Cahier des Charges

Projet d'Analyse et de Prédiction de Données de Formule 1

Hajda Altin
Telley Cyril

28 Avril 2025

Table des matières

1 But général du projet

Développer un système automatisé pour la collecte, l'analyse et la prédiction des données issues de la Formule 1. Ce système inclura une interface utilisateur web (développée avec `Dash` ou `Streamlit`) permettant la visualisation des analyses et des prédictions générées.

2 Contexte et objectifs

Contexte : La Formule 1 est un sport générant une quantité importante de données à chaque événement. L'analyse de ces données historiques peut révéler des tendances, des schémas de performance et fournir des informations utiles pour comprendre les résultats passés et potentiellement anticiper les performances futures. Ce projet vise à exploiter ces données pour créer des outils d'analyse et de prédiction.

Objectifs spécifiques : Les objectifs spécifiques de ce projet sont :

1. Mettre en place un système de web scraping pour extraire les résultats des sessions (Essais Libres, Qualifications, Courses, Qualifications Sprints, Courses Sprints) depuis le site officiel `formula1.com`.
2. Intégrer l'API `FastF1` (via sa bibliothèque Python) pour acquérir des données télémétriques et contextuelles détaillées complémentaires.
3. Concevoir et implémenter une méthode de stockage structurée pour les données collectées, initialement sous forme de fichiers `CSV`.
4. Développer un pipeline de nettoyage et de prétraitement des données, incluant la fusion cohérente des informations provenant des différentes sources (`formula1.com` et `FastF1`).
5. Élaborer et entraîner des modèles de Machine Learning capables de prédire divers aspects des courses futures (ex : classement final, composition du podium, détenteur du meilleur tour en course).
6. Mettre en place un processus pour générer et actualiser ces prédictions avant chaque Grand Prix.
7. Créer une interface utilisateur web interactive (avec `Dash` ou `Streamlit`) pour permettre la visualisation des données historiques, des analyses et des prédictions générées par les modèles.

3 Présentation des données à utiliser

3.1 Sources de données

Le projet s'appuiera sur les sources de données principales suivantes :

- Site Web Officiel : `formula1.com` (pour les résultats des sessions).
- API `FastF1` : Bibliothèque Python `FastF1` (pour les données télémétriques, météo, pneus, etc.).

3.2 Description des données (attributs, quantité)

Période temporelle couverte : Les données couvriront a minima les 5 dernières saisons de Formule 1, ainsi que la saison en cours, en fonction du temps disponible pour la collecte.

Attributs envisagés par session : (Liste non exhaustive, à affiner lors de l'exploration)

- *Informations générales :* Année, nom/identifiant du Grand Prix, nom du circuit, dates, type de session (FP1, FP2, FP3, Q, SQ, R, SR).
- *Résultats (Classement) :* Position finale, numéro de voiture, nom complet du pilote, nom de l'écurie, temps final ou écart au leader, nombre de tours complétés, statut (ex : Fini, Abandon, +1 Lap), points marqués.
- *Données de Qualification / Sprint Qualification :* Temps réalisés dans chaque segment (Q1, Q2, Q3), classement final de la séance.
- *Temps au tour :* Meilleurs temps au tour par pilote pour chaque session, potentiellement l'historique des temps au tour (via `FastF1`).
- *Arrêts aux stands (Pit Stops) :* Nombre total d'arrêts, tour de chaque arrêt, durée de l'immobilisation (si disponible via `FastF1`).
- *Données `FastF1` complémentaires :* Données de télémétrie (vitesse, RPM, rapport engagé, position sur la piste), informations sur les pneus (type de gomme utilisé, usure/tours effectués), temps par secteur, conditions météorologiques (température air/piste, pluie).

3.3 Méthodes d'extraction et de stockage

`formula1.com` **(Web Scraping)** :

- Bibliothèques Python : `requests` pour les requêtes HTTP et `BeautifulSoup4` (ou `lxml`) pour l'analyse syntaxique (parsing) du HTML.
- Logique de navigation : Développement d'un script pour parcourir les archives par année, Grand Prix et type de session de manière automatisée.

`FastF1` **(API Client)** :

- Utilisation directe de la bibliothèque Python `FastF1` et de ses fonctions pour requêter et récupérer les données structurées (télémétrie, pneus, météo, etc.).

Stockage Initial :

- Format : Sauvegarde des données brutes extraites et des données traitées dans des fichiers au format `CSV`.
- Organisation : Structuration des fichiers par année, type de session ou autre critère pertinent.

4 Architecture globale et technologies envisagées

L'architecture du système est conçue pour être modulaire, avec des composants distincts et faiblement couplés afin de faciliter le développement, les tests et la maintenance future.

4.1 Modules principaux

Le système sera composé des modules logiques suivants :

Module	Rôle Principal
Module d'Extraction	Collecte des données depuis `formula1.com` (Scraping) et l'API `FastF1`.
Module de Gestion des Données	Lecture, écriture, validation et fusion des fichiers de données brutes et traitées (format `CSV`).
Module de Traitement	Nettoyage des données, prétraitement, et ingénierie de caractéristiques (feature engineering).
Module de Modélisation	Entraînement, évaluation, optimisation des hyperparamètres et sauvegarde des modèles de Machine Learning (ex : via `scikit-learn`).
Module de Prédiction	Chargement des modèles entraînés et génération des prédictions pour les nouvelles courses ou sessions.
Module d'Interface Utilisateur	Application Web (`Dash` ou `Streamlit`) pour la consultation interactive des données, analyses et prédictions.

4.2 Flux de données principal

Le flux de traitement des données suivra approximativement les étapes séquentielles suivantes :

(a) Extraction (Scraper / `FastF1` API) → Stockage initial (`CSV` bruts).

(b) Lecture (`CSV` bruts) → Traitement (Nettoyage, Fusion, Feature Engineering) → Stockage traité (`CSV` propres).

(c) Lecture (`CSV` propres) → Entraînement Modèles ML → Sauvegarde Modèles (fichiers `pickle`/`joblib`).

(d) Chargement (Modèles + Données nouvelles courses) → Script de Prédiction → Stockage Prédictions (fichier `CSV`/`JSON` ou autre).

(e) Lecture (Données Traitées + Prédictions) → Affichage via Interface Utilisateur.

4.3 Stockage

Le stockage principal des données et des modèles s'appuiera sur le système de fichiers local. Les données seront organisées en fichiers `CSV`. Les modèles de Machine Learning entraînés seront sauvegardés sous forme de fichiers sérialisés (par exemple, via `pickle` ou `joblib`).

4.4 Technologies envisagées

Les technologies suivantes sont pressenties pour la réalisation du projet :

Domaine	Outils / Bibliothèques
Langage de programmation	`Python`.
Manipulation de données	`Pandas`, `NumPy`.
Web Scraping	`Requests`, `BeautifulSoup4` ou `lxml`.
API F1	`FastF1`.
Machine Learning	`Scikit-learn` (pipelines, métriques, modèles de base, optimisation d'hyperparamètres). `XGBoost`, `LightGBM`, `CatBoost`
Interface Utilisateur (Web App)	`Dash` (basé sur Plotly et Flask) *ou* `Streamlit`.
Gestion de l'environnement	`venv` (intégré à Python) ou `conda`. Gestion des dépendances via un fichier `requirements.txt` ou `environment.yml`.

5 Techniques, méthodes et algorithmes envisagés

Analyse Exploratoire des Données (EDA) :

- Analyse statistique descriptive : Calcul des moyennes, médianes, écarts-types, quantiles des temps au tour, écarts, nombre d'arrêts, etc.
- Statistiques agrégées par pilote, équipe, circuit, saison pour identifier des tendances.
- Visualisations (réalisées avec des bibliothèques comme `Matplotlib`, `Seaborn`, `Plotly`, potentiellement via des notebooks Jupyter pour l'exploration ou intégrées dans l'UI finale) : Histogrammes, box plots, scatter plots, séries temporelles pour suivre l'évolution des classements, comparer les performances, étudier les corrélations.

Nettoyage et Prétraitement :

- Gestion des valeurs manquantes (NaN) : Application de techniques de suppression (ex : lignes incomplètes) ou d'imputation (ex : remplacement par la moyenne, médiane, mode, ou autres méthodes d'imputation).
- Standardisation des formats : Conversion des temps (en secondes), unification des formats de date/heure, normalisation des noms (pilotes, équipes).
- Encodage des variables catégorielles : Transformation des variables non numériques (noms de pilotes, équipes, circuits) en représentations numériques utilisables par les modèles ML (ex : One-Hot Encoding, Label Encoding, Target Encoding).
- Mise à l'échelle des variables numériques (Normalisation/Standardisation) : Utilisation d'outils comme `StandardScaler` ou `MinMaxScaler` de `scikit-learn`.

Ingénierie de Caractéristiques (Feature Engineering) :

- Création de variables basées sur l'historique : Performance récente du pilote/équipe (moyenne des points/positions sur les N dernières courses), performance historique sur le circuit considéré, résultats des qualifications de la course actuelle, position sur la grille de départ.
- Intégration des caractéristiques du circuit (ex : longueur, type - urbain/rapide, nombre de virages, sens de rotation).
- Données détaillées issues des sessions antérieures ou de `FastF1` : Moyenne/médiane/écart-type des temps au tour en essais/qualifications, nombre d'arrêts aux stands effectués.

- Variables relatives : Calcul de l'écart de performance par rapport au coéquipier, classement relatif dans le championnat avant la course.

Modélisation (Machine Learning) :

- *Priorité : Problèmes de Classification* :
  - Prédiction du Podium (Top 3) : Approche binaire (pilote X sera-t-il dans le top 3 ? Oui/Non) ou multi-classe (prédiction directe des 3 premiers - potentiellement plus complexe). Algorithmes envisagés : Régression Logistique, Forêts Aléatoires (Random Forest Classifier), Gradient Boosting (`LightGBM`/`XGBoost`).
  - Prédiction du Vainqueur : Problème de classification multi-classe (un pilote parmi N).
  - Prédiction du Top 10 (Points) : Approche binaire (pilote X finira-t-il dans les points ?).
- *Si le temps le permet : Problèmes de Classement ou Régression* :
  - Prédiction du Classement Complet : Peut être abordé comme une série de classifications binaires, une classification multi-classe ordonnée, ou une régression sur la position finale. Des algorithmes de type "Learning to Rank" pourraient être explorés.
  - Prédiction du Nombre d'Arrêts aux Stands : Problème de régression (prédire un nombre entier).
  - Prédiction du Meilleur Tour en Course : Classification multi-classe.

Optimisation et Évaluation des Modèles :

- Métriques adaptées au problème :
  - Classification : Exactitude (Accuracy), Précision, Rappel (Recall), F1-Score (pondéré ou macro), Log Loss (pour les probabilités), Aire sous la courbe ROC (AUC-ROC), Matrice de confusion. Pour le podium, des métriques spécifiques comme "pourcentage de podiums correctement prédits" ou "présence dans le top 3 prédit".
  - Régression : Erreur Absolue Moyenne (MAE), Erreur Quadratique Moyenne (RMSE), R.
- Techniques de validation : La validation croisée temporelle (Time Series Cross-Validation) est importante. Par exemple : entraîner sur les saisons $1..N-2$, valider/optimiser sur la saison $N-1$, tester sur la saison N. Ceci simule le processus de prédiction où le modèle n'a pas accès aux données futures. Mise de côté d'un jeu de test (ex : la saison la plus récente non utilisée pour l'entraînement/validation) pour l'évaluation finale.
- Optimisation des hyperparamètres : Utilisation de techniques comme Grid Search, Random Search, ou optimisation bayésienne (avec des outils comme `Optuna` ou `Hyperopt`) sur l'ensemble de validation défini par la validation croisée temporelle.

6 Résultats attendus

À l'issue du projet (échéance), les livrables suivants sont attendus :

- Un ensemble de scripts `Python` fonctionnels et commentés pour l'extraction des données (scraping/API), leur traitement, l'entraînement d'au moins un modèle prédictif prioritaire (ex : prédiction du podium) et la génération de prédictions pour de nouvelles courses.
- Une collection organisée de fichiers `CSV` contenant les données F1 structurées et nettoyées pour les saisons ciblées.
- Au moins un modèle de Machine Learning entraîné, optimisé, évalué et sauvegardé (ex : fichier `pickle`/`joblib`) pour la tâche de prédiction principale définie.

- Des fichiers de sortie (ex : `CSV`, `JSON`) contenant les prédictions générées pour les courses à venir (ou un exemple basé sur des données historiques pour la démonstration).
- Une application web locale (`Dash` ou `Streamlit`) fonctionnelle permettant à minima :
  - La consultation et le filtrage des données historiques.
  - L'affichage des prédictions générées par le modèle principal (ex : probabilités pour le podium).
  - (Optionnel) Quelques visualisations pertinentes issues de l'EDA ou de l'analyse des résultats du modèle.
- Une documentation technique (fichier `README.md` ou documentation dédiée) décrivant l'architecture du projet, les instructions d'installation des dépendances (`requirements.txt`/`environment.yml`), et le mode d'emploi pour exécuter les différentes composantes (extraction, entraînement, prédiction, application web).

7 Risques et Stratégies d'Atténuation

Risques Identifiés et Stratégies : • *Maintenance/Blocage du Scraper :* Le site `formula1.com` peut changer sa structure ou implémenter des mesures anti-scraping.

Atténuation : Code de scraping modulaire et commenté pour faciliter les mises à jour, utilisation de délais entre les requêtes, gestion des erreurs attentive, user-agents variables, monitoring. Prioriser l'utilisation de `FastF1` lorsque possible.

- *Qualité/Disponibilité/Limitations des Données :* Incohérences, erreurs, délais dans les sources (`formula1.com`, API `FastF1`). Limitations de l'API `FastF1`.

  Atténuation : Validation attentive des données lors du nettoyage, détection d'anomalies, documentation des incohérences. Ne pas dépendre exclusivement d'une seule source pour les informations critiques si possible. Comprendre et documenter les limitations de `FastF1`.

- *Complexité du Nettoyage/Traitement :* La fusion et le nettoyage peuvent être chronophages.

  Atténuation : Allouer du temps dans la planification (Phase 2), procéder par itérations, documenter les étapes de nettoyage, utiliser des outils de profiling (`Pandas Profiling`) pour identifier les problèmes.

- *Performance des Modèles :* Difficulté de la prédiction en F1.

  Atténuation : Se concentrer sur une ingénierie de caractéristiques pertinente, utiliser les techniques de validation définies, définir des objectifs de performance réalistes, communiquer sur les limites des prédictions. Commencer par des modèles plus simples et itérer.

- *Gestion du Temps / Périmètre :* Le planning peut être serré pour la durée allouée.

  Atténuation : Priorisation des objectifs (Podium > Reste), communication entre les membres de l'équipe, découpage en tâches, flexibilité pour ajuster le périmètre si nécessaire.

8 Planification prévisionnelle des étapes du projet

Le projet a débuté le 4 Avril 2025 et se terminera le 16 Juin 2025. Le planning prévisionnel suivant, réparti sur environ 10 semaines, est proposé :

Phase 1 : *Initialisation & Exploration (env. 1.5 semaines : 4 Avril 2025 – 25 Avril 2025)*

- Analyse des sources de données (formula1.com, documentation `FastF1`).
- Définition de la structure des fichiers `CSV`.
- Développement d'un prototype du scraper pour les fonctionnalités essentielles.
- Finalisation du Cahier des Charges (ce document).
- Configuration de l'environnement de développement (`Python`, `venv`/`conda`, `Git`).
- Exploration de `FastF1` et collecte de données tests.

Phase 2 : *Développement Core Data Pipeline (env. 2.5 semaines : 28 Avril – 16 Mai 2025)*

- Finalisation du scraper web.
- Intégration de l'extraction de données via l'API `FastF1`.
- Implémentation des scripts de gestion des fichiers `CSV` (lecture/écriture, structure de dossiers).
- Lancement de la collecte des données historiques (focus sur les 5-6 dernières saisons).
- Implémentation du pipeline de nettoyage et de prétraitement (`Pandas`). Fusion et validation des données.

Phase 3 : *Modélisation & Prédiction (env. 2.5 semaines : 19 Mai – 30 Mai 2025)*

- Analyse Exploratoire des Données (EDA) sur les données nettoyées.
- Ingénierie de caractéristiques (Feature Engineering) et sélection.
- Développement du pipeline de modélisation (incluant prétraitement spécifique au modèle).
- Entraînement et évaluation (validation croisée temporelle) des modèles prioritaires (ex : classification podium).
- Optimisation des hyperparamètres des modèles retenus.
- Comparaison et sélection des meilleurs modèles.
- Sauvegarde des modèles entraînés et des pipelines.
- Développement du script de prédiction pour un nouveau Grand Prix.

Phase 4 : *Développement Interface Utilisateur (env. 1.5 semaines : 2 Juin – 13 Juin 2025)*

- Mise en place de l'architecture de l'application (`Dash` ou `Streamlit`).
- Développement des composants UI principaux :
  * Tableau interactif pour visualiser/filtrer les données historiques.
  * Section pour afficher les prédictions (ex : probabilités, classement prédit).
  * (Optionnel si temps) Graphiques issus de l'EDA ou pour illustrer les prédictions.
- Intégration de l'application avec les scripts de lecture des données et des prédictions. Test de l'interface.

Phase 5 : *Tests, Raffinement & Finalisation (env. 1 semaine : 9 Juin –)*

- Tests d'intégration de l'ensemble du pipeline (Extraction → Traitement → Prédiction → UI).
- Corrections des bugs et améliorations.

– Amélioration de l'UI et de la lisibilité/qualité du code (commentaires, docstrings).
– Rédaction de la documentation (`README.md`).
– Nettoyage du code et du repository `Git`.
– Préparation de la démonstration/présentation.
– Livraison du projet (code source, données traitées, modèles, documentation).