University of Brighton

Department of Computer Science

CI603 Data Mining

Data Mining Coursework

Module leader: Dr Gulden Uchyigit

A report submitted in partial fulfilment of the requirements of
the University of Brighton for the degree of
Bachelor of Science in *Computer Science*

26/05/2025

Executive summary

This report presents a data mining analysis of high-dimensional telemetry and race session data extracted from the F1 2020 simulation platform. The primary objective was to develop an analytical pipeline that transforms raw telemetry data into actionable insights, supporting strategic decision-making and performance profiling in motorsport environments.

Leveraging over one million telemetry records per session across 22 race simulations, this project implemented a scalable analysis pipeline within the Databricks environment using PySpark. The analytical workflow included data preprocessing, exploratory data analysis, feature engineering, machine learning, and rule-based inference methods. The aim was to extract actionable intelligence from raw simulation telemetry, proving suitability for real-world applications.

A key stage in the pipeline was the application of Principal Component Analysis for dimensionality reduction. This technique successfully retained 73% of the variance in lap time data using only two principal components, significantly reducing computational overhead whilst preserving critical information for subsequent tasks such as clustering and visualisation.

Clustering using the K-Means algorithm allowed for effective segmentation of driver and team performance. With a high Silhouette score of 0.975, the clustering results demonstrated strong internal cohesion and external separation, highlighting clear groupings within race telemetry and lap time metrics.

In the classification stage, a Random Forest model was employed to predict tyre compound usage based on telemetry features. The model achieved an accuracy of 74% with balanced precision and recall. Notably, visualisation of tyre surface and inner temperatures showed strong alignment with predicted compounds, enhancing interpretability.

The final analytical layer employed association rule mining using FP-Growth. This uncovered over 14,000 meaningful rules, with an average confidence of 72% and strong lift values. These rules revealed dependencies between driving inputs, environmental factors, and tyre strategy, supporting rule-based explainability and tactical insights.

In conclusion, this project demonstrates the value of an integrated data mining approach to extract knowledge from complex telemetry data. The findings validate that machine learning, when paired with statistical and rule-based techniques, can uncover hidden patterns, guide strategic decisions, and provide interpretable analytics. These capabilities are directly applicable to competitive domains where real-time data-driven decision-making is crucial. The project offers a replicable blueprint for adopting data mining in simulation environments or real-world telemetry systems across industries.

Report's total word count: 2738 words

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This project explores the application of data mining techniques to high-resolution telemetry and race data from F1 2020, a racing simulation developed by Codemasters. The dataset, sourced from Kaggle (Mine, 2020), includes detailed records across 22 race sessions, capturing participant details, session metadata, lap times, and granular telemetry such as speed, throttle, and G-forces. Due to the data's volume and complexity, the project is implemented using PySpark within a Databricks environment, enabling scalable processing and analysis (Apache Spark, 2025, Databricks, 2025).

Primary aim: To apply data mining techniques to telemetry data to derive actionable insights, helping to understand race dynamics, driver behaviour, and strategies.

Objectives:

- To clean and prepare multi-session telemetry data for analysis.
- To perform dimensionality reduction to simplify high-dimensional telemetry and timing data whilst maintaining variance.
- To apply clustering and classification techniques to profile driver/team performance and predict tyre strategies.
- To extract interpretable associations between telemetry patterns and race conditions using rule mining.
- To provide meaningful visualisations for exploratory analysis, clustering results, and classification outputs to support interpretability and decision-making.
- To demonstrate the practical utility of data mining in motorsport analytics.

Chapter 2

Dataset

2.1 Source

The datasets used in this project were sourced from Kaggle (Mine, 2020) and were generated by the F1 2020 simulation game developed by Codemasters. The data is derived from in-game telemetry exported via UDP streams during simulated races. This telemetry data captures real-time information on car dynamics, driver inputs, and environmental conditions across each frame of the race, making it highly detailed and time-sensitive.

Each race session has been simulated using consistent parameters, ensuring comparability across the datasets. These include an AI difficulty setting of 90, fixed weather conditions, 50% race length, and the absence of safety cars, including virtual. The starting order was randomly assigned, car damage settings were set to reduced, and standardised fuel strategies were applied (three additional laps of fuel at the start). This created a realistic yet controlled racing simulation environment suitable for analytical modelling. This structured and uniform configuration enhances the dataset's reliability for exploring driver performance, vehicle behaviour, and race strategies through data mining techniques.

2.2 Structure

The dataset is organised into 22 separate race sessions, each containing four structured data files containing distinct aspects of the race environment and driver performance. These files are consistently named and follow a unified schema across all sessions, enabling scalable analysis and streamlined integration.

Dataset	**Rows**	**Columns**	**Key attributes**
Participant data	20	6	pilot_index, driverId, teamId
Session data	1	6	weather, trackTemperature, trackId
Race time data	561	9	LapTime, sector1TimeInMS, sector2TimeInMS, sector3TimeInMS, carPosition, currentLapNum
Telemetry data	1,113,180	56	speed, throttle, steer, brake, clutch, gear, engineRPM, actualTyreCompound, tyresPressure

Table 2.1: An overview of the datasets from the F1 2020 simulation game

These datasets are loaded into PySpark DataFrames and cleaned before saving to the Parquet format for efficient retrieval. The telemetry data, being the largest dataset, provides the foundation for time-series analysis, clustering, and classification tasks. The schema is consistent across sessions, with identifiers facilitating joins across datasets. This modular and consistent structure supports both individual session analysis and cross-session aggregation, making it well-suited for large-scale data mining applications.

2.3 Pre-processing

Data preprocessing is a critical step in preparing the F1 2020 telemetry and race data for effective analysis. Although each race session requires specific cleaning operations, the consistency of the dataset schema across all sessions allows a unified cleaning pipeline to be applied to all sessions.

Task	**Description**
Parse slash-separated string arrays	Convert strings into numerical arrays for variables such as tyre temperatures, pressures, and surface types.
Normalise string fields	Trim the whitespaces, convert to lowercase, and replace spaces with underscores. This normalisation reduces variability caused by inconsistent text formatting.
Dropping duplicate records	Removing duplicate entries ensures data integrity.
Add session identifiers	Adds `session_id` field to each dataset to facilitate merging and comparative analysis across multiple sessions.
Dropping irrelevant or invariant fields	Columns such as `aiControlled` and `maxRPM` can be dropped from participant data as they do not contribute towards the analysis.
Maintain order	After cleaning, the records are sorted by key columns (e.g., `pilot_index`, `frameIdentifier`), preserving temporal and logical data integrity.

Table 2.2: Pre-processing tasks and their descriptions

The array parsing and string normalisation are implemented using a PySpark user-defined function (UDF). Due to the large volume of telemetry data, exceeding one million rows per session, the cleaning process is computationally intensive and time-consuming. To optimise subsequent analyses, the cleaned data for each session and data category is saved in Parquet format, which supports efficient storage and retrieval. The cleaning pipeline ensures that the data is consistently prepared for tasks such as exploratory data analysis, feature engineering, and machine learning.

```python
# UDF for converting strings to arrays
@udf(returnType=ArrayType(FloatType()))
def parse_slash_array(s):
    # --- Omitted: input validation ---
        return [float(x) for x in re.findall(r"\d+(?:\.\d+)?", s)]

# UDF for normalising strings
@udf(returnType=StringType())
def normalize_string(s):
    # --- Omitted: input validation ---
    return s.strip().lower().replace(" ", "_")

```

```
# Applies the string normalisation to all string columns
def normalize_all_string_columns(df):
    for field in df.schema.fields:
        if isinstance(field.dataType, StringType):
            df = df.withColumn(field.name, normalize_string(F.col(field.name)))
    return df

# Reusable function for cleaning datasets
def base_clean(df, session_id, normalize_strings=False):
    df = df.dropDuplicates()
    if normalize_strings:
        df = normalize_all_string_columns(df)
    return df.withColumn("session_id", F.lit(session_id))
```

Listing 2.1: Data cleaning functions

Chapter 3

Implementation

3.1 Exploratory Data Analysis (EDA)

Exploratory Data Analysis is essential for understanding the structure, quality, and distributions of the F1 telemetry and race datasets before applying more complex modelling techniques. Given the consistent schema across all 22 sessions, EDA operations can be efficiently applied to either individual sessions or across all sessions.

3.1.1 Dataset metadata

A range of reusable functions facilitates the identification of dataset metadata for each session (see Appendix A).

EDA function	Description
Schema	Shows field names and data types for verification.
Summary	Calculates count, mean, standard deviation, minimum, and maximum values for numerical columns.
Sample records	Shows the first data entries for initial data inspection.
Count	Prints the number of rows and columns to gain an understanding of the dimensionality.
Missing values	Displays the number of missing values.

Table 3.1: EDA metadata functions and their descriptions

The implementation of these EDA functions leverages a generic `apply_to_sessions()` method, which applies any given function to either all sessions or just the first session (`Session_1`). This modular design ensures consistency and scalability in metadata exploration across the full dataset.

```python
# Apply a function to either all sessions or just Session_1
def apply_to_sessions(data_dict, func, print_all=False, **kwargs):
    if print_all:
        for session_name, session_data in data_dict.items():
            print(f"\nRunning on {session_name}...")
            func(session_name, session_data, **kwargs)
    else:
        session_name = "Session_1"
        if session_name not in data_dict:
            print("Session_1 not found in data.")
```

```
            return
        print(f"\nRunning on {session_name}...")
        func(session_name, data_dict[session_name], **kwargs)

# Print the schema's of the dataset in a race session
def print_session_schemas(session_name, session_data):
    # --- Omitted: Print statements and category loop ---
            df.printSchema()

apply_to_sessions(cleaned_data, print_session_schemas)

# Print the description of the dataset, ignoring null values
def print_session_summary(session_name, session_data):
    # --- Omitted: Print statements and category loop ---
            df.dropna()
            df.describe().show()

apply_to_sessions(cleaned_data, print_session_summary)

# Print first entries of the datasets
def print_session_first_entries(session_name, session_data,
    number_of_entries=5):
    # --- Omitted: Print statements and category loop ---
            df.show(number_of_entries)

apply_to_sessions(cleaned_data, print_session_first_entries)

# Print dimensionality of the datasets
def print_session_count(session_name, session_data):
    # --- Omitted: Print statements and category loop ---
            print(f"\n{category} dataset size: {df.count()} rows x
    {len(df.columns)} columns")

apply_to_sessions(dfs, print_session_count)

# Print missing values
def print_missing_values(session_name, session_data):
    # --- Omitted: Print statements, category loop, column is null
    count ---
        print(f"\n{category} missing values:")
        for col_name, missing_count in missing_values.items():
            print(f" - {col_name}: {missing_count} missing values")

apply_to_sessions(cleaned_data, print_missing_values)
```

Listing 3.1: EDA dataset metadata code (Feng, 2021)

3.1.2 Visualisations

Visual analysis is performed using Databricks' built-in visualisation framework, accessed through the `display()` function (Databricks, 2025). This integration offers interactive plotting, with support for various plot types such as line charts, scatter plots, histograms, and box plots. The plots provide insights into temporal dynamics and distributions.

EDA function	**Description**
Column plotting	Plots specified columns, enabling focused inspection of relevant data (see figure 3.1)
Cross-session average plots	Plots aggregated mean metrics such as lap and sector times per driver, facilitating the identification of performance trends over multiple races (see figure 3.2).
Numerical distributions	Histograms highlight the frequency and spread of key metrics, such as lap times (see figure 3.3).
Outlier detection	Box plots highlight extreme values, which potentially identify anomalies (see figure 3.4).

Table 3.2: EDA plotting functions and their descriptions

Collectively, these visualisations complement the metadata analysis by providing insightful plots that guide subsequent feature engineering and machine learning stages.

```python
# plots the sepcified columns of the dataset
def plot_data(cleaned_data, category, suffix, display_cols=[]):
    # --- Omitted: Dataset selection ---
    display(df.selectExpr(*display_cols))

plot_data(cleaned_data, categories[3], session_suffixes[0], ["throttle", "speed", "brake"])

# Plot average telemetry data for each session
def plot_average(cleaned_data, category, avg_cols=[]):
    # --- Omitted: Aggregation, grouping, average column selection ---
    display(result_df.selectExpr(*select_expr))

avg_columns = ["LapTime", "sector1TimeInMS", "sector2TimeInMS", "sector3TimeInMS"]
plot_average(cleaned_data, categories[2], avg_columns)

# Plot numerical distribution by selecting numerical columns for histogram
def plot_numerical_distribution(df):
    # --- Omitted: Int and double column selection ---
    display(df.selectExpr(*num_cols))

plot_numerical_distribution(cleaned_data["Session_1"][categories[2]])

# Identifies outliers by selecting numerical columns for box plot
def plot_outliers(df):
    # --- Omitted: Int and double column selection ---
    display(df.selectExpr(*num_cols))

plot_numerical_distribution(cleaned_data["Session_1"][categories[2]])
```

Listing 3.2: EDA plotting functions (Feng, 2021)

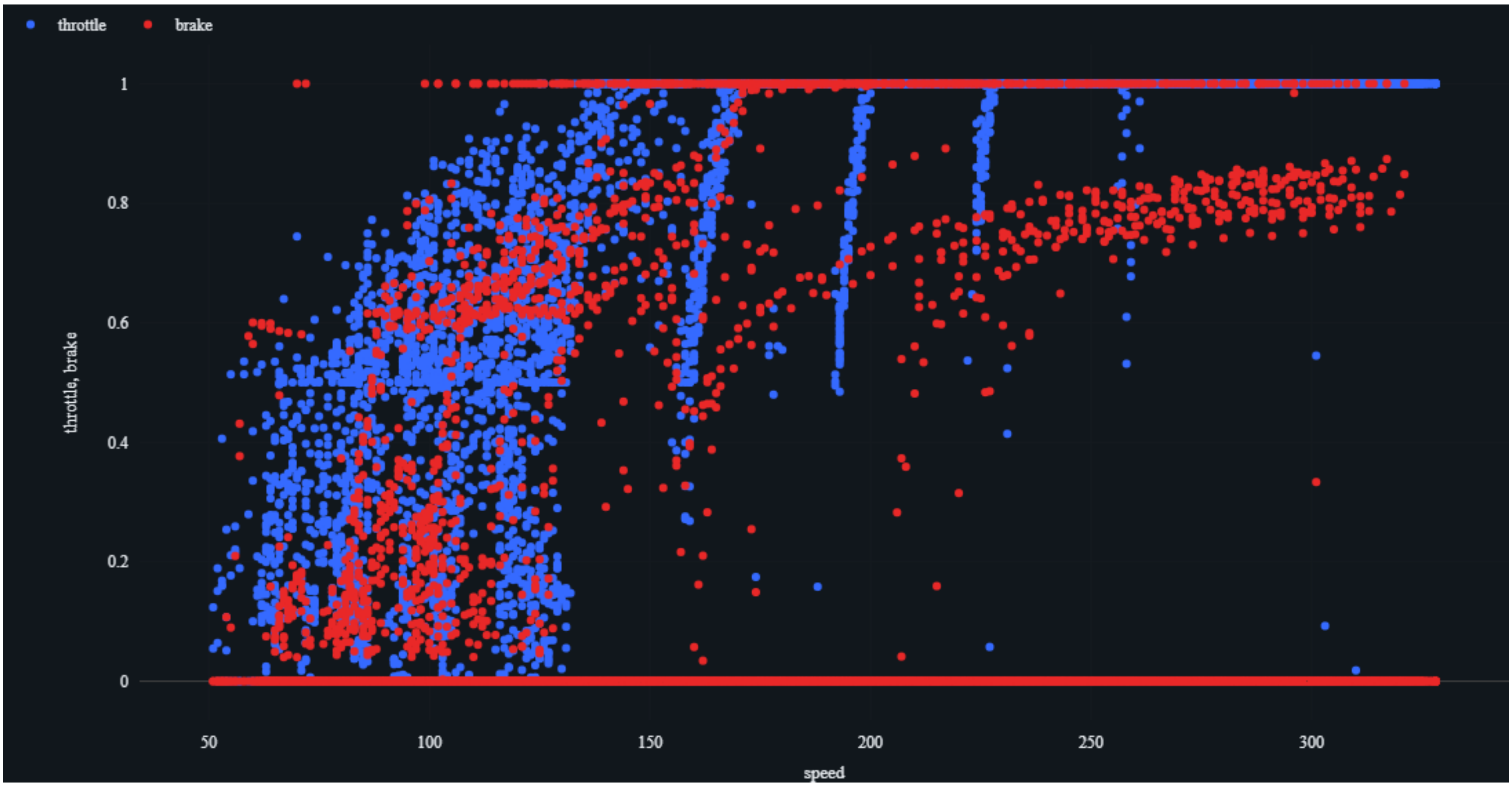

Figure 3.1: Scatter plot of speed against throttle and brake values.

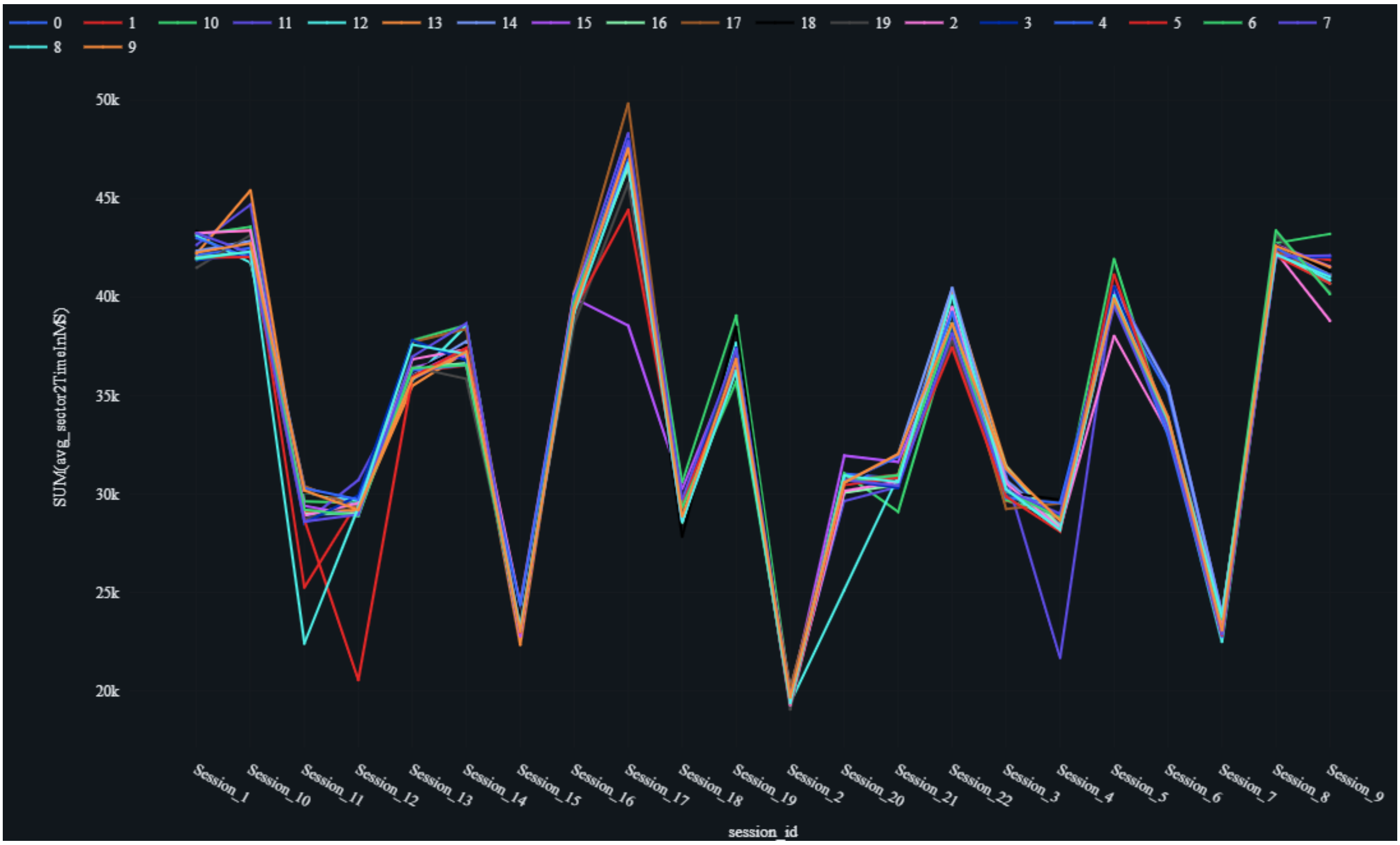

Figure 3.2: Line chart of average sector 2 time for each session grouped by driver.

Figure 3.3: Histogram of sector 2 time in milliseconds.

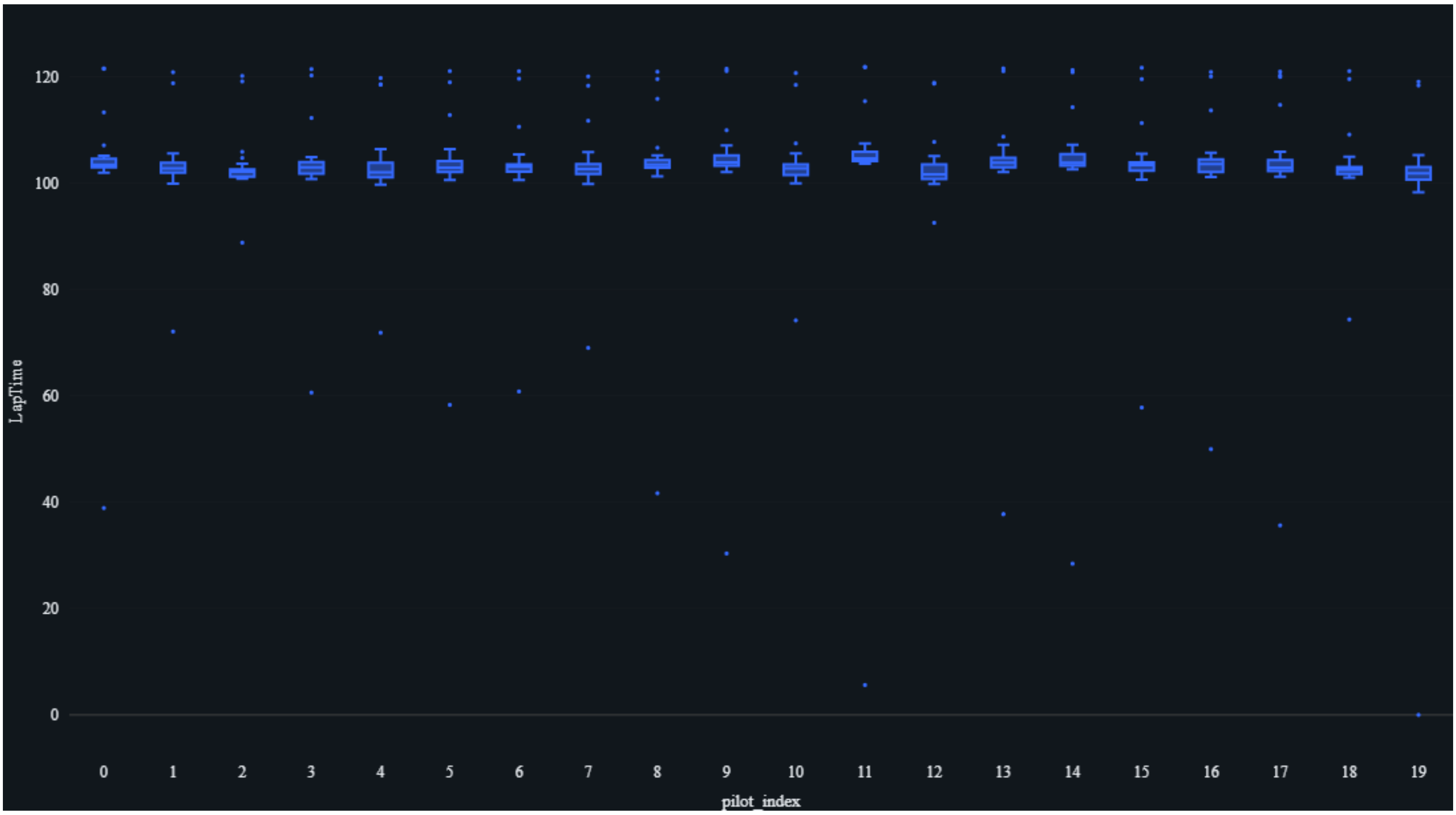

Figure 3.4: A box plot of lap times for each driver.

3.2 Feature engineering

Feature engineering is a crucial stage of data mining. This step converts raw data into suitable representations that improve the performance of machine learning models. This step is particularly important when working with high-dimensional datasets such as telemetry data, which consists of multiple time series and sensor variables. Without careful feature engineering, the presence of redundant or correlated features can hide underlying patterns and reduce the effectiveness of processes such as clustering and classification.

3.2.1 Principal Component Analysis (PCA)

Principal Component Analysis is a statistical method that reduces the number of variables in a dataset by projecting the data onto a set of principal components that capture the greatest variance. The first component captures the direction of maximum variance, the second captures the next highest variance orthogonal to the first, and so on (Uchyigit, 2025*b*). This makes PCA especially useful for multivariate telemetry data, where many features may be correlated. PCA is implemented in this project using a PySpark ML pipeline, ensuring scalability and modular integration with the larger analysis workflow. The implementation consists of three main stages:

Stage	Description
Vector assembly	The selected input features are combined into a single vector column using `VectorAssembler`.
Standardisation	All features are standardised using `StandardScaler` to ensure each contributes equally to the PCA transformation.
PCA transformation	The PCA model reduces the standardised features to k principal components, which are added as new columns for analysis.

Table 3.3: PCA pipeline stages and their descriptions

The PCA pipeline is contained in the `run_pca()` function, which takes the cleaned data and produces a transformed DataFrame containing the principal components. The number of components k is configurable based on the analysis task. This function can be applied to simplify complex data for visualisation or improve clustering performance by reducing noise and redundant dimensions.

```python
# Run PCA using the pipeline with k number of components
def run_pca(cleaned_data, suffix, category, k=3, features=None):
    # --- Omitted: Dataset selection and drop missing values ---
    assembler = VectorAssembler(inputCols=features, outputCol="
    raw_features")
    scaler = StandardScaler(inputCol="raw_features", outputCol="
    pca_features", withMean=True, withStd=True)
    pca = PCA(k=k, inputCol="pca_features", outputCol="pca_vector")

    pipeline = Pipeline(stages=[assembler, scaler, pca])
    model = pipeline.fit(df)
    result = model.transform(df)

    # --- Omitted: Conversion from PCA vector to array and
    individual columns ---

```

```
    pca_model = model.stages[-1]
    return pca_model, result
```

Listing 3.3: PCA function (Feng, 2021)

3.3 Clustering

Clustering is an unsupervised learning technique used to group observations in a dataset based on similarities, without the use of predefined labels. In the context of F1 data, clustering allows for the identification of underlying performance trends among drivers or teams, such as driving style, tyre usage, or sector performance consistency. These insights are valuable for strategic analysis, post-race evaluation, and driver profiling.

3.3.1 K-Means

The project uses K-Means clustering, a widely used algorithm that groups data into k non-overlapping clusters by minimising intra-cluster variance (Tan et al., 2019). K-Means is well-suited for this application due to its scalability, simplicity, and effectiveness when working with numerical, continuous telemetry features. Given the high dimensionality of telemetry and lap data, K-Means allows for compact performance differentiation, especially after applying dimensionality reduction (see section 3.2). For this project, clustering is implemented using a pipeline, and supports three input configurations: Telemetry data, lap time data, or a combination of both. The `perform_cluster_analysis` function supports both clustering by driver or by team, providing flexibility in performance grouping.

Stage	**Description**
Data preparation	Based on the analysis type, relevant datasets and features are selected and joined.
Missing value handling	Records with null values in selected features are removed.
Optional grouping	Data can be grouped by driver (`pilot_index`) or team (`teamId`) by calculating average values.
Vector assembly and scaling	Features are assembled into a single vector and standardised to ensure uniform scaling.
Model fitting	A K-Means model is applied to the scaled data, and cluster labels (`cluster_prediction`) are assigned to each record.

Table 3.4: Clustering stages and their descriptions

The number of clusters k is user-defined but can be optimised using the `find_no_of_clusters()` function. This method iteratively evaluates cluster quality using the Silhouette score, a metric that represents the cohesion of clusters and the separation between them (Feng, 2021). The optimal k is determined based on the maximum Silhouette score within a specified range.

Clustering enables natural performance grouping in the data that may not be obvious through direct observation. By applying clustering to multiple configurations (telemetry, lap times, and combined), the analysis covers both temporal driving patterns and lap-specific driving styles. This clustering approach improves the interpretation of driver behaviour and team performance across different metrics.

```
# Perform clustering analysis on the datasets with optional grouping
def perform_cluster_analysis(cleaned_data, suffix, analysis_type="combined", clusters=3, group_by=None, features=None):
    # --- Omitted: Input validation, dataset and feature selection, drop missing values, and dataset joining ---

    # Group by driver or team
    if group_by == "driver":
        df = df.groupBy("pilot_index").agg(*[F.avg(col).alias(f'avg_{col}') for col in feature_list])
        df = df.join(participant_data, on="pilot_index", how="left")
        feature_list = [f'avg_{col}' for col in feature_list]

    elif group_by == "team":
        df = df.join(participant_data, on="pilot_index", how="left")
        df = df.groupBy("teamId").agg(*[F.avg(col).alias(f'avg_{col}') for col in feature_list])
        feature_list = [f'avg_{col}' for col in feature_list]

    # --- Omitted: Dataset joining ---

    # Vector assembler, standard scaler, and k-means pipeline
    assembler = VectorAssembler(inputCols=feature_list, outputCol="cluster_features")
    scaler = StandardScaler(inputCol="cluster_features", outputCol="scaled_features", withMean=True, withStd=True)
    kmeans = KMeans(k=clusters, seed=1, featuresCol="scaled_features", predictionCol="cluster_prediction")

    pipeline = Pipeline(stages=[assembler, scaler, kmeans])
    model = pipeline.fit(df)
    result = model.transform(df)

    return result

# Iteratively find the most suitable number of clusters (range (2,10))
def find_no_of_clusters(cleaned_data, suffix, analysis_type="combined", group_by=None, features=None):
  evaluator = ClusteringEvaluator(
    predictionCol='cluster_prediction',
    featuresCol='scaled_features',
    metricName='silhouette',
    distanceMeasure='squaredEuclidean'
  )

  silhouette_scores = {}

  for k in range(2,10):
    result_df = perform_cluster_analysis(cleaned_data, suffix, analysis_type, k, group_by, features)
    score=evaluator.evaluate(result_df)
```

```
        silhouette_scores[k] = score
        print(f'Silhouette Score for k = {k} is {score}')

    best_k = max(silhouette_scores, key=silhouette_scores.get)
    best_score = silhouette_scores[best_k]
    print(f"\nBest k = {best_k} with silhouette score = {best_score:.4f}")
    return best_k

best_k = find_no_of_clusters(cleaned_data, session_suffixes[0], analysis_type="lap_time")
```

Listing 3.4: Clustering code (Feng, 2021)

3.4 Classification

Classification is a supervised machine learning technique that involves predicting discrete target labels based on a set of input features. In the context of this project, classification is used to predict the tyre compound used during a session based on telemetry and environmental data. Tyre strategy is one of the most critical variables influencing race performance in Formula 1, and the ability to derive a tyre compound from telemetry data can support both retrospective analysis and strategic forecasting.

3.4.1 Random Forest

The project utilises the Random Forest algorithm, an ensemble learning method based on decision trees. A Random Forest builds multiple decision trees during training and outputs the class that is most common in the individual trees (Tan et al., 2019). This approach can handle high-dimensional data and offers robustness to noise and over-fitting, as averaging over multiple trees reduces variance. Random Forest is particularly appropriate for this use case because the underlying relationship between telemetry data and tyre compound is non-linear and potentially complex. Classification is implemented using features derived from both car telemetry and session environmental data.

Stage	**Description**
Data integration	Telemetry and session data are retrieved and joined using the session ID to form a complete dataset.
Feature aggregation	Telemetry variables that are stored as arrays, such as tyre temperature and wear, are aggregated to average values for simplicity and compatibility.
Label encoding	Maps `actualTyreCompound` to numeric labels using `StringIndexer`.
Vectorisation	The input features are combined into a single vector column.
Data splitting	The data is randomly split into training and testing sets (80/20 split).
Model training	The Random Forest classifier model is fit to the training dataset.
Prediction	Apply the trained model to the test dataset to generate prediction labels.

Table 3.5: Classification stages and their descriptions

This classification task complements earlier data mining stages by validating whether telemetry and environmental data can reliably predict tyre strategy. In real-world applications, such models could assist teams in analysing competitor strategies, enhance commentary, or support data-driven strategy decisions.

```python
# Classify session data using a classification pipeline and random split
def classify_session_data(df, feature_cols, target_col, label_col="label", prediction_col="prediction"):
    # --- Omitted: Input validation, column selection, drop missing values ---

    indexer = StringIndexer(inputCol=target_col, outputCol=label_col)
    assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")
    clf = RandomForestClassifier(featuresCol="features", labelCol=label_col, predictionCol=prediction_col, seed=1)

    pipeline = Pipeline(stages=[indexer, assembler, clf])

    train_df, test_df = df.randomSplit([0.8, 0.2], seed=1)

    model = pipeline.fit(train_df)
    predictions = model.transform(test_df)

    return model, predictions

# Classify tyre compound using telemetry data
def classify_tyre_compound(cleaned_data, suffix):
    # --- Omitted: Data retrieval and joining ---

    # Aggregate array columns to a single average value
    df = df.withColumn(
        "avg_tyresSurfaceTemperature",
        F.expr("aggregate(tyresSurfaceTemperature, 0D, (acc, x) -> acc + x) / 4")
    )
    # --- Omitted: Additional aggregation, feature and target selection ---

    return classify_session_data(df,feature_cols=features, target_col=target)
```

Listing 3.5: Classification code (Feng, 2021)

3.5 Association rule mining

Association rule mining is a data mining technique used to discover meaningful relationships between variables in large datasets. In the context of F1 telemetry, association rules can reveal underlying connections between driving behaviours, environmental conditions, and tyre strategies. These patterns are particularly valuable for gaining interpretability, identifying strategies, and supporting data-driven decision making.

3.5.1 FP-Growth

This project leverages the Frequent Pattern Growth (FP-Growth) algorithm to perform association rule mining. FP-Growth uses a prefix tree structure (FP-tree) to compress and navigate the dataset more efficiently (Tan et al., 2019, Uchyigit, 2025*a*). FP-Growth is particularly well-suited for this project, as the algorithm is computationally efficient, can handle high-dimensional categorical data effectively, and can generate interpretable and actionable rules. Given that the telemetry data is naturally transactional and contextual, association rule mining offers a unique perspective not captured by supervised learning or clustering alone.

The discretisation process converts continuous telemetry signals into meaningful ranges, with category cut-off values derived from EDA calculations such as mean, standard deviation, minimum and maximum values. Most features are discretised into three categories (low, medium, and high) to capture general behavioural patterns. However, certain features, such as gear and weather, are divided into more categories to enhance the granularity and accuracy of the association rules. These discretised variables, combined into transactional arrays, are passed to the `FPGrowth` model, which is configured with a default minimum support of 0.01 and minimum confidence of 0.3. As a result, rules are only generated if they are sufficiently frequent and reliable across the session data.

Stage	**Description**
Feature selection	Choose relevant telemetry and session features, such as speed, throttle, and gear.
Data integration	Retrieve and join telemetry data with session data on `session_id`.
Discretisation	Bin numerical features into labelled categories (e.g., `speed=fast`).
Label generation	Convert the `actualTyreCompound` into a labelled format (e.g., `tyre=soft`).
Transaction assembly	Aggregate categorical labels into an `items` array representing each observation.
Model training	Apply FP-Growth to identify frequent itemsets and generate association rules.

Table 3.6: Association stages and their descriptions

Association rule mining complements the earlier stages by providing rule-based interpretability. FP-Growth identifies explicit logical relationships between telemetry conditions and tyre selections. In practice, this approach enables the identification of specific combinations of telemetry variables that frequently result in the use of a particular tyre compound. It also reveals behavioural patterns that differentiate the tyre compounds, providing additional insights into tyre strategy. This interpretability aligns closely with the objectives of real-world motorsport analytics, where understanding the relationships behind strategic choices is just as critical as making accurate predictions.

```
# Run FP growth on the session data
def run_fpgrowth_on_session(df, min_support=0.01, min_confidence=0.3):
    fpg = FPGrowth(itemsCol="items", minSupport=min_support, minConfidence=min_confidence)
    model = fpg.fit(df)
    return model.freqItemsets, model.associationRules
```

```

# Run association rule mining on telemetry data
def run_association(cleaned_data, suffix, cols_to_include=None):
    # --- Omitted: feature selection, data retrieval and joining, drop missing values ---

    # Column binning
    if "speed" in cols_to_include:
        df = df.withColumn("speed_cat",
            F.when(F.col("speed") < 100, "speed=slow")
            .when(F.col("speed") < 200, "speed=medium")
            .otherwise("speed=fast")
        )

    # --- Omitted: additional discretisation ---

    # Add label to tyre compound
    if "actualTyreCompound" in cols_to_include:
        df = df.withColumn("actualTyreCompound_cat",
            F.concat_ws("=", F.lit("tyre"), F.col("actualTyreCompound"))
        )

    # Combine columns to include into a single column
    df = df.withColumn("items", F.array(*[F.col(f"{c}_cat") for c in cols_to_include]))

    return run_fpgrowth_on_session(df)
```

Listing 3.6: Association rule code (Feng, 2021)

Chapter 4

Evaluation

4.1 Dimensionality reduction

Evaluating the effectiveness of Principal Component Analysis (PCA) is essential to ensure that the dimensionality reduction retains sufficient information from the original features. Since the principal components aim to maximise variance, the quality of the transformation is assessed through the explained variance. This metric indicates how much of the original dataset's variability is preserved by each component, and thus how informative the reduced representation is. A well-performing PCA transformation will typically result in a high cumulative explained variance in the first few components, suggesting that most of the data's structure can be retained with minimal loss. This is particularly important when PCA is followed by clustering or visualisation, as it ensures that meaningful patterns are not removed.

In this project, PCA evaluation is performed using the `evaluation_reduction` function, which calculates the explained variance for the top k components. The function returns the proportion of variance captured by each principal component and their total, providing a quantitative basis for deciding how many components to retain (see Appendix B.1).

Metric	Value	Description
Total variance	0.729	73% of the lap time variability is retained with only 2 principal components.
PC1 variance	0.522	The first principal component contains 52% of the features' variability.
PC2 variance	0.207	The second principal component contains 21% of the features' variability.

Table 4.1: PCA evaluation results of lap time data with 2 principal components

```python
# Evaluates PCA by calculate the variance
def evaluate_reduction(cleaned_data, suffix, category, features, k
    =3):
  pca_model, pca_result = run_pca(cleaned_data, suffix, category, k
    , features)

  explained_variance = pca_model.explainedVariance.toArray()

  result = {"total_variance": sum(explained_variance)}
  for i in range(k):
    result[f"PC{i+1}_variance"] = {explained_variance[i]}
```

```

  return result

# Feature selection for lap times and telemetry
features_time = ["LapTime", ..., "carPosition"]
featuers_telem = ["throttle", ..., "worldVelocityZ"]

# Run evaluation
result = evaluate_reduction(cleaned_data=cleaned_data, suffix=
    session_suffixes[0], category=categories[2], features=
    features_time, k=2)
print(result)
```

Listing 4.1: PCA evaluation code (Feng, 2021)

4.2 Clustering

The performance of clustering models is evaluated using the Silhouette score. A higher Silhouette score indicates more cohesive and well-separated clusters, with a maximum value of 1.0. In this project, the `evaluate_clustering()` function is used to calculate the Silhouette score and assess the quality of the resulting cluster assignments by the K-Means algorithm. Additionally, the function analyses cluster distribution. This helps ensure that the clustering model is not only quantitatively valid but also practically meaningful (see Appendix B.2).

Metric	**Value**	**Description**
Silhouette score	0.975	Shows excellent internal cohesion and external separation based on the scaled features and cluster predictions.
Cluster distribution	89.3%, 7.49%, 1.07%, 0.89%, 0.36%, 0.89%	Represents the distribution of points across clusters, indicating that the majority of lap time data is significantly similar.

Table 4.2: Clustering evaluation results for lap time clustering with $k = 6$

```
# Use clustering evaluator to calculate silhouette score
def evaluate_clustering(df, features_col="scaled_features",
    prediction_col="cluster_prediction"):
    evaluator = ClusteringEvaluator(
        featuresCol=features_col,
        predictionCol=prediction_col,
        metricName="silhouette"
    )
    silhouette = evaluator.evaluate(df)

    cluster_sizes = df.groupBy(prediction_col).count().orderBy(
    prediction_col)
    cluster_distribution = cluster_sizes.withColumn(
        "percentage",
        F.round((F.col("count") / df.count()) * 100, 2)
    )

    # --- Omitted: print and return statements ---

```

```python
df = perform_cluster_analysis(cleaned_data, session_suffixes[0], "lap_time", 6)
evaluate_clustering(df)

# Plot clustering results of two feature columns
def plot_cluster(cleaned_data, suffix, analysis_type, clusters=3, group_by=None, features=[]):
    df = perform_cluster_analysis(cleaned_data, suffix, analysis_type, clusters, group_by, features)
    if group_by == "team":
        features.append("teamId")
    else:
        features.append("pilot_index")
    features.append("cluster_prediction")
    display(df.selectExpr(*features)) # Scatter plot grouped by cluster predictions

plot_cluster(cleaned_data, session_suffixes[0], "lap_time", clusters=4, features=["sector2TimeInMS", "sector3TimeInMS"])

# Apply PCA before clustering, display results.
def plot_pca_cluster(cleaned_data, suffix, category, pca_k=2, clusters=3, group_by=None, display_cols=["PC1", "PC2", "cluster_prediction"]):
    # --- Omitted: feature selection ---
    pca_model, pca_result = run_pca(cleaned_data, suffix, category, 2, features)
    df = perform_cluster_analysis(pca_result, suffix, "PCA", clusters, group_by, features=["PC1", "PC2"])
    display(df.selectExpr(*display_cols)) # Scatter plot grouped by cluster predictions

plot_pca_cluster(cleaned_data, session_suffixes[0], categories[3], clusters=3)
```

Listing 4.2: Clustering evaluation code (Feng, 2021)

Further analysis was supported visually through scatter plots of selected features. A 2D scatter plot of `sector2TimeInMS` and `sector2TimeInMs` (see figure 4.1) shows how lap time features are grouped into clusters. Additionally, clustering was applied to PCA-reduced telemetry data using PC1 and PC2, resulting in a clear visual representation of clusters (see figure 4.2). This confirms the effectiveness of PCA as a pre-processing step for high-dimensional telemetry clustering.

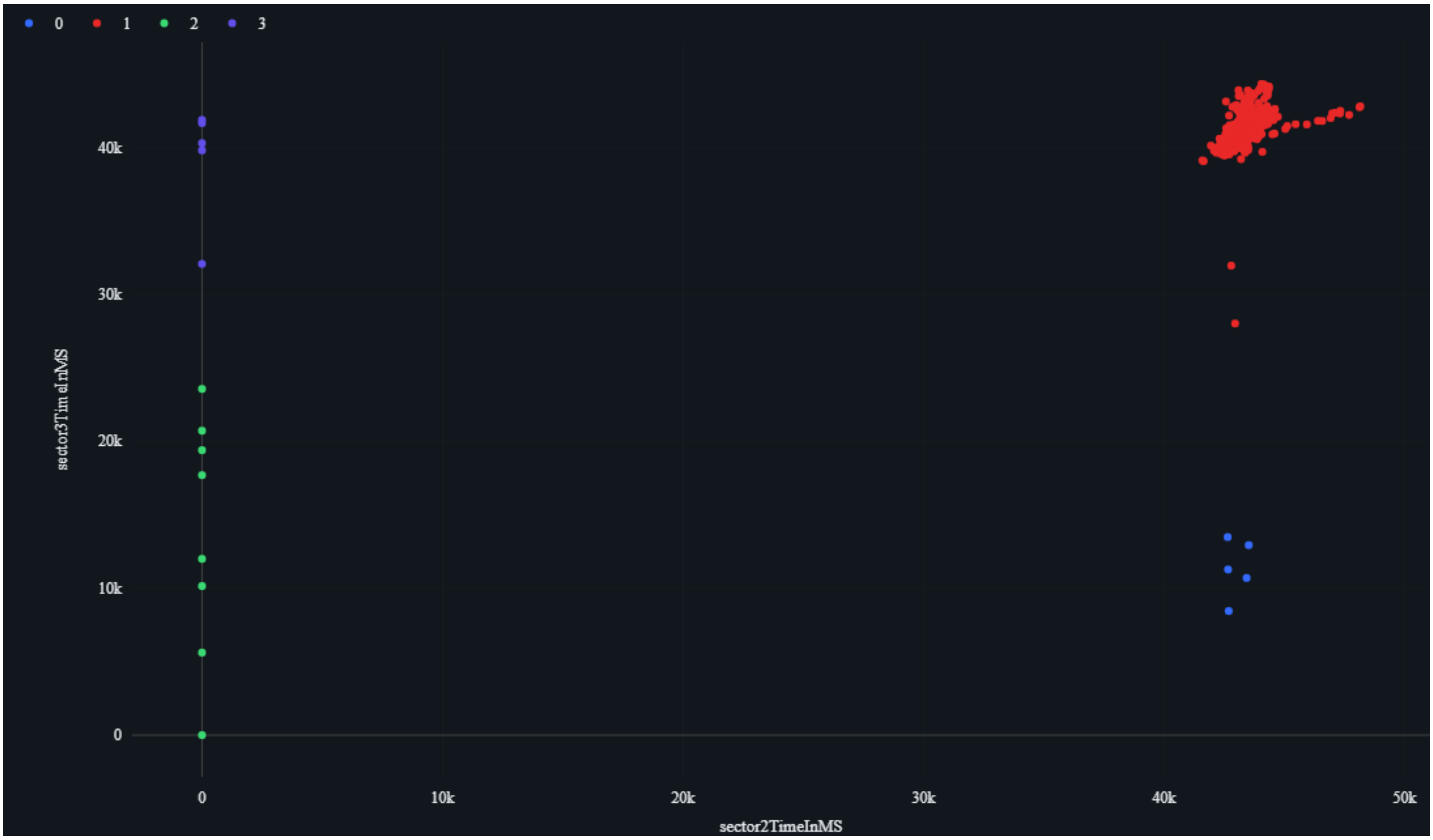

Figure 4.1: Scatter plot of sector 2 and 3 timings grouped by cluster analysis results.

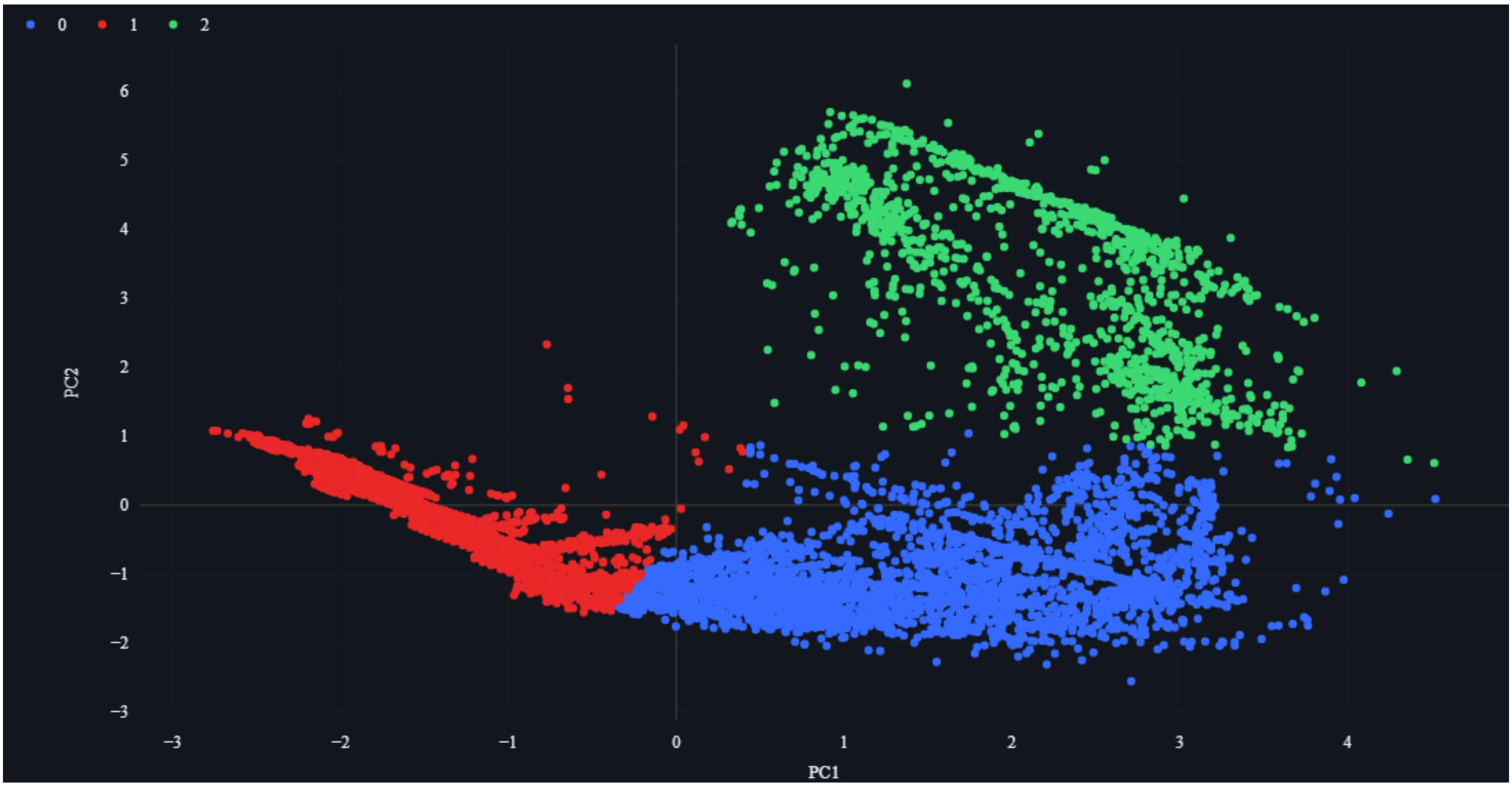

Figure 4.2: Scatter plot of PC1 and PC2 telemetry data grouped by cluster analysis results.

4.3 Classification

The performance of the classification model is evaluated using a combination of standard metrics suitable for multiclass classification, including accuracy, F1 score, precision, and recall (see Table 4.3). These metrics provide a comprehensive view of the model's predictive ability, particularly in the context of class imbalances. The evaluation is conducted using PySpark's `MulticlassClassificationEvaluator`, which applies each metric to the predicted and actual class labels produced by the classification pipeline (see Appendix B.3).

Metric	**Value**	**Description**
Accuracy	0.739	Approximately 74% of predictions matched the actual tyre compound.
F1 score	0.737	The score suggests a balanced trade-off between precision and recall.
Precision	0.737	Proportion of correctly predicted compounds among all predicted compounds.
Recall	0.739	Proportion of correctly predicted compounds among all actual compounds.
True positives	62,364	Number of correctly predicted tyre observations (label = 1.0).
False positives	33,361	Number of incorrectly predicted tyre observations (label = 0.0).
False negatives	24,740	Number of incorrectly predicted tyre observations (label = 1.0).
True negatives	101,976	Number of correctly predicted tyre observations (label = 0.0).

Table 4.3: Classification evaluation results for classifying actual tyre compound

```python
# Use multiclass classification evaluator to calculate accuracy and f1 score
def evaluate_classification(predictions, label_col="label", pred_col="prediction"):
    evaluator = MulticlassClassificationEvaluator(labelCol=label_col, predictionCol=pred_col)

    metrics = {
        "accuracy": evaluator.evaluate(predictions, {evaluator.metricName: "accuracy"}),
        "f1": evaluator.evaluate(predictions, {evaluator.metricName: "f1"}),
        "precision": evaluator.evaluate(predictions, {evaluator.metricName: "weightedPrecision"}),
        "recall": evaluator.evaluate(predictions, {evaluator.metricName: "weightedRecall"})
    }

    confusion_matrix = predictions.groupBy(label_col, pred_col).count()

    # --- Omitted: print and return statements ---
```

```
model, predictions = classify_tyre_compound(cleaned_data,
    session_suffixes[0])
evaluate_classification(predictions)

# plot classification predictions
def plot_classification(cleaned_data, suffix):
     model, predictions = classify_tyre_compound(cleaned_data,
    suffix)
     display(predictions) # Scatter plot grouped by classification

plot_classification(cleaned_data, session_suffixes[0])
```

Listing 4.3: Classification evaluation code (Feng, 2021)

A scatter plot of average tyre surface temperature against tyre inner temperature, grouped by predicted tyre compound (see figure 4.3), further illustrates effectiveness. The visualisation reveals a clear separation between predicted tyre compounds. This aligns with the physical characteristics of different compounds, supporting the model's predictive logic and enhancing the interpretability of its outputs.

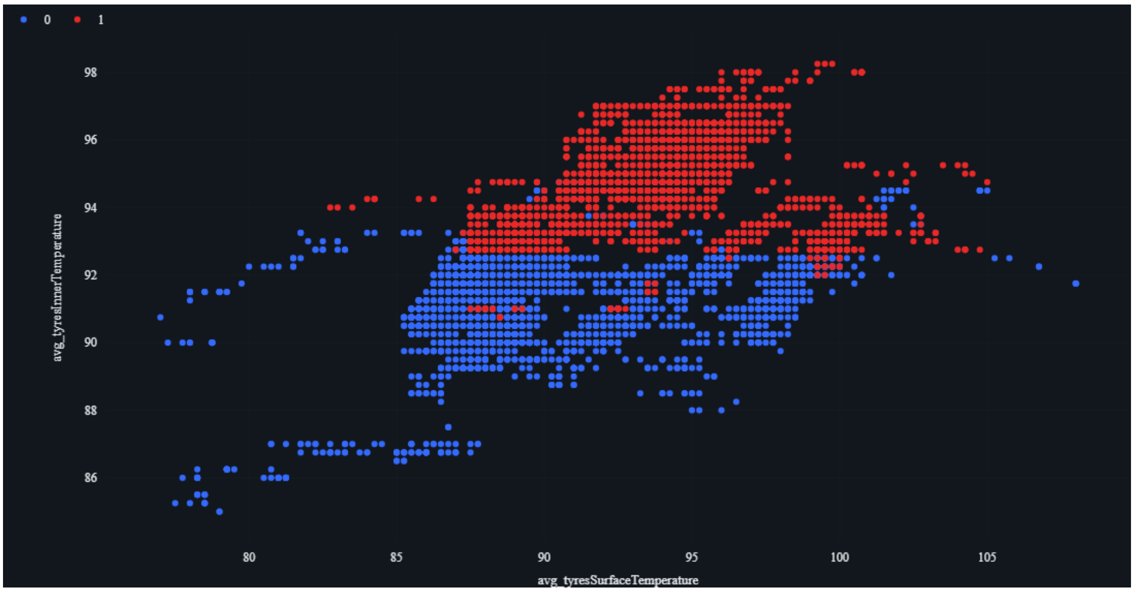

Figure 4.3: Scatter plot of tyre surface temperature and tyre inner temperature grouped by classification tyre compound prediction.

4.4 Association rule

To assess the effectiveness of the association rule mining process, several key indicators were computed based on the output of the FP-Growth algorithm. The evaluation focused on the number and quality of both frequent itemsets and rules generated. Metrics such as confidence, lift, and support were used to measure rule strength, reliability, and significance, whilst ranked summaries of the strongest rules provide additional insight into the nature of the discovered relationships. These metrics help determine whether the mined rules are not only statistically significant but also contextually relevant to the domain (see Appendix B.4).

Metric	Value	Description
Total frequent itemsets	3,719	The total number of frequent itemsets generated by the FP-Growth algorithm.
Total rules generated	14,415	The total number of association rules itemsets generated by the FP-Growth algorithm.
Average confidence	0.717	Rules were correct nearly 72% of the time when their antecedent occurred.
Average lift	1.359	Most rules revealed strong and statistically significant associations between telemetry conditions and outcomes.

Table 4.4: Association rule mining evaluation results based on telemetry and session data

```
# Print number of itemsets and rules, top 20 rules order by
    confidence
def evaluate_association(cleaned_data, suffix, cols_to_include=None
    ):
     itemsets, rules = run_association(cleaned_data, suffix,
    cols_to_include)

     total_itemsets = itemsets.count()
     total_rules = rules.count()
     top_confidence = rules.orderBy("confidence", ascending=False).
    limit(10)
     top_lift = rules.orderBy("lift", ascending=False).limit(10)
     top_support = rules.orderBy("support", ascending=False).limit
    (10)

     # --- Omitted: print and return statements ---

evaluate_association(cleaned_data, session_suffixes[0])
```

Listing 4.4: Association evaluation code (Feng, 2021)

These results confirm that the FP-Growth model successfully captured both intuitive and nuanced behavioural patterns within the telemetry data. For example, top lift rules showed particularly strong associations between low-throttle, high-speed driving and hard braking, with lift values exceeding 8.4, indicating a strong dependency beyond chance.

References

Apache Spark (2025), 'Apache Spark Python API Documentation'. Accessed on: 10/02/2025.
URL: *https://spark.apache.org/docs/latest/api/python/index.html*

Databricks (2025), 'Databricks AWS Documentation'. Accessed on: 24/02/2025.
URL: *https://docs.databricks.com/aws/en/*

Feng, W. (2021), 'Learning apache spark with python'. Accessed on: 24/02/2025.
URL: *https://runawayhorse001.github.io/LearningApacheSpark/pyspark.pdf*

Mine, N. (2020), 'F1 2020 Race Data'. Accessed on: 03/02/2025.
URL: *https://www.kaggle.com/datasets/coni57/f1-2020-race-data*

Tan, P.-N., Steinbach, M., Kumar, V. and Karpatne, A. (2019), *Introduction to Data Mining*, 2 edn, Pearson Education, Limited.

Uchyigit, D. G. (2025*a*), Association rule mining, fp-growth algorithm. Lecture slides for module CI603 Data Mining, Lecture 2.

Uchyigit, D. G. (2025*b*), Principal component analysis. Lecture slides for module CI603 Data Mining, Lecture 8.

Appendix A

EDA metadata results

A.1 Schema

```
Running on Session_1...
Schema for Session_1:
  ParticipantData:
root
 |-- pilot_index: integer (nullable = true)
 |-- driverId: string (nullable = true)
 |-- teamId: string (nullable = true)
 |-- idleRPM: integer (nullable = true)
 |-- session_id: integer (nullable = true)

  SessionData:
root
 |-- weather: integer (nullable = true)
 |-- trackTemperature: integer (nullable = true)
 |-- airTemperature: integer (nullable = true)
 |-- totalLaps: integer (nullable = true)
 |-- trackLength: integer (nullable = true)
 |-- trackId: string (nullable = true)
 |-- session_id: integer (nullable = true)

  RaceTimeData:
root
 |-- frameIdentifierStart: integer (nullable = true)
 |-- frameIdentifierStop: integer (nullable = true)
 |-- pilot_index: integer (nullable = true)
 |-- LapTime: double (nullable = true)
 |-- sector1TimeInMS: integer (nullable = true)
 |-- sector2TimeInMS: integer (nullable = true)
 |-- sector3TimeInMS: integer (nullable = true)
 |-- carPosition: integer (nullable = true)
 |-- currentLapNum: integer (nullable = true)
 |-- session_id: integer (nullable = true)

  TelemetryData:
```

```
root
 |-- sessionTime: double (nullable = true)
 |-- frameIdentifier: integer (nullable = true)
 |-- pilot_index: integer (nullable = true)
 |-- worldPositionX: double (nullable = true)
 |-- worldPositionY: double (nullable = true)
 |-- worldPositionZ: double (nullable = true)
 |-- worldVelocityX: double (nullable = true)
 |-- worldVelocityY: double (nullable = true)
 |-- worldVelocityZ: double (nullable = true)
 |-- worldForwardDirX: integer (nullable = true)
 |-- worldForwardDirY: integer (nullable = true)
 |-- worldForwardDirZ: integer (nullable = true)
 |-- worldRightDirX: integer (nullable = true)
 |-- worldRightDirY: integer (nullable = true)
 |-- worldRightDirZ: integer (nullable = true)
 |-- gForceLateral: double (nullable = true)
 |-- gForceLongitudinal: double (nullable = true)
 |-- gForceVertical: double (nullable = true)
 |-- yaw: double (nullable = true)
 |-- pitch: double (nullable = true)
 |-- roll: double (nullable = true)
 |-- speed: double (nullable = true)
 |-- throttle: double (nullable = true)
 |-- steer: double (nullable = true)
 |-- brake: double (nullable = true)
 |-- clutch: double (nullable = true)
 |-- gear: double (nullable = true)
 |-- engineRPM: double (nullable = true)
 |-- drs: double (nullable = true)
 |-- brakesTemperature: array (nullable = true)
 |    |-- element: float (containsNull = true)
 |-- tyresSurfaceTemperature: array (nullable = true)
 |    |-- element: float (containsNull = true)
 |-- tyresInnerTemperature: array (nullable = true)
 |    |-- element: float (containsNull = true)
 |-- engineTemperature: double (nullable = true)
 |-- tyresPressure: array (nullable = true)
 |    |-- element: float (containsNull = true)
 |-- surfaceType: array (nullable = true)
 |    |-- element: float (containsNull = true)
 |-- fuelMix: double (nullable = true)
 |-- pitLimiterStatus: double (nullable = true)
 |-- fuelInTank: double (nullable = true)
 |-- fuelRemainingLaps: double (nullable = true)
 |-- tyresWear: array (nullable = true)
 |    |-- element: float (containsNull = true)
 |-- actualTyreCompound: string (nullable = true)
 |-- tyresDamage: array (nullable = true)
```

```
 |    |-- element: float (containsNull = true)
 |-- ersStoreEnergy: double (nullable = true)
 |-- ersDeployMode: double (nullable = true)
 |-- ersHarvestedThisLapMGUK: double (nullable = true)
 |-- ersHarvestedThisLapMGUH: double (nullable = true)
 |-- ersDeployedThisLap: double (nullable = true)
 |-- carPosition: integer (nullable = true)
 |-- currentLapTime: double (nullable = true)
 |-- currentLapNum: integer (nullable = true)
 |-- lapDistance: double (nullable = true)
 |-- totalDistance: double (nullable = true)
 |-- pitStatus: string (nullable = true)
 |-- sector: integer (nullable = true)
 |-- driverStatus: string (nullable = true)
 |-- resultStatus: string (nullable = true)
 |-- session_id: integer (nullable = true)
```

A.2 Summary

```
Running on Session_1...

Summary for Session_1:

Checking ParticipantData dataset:
+-------+-----------------+----------------+----------+---------------+----------+
|summary|      pilot_index|        driverId|    teamId|        idleRPM|session_id|
+-------+-----------------+----------------+----------+---------------+----------+
|  count|               20|              20|        20|             20|        20|
|   mean|              9.5|            null|      null|         4039.6|       1.0|
| stddev|5.916079783099616|            null|      null|341.37838244388|       0.0|
|    min|                0|alexander_albon|alfa_romeo|           3499|         1|
|    max|               19|valtteri_bottas|  williams|           4300|         1|
+-------+-----------------+----------------+----------+---------------+----------+


Checking SessionData dataset:
+-------+-------+----------------+--------------+---------+-----------+----------+----------+
|summary|weather|trackTemperature|airTemperature|totalLaps|trackLength|   trackId|session_id|
+-------+-------+----------------+--------------+---------+-----------+----------+----------+
|  count|      1|               1|             1|        1|          1|         1|         1|
|   mean|    0.0|            32.0|          26.0|     28.0|     5547.0|      null|       1.0|
| stddev|   null|            null|          null|     null|       null|      null|      null|
|    min|      0|              32|            26|       28|       5547|yas_marina|         1|
|    max|      0|              32|            26|       28|       5547|yas_marina|         1|
+-------+-------+----------------+--------------+---------+-----------+----------+----------+


Checking RaceTimeData dataset:
```

summary	frameIdentifierStart	frameIdentifierStop	pilot_index	LapTime	sector1TimeInMS
count	561	561	561	561	561
mean	30537.313725490196	32732.076648841354	9.516934046345812	102.7155347950089	19737.4064171123
stddev	18134.350311572478	18032.41809787624	5.78021396146243	11.797555036767218	5104.91725295623
min	0	2304	0	0.0	0
max	61593	61593	19	121.84802	38233

sector2TimeInMS	sector3TimeInMS	carPosition	currentLapNum	session_id
561	561	561	561	561
42450.22459893048	40527.40285204991	10.483065953654188	14.52584670231729	1.0
6583.820226740453	4548.939313368133	5.78021396146243	8.100912120471417	0.0
0	0	1	1	1
48154	44319	20	29	1

Checking TelemetryData dataset:

summary	sessionTime	frameIdentifier	pilot_index	worldPositionX	worldPositionY
count	1113180	1113180	1113180	1113180	1113180
mean	1454.1806857624258	30955.962683483354	9.5	-118.4211698882486	5.111965461353947
stddev	838.0660056360833	17758.76495261517	5.766283887341135	477.2495074013966	2.045356833283135
min	0.01101	0	0	-726.21606	-1.88867
max	2904.25439	61593	19	834.65466	13.44968

worldPositionZ	worldVelocityX	worldVelocityY	worldVelocityZ	worldForwardDirX
1113180	1113180	1113180	1113180	1113180
28.134711612973902	-0.11384003928385483	3.971814980505978...	0.08447690445391311	3335.0187301245082
301.95107595571045	41.3058750967056	0.714375001977161	38.44156043911875	23434.685159763794
-314.91916	-92.58421	-3.53851	-76.43197	-32767
657.55359	81.70893	4.07681	89.44661	32766

worldForwardDirY	worldForwardDirZ	worldRightDirX	worldRightDirY	worldRightDirZ	gForceLateral
1113180	1113180	1113180	1113180	1113180	1113180
49.243800643202356	-135.86594890314234	136.23225803553782	184.91690023176844	3334.9412754451214	0.12108981014750567
442.4237039789084	22651.99251066368	22650.91718083433	427.91960721998703	23435.30518579091	1.5437679149649677
-3270	-32767	-32766	-2142	-32767	-16.16686
2343	32766	32767	2380	32766	17.97173

gForceLongitudinal	gForceVertical	yaw	pitch	roll
1113180	1113180	1113180	1113180	1113180
0.007252874934871201	0.002573809509692...	0.3053849198871694	0.00208042066871485	-0.00564769123591874
1.2020397877584408	0.2049306712822592	1.7752489775402105	0.015028569354027705	0.013072013371835367
-13.24528	-3.06711	-3.14159	-0.06591	-0.07272
10.81696	3.46392	3.14159	0.06999	0.06542

speed	throttle	steer	brake	clutch	gear
1113140	1113140	1113140	1113140	1113140	1113140
187.42319474639308	0.6181204417773135	-0.04126194314282198	0.0959791076683973	0.016529816554970624	4.5285884974037405
77.17248184686964	0.4287409184882302	0.23656187071084894	0.2524421022102898	1.2855776543599093	2.216158204510602
0.0	0.0	-1.0	0.0	0.0	0.0
333.0	1.0	1.0	1.0	100.0	8.0

engineRPM	drs	engineTemperature	fuelMix	pitLimiterStatus	fuelInTank
1113140	1113140	1113140	1113100	1113100	1113100
10365.261132472106	0.05518892502290817	90.70294841619203	1.31739556194412	0.014664450633366275	30.12073221016145
1128.5885960496346	0.22834875611885233	5.865131049254725	0.5642715472751436	0.12020572990597053	16.661782544391308
3480.0	0.0	89.0	0.0	0.0	0.73334
13025.0	1.0	128.0	2.0	1.0	61.69364

fuelRemainingLaps	actualTyreCompound	ersStoreEnergy	ersDeployMode	ersHarvestedThisLapMGUK
1113100	1113100	1113100	1113100	1113100
1.2837225181834544	null	1806766.9042802476	1.1147165573623214	532864.7823205276
1.0042049793456325	null	625397.4324571986	0.31867971392454053	332567.03435290785
-0.86009	medium	286561.84375	1.0	0.0
3.93183	soft	4000000.0	2.0	1278835.125

```
+----------------------+-----------------+-----------------+----------------+-----------------+-----------------+
|ersHarvestedThisLapMGUH|ersDeployedThisLap|       carPosition|   currentLapTime|     currentLapNum|       lapDistance|
+----------------------+-----------------+-----------------+----------------+-----------------+-----------------+
|               1113100|          1113100|          1113180|         1113180|          1113180|          1113180|
|      643592.113166892|1364199.1716915332|             10.5|51.97768310813166|14.215152985141666| 2849.900960048742|
|     411776.7205358306| 861674.6449259303|5.7662838873411415| 30.4000736948662| 7.938015985390431|1634.7144908459093|
|                   0.0|              0.0|                1|             0.0|                1|           -51.625|
|             1875489.5|       4000780.75|               20|       121.83666|               29|        5547.11719|
+----------------------+-----------------+-----------------+----------------+-----------------+-----------------+

+---------------+-----------+-----------------+-----------+------------+----------+
|   totalDistance|  pitStatus|           sector|driverStatus|resultStatus|session_id|
+---------------+-----------+-----------------+-----------+------------+----------+
|         1113180|      16321|          1113180|    1113180|     1113180|   1113180|
|76155.9092728247|       null| 1.197817064625667|       null|        null|       1.0|
|44043.4554272805|       null|0.7366130024992832|       null|        null|       0.0|
|         -51.625|in_pit_area|                0|     in_lap|      active|         1|
|     155319.78125|    pitting|                2|    out_lap|      active|         1|
+---------------+-----------+-----------------+-----------+------------+----------+
```

A.3 The first entries

```
Running on Session_1...

First 5 entries for Session_1:

Printing 5 entries for ParticipantData dataset:
```

```
+-----------+----------------+---------------+-------+----------+
|pilot_index|        driverId|         teamId|idleRPM|session_id|
+-----------+----------------+---------------+-------+----------+
|          0|    pierre_gasly|     toro_rosso|   3499|         1|
|          1| charles_leclerc|        ferrari|   4300|         1|
|          2|  max_verstappen|red_bull_racing|   3499|         1|
|          3|    lando_norris|        mclaren|   3799|         1|
|          4|sebastian_vettel|        ferrari|   4300|         1|
+-----------+----------------+---------------+-------+----------+
only showing top 5 rows


Printing 5 entries for SessionData dataset:
+-------+----------------+--------------+---------+-----------+----------+----------+
|weather|trackTemperature|airTemperature|totalLaps|trackLength|   trackId|session_id|
+-------+----------------+--------------+---------+-----------+----------+----------+
|      0|              32|            26|       28|       5547|yas_marina|         1|
+-------+----------------+--------------+---------+-----------+----------+----------+


Printing 5 entries for RaceTimeData dataset:
+--------------------+-------------------+-----------+---------+---------------+---------------+---------------+
|frameIdentifierStart|frameIdentifierStop|pilot_index|  LapTime|sector1TimeInMS|sector2TimeInMS|sector3TimeInMS|
+--------------------+-------------------+-----------+---------+---------------+---------------+---------------+
|                   0|               2468|          0|113.29473|          23686|          47302|          42306|
|                2470|               4720|          0|103.73251|          18314|          43825|          41593|
|                4721|               7046|          0|103.59639|          18408|          43450|          41738|
|                7047|               9241|          0|104.06097|          18423|          43749|          41888|
|                9243|              11494|          0|104.15616|          18486|          43618|          42052|
+--------------------+-------------------+-----------+---------+---------------+---------------+---------------+
```

```
+-----------+-------------+----------+
|carPosition|currentLapNum|session_id|
+-----------+-------------+----------+
|         15|            1|         1|
|         15|            2|         1|
|         15|            3|         1|
|         15|            4|         1|
|         15|            5|         1|
+-----------+-------------+----------+
only showing top 5 rows

Printing 5 entries for TelemetryData dataset:
+-----------+---------------+-----------+--------------+--------------+--------------+--------------+--------------+
|sessionTime|frameIdentifier|pilot_index|worldPositionX|worldPositionY|worldPositionZ|worldVelocityX|worldVelocityY|
+-----------+---------------+-----------+--------------+--------------+--------------+--------------+--------------+
| 1427.42664|          30468|          2|    -725.17249|       4.39897|     580.46643|      -0.12031|      -0.03396|
| 1427.42664|          30468|          3|      44.93525|       5.93634|    -307.54974|     -83.84908|      -0.00885|
| 1427.42664|          30468|          4|     170.28351|       5.95592|    -308.86057|     -82.49887|      -0.00856|
| 1427.42664|          30468|          5|     344.89288|       5.98473|    -309.90738|      -81.3914|      -0.00964|
| 1427.42664|          30468|          6|    -379.04831|       4.00352|    -301.15857|     -19.89058|       0.03777|
+-----------+---------------+-----------+--------------+--------------+--------------+--------------+--------------+

+--------------+----------------+----------------+----------------+--------------+--------------+--------------+
|worldVelocityZ|worldForwardDirX|worldForwardDirY|worldForwardDirZ|worldRightDirX|worldRightDirY|worldRightDirZ|
+--------------+----------------+----------------+----------------+--------------+--------------+--------------+
|      40.76767|             -52|            -133|           32766|        -32753|           925|           -48|
|       0.81945|          -32765|              85|             316|          -316|             0|        -32765|
|       0.79761|          -32765|              89|             313|          -313|             0|        -32765|
|       0.74661|          -32765|              56|             302|          -302|             0|        -32765|
|      10.11578|          -29686|              -1|           13869|        -13869|           265|        -29685|
+--------------+----------------+----------------+----------------+--------------+--------------+--------------+
```

gForceLateral	gForceLongitudinal	gForceVertical	yaw	pitch	roll	speed	throttle	steer	brake	clutch	gear
1.68183	-2.81545	0.01295	-0.0016	0.02824	-0.02825	146.0	0.0	-0.14312	0.80079	0.0	3.0
-0.02033	0.08644	-6.6E-4	-1.56113	0.0026	-3.0E-5	301.0	1.0	0.0	0.0	0.0	8.0
0.01264	0.11058	-2.0E-5	-1.56123	0.00272	1.0E-5	297.0	1.0	-0.00123	0.0	0.0	8.0
-6.0E-4	0.25253	9.6E-4	-1.56156	0.00173	0.0	293.0	1.0	0.0	0.0	0.0	7.0
1.82556	-0.66331	-0.01596	-1.13374	0.00338	-0.0081	80.0	0.0	-0.60385	0.06614	0.0	1.0

engineRPM	drs	brakesTemperature	tyresSurfaceTemperature	tyresInnerTemperature	engineTemperature
10772.0	0.0	[992.0, 993.0, 10...	[86.0, 89.0, 91.0...	[91.0, 92.0, 93.0...	89.0
10529.0	0.0	[395.0, 396.0, 25...	[87.0, 88.0, 88.0...	[91.0, 92.0, 94.0...	89.0
10358.0	0.0	[451.0, 452.0, 31...	[90.0, 90.0, 88.0...	[88.0, 89.0, 90.0...	89.0
11762.0	1.0	[525.0, 527.0, 39...	[85.0, 85.0, 84.0...	[90.0, 91.0, 93.0...	89.0
9992.0	0.0	[1015.0, 1016.0, ...	[83.0, 85.0, 90.0...	[90.0, 90.0, 93.0...	89.0

tyresPressure	surfaceType	fuelMix	pitLimiterStatus	fuelInTank	fuelRemainingLaps	tyresWear
[21.1, 21.1, 23.0...	[0.0, 0.0, 0.0, 0.0]	1.0	0.0	28.38136	0.33065	[24.0, 24.0, 22.0...
[21.1, 21.1, 23.0...	[0.0, 0.0, 0.0, 0.0]	2.0	0.0	29.67995	0.74605	[11.0, 11.0, 9.0,...
[21.1, 21.1, 23.0...	[0.0, 0.0, 0.0, 0.0]	1.0	0.0	28.57323	0.15002	[1.0, 1.0, 1.0, 1.0]
[21.1, 21.1, 23.0...	[0.0, 0.0, 0.0, 0.0]	1.0	0.0	29.38491	0.5391	[11.0, 11.0, 10.0...
[21.1, 21.1, 23.0...	[0.0, 0.0, 0.0, 0.0]	2.0	0.0	28.3111	0.11334	[10.0, 10.0, 9.0,...

```
+-----------------+--------------------+-------------+-------------+-----------------------+-----------------------+
|actualTyreCompound|         tyresDamage|ersStoreEnergy|ersDeployMode|ersHarvestedThisLapMGUK|ersHarvestedThisLapMGUH|
+-----------------+--------------------+-------------+-------------+-----------------------+-----------------------+
|             soft|[24.0, 24.0, 22.0...|  1133022.875|          1.0|             683356.375|            647346.6875|
|           medium|[11.0, 11.0, 9.0,...|  1254613.125|          1.0|            334014.8125|           410718.34375|
|             soft|[1.0, 1.0, 1.0, 1.0]|  1351182.875|          1.0|           251932.26562|              844681.75|
|           medium|[11.0, 11.0, 10.0...|  1947022.375|          1.0|           337044.40625|             463771.125|
|           medium|[10.0, 10.0, 9.0,...|  1416593.375|          1.0|            536471.9375|             508305.625|
+-----------------+--------------------+-------------+-------------+-----------------------+-----------------------+

+-----------------+-----------+-------------+-------------+-----------+-------------+---------+------+------------+
|ersDeployedThisLap|carPosition|currentLapTime|currentLapNum|lapDistance|totalDistance|pitStatus|sector|driverStatus|
+-----------------+-----------+-------------+-------------+-----------+-------------+---------+------+------------+
|      1510565.875|          2|     62.36975|           14| 3756.22656|  75868.75781|     null|     2|    on_track|
|      834218.3125|          9|     40.02254|           14| 2328.45312|  74440.98438|     null|     1|    on_track|
|      1292466.875|         10|     56.02494|           14| 2203.10156|  74315.63281|     null|     1|     out_lap|
|        1121345.5|         13|     36.62075|           14| 2028.48438|  74141.01562|     null|     1|    on_track|
|      1667981.625|          7|     45.77728|           14| 2752.57812|  74865.10938|     null|     1|    on_track|
+-----------------+-----------+-------------+-------------+-----------+-------------+---------+------+------------+

+------------+----------+
|resultStatus|session_id|
+------------+----------+
|      active|         1|
|      active|         1|
|      active|         1|
|      active|         1|
|      active|         1|
+------------+----------+
only showing top 5 rows
```

A.4 Count

```
Running on Session_1...

Count for Session_1:

ParticipantData dataset size: 20 rows x 6 columns

SessionData dataset size: 1 rows x 6 columns

RaceTimeData dataset size: 561 rows x 9 columns

TelemetryData dataset size: 1113180 rows x 56 columns
```

A.5 Missing values

```
Running on Session_1...

Missing values for Session_1

ParticipantData missing values:

SessionData missing values:

RaceTimeData missing values:

TelemetryData missing values:
 - speed: 40 missing values
 - throttle: 40 missing values
 - steer: 40 missing values
 - brake: 40 missing values
 - clutch: 40 missing values
 - gear: 40 missing values
 - engineRPM: 40 missing values
 - drs: 40 missing values
 - engineTemperature: 40 missing values
 - fuelMix: 80 missing values
 - pitLimiterStatus: 80 missing values
 - fuelInTank: 80 missing values
 - fuelRemainingLaps: 80 missing values
 - ersStoreEnergy: 80 missing values
 - ersDeployMode: 80 missing values
 - ersHarvestedThisLapMGUK: 80 missing values
 - ersHarvestedThisLapMGUH: 80 missing values
 - ersDeployedThisLap: 80 missing values
 - actualTyreCompound: 80 missing values
 - pitStatus: 1096859 missing values
```

Appendix B

Evaluation results

B.1 Dimensionality reduction

```
+-----------+-----------------------------------------+
|pilot_index|pca_vector                               |
+-----------+-----------------------------------------+
|0          |[-1.3574925364474593,0.3762422177015506] |
|0          |[-0.19793439368670532,0.8558054825422172]|
|0          |[-0.18215023141132117,0.8491145055368187]|
|0          |[-0.24805223608956276,0.8585266054847129]|
|0          |[-0.2643495628755316,0.8587836612756016] |
+-----------+-----------------------------------------+
only showing top 5 rows

’total_variance’: 0.7288847302794471,
’PC1_variance’: 0.5218527363684351,
’PC2_variance’: 0.20703199391101204
```

B.2 Clustering

```
Clustering evaluation metrics:
Silhouette score: 0.975

Cluster distribution:
+------------------+-----+----------+
|cluster_prediction|count|percentage|
+------------------+-----+----------+
|                 0|  501|      89.3|
|                 1|   42|      7.49|
|                 2|    6|      1.07|
|                 3|    5|      0.89|
|                 4|    2|      0.36|
|                 5|    5|      0.89|
+------------------+-----+----------+
```

B.3 Classification

```
Classification metrics:
Accuracy: 0.739
F1 Score: 0.737
Precision: 0.737
Recall: 0.739

Confusion Matrix:
+-----+----------+------+
|label|prediction| count|
+-----+----------+------+
|  1.0|       1.0| 62364|
|  0.0|       1.0| 24740|
|  1.0|       0.0| 33361|
|  0.0|       0.0|101976|
+-----+----------+------+
```

B.4 Association rule

```
Association rule mining metrics:
Total frequent itemsets: 3719
Total rules generated: 14415
Average confidence: 0.717
Average lift: 1.359
```

Top 10 Rules by Confidence:

antecedent	consequent	confidence
[gear=medium, fuel=medium, speed=fast, throttle=high, weather=clear, track=warm]	[brake=none]	1.0
[throttle=medium, speed=medium, tyre=soft, track=warm]	[weather=clear]	1.0
[gear=medium, fuel=low, throttle=low, speed=medium]	[weather=clear]	1.0
[speed=medium, tyre=soft, throttle=high]	[brake=none]	1.0
[speed=slow, fuel=low, throttle=low, brake=none, weather=clear]	[gear=low]	1.0
[gear=low, fuel=high, throttle=high, track=warm]	[weather=clear]	1.0
[brake=light, speed=medium, tyre=medium]	[weather=clear]	1.0
[speed=medium, tyre=soft, throttle=high]	[weather=clear]	1.0
[throttle=medium, gear=low, track=warm]	[brake=none]	1.0
[tyre=soft, throttle=high]	[track=warm]	1.0

lift	support
1.1719469733740302	0.0187323690593837
1.0	0.034987871709639747
1.0	0.011643158745844937
1.1719469733740302	0.06163058125954542

```
|2.4588461912293567|0.014811786901446411|
|1.0               |0.03212200161710538 |
|1.0               |0.015004042763453419|
|1.0               |0.06163058125954542 |
|1.1719469733740302|0.1057119755637409  |
|1.0               |0.23782768843769653 |
+------------------+--------------------+

Top 10 Rules by Lift:
+-----------------------------------------------------------------------------+------------+------------------+
|antecedent                                                                   |consequent  |confidence        |
+-----------------------------------------------------------------------------+------------+------------------+
|[throttle=low, fuel=medium, speed=fast, weather=clear]                       |[brake=hard]|0.9021543464374892|
|[throttle=low, fuel=medium, speed=fast, track=warm]                          |[brake=hard]|0.9021543464374892|
|[throttle=low, fuel=medium, speed=fast]                                      |[brake=hard]|0.9021543464374892|
|[throttle=low, fuel=medium, speed=fast, weather=clear, track=warm]           |[brake=hard]|0.9021543464374892|
|[throttle=low, gear=high, speed=fast, tyre=medium, track=warm]               |[brake=hard]|0.8949367088607595|
|[throttle=low, gear=high, speed=fast, tyre=medium, weather=clear, track=warm]|[brake=hard]|0.8949367088607595|
|[throttle=low, gear=high, speed=fast, tyre=medium]                           |[brake=hard]|0.8949367088607595|
|[throttle=low, gear=high, speed=fast, tyre=medium, weather=clear]            |[brake=hard]|0.8949367088607595|
|[throttle=low, gear=high, tyre=medium, weather=clear]                        |[brake=hard]|0.8949233183175583|
|[throttle=low, gear=high, tyre=medium, track=warm]                           |[brake=hard]|0.8949233183175583|
+-----------------------------------------------------------------------------+------------+------------------+

+-----------------+--------------------+
|lift             |support             |
+-----------------+--------------------+
|8.443450429405026|0.013957416224957326|
|8.443450429405026|0.013957416224957326|
|8.443450429405026|0.013957416224957326|
|8.443450429405026|0.013957416224957326|
```

```
|8.375899056031745|0.015879076453148865|
|8.375899056031745|0.015879076453148865|
|8.375899056031745|0.015879076453148865|
|8.375899056031745|0.015879076453148865|
|8.375773731148938|0.01588446680442009 |
|8.375773731148938|0.01588446680442009 |
+-----------------+--------------------+
```

Top 10 Rules by Support:

```
+----------------------------+---------------+------------------+----+------------------+
|antecedent                  |consequent     |confidence        |lift|support           |
+----------------------------+---------------+------------------+----+------------------+
|[weather=clear]             |[track=warm]   |1.0               |1.0 |1.0               |
|[track=warm]                |[weather=clear]|1.0               |1.0 |1.0               |
|[weather=clear, track=warm]|[brake=none]   |0.8532809271404187|1.0 |0.8532809271404187|
|[track=warm]                |[brake=none]   |0.8532809271404187|1.0 |0.8532809271404187|
|[weather=clear]             |[brake=none]   |0.8532809271404187|1.0 |0.8532809271404187|
|[brake=none, weather=clear]|[track=warm]   |1.0               |1.0 |0.8532809271404187|
|[brake=none]                |[weather=clear]|1.0               |1.0 |0.8532809271404187|
|[brake=none, track=warm]    |[weather=clear]|1.0               |1.0 |0.8532809271404187|
|[brake=none]                |[track=warm]   |1.0               |1.0 |0.8532809271404187|
|[track=warm]                |[tyre=medium]  |0.5706495373281826|1.0 |0.5706495373281826|
+----------------------------+---------------+------------------+----+------------------+
```