.")



# ==============================================================================
# Step 7: Pre-Modeling Data Preparation
# ==============================================================================
print("Preparing data for modeling...")

# Defining Predictors (X) and Target (y)
X = final_df.drop(columns=['fastestLapTime_ms', 'positionOrder', 'podium',
'driverRef', 'constructor_name'])
y1 = final_df['fastestLapTime_ms'] # Q1
y2 = final_df['positionOrder'] # Q2
y3 = final_df['podium'] # Q3

print(f"X shape: {X.shape}, y1 shape: {y1.shape}")

# Handle missing values
imputer = SimpleImputer(strategy='median')
X_imputed = imputer.fit_transform(X)
X_imputed = pd.DataFrame(X_imputed, columns=X.columns) # Convert back to
DataFrame in order to keep column names
print("Missing data imputed using median.")

# Split into training and testing sets
X_train, X_test, y1_train, y1_test, y2_train, y2_test, y3_train, y3_test =
train_test_split(
```

```
    X_imputed, y1, y2, y3, test_size=0.3, random_state=7406
)
print(f"Data split: {len(X_train)} training samples and {len(X_test)} testing
samples.")

# Standardize features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
print("Feature scaling complete. Data is ready for modeling.")
print("\nSample of Scaled Training Data:")
print(X_train_scaled[:5])
```

Bibliography and Credits

1. *Formula 1 World Championship (1950 - 2024) Dataset*. Kaggle. Retrieved from: https://www.kaggle.com/datasets/rohanrao/formula-1-world-championship-1950-2020