Decision Trees and Random Forest

Formula 1

What are Decision Trees?

- Decision trees are a type of supervised learning algorithm that are widely used for both classification and regression tasks.
- They model decisions and their possible consequences by creating a tree-like structure of branches and nodes

What is the Random Forest algorithm?

- The Random Forest algorithm is a sophisticated machine learning method that uses ensemble learning to build and combine multiple decision trees to produce more accurate and stable predictions

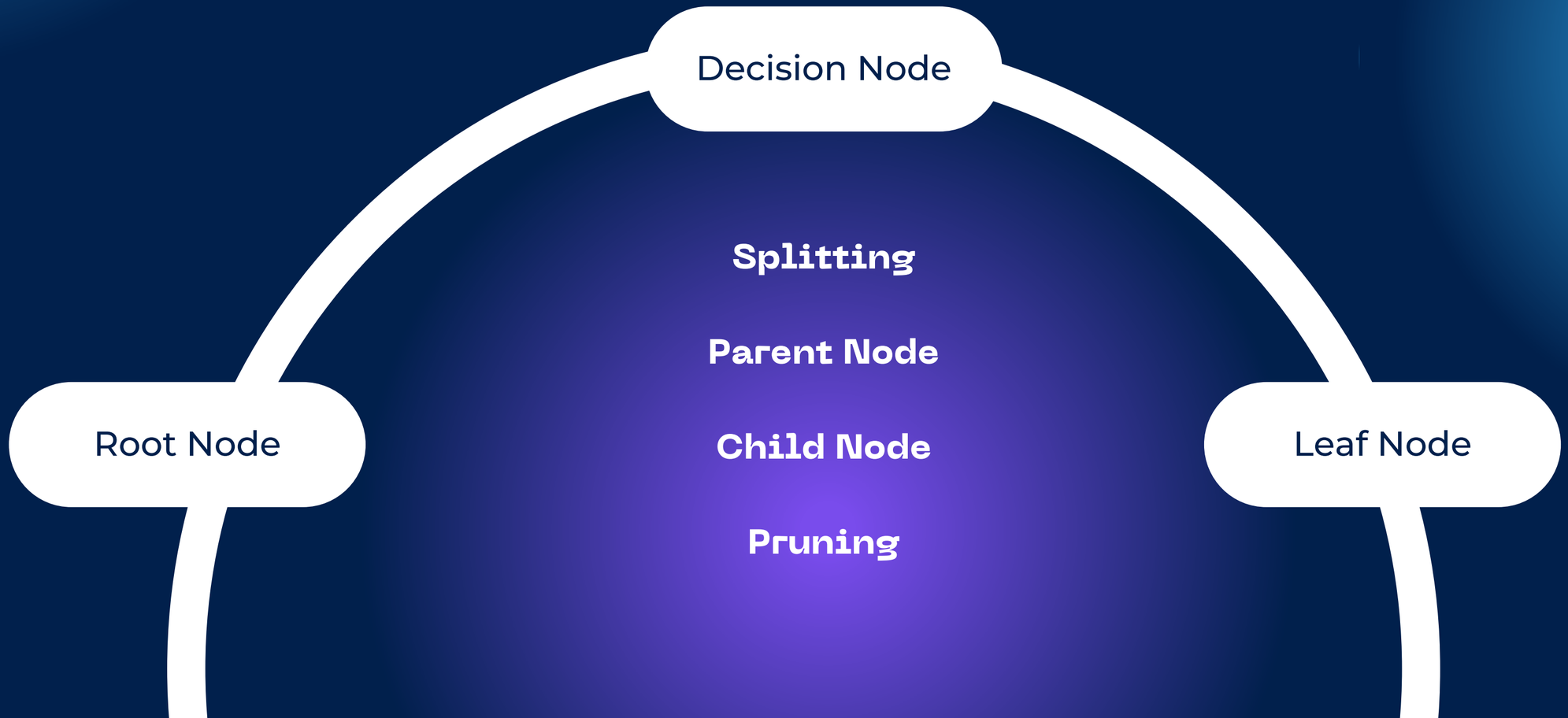
Why are they needed?

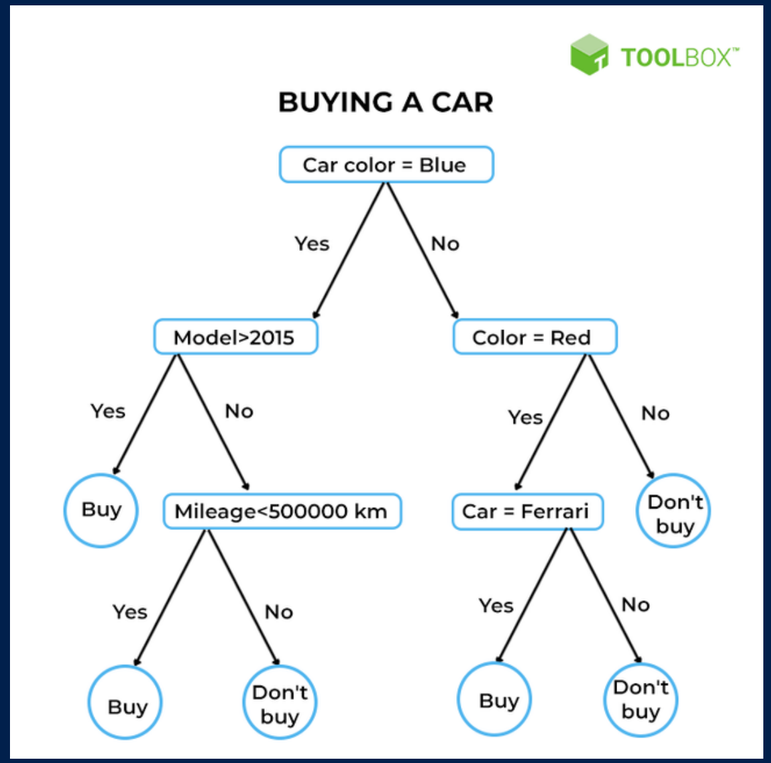
- Decision trees and Random Forests are both popular machine learning methods that have broad applications across many industries and disciplines.
- They are particularly favored for their interpretability and effectiveness in handling a variety of data types and structures.

AGENDA

- Understanding Decision Trees
- Introduction to Random Forests
- Key parameters of Random Forests
- Advantages of Random Forests
- Dataset Introduction: Formula 1 Weather Forecast
- Exploratory data analysis
- Implementation of Random Forest
- Model Evaluation
- Case Study: Using Random Forest for Weather Classification in Formula One
- Comparison with other techniques
- Conclusion

Understanding Decision trees





Root Node: The top-most node that holds the most essential attribute for the training data.

Decision Node: If a sub node is split further, each condition is represented by the decision node.

Leaf Node: A node that cannot be split any further

Parent node: If a node is split into sub nodes, that node is the parent of te new sub nodes.

Child Node: The sub nodes derived from the parent node.

Pruning: Removing some branches of the trees.

Splitting criteria:

Gini Impurity

- Often used in classification
- It measures the impurity of a node.
- A node is "pure" (Gini = 0) if all its records belong to the same class.

Entropy

- Another measure used in information gain as a splitting criterion.
- It indicates the level of uncertainty or disorder.

Information Gain

- Information Gain is the effectiveness of an attribute to retain entropy that may be lost due to partitioning.
- The attribute with the highest information gain is chosen as next node.

Mathematical Foundation

- **Gini Impurity:**
  - It is the probability of misclassifying a randomly chosen element in a set
  - The range of the Gini index is [0, 1]

$$GI = 1 - \sum_{i=1}^{n} (p)^2$$

$$GI = 1 - \left[(P_{(+)})^2 + (P_{(-)})^2 \right]$$

Mathematical Foundation

- **Entropy:**
  - Average amount of information needed to classify any observation in the data
  - Termed as ‘Uncertainty’, if entropy is higher confidence in correctly classifying observation is lower and vice-versa

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

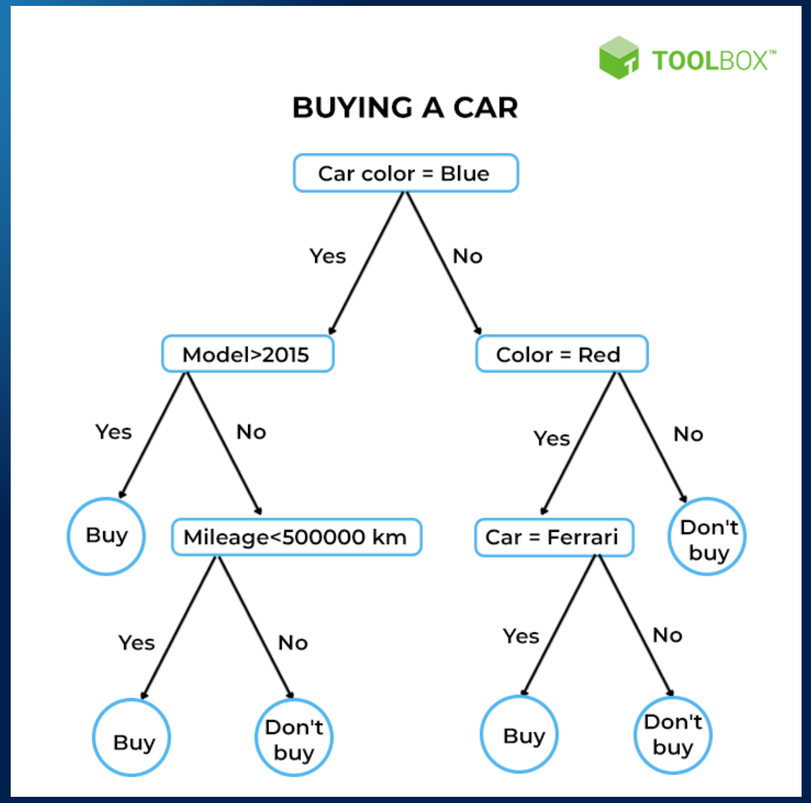
where c is the number of output classes.

Mathematical Foundation

- **Information Gain:**
  - Splitting a dataset results in loss of information
  - IG is the measure of an attribute to retain this lost entropy/information

$$IG(S, A) = Entropy(S) - \sum_{v=Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Mechanism Of Decision Trees



Car	Car Color	Model Year	Mileage	Target
Ferrari	Red	2015	100000	Buy
Porsche	Blue	2021	300000	Dont Buy
Nissan	Black	2017	150000	Dont Buy

$$GI = 1 - \sum_{i=1}^{n} (p)^2$$

$$GI = 1 - \left[(P_{(+)})^2 + (P_{(-)})^2 \right]$$

Calculate Gini Impurity for attributes to identify root nodes and further split the node based on the same criteria

Select appropriate root node and parent nodes based on impurity values to create a tree structure.

Classify target variable by running each data sample down the tree.

Gini impurity for 'Car Color' attribute Per Leaf:

$$GI_{\text{car color}} = 1 - [P(Yes)^2 - P(No)^2]$$

Gini impurity for 'Car' attribute Per Leaf: :

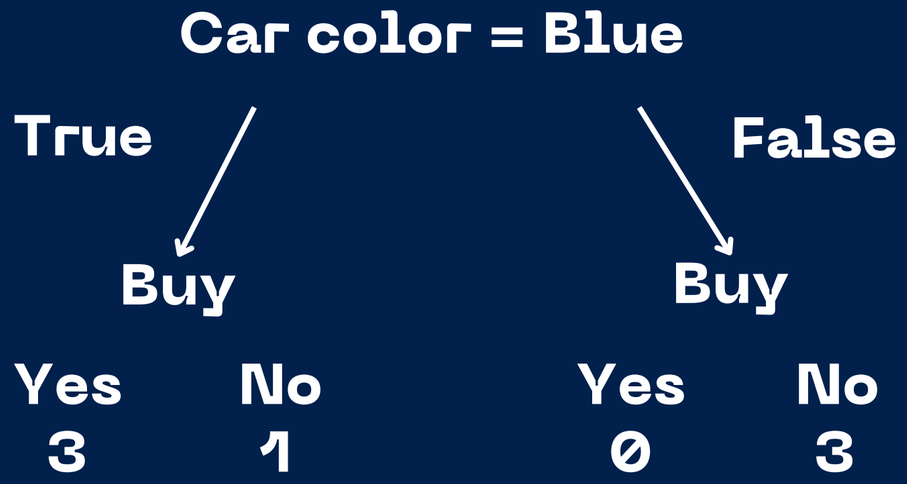
$$GI_{\text{car}} = 1 - [P(Yes)^2 - P(No)^2]$$

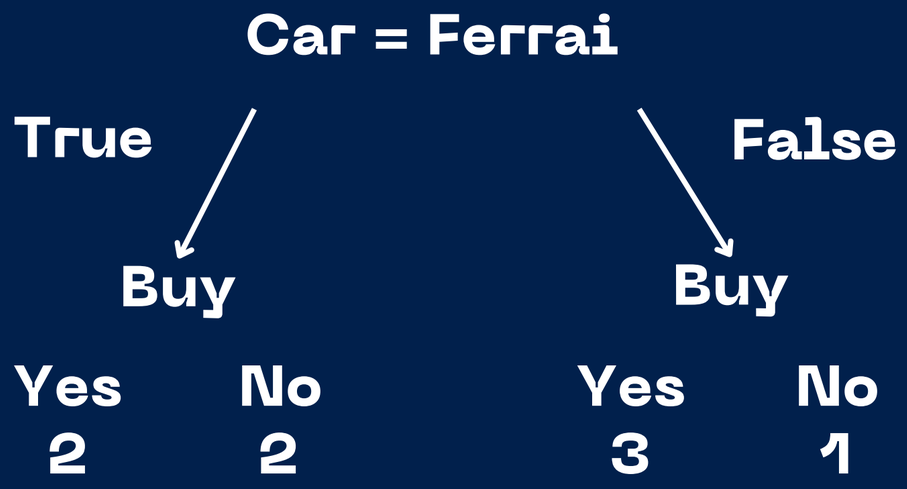
What is impurity?
It is the 'mixedness' of an ensemble of items. It represents the probability of an item in that ensemble being incorrectly labeled if it was randomly labaled according to the distribution of the set.

An attribute is randomly selected and the impurity of its leaves is calculated

The weighted average of the gini impurity of Leaf Nodes is taken to calculate Total Gini Impurity for an attribute.

The attribute with lowest gini impurity is selected as the Root Node





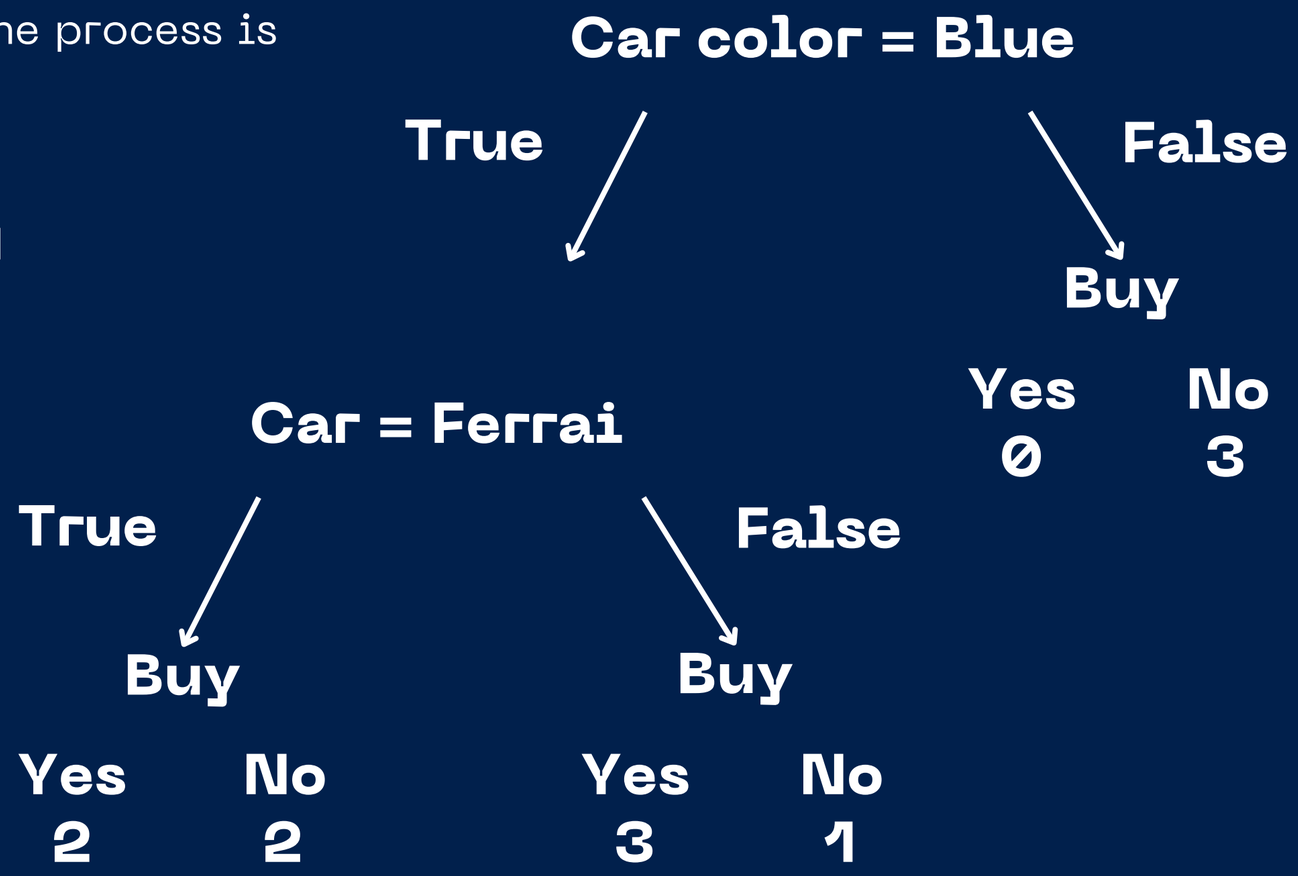
Mechanism Of Decision Trees

The Root Node is then split into sub nodes and the process is repeated

The impurity in one of the leaf nodes is resovled by splitting it based other attributes with lowest Gini Impurity

Output of the leaf is then assigned
It is the majority of of votes in that leaf

this results in a classification decision tree which classifies any data sample that is run through it



Key Parameters:

Max depth (max_depth)

- Defines the maximum depth of the tree
- Limits the number of levels allowed in the tree

Criterion

- function to choose the method for splitting node
- ‘Gini’ for Gini Impurity and ‘entropy’

Min_sample_split

- The minimum number of samples a node needs to have before it can split

Advantages of Decision Trees

Interpretability

- Easy-to-understand
- Transparent
- Easy to explain to non-technical stakeholders

Non-Parametric

- Does not require assumptions about the distribution of the data

Handle multiple datatype

- Can manage categorical as well as numeric data

Drawbacks of Decision Trees

Overfitting

- May result in overly complex trees that do not perform well on new sample data

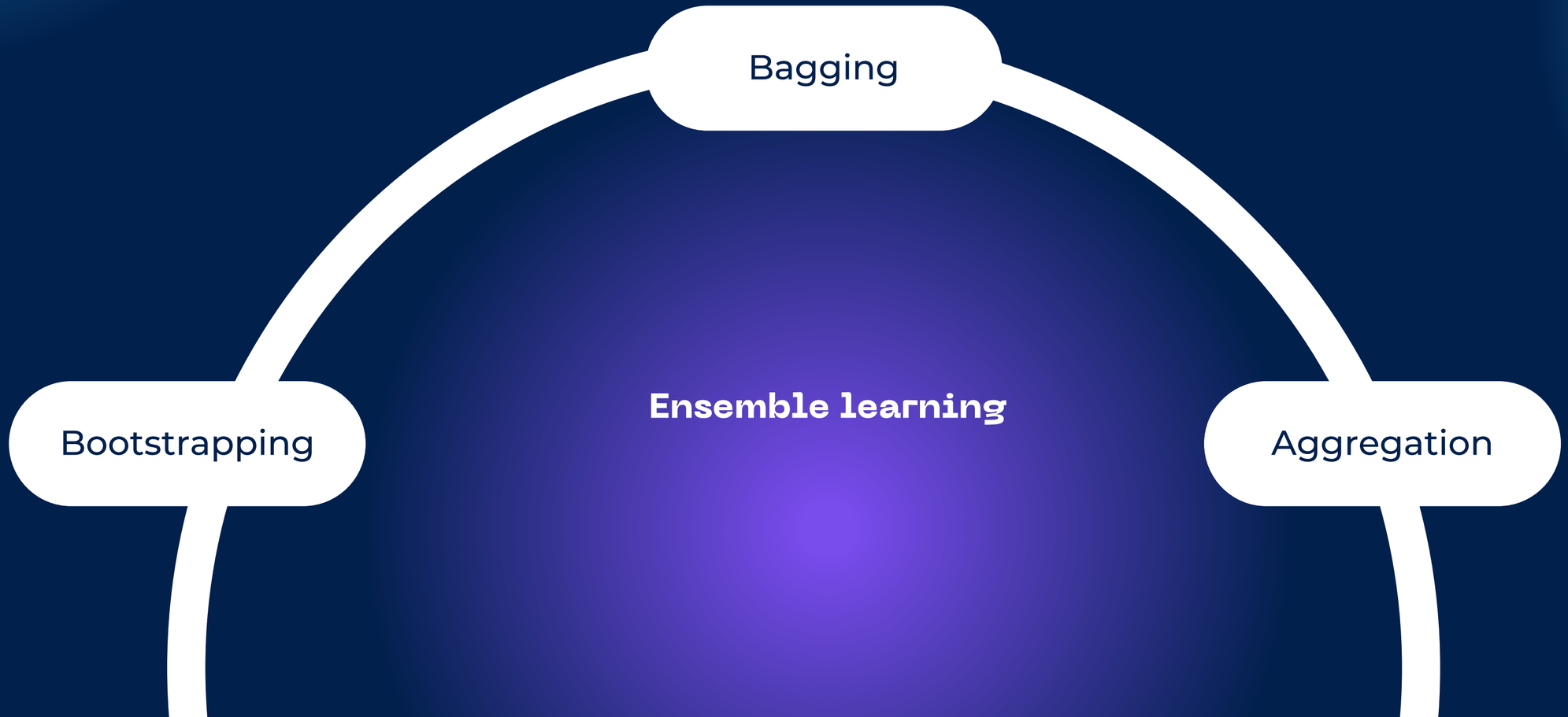
Variance

- It is a high variance classifier
- This means that it is sensitive to changes in data
- Small changes in data may have drastic impact on the tree

Bias

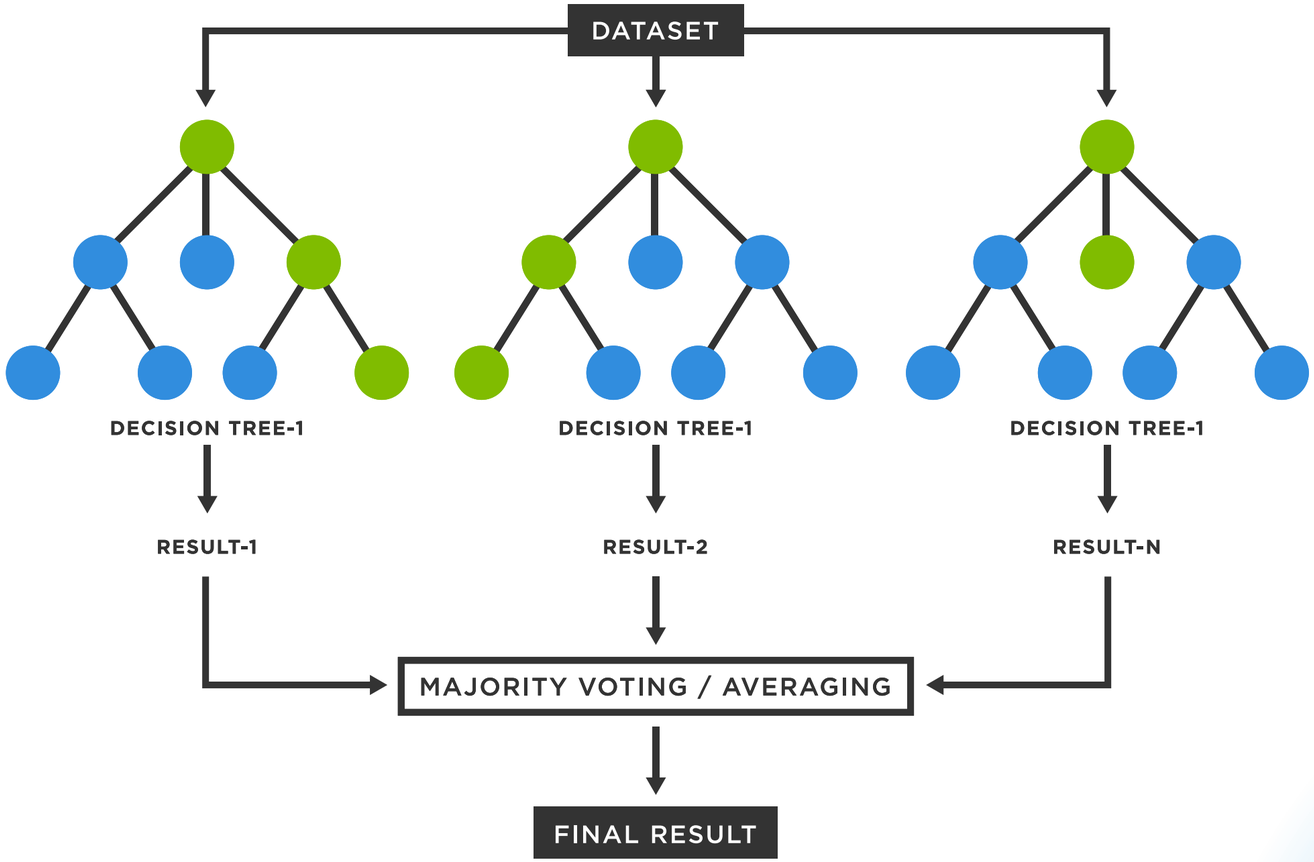
- May result in biased trees if a class dominates

Introduction To Random Forest



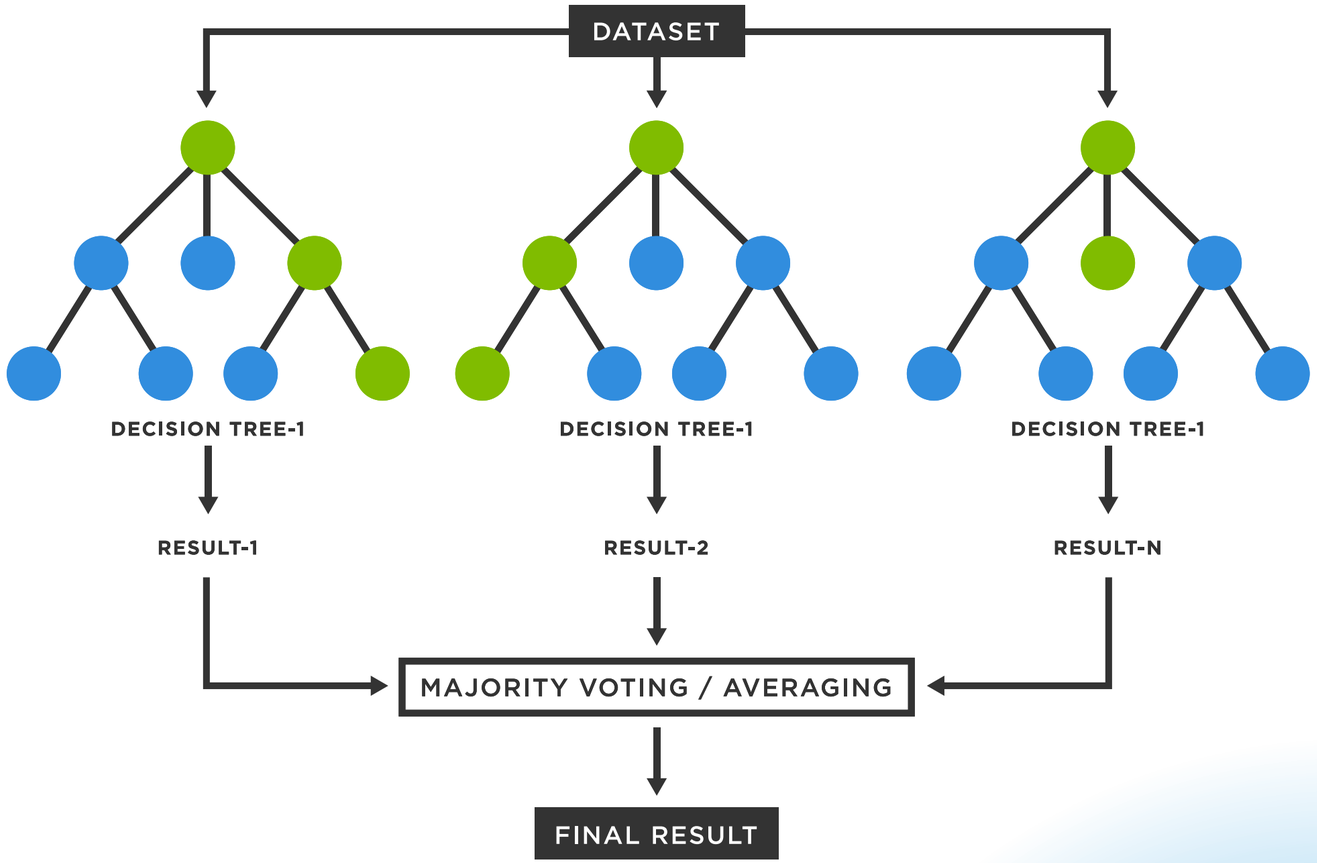
Random Forest Classifier

Robust and versatile algorithm capable of both classification and regression



Mechanism of Random Forest Algorithm

- Creating a Bootstrapped Dataset
- Creating Decision Trees from Bootstrapped Dataset
- Classifying bootstrapped dataset
- Aggregation of classification results (Bagging)
- Classification of Out–Of–Bag data



Mechanism Of Random Forest

Car	Car Color	Model Year	Mileage	Target
Ferrari	Red	2015	100000	Buy
Porsche	Blue	2021	300000	Dont Buy
Nissan	Blue	2017	150000	Dont Buy
Ferrari	Red	2020	300000	Buy
Nissan	Blue	2024	1700000	Buy

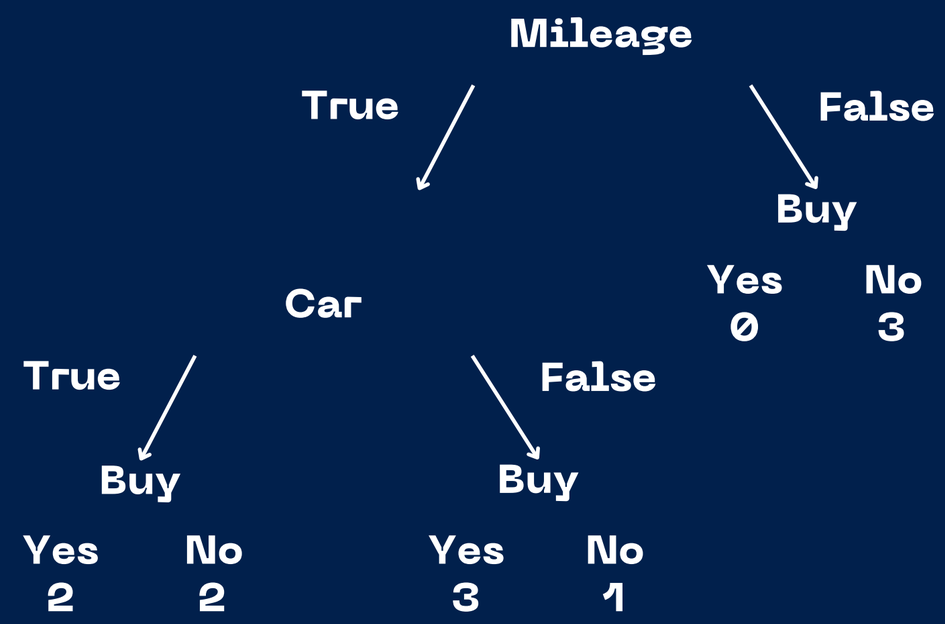
Car	Car Color	Model Year	Mileage	Target
Porsche	Blue	2021	300000	Dont Buy
Toyota Supra	Red	2020	300000	Buy
Porsche	Blue	2021	300000	Dont Buy

Creating a Bootstrapped Dataset:

- Random samples are selected from the dataset to create a sample subset
- Allows repetition of a sample in the same subset while some samples may be left out

Mechanism Of Random Forest

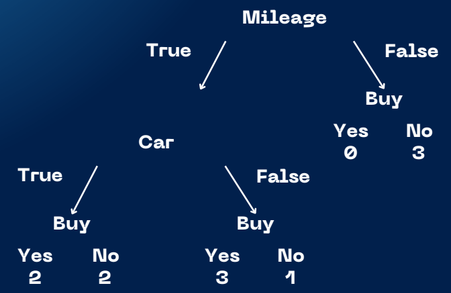
Car	Car Color	Model Year	Mileage	Target
Ferrari	Red	2015	100000	Buy
Porsche	Blue	2021	300000	Dont Buy
Nissan	Blue	2017	150000	Dont Buy
Ferrari	Red	2020	300000	Buy
Nissan	Blue	2024	1700000	Buy

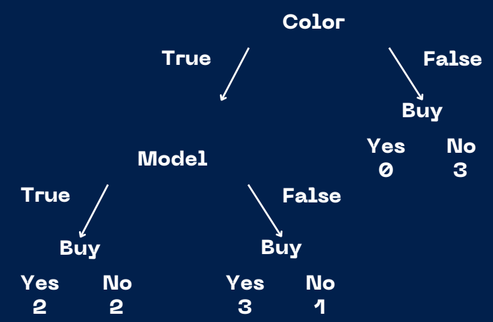


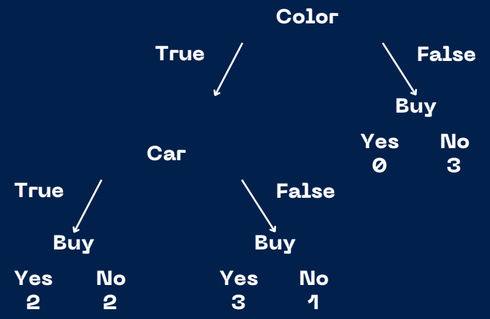
Creating Decision trees from Bootstrapped Dataset:

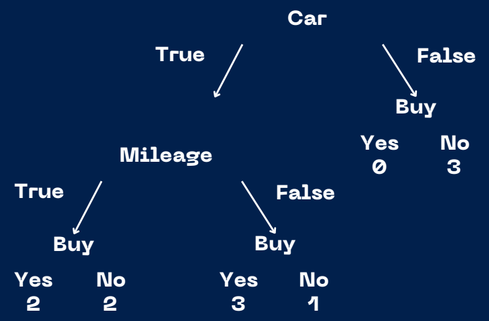
- Randomly select a subset of variables (columns) for splitting the root node
- Continue random selection of a subset of variables to split branches of the tree

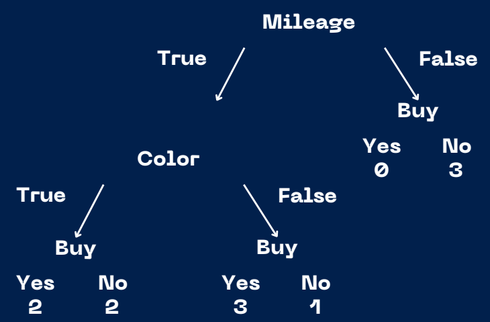
Mechanism Of Random Forest

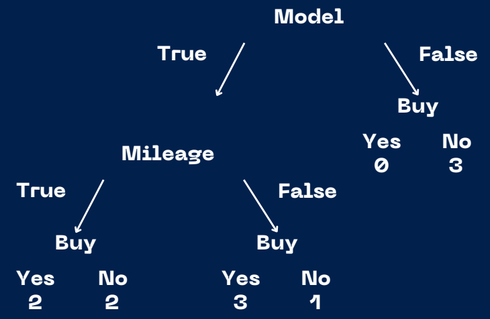








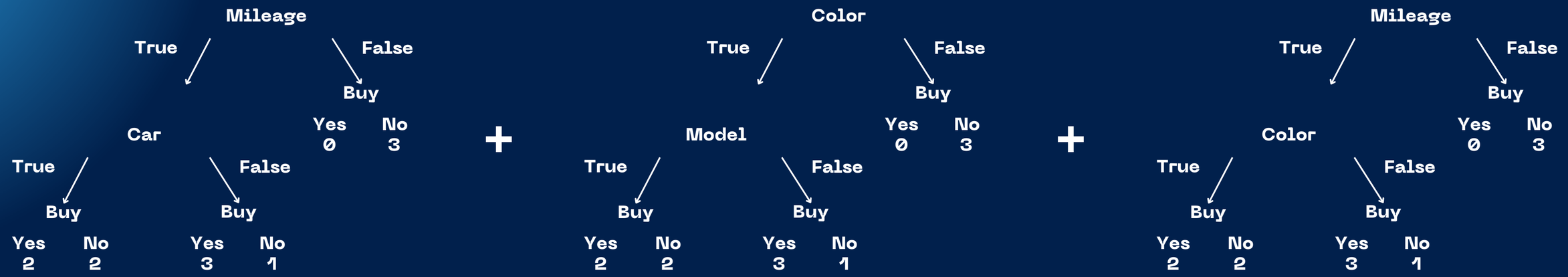




Creating Decison Trees from bootstrapped dataset:

- Create many such trees repeating the process of selecting random variables to split root node and branch nodes

Mechanism Of Random Forest



Bagging:

- The bootstrapped dataset is then run through the numerous decision trees and the classification results are aggregated

Mechanism Of Random Forest

Car	Car Color	Model Year	Mileage	Target
Ferrari	Red	2015	100000	Buy
Porsche	Blue	2021	300000	Dont Buy
Nissan	Blue	2017	150000	Dont Buy
Ferrari	Red	2020	300000	Buy
Nissan	Blue	2024	1700000	Buy

Car	Car Color	Model Year	Mileage	Target
Porsche	Blue	2021	300000	Dont Buy
Ferrari	Red	2015	100000	Buy
Porsche	Blue	2021	300000	Dont Buy

Out-Of-Bag Data:

Car	Car Color	Model Year	Mileage	Target
Nissan	Blue	2017	150000	Dont Buy

Car	Car Color	Model Year	Mileage	Target
Nissan	Blue	2017	150000	Dont Buy

Mechanism Of Random Forest

Bootstrapped Dataset:

Car	Car Color	Model Year	Mileage	Target
Porsche	Blue	2021	300000	Dont Buy
Ferrari	Red	2015	100000	Buy
Porsche	Blue	2021	300000	Dont Buy

Out-Of-Bag Data:

Car	Car Color	Model Year	Mileage	Target
Nissan	Blue	2017	150000	Dont Buy

Car	Car Color	Model Year	Mileage	Target
Nissan	Blue	2017	150000	Dont Buy

Out-Of-Bag Data:

- Entries that did not make it into the bootstrapped dataset
- They are run through all the trees and classification results are collected
- Accuracy of Random forest mdoel can be measured on the correct classification of out-of-bag samples
- Difference in actual and predicted values for out-of-bag data is called 'Out-Of-Bag Error'

Advantages of Random Forest

Accuracy

- Produce high accuracy
- Due to diverse trees
- Method of averaging reduces variance

Robustness

- Robust to outliers and non linear data
- Uses multiple trees derived from subsets of data

Versatility

- Can be used for both classification and regression tasks

Drawbacks of Random Forest

Complexity

- Computationally more expensive
- Due to larger processing time of multiple trees

Interpretability

- Complexity of Random Forest makes it harder to interpret
- Due to involvement of multiple trees

Overfitting

- Although less prone to overfitting, can still overfit noisy data

Case Study: Understanding Random Forest Via Formula One Weather Prediction

- Data Description
- Problem Statement
- Random Forest Implementation
- Model Optimization
- Model Evaluation
- Model Comparison

Data Description:

- **Dataset Source:**
  - This dataset appears to be derived from telemetry and weather monitoring systems used during Formula One races. The data is structured to assist in analyzing the conditions and performance metrics relevant to the racing events.

- **Dataset Size:**
  - Entries: Over 3,572,328 entries, indicating a substantial volume of data collected over multiple races or sessions.

Key Features:

- M_SESSION_UID: Unique identifier for the session (Float).
- M_SESSION_TIME: Timestamp within the session (Float).
- M_FRAME_IDENTIFIER: Identifier for a particular frame (Integer).
- M_PLAYER_CAR_INDEX: Index of the player's car (Integer).
- M_SECONDARY_PLAYER_CAR_INDEX: Index of the secondary player's car (Integer).
- M_TRACK_TEMPERATURE: Temperature of the racetrack (Integer, in degrees Celsius).
- M_AIR_TEMPERATURE: Ambient air temperature (Integer, in degrees Celsius).
- M_TOTAL_LAPS: Total number of laps in the session (Float).
- M_TRACK_ID: Identifier for the racetrack (Integer).
- M_WEATHER: Categorical representation of weather conditions (Integer).
- M_RAIN_PERCENTAGE: Percentage chance of rain (Float).

Target Variable:

M_WEATHER: This categorical variable describes the weather condition during the race, which is the primary variable of interest for predictions.

M_WEATHER OUTPUT VALUES:
0 = clear, 1 = light cloud, 2 = overcast, 3 = light rain, 4 = heavy rain, 5 = storm

Problem Statement:

- **Background:**
  - Formula One racing is a sport where weather conditions can significantly impact race strategy and outcomes.
  - Understanding and predicting weather conditions accurately during races is crucial for team strategists to make informed decisions regarding tire choices, pit stops, and race tactics
  - The complex dynamics of weather variables and their interactions pose a challenge in developing robust predictive models

- **Objective:**
  - The primary objective of this study is to develop a predictive model using historical Formula
  - This model aims to provide accurate weather predictions that can help race teams optimize their strategies in response to changing weather conditions.

Building Random Forest Model:

In [127]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

target = 'M_WEATHER'
# Let's assume 'df_scaled' is your preprocessed and scaled DataFrame
X = df_agg.drop(columns = target)  # Exclude target variable and closely related features
y = df_agg[target]

X_train, X_test, y_train, y_test = train_test_split(X, y, shuffle=False, test_size=0.2, ra
```

In [128]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Initialize Random Forest regressor
rf_model = RandomForestClassifier(n_estimators = 100, random_state=42)
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)
```

Model V1 Evaluation:

accuracy			0.79	54286
macro avg	0.64	0.56	0.57	54286
weighted avg	0.87	0.79	0.81	54286

- **Accuracy (79%):**
  - The model correctly predicted the weather 79 percent of the time. This suggests that the model was fairly accurate
- **Recall (56%):**
  - Model correctly identified 56 percent of positive cases
- **Precision (64%):**
  - On average, each class's precision (the ability of the classifier not to label as positive a sample that is negative) is 64%

Model Optimization:

Feature Selection:

Top 10 features were selected using the ANOVA F-Test to improve the accuracy of our random forest model in classifying the weather data

```
In [130]: from sklearn.feature_selection import f_regression, SelectKBest

          # Apply SelectKBest with ANOVA F-test
          # You can adjust 'k' to select the number of top features
          selector = SelectKBest(score_func=f_regression, k=15)  # Or use k=10 for top 10 features
          X_new = selector.fit_transform(X, y)

          # Get the selected feature names
          selected_features = X.columns[selector.get_support()]

          print("Selected features:", selected_features)
```

```
Selected features: Index(['M_PLAYER_CAR_INDEX', 'M_PIT_STOP_WINDOW_IDEAL_LAP',
       'M_TRACK_TEMPERATURE', 'M_TRACK_LENGTH', 'M_AIR_TEMPERATURE',
       'M_TRACK_ID', 'M_PIT_STOP_WINDOW_LATEST_LAP', 'M_SESSION_DURATION',
       'M_PIT_STOP_REJOIN_POSITION', 'M_WEATHER_FORECAST_SAMPLES_M_WEATHER',
       'M_RAIN_PERCENTAGE', 'M_AI_DIFFICULTY', 'M_TOTAL_LAPS', 'hour', 'min'],
      dtype='object')
```

Model V2 Evaluation:

accuracy			0.91	54286
macro avg	0.69	0.68	0.69	54286
weighted avg	0.90	0.91	0.91	54286

- **Accuracy (91%):**
  - The model correctly predicted the weather 91 percent of the time. This suggests that the model was very accurate at classifying weather
- **Recall (68%):**
  - Model correctly identified 56 percent of positive cases
- **Precision (69%):**
  - On average, each class's precision (the ability of the classifier not to label as positive a sample that is negative) is 64%

Final Model Evaluation:

Overall Performance:

- The Random Forest model performs exceptionally well for the most common classes (0 and 1), with high precision, recall, and F1-scores, indicating robustness and reliability in predictions for these classes.
- However, the model completely fails to identify instances of Class 2, which could be a point of concern, especially if predictions for this class are crucial.
- This could be due to various factors such as class imbalance, inadequate feature representation for Class 2, or insufficient training data for this class.
- The performance on Class 5, while perfect, should be interpreted with caution due to the small number of instances.

Random Forest Classification Report:

	precision	recall	f1-score	support
0.0	0.93	0.96	0.94	41921
1.0	0.84	0.78	0.81	11760
2.0	0.00	0.00	0.00	501
5.0	1.00	1.00	1.00	104
accuracy			0.91	54286
macro avg	0.69	0.68	0.69	54286
weighted avg	0.90	0.91	0.91	54286

Accuracy: 0.9112662564933869

Complexity

- The individual complexity of each tree depends on various parameters like max_depth, min_samples_split, and min_samples_leaf. Deeper trees, which result from higher max_depth or lower min_samples_leaf, increase the complexity of the model.

Scalability

- Random Forest can handle large datasets with thousands of input variables without variable deletion
- More trees require more memory and more processing power.
- One of the significant advantages of Random Forest is that it inherently allows for parallel processing

Computational Considerations

- The training time for Random Forest can be quite high compared to simpler model
- Each tree in the forest is an independent model, and storing many large decision trees in memory can be resource-intensive
- While training might be slow, predictions with Random Forest are relatively fast

Model Comparison: Logistic regression VS Random Forest

- Logitsic Regression Introduction
- Logitsic Regression Implementation
- Logisitc Regression VS Random Forest

Logitsic Regression Implementation:

```
from sklearn.linear_model import LogisticRegression
```

In [45]:
```
log_reg = LogisticRegression(max_iter=1000, random_state=42)
log_reg.fit(X_train, y_train)
```

[46]:
```
log_reg_pred = log_reg.predict(X_test)
```

Logistic Regression VS Random Forest

Accuracy: Random Forest outperforms Logistic Regression significantly, with over 91% accuracy compared to approximately 69%.
Precision and Recall: Random Forest also shows superior precision and recall for the major classes (0 and 1), and manages to recognize the rare Class 5 effectively.
Model Suitability: The Random Forest model's performance suggests that it is better suited for handling imbalanced data, capturing more complex relationships, and providing robust predictions across most classes.
Computational Complexity: Random Forest is typically more computationally intensive than Logistic Regression, which might be a consideration in some practical applications.

Logistic Regression Classification Report:

	precision	recall	f1-score	support
0.0	0.96	0.63	0.76	41921
1.0	0.41	0.94	0.57	11760
2.0	0.00	0.00	0.00	501
5.0	0.00	0.00	0.00	104
accuracy			0.69	54286
macro avg	0.34	0.39	0.33	54286
weighted avg	0.83	0.69	0.71	54286

Accuracy: 0.6893858453376561
ROC AUC: 0.6832725742629345

Random Forest Classification Report:

	precision	recall	f1-score	support
0.0	0.93	0.96	0.94	41921
1.0	0.84	0.78	0.81	11760
2.0	0.00	0.00	0.00	501
5.0	1.00	1.00	1.00	104
accuracy			0.91	54286
macro avg	0.69	0.68	0.69	54286
weighted avg	0.90	0.91	0.91	54286

Accuracy: 0.9112662564933869

Model Improvement:

Model Tuning:

Further tuning the hyperparameters of the Random Forest could potentially improve recall for Class 1 and overall performance.

Increase Data Collection:

For underperforming classes, especially Class 2, collecting more data could help in training the model more effectively.

Conclusion

- The analysis successfully demonstrates how to preprocess, analyze, and model complex datasets and perform random forest clasification in a real-world context.
- The Random Forest model, after careful tuning and selection of features, provides robust predictions of weather conditions during Formula One races, which can significantly aid in strategic decision-making for racing teams.
- The document showcases effective data handling techniques, from initial cleaning to advanced model evaluation, highlighting the potential for machine learning in sports analytics, particularly under variable conditions like weather.

THANK YOU