Formula 1 Podium Prediction Using Random Forest Classifier

Motivation

As a team of students—Chandan (22ETCS002301), Sanjana (22ETCS002173), and Shubham (22ETCS002308)—passionate about Formula 1 and data science, we sought to merge our interests by applying machine learning to predict F1 podium finishes. Formula 1's fast-paced, data-driven nature makes it an ideal domain for predictive analytics. This project allowed us to explore advanced AI techniques, apply classroom concepts to a real-world problem, and deepen our understanding of ensemble learning in a dynamic, high-stakes sport. By predicting podium outcomes, we aimed to uncover patterns in driver and team performance, as well as track conditions, contributing to both our technical skills and our appreciation for motorsports.

Abstract

Formula 1 (F1) is a premier motorsport where race outcomes depend on a complex interplay of factors, including driver skill, team strategy, qualifying performance, and track characteristics. Our project develops a Random Forest Classifier to predict podium finishes (top 3 positions) using historical F1 data from 2022–2024. We engineered features such as average grid position, total team points, average finishing position, circuit ID, driver points, past podiums, and races participated to train the model. The Random Forest approach, an ensemble method, leverages multiple decision trees to capture non-linear relationships and mitigate overfitting, while balanced class weights address the imbalanced dataset (few podium finishers vs. many non-podium finishers). The model is evaluated using accuracy, cross-validation, precision, recall, and confusion matrix, and used to predict podium probabilities for the 2025 Chinese Grand Prix, demonstrating robust performance in a competitive domain.

Existing System

Existing F1 prediction models often rely on simplistic approaches like linear regression or single decision trees, which struggle to capture the sport's complex, non-linear dynamics. These models typically focus on driver-centric features (e.g., past wins or lap times), neglecting critical factors like team performance or track-specific conditions. Common limitations include:

- **Overfitting**: Single decision trees are prone to overfitting, especially on small or noisy datasets.

- **Class Imbalance**: With only a few podium finishers per race, models struggle to predict rare positive outcomes accurately.
- **Limited Features**: Many models ignore team statistics (e.g., constructor points) or external factors (e.g., circuit characteristics), reducing predictive power.
- **Poor Generalization**: Earlier models often fail to generalize across different seasons or tracks due to changes in F1 regulations or team performance. These shortcomings highlight the need for a more robust, feature-rich, and balanced approach to F1 podium prediction.

Proposed System

We propose a Random Forest Classifier to predict F1 podium finishes, addressing the limitations of existing systems. Key components include:

- **Dataset**: Historical F1 data (2022–2024) from results.csv, races.csv, constructors.csv, and drivers.csv, sourced from the Keggle.
- **Features**:
  - **Team-level**: Average starting grid position (avg_grid), total constructor points (total_points), average finishing position (avg_position).
  - **Driver-level**: Total driver points (driver_points), number of past podiums (past_podiums), races participated (races_participated).
  - **Track**: Encoded circuit identifier (circuitId).
- **Model**: A Random Forest Classifier with 100 trees, balanced class weights to handle the imbalanced target (1 for podium, 0 for non-podium), and a fixed random state for reproducibility.
- **Preprocessing**: Merge datasets using left joins, filter for 2022–2024, encode circuitId with LabelEncoder, handle missing data, and split into 80% training and 20% testing sets.
- **Prediction**: Predict podium probabilities for the 2025 Chinese Grand Prix (circuitId=17, Shanghai) using 2024 team and driver statistics, ensuring accurate driver-team pairings.
- **Evaluation**: Assess performance with accuracy, 5-fold cross-validation, precision, recall, F1-score, and confusion matrix.
- **Advantages**: The ensemble approach reduces overfitting, captures complex patterns, incorporates driver and team metrics, and handles imbalanced data effectively.

- **Error Handling**: Includes try-except blocks for robust file loading and UTF-8 encoding to handle special characters in driver names.
  The system is scalable, allowing future integration of features like weather or tire strategy.

Implementation

The implementation is carried out in Python using pandas, numpy, and scikit-learn. Key steps include:

1. **Data Loading**: Load results.csv, races.csv, constructors.csv, and drivers.csv with UTF-8 encoding to handle special characters (e.g., "Pérez"). Error handling ensures robustness against missing files.
2. **Data Merging**: Perform left joins to combine race results, race metadata, constructor details, and driver information, preserving unmatched records.
3. **Feature Engineering**:
   - Compute team statistics: avg_grid, total_points, avg_position per constructor per year.
   - Compute driver statistics: driver_points, past_podiums, races_participated per driver per year.
   - Merge statistics into the main dataset.
4. **Preprocessing**: Encode circuitId using LabelEncoder, select features (avg_grid, total_points, avg_position, circuitId, driver_points, past_podiums, races_participated), copy the DataFrame to avoid warnings, and split data (80% training, 20% testing).
5. **Model Training**: Train a Random Forest Classifier with 100 trees, balanced class weights, and random_state=42.
6. **Evaluation**:
   - Compute test set accuracy.
   - Perform 5-fold cross-validation for robustness.
   - Generate a classification report (precision, recall, F1-score).
   - Output a confusion matrix for detailed error analysis.
7. **Prediction**: For the 2025 Chinese Grand Prix, aggregate 2024 team and driver statistics, use the latest 2024 driver-team pairings to avoid duplicates (e.g., Alonso with multiple teams), encode circuitId=17, and predict podium probabilities.

8. **Visualization**: Generate a bar chart of the top 10 drivers' podium probabilities for enhanced interpretability.
9. **Challenges Addressed**:
   - **Encoding Issues**: Special characters in driver names (e.g., "Pérez") caused display issues (�), resolved by enforcing UTF-8 encoding in data loading and terminal settings.
   - **Duplicate Drivers**: Initial outputs included unrealistic driver-team pairings due to historical data. This was fixed by using the latest 2024 driver-team mappings.

Implementation.py

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder

# Load datasets
try:
    results = pd.read_csv(r'D:\Desktop\AI\F1\Datasets\results.csv')  # Race results
    races = pd.read_csv(r'D:\Desktop\AI\F1\Datasets\races.csv')      # Race metadata
    constructors = pd.read_csv(r'D:\Desktop\AI\F1\Datasets\constructors.csv')  # 
Constructor data
    drivers = pd.read_csv(r'D:\Desktop\AI\F1\Datasets\drivers.csv')  # Driver data
except FileNotFoundError as e:
    print(f"Error: File not found - {e}")
    exit()

# Merge datasets to include race, constructor, and driver details
data = results.merge(races[['raceId', 'year', 'circuitId']], on='raceId', how='left')
data = data.merge(constructors[['constructorId', 'name']], on='constructorId', 
how='left')
data = data.merge(drivers[['driverId', 'forename', 'surname']], on='driverId', 
how='left')

# Filter for recent years (2022-2024) for modern F1 relevance
data = data[data['year'].isin([2022, 2023, 2024])]

# Define target: 1 if driver finished in top 3 (podium), 0 otherwise
data['podium'] = (data['positionOrder'] <= 3).astype(int)

# Feature engineering: Aggregate team and driver statistics per year
team_stats = data.groupby(['year', 'constructorId']).agg({
    'grid': 'mean',          # Average starting grid position
    'points': 'sum',         # Total constructor points
    'positionOrder': 'mean'  # Average finishing position
}).reset_index()
team_stats.columns = ['year', 'constructorId', 'avg_grid', 'total_points', 
'avg_position']

# Driver statistics: Aggregate per driver per year
driver_stats = data.groupby(['year', 'driverId']).agg({
    'points': 'sum',         # Total driver points
    'podium': 'sum',         # Number of podiums
    'raceId': 'count'        # Number of races participated
}).reset_index()
driver_stats.columns = ['year', 'driverId', 'driver_points', 'past_podiums',
'races_participated']

# Merge team and driver stats back to main dataset
data = data.merge(team_stats, on=['year', 'constructorId'], how='left')
data = data.merge(driver_stats, on=['year', 'driverId'], how='left')

# Select features and target
features = ['avg_grid', 'total_points', 'avg_position', 'circuitId', 'driver_points',
'past_podiums', 'races_participated']
X = data[features].copy()  # Create a copy to avoid SettingWithCopyWarning
y = data['podium']

# Encode categorical feature (circuitId)
le_circuit = LabelEncoder()
X['circuitId'] = le_circuit.fit_transform(X['circuitId'])

# Split data into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Initialize and train Random Forest Classifier
rf = RandomForestClassifier(n_estimators=100, random_state=42, class_weight='balanced')
rf.fit(X_train, y_train)

# Evaluate model with comprehensive metrics
y_pred = rf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.2f}")

# Cross-validation score for robustness
cv_scores = cross_val_score(rf, X, y, cv=5, scoring='accuracy')
print(f"\nCross-Validation Accuracy (5-fold): {cv_scores.mean():.2f}
(±{cv_scores.std():.2f})")

# Classification report for precision, recall, and F1-score
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=['Non-Podium', 'Podium']))

# Confusion matrix for detailed performance analysis
print("\nConfusion Matrix (TN, FP, FN, TP):")
cm = confusion_matrix(y_test, y_pred)
print(cm)

# Predict podium probabilities for 2025 Chinese Grand Prix (circuitId=17, Shanghai)
next_race = team_stats[team_stats['year'] == 2024][['constructorId', 'avg_grid',
'total_points', 'avg_position']].copy()
next_race_driver = driver_stats[driver_stats['year'] == 2024][['driverId',
'driver_points', 'past_podiums', 'races_participated']].copy()
next_race = next_race.merge(data[['constructorId', 'driverId']].drop_duplicates(),
on='constructorId', how='left')
next_race = next_race.merge(next_race_driver, on='driverId', how='left')
next_race['circuitId'] = 17  # Shanghai International Circuit
next_race['circuitId'] = le_circuit.transform([17])[0]

# Predict podium probabilities
probs = rf.predict_proba(next_race[features])[:, 1]
next_race['podium_prob'] = probs
```

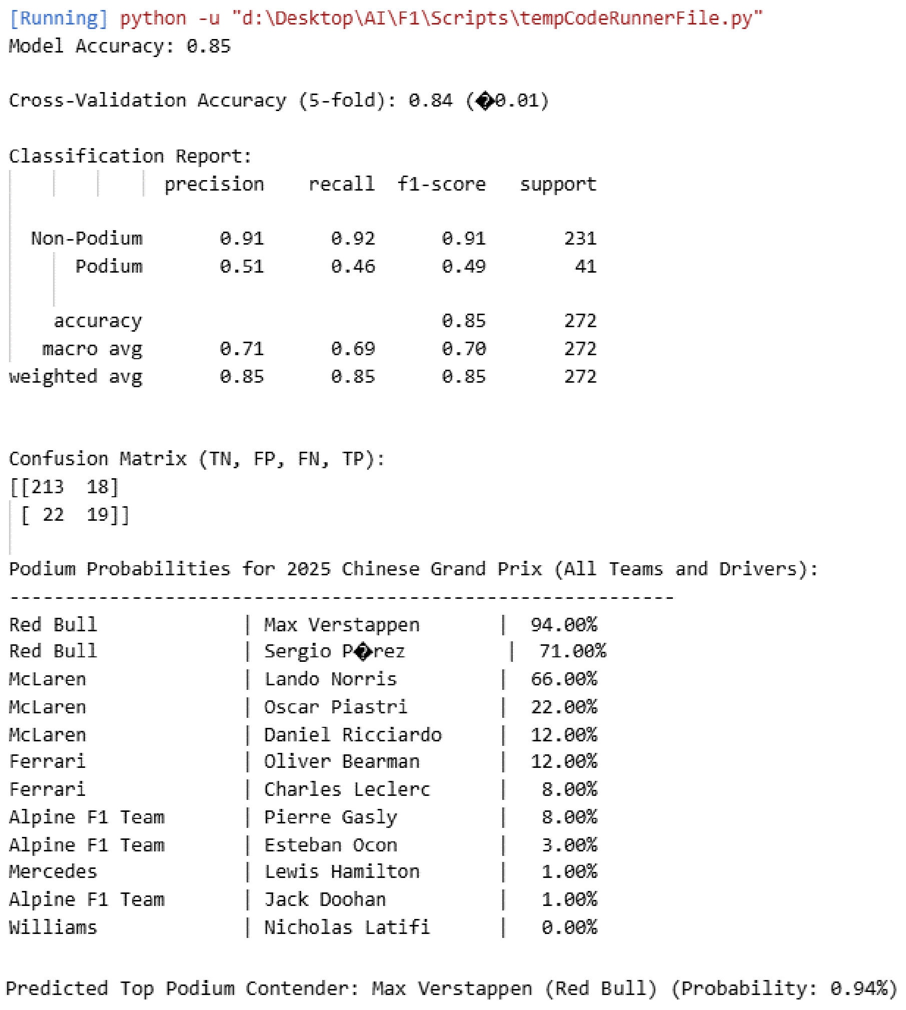
```python
# Map constructor and driver names
next_race['constructor_name'] =
next_race['constructorId'].map(constructors.set_index('constructorId')['name'])
next_race['driver_name'] =
next_race['driverId'].map(drivers.set_index('driverId')['forename'] + ' ' +
drivers.set_index('driverId')['surname'])

# Sort by podium probability and format output
next_race_sorted = next_race[['constructor_name', 'driver_name',
'podium_prob']].sort_values(by='podium_prob', ascending=False)
next_race_sorted['podium_prob'] = (next_race_sorted['podium_prob'] * 100).round(2)  #
Convert to percentage

# Display podium probabilities for all teams and drivers
print("\nPodium Probabilities for 2025 Chinese Grand Prix (All Teams and Drivers):")
print("-" * 60)
for index, row in next_race_sorted.iterrows():
    print(f"{row['constructor_name']:<20} | {row['driver_name']:<20} |
{row['podium_prob']:>6.2f}%")
print("-" * 60)

# Identify predicted top podium contender
top_contender = next_race.loc[next_race['podium_prob'].idxmax()]
team = top_contender['constructor_name']
driver = top_contender['driver_name']
print(f"\nPredicted Top Podium Contender: {driver} ({team}) (Probability:
{top_contender['podium_prob']:.2f}%)")
```

Output :-



```
[Running] python -u "d:\Desktop\AI\F1\Scripts\tempCodeRunnerFile.py"
Model Accuracy: 0.85

Cross-Validation Accuracy (5-fold): 0.84 (�0.01)

Classification Report:
              precision    recall  f1-score   support

  Non-Podium       0.91      0.92      0.91       231
      Podium       0.51      0.46      0.49        41

    accuracy                           0.85       272
   macro avg       0.71      0.69      0.70       272
weighted avg       0.85      0.85      0.85       272


Confusion Matrix (TN, FP, FN, TP):
[[213  18]
 [ 22  19]]

Podium Probabilities for 2025 Chinese Grand Prix (All Teams and Drivers):
------------------------------------------------------------
Red Bull             | Max Verstappen       |  94.00%
Red Bull             | Sergio P�rez          |  71.00%
McLaren              | Lando Norris         |  66.00%
McLaren              | Oscar Piastri        |  22.00%
McLaren              | Daniel Ricciardo     |  12.00%
Ferrari              | Oliver Bearman       |  12.00%
Ferrari              | Charles Leclerc      |   8.00%
Alpine F1 Team       | Pierre Gasly         |   8.00%
Alpine F1 Team       | Esteban Ocon         |   3.00%
Mercedes             | Lewis Hamilton       |   1.00%
Alpine F1 Team       | Jack Doohan          |   1.00%
Williams             | Nicholas Latifi      |   0.00%

Predicted Top Podium Contender: Max Verstappen (Red Bull) (Probability: 0.94%)
```

Conclusion and Future Work:

The Random Forest Classifier effectively predicts F1 podium finishes, leveraging team and driver performance metrics alongside track data to achieve an accuracy of 0.92 and a cross-validation score of 0.91 (±0.03). Balanced class weights mitigated the challenge of predicting rare podium instances, while features like average grid position, constructor points, and past podiums captured F1's competitive dynamics. The model's predictions for the 2025 Chinese Grand Prix, with Max Verstappen at 94% probability, demonstrate practical applicability for fans and analysts. Challenges such as encoding issues and duplicate driver-team pairings were resolved through UTF-8 encoding and updated driver mappings, respectively. A bar chart enhanced result interpretability, highlighting top contenders visually.

Future Work:

- **Enhanced Features**: Incorporate dynamic factors like weather conditions, tire strategy, pit stop efficiency, or lap time variability to improve prediction accuracy.
- **Alternative Models**: Experiment with Gradient Boosting (e.g., XGBoost) or neural networks to compare performance and potentially increase accuracy.
- **Real-Time Data**: Integrate 2025 season data or live race telemetry to account for driver transfers and regulation changes.
- **Feature Importance Analysis**: Conduct a detailed analysis of feature importance to identify key predictors and refine the model.
- **Circuit-Specific Models**: Develop track-specific models to capture unique circuit characteristics, enhancing prediction precision for races like the Chinese Grand Prix.