

TABLE OF CONTENTS

PROBLEM SETTING:	3
PROJECT OBJECTIVE:	3
DATA SOURCE:	3
DATA DESCRIPTION:	4
EXPLORATORY DATA ANALYSIS:	5
1. Driver Experience (Years_Active):	5
2. Nationality:	5
3. Pole Positions and Podiums:	6
4. Years of Experience and Winning Championships	7
5. Fastest Laps and Win Rate:	7
6. Championships:	8
7. Pole Position Rate:	8
8. Race Wins and Pole Positions:	9
9. Heatmap to indicate Pearson's coefficient among continuous features:	9
DATA TRANSFORMATION	10
Data Pre-processing:.....	10
1. Balancing target feature data using SMOTE:	10
2. Dimension Reduction using Principal Component Analysis:	11
DATA MINING MODELS:	12
1. Naive Bayes classifier:.....	12
2. Logistic Regression:	13
3. Neural network	14
4. KNN	15
Accuracy Comparisons:	17
ROC Curve:.....	17
RECOMMENDATION	19
FUTURE SCOPE	19
CONCLUSION	20

PROBLEM SETTING:

The project aims to create an accurate machine learning model to predict the winner of the F1 championship based on performance and non-performance factors.

PROJECT OBJECTIVE:

Using a machine learning model, we can predict which racer has the potential to win the F1 Championship based on their past performance. The model will analyze statistics such as race starts, pole positions, race wins, podiums, fastest laps, and championship points earned. Additionally, the model will consider whether the driver is currently active in the sport, the year and decade of their earliest championship win, and various percentage rates such as pole rate, start rate, win rate, podium rate, and fast lap rate. By considering all these factors, the ML model will provide a prediction for the potential F1 Championship winner.

DATA SOURCE:

We have sourced the dataset from Kaggle. Please see the link:

<https://www.kaggle.com/datasets/dubradave/formula-1-drivers-dataset?resource=download>

DATA DESCRIPTION:

- **Championships:** the number of championships won by the team or driver.
- **Race_Entries:** the number of times the team or driver has entered a race.
- **Race_Starts:** the number of times the team or driver has started a race.
- **Pole_Positions:** the number of times the team or driver has qualified in pole position(i.e., starting the race in first place).
- **Race_Wins:** the number of times the team or driver has won a race.
- **Podiums:** the number of times the team or driver has finished on the podium (i.e., in the top three positions).
- **Fastest Laps:** the number of times the team or driver has set the fastest lap in a race.
- **Points:** the total number of championship points earned by the team or driver.
- **Active:** A Boolean value indicating whether the team or driver is currently active in the sport.
- **Championship Years:** the year of the earliest championship won by the team or driver.
- **Decade:** the decade of the earliest championship won by the team or driver.
- **Pole_Rate:** the percentage of races in which the team or driver has qualified in pole position.
- **Start_Rate:** the percentage of race entries in which the team or driver has started the race.
- **Win_Rate:** the percentage of races in which the team or driver has won.
- **Podium_Rate:** the percentage of races in which the team or driver has finished on the podium.
- **FastLap_Rate:** the percentage of races in which the team or driver has set the fastest lap.
- **Points_Per_Entry:** the average number of championship points earned per race entry.
- **Years_Active:** the number of years in which the team or driver has competed in the sport.
- **Champion:** A Boolean value indicating whether the team or driver has ever won a championship.

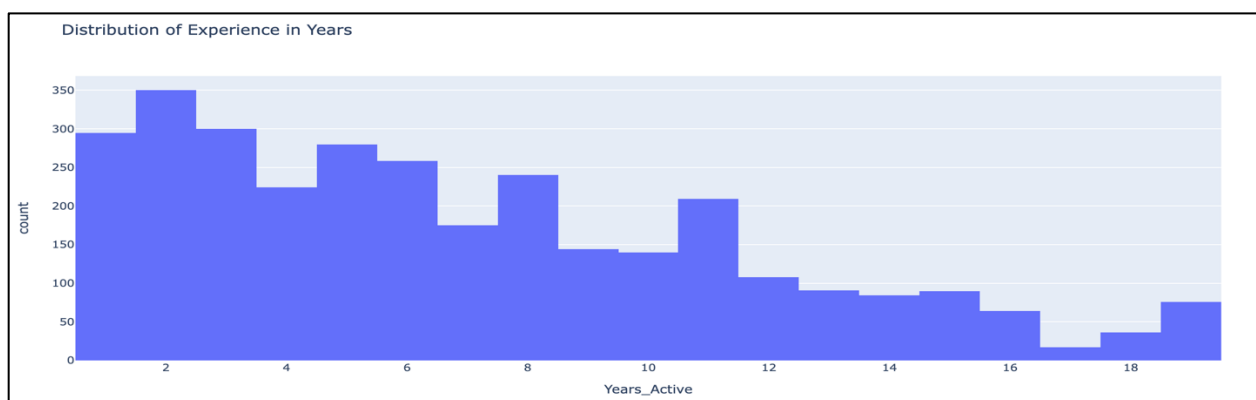
EXPLORATORY DATA ANALYSIS:

Exploratory Data Analysis (EDA) is a process of analyzing and summarizing data sets to gain insights and understanding of the underlying patterns, distributions, and relationships between variables. It involves a range of statistical and graphical techniques to help identify patterns, outliers, and relationships in the data. The goal of EDA is to discover interesting and informative features in the data that may guide further analysis, and to develop an intuitive understanding of the data. This is typically done through a process of visualization and summary statistics, such as histograms, box plots, scatter plots, correlation matrices, and other graphical and numerical methods.

In our dataset, we charted a few plots to explore the nature and develop an understanding of certain features.

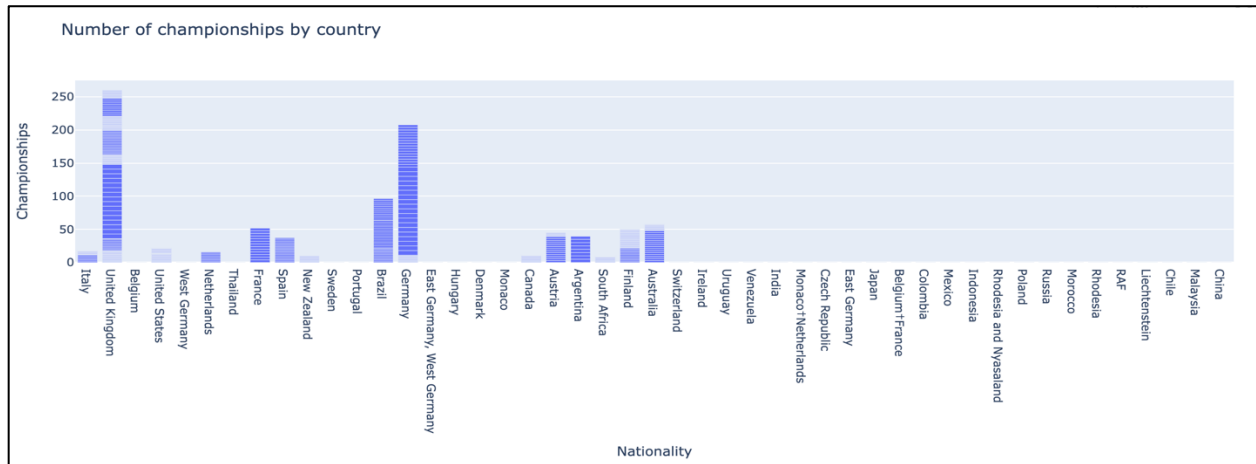
1. Driver Experience (Years_Active):

From the following histogram we can infer that the data for this feature is skewed and most of the drivers have about 2-4 years of experience. This data can be useful to model our training data as experience plays a major role in winning championships.



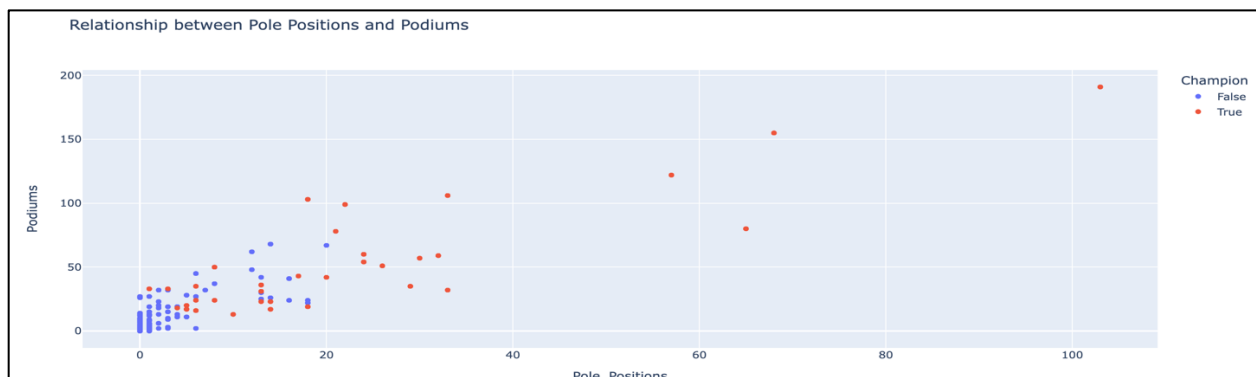
2. Nationality:

From the following bar chart, we can infer that United Kingdom has the highest number of championships by nationality. This gives us an understanding that many drivers are from the UK, and they tend to join good teams that will eventually get them a championship.



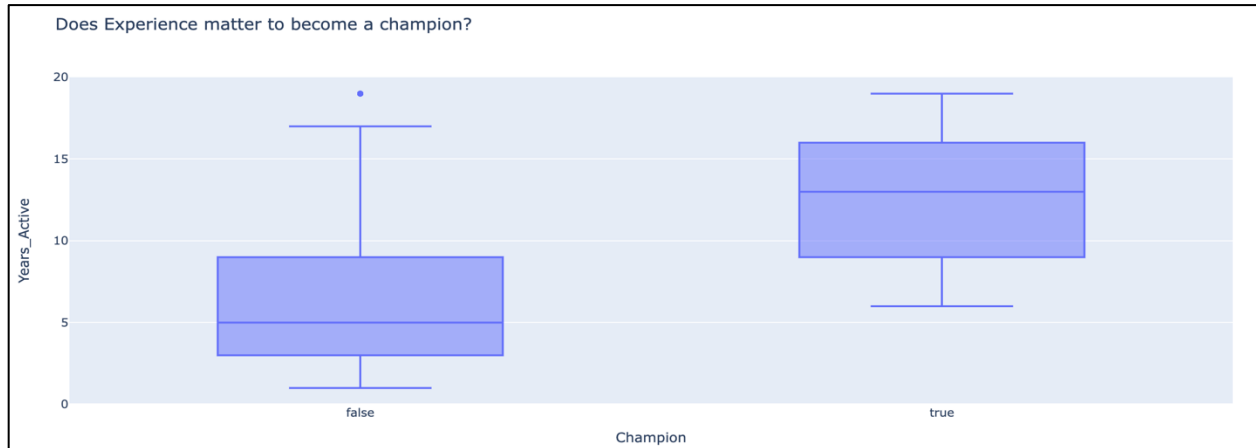
3. Pole Positions and Podiums:

A Pole position is defined as the first position in formula one terms. This is decided in the qualifying race which happens a day before the actual race. A Podium is defined as securing either position in the top three spots after the actual race is finished. These features are the frequencies which drivers have secured in their tenure. The following scatter plot tells us that drivers who had more pole positions were likely to have more podiums and hence a good chance of becoming a champion.



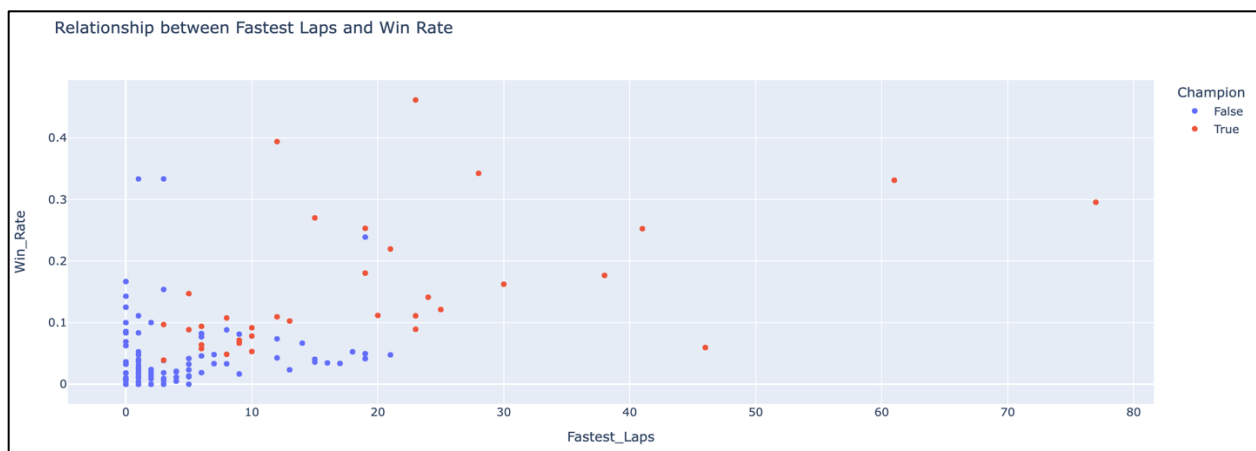
4. Years of Experience and Winning Championships

We plot a box plot to determine how years of experience affects winning championships. From the following box plot we can infer that drivers who had more experience were winning more championships. Drivers who had 10 to 15 years of experience approximately were likely to become champions.



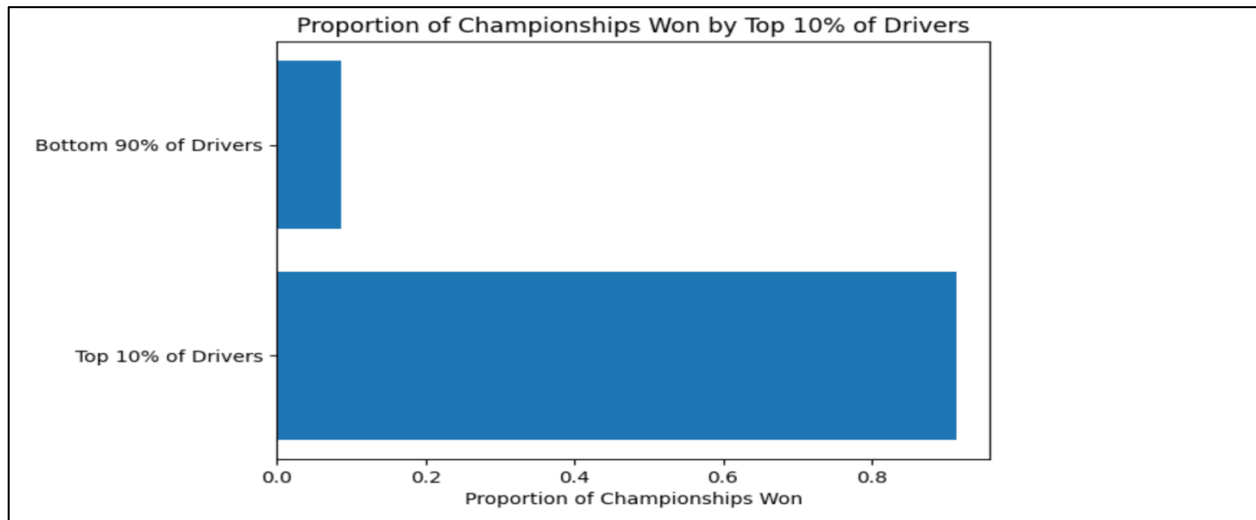
5. Fastest Laps and Win Rate:

A fastest lap is the best performed lap by the driver in each race. Performance is measured by time and the feature in our dataset is the frequency in which each driver has had fastest laps in the races overall. Win rate is determined by the number of times the driver secures a podium and the total number of races participated. The below scatter plot shows us that Drivers who had more than 10 fastest laps were more likely to be champions.



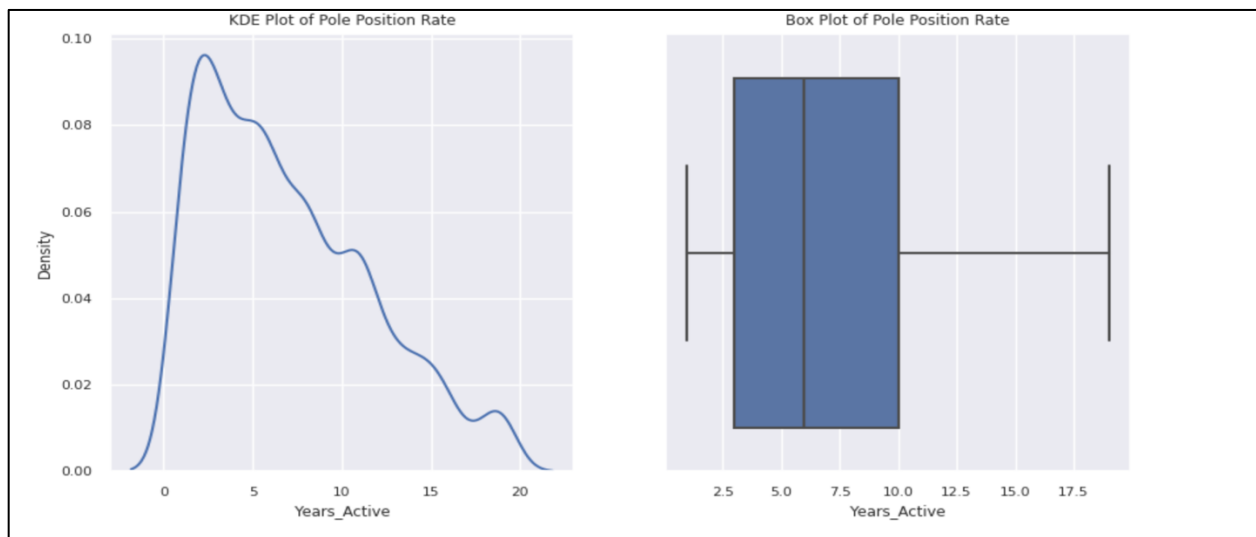
6. Championships:

A championship is secured by a driver who collects most of the points throughout the season. After aggregating the championship feature in our dataset, we were able to conclude that the top 10.0% of drivers have won 91.35% of all championships.



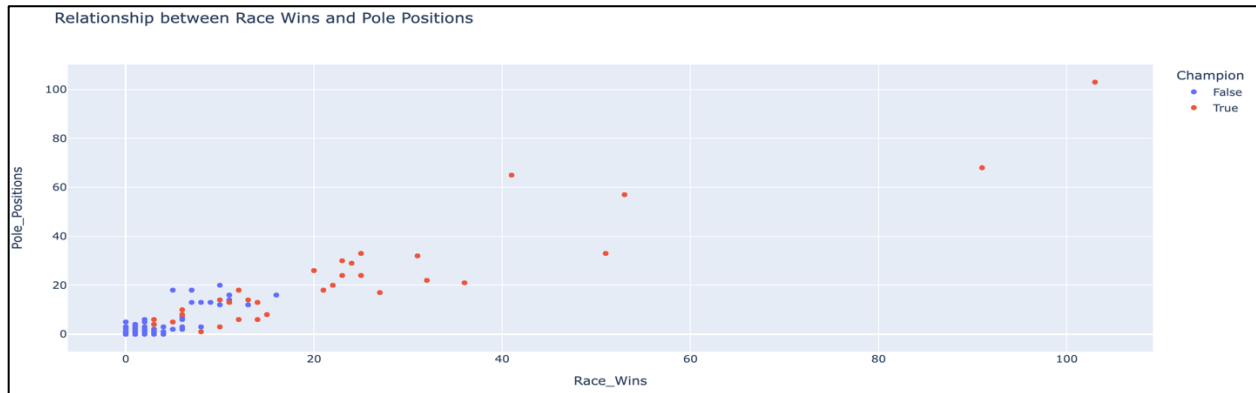
7. Pole Position Rate:

Pole position rate is determined by the ratio between the number of times a driver secures the first position in the qualifying race and the total number of races he/she participates in.



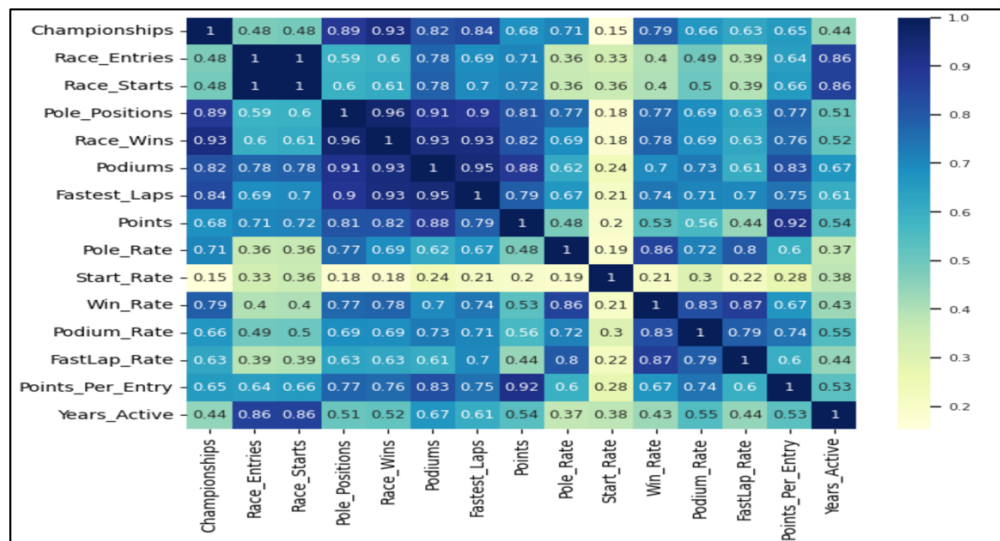
8. Race Wins and Pole Positions:

From the below scatter plot we can infer that the more pole positions a driver will secure, the more likely the driver will be winning that race. Winning more races will likely lead to securing a championship.



9. Heatmap to indicate Pearson's coefficient among continuous features:

Pearson's correlation coefficient is a measure of the strength and direction of the linear relationship between two variables. A correlation coefficient of +1 indicates a perfect positive linear relationship, meaning that as one variable increases, the other variable increases proportionally. A correlation coefficient of -1 indicates a perfect negative linear relationship, meaning that as one variable increases, the other variable decreases proportionally. A correlation coefficient of 0 indicates no linear relationship between the two variables.



DATA TRANSFORMATION

Data Pre-processing:

1. Balancing target feature data using SMOTE:

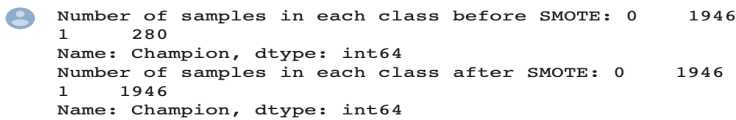
SMOTE (Synthetic Minority Over-sampling Technique) is a method used to address the problem of imbalanced datasets. Imbalanced datasets refer to datasets where one class (the minority class) has significantly fewer instances than the other class (the majority class). This can lead to biased models that perform poorly in the minority class. SMOTE works by generating synthetic samples of the minority class. It does this by randomly selecting a minority class instance and then selecting one of its nearest neighbors. It then creates a new instance by interpolating between the two instances.

This process is repeated until the desired number of synthetic instances is generated. The idea behind SMOTE is that by generating synthetic instances, the minority class will be better represented in the dataset and the model will be able to learn from these instances. This can improve the performance of the model in the minority class.

```
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=44)

# Apply SMOTE to the training data
sm = SMOTE(random_state=42)
X_resampled, y_resampled = sm.fit_resample(X_train, y_train)

# Print the number of samples in each class before and after SMOTE
print('Number of samples in each class before SMOTE:', y_train.value_counts())
print('Number of samples in each class after SMOTE:', y_resampled.value_counts())
```



```
Number of samples in each class before SMOTE: 0    1946
1         280
Name: Champion, dtype: int64
Number of samples in each class after SMOTE: 0    1946
1         1946
Name: Champion, dtype: int64
```

In summary, SMOTE is used to address the problem of imbalanced datasets by generating synthetic instances of the minority class. It is implemented using libraries such as imbalanced-learn in Python.

2. Dimension Reduction using Principal Component Analysis:

PCA (Principal Component Analysis) is a widely used dimensionality reduction technique that is used to reduce the number of features in a dataset while retaining most of the information. PCA works by identifying the directions (principal components) in which the data varies the most and projecting the data onto a lower-dimensional space spanned by these principal components.

To implement PCA, we first standardize the data by subtracting the mean and dividing it by the standard deviation. We then compute the eigenvectors and eigenvalues of the covariance matrix of the standardized data. The eigenvectors represent the principal components of the data, and the eigenvalues represent the amount of variance explained by each principal component.

We can then project the data onto a lower-dimensional space spanned by the principal components. We can choose the number of principal components to retain based on the amount of variance explained by each component or based on a desired level of dimensionality reduction.

```
from sklearn.decomposition import PCA

# Apply PCA to the resampled training data
pca = PCA(n_components=10)
X_pca = pca.fit_transform(X_resampled)

# Combine the PCA components with the target variable
pca_df = pd.DataFrame(data=X_pca, columns=['PC{}'.format(i) for i in range(1, 11)])
pca_df['Champion'] = y_resampled.values

# View the first few rows of the PCA data with the target variable attached
print(pca_df.head())
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	\
0	169.681046	-15.572245	-4.223166	1.050537	-0.056401	1.549158	-0.451221	
1	-365.410211	9.542497	-4.688968	0.678440	-0.369120	1.428407	0.022850	
2	-286.453234	21.800868	-5.419524	0.983663	-0.382058	1.242739	0.590954	
3	175.645977	-5.551220	-4.495558	0.981997	-0.399666	0.970480	-0.050602	
4	-390.384436	3.634371	-1.643912	-3.073142	-0.737472	-1.306841	-0.473433	

	PC8	PC9	PC10	Champion
0	0.537667	0.863609	0.004107	0
1	0.199484	0.398764	-0.375745	0
2	-0.260296	-0.895497	-0.595801	0
3	-0.063202	-0.062650	-0.146663	0
4	-0.853984	0.463997	-0.171243	0

In summary, PCA is a dimensionality reduction technique used to reduce the number of features in a dataset while retaining most of the information. It is implemented using libraries such as scikit-learn in Python.

DATA MINING MODELS:

To build the best possible classification model for our dataset, we explored various data preprocessing techniques. Specifically, we worked with raw data, standardized data, and PCA-transformed data. To determine the optimal classification model, we tested five different models on our dataset.

To ensure that the model generalizes well to unseen data, we split the data into an 80% training set and a 20% testing set. Initially, we ran the model on the raw data without the best parameters, but we realized that our data was imbalanced, with one class having significantly more samples than the other.

To address this issue, we used SMOTE (Synthetic Minority Over-sampling Technique) to balance the data. We then applied PCA to the SMOTE data to reduce the dimensionality of the data and remove any redundant features that might not contribute much to the classification.

Finally, we applied the five different classification models to the PCA-transformed SMOTE data to determine which model performed the best on our dataset. By utilizing these various techniques and testing multiple models, we aimed to build the most effective classification model for our specific dataset.

1. Naive Bayes classifier:

Naive Bayes is a machine learning algorithm that is based on the Bayes theorem, which assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. This assumption is called the "naive" assumption, hence the name Naive Bayes. It is commonly used in text classification tasks, such as spam filtering, sentiment analysis, and topic classification, due to its simplicity and speed. It works by calculating the probability of each class given the input data and selecting the class with the highest probability as the output.

One of the key advantages of Naive Bayes is that it requires a relatively small amount of training data compared to other machine learning algorithms. It is also computationally efficient and can handle high-dimensional data with ease. The outcome of a Naive Bayes model is a prediction of the class label for the input data. The accuracy of the model depends on the quality of the training data, the features selected, and the assumptions made by the algorithm. Naive Bayes may not be appropriate for all types of data, particularly when the assumption of independence between features is not met. However, in many cases, it can provide accurate and fast predictions with minimal training data.

Data Type	Accuracy
Original Dataset	87.53 %
Smote Dataset	36.64 %

True/ False	precision	recall	f1-score	support
1	0.50	1.0	0.50	1946
0	0.0	0.0	0.0	1946

2. Logistic Regression:

Logistic Regression is a machine learning algorithm used for classification tasks. It is a statistical method used to predict the probability of a binary outcome (i.e., two possible classes) based on one or more predictor variables. The algorithm is called "logistic" because it uses a logistic function to model the probability of the output.

Logistic Regression is widely used in various applications such as medical diagnosis, marketing analysis, and credit scoring. It is especially useful when the output variable is categorical, and the data is linearly separable, which means that there is a clear boundary that can separate the classes.

The effectiveness of the logistic regression model depends on numerous factors such as the quality of the training data, the feature selection, and the choice of the hyperparameters. Logistic regression is often used as a baseline model for classification tasks, as it is relatively simple and easy to interpret. However, it may not perform as well as more complex models on more complicated datasets.

Data Type	Accuracy
Original Dataset	92.04 %
SMOTE Dataset	96.09%

True/ False	precision	recall	f1-score	support
1	0.97	0.98	0.98	1946
0	0.87	0.81	0.84	280

3. Neural network

A neural network is a machine learning model that is inspired by the structure and function of the human brain. It is composed of interconnected nodes or "neurons" that work together to process information and make predictions. Neural networks can learn complex relationships between input and output data, making them useful for a wide range of tasks such as image classification, speech recognition, and natural language processing.

Neural networks are particularly useful in cases where traditional machine learning models, such as logistic regression, may not be sufficient. They can handle substantial amounts of data and automatically extract relevant features from the input. Additionally, neural networks can perform well even when the relationship between the input and output variables is non-linear and complex.

The outcome of a neural network model is a prediction of the output variable based on the input data. The accuracy of the model depends on several factors such as the size of the neural network, the number of layers, the activation functions used, and the choice of hyperparameters. Neural networks can be trained using a variety of optimization algorithms such as gradient descent and backpropagation.

One of the main advantages of neural networks is their ability to learn and adapt to new data. Once trained, the neural network can make accurate predictions on new, unseen data. However, neural networks can be computationally intensive and may require a large amount of training data and time to train. Additionally, the output of a neural network can be difficult to interpret, making it harder to understand how the model arrived at its predictions.

Data Type	Accuracy
Smote Dataset	98.32%
Normal Dataset	98.01%

True/ False	precision	recall	f1-score	support
0	0.99	0.99	0.99	836
1	0.92	0.92	0.92	119

4. KNN

K-Nearest Neighbors (KNN) is a machine learning algorithm used for classification and regression tasks. It is a non-parametric algorithm that works by finding the k closest points in the training dataset to the input point and using their labels to predict the label of the input. KNN is particularly useful when there is no prior knowledge about the distribution of the data, making it applicable to a wide range of datasets.

KNN works by measuring the distance between the input point and the k closest points in the training dataset. The output of the model is the majority class label among the k closest points. The value of k can be chosen based on the size and complexity of the dataset, and it can affect the performance of the model.

KNN is often used in applications such as image recognition, recommendation systems, and text classification. It is particularly useful when the input data has a simple structure, and the relationships between the input and output variables are straight forward. KNN can also be used for regression tasks by taking the average of the k closest points instead of the majority class label.

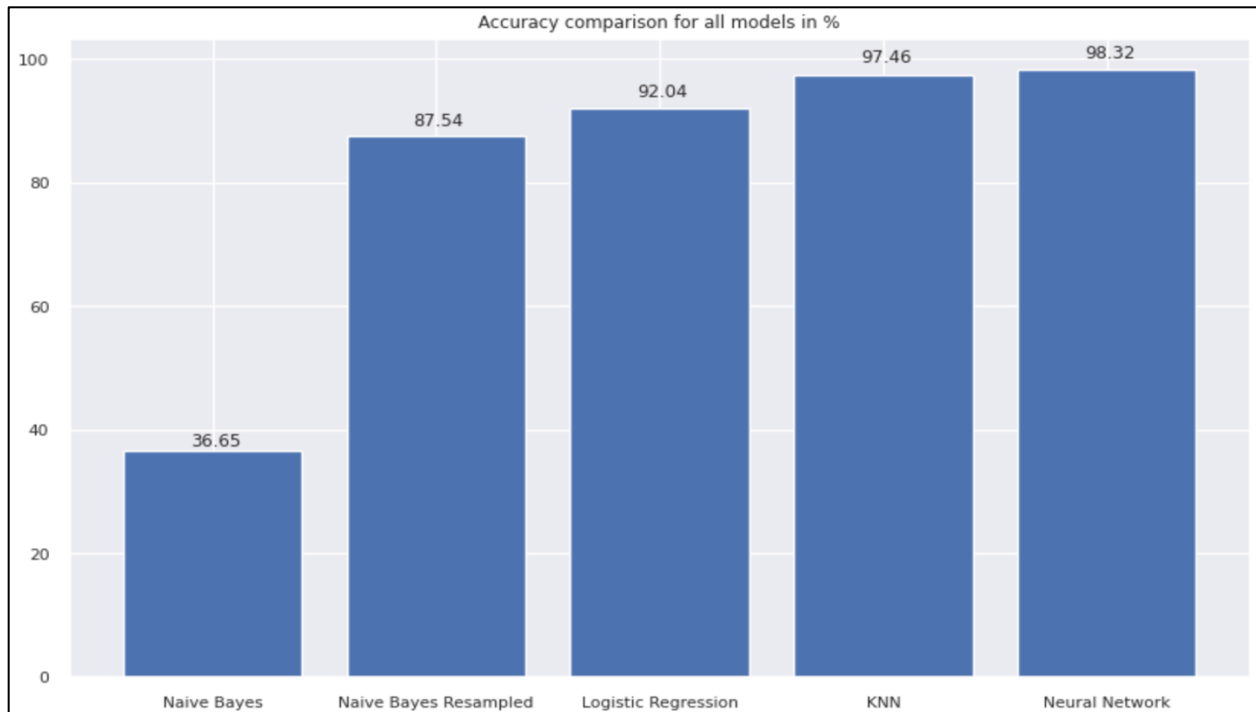
The outcome of a KNN model is the predicted class label or the predicted value based on the input data. The accuracy of the model depends on a range of factors such as the distance metric used, the value of k, and the quality of the training data. KNN can be computationally intensive, especially when the dataset is large, making it less suitable for real-time applications.

Data Type	Accuracy
Smote Dataset	97.45%
Normal Dataset	98.01%

True/ False	precision	recall	f1-score	support
1	1.00	0.95	0.97	1946
0	0.95	1.00	0.98	1946

Accuracy Comparisons:

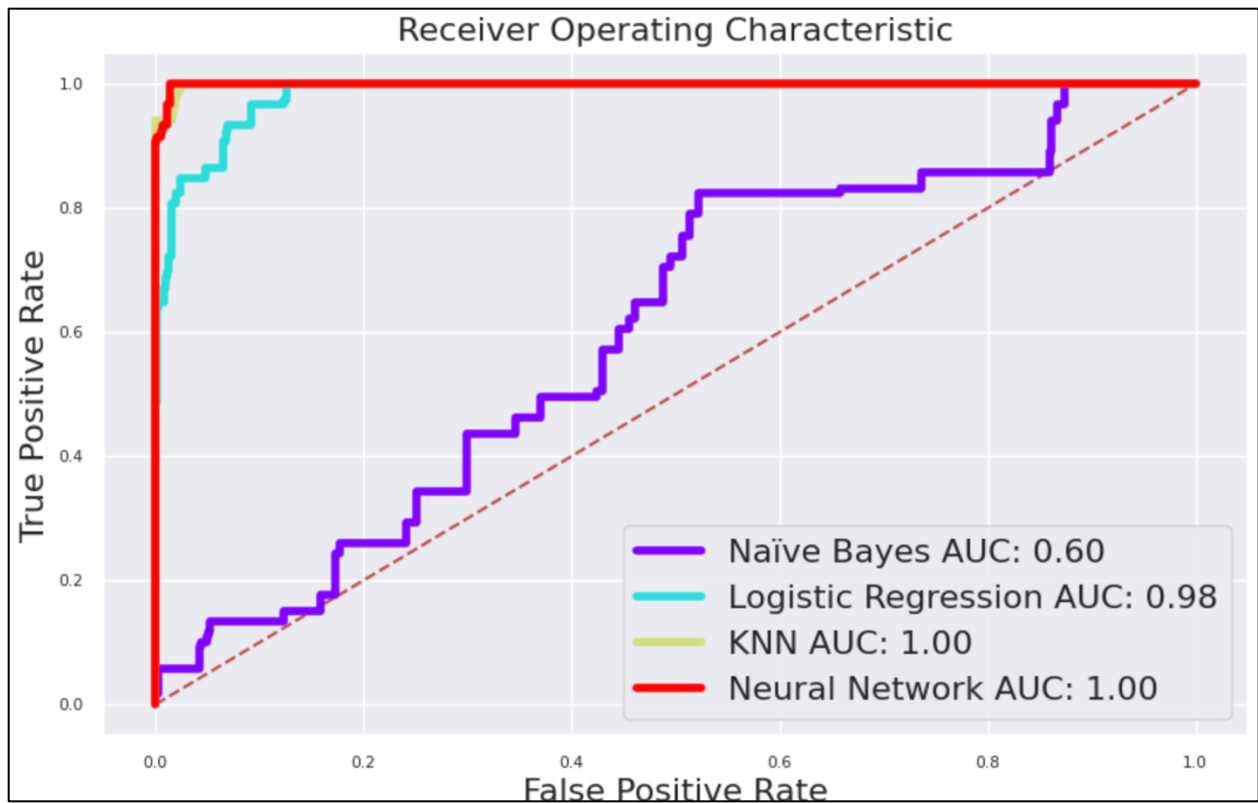
After recording accuracies of all the models, we visualized the comparisons using a bar chart to recommend the most optimal model.



ROC Curve:

A ROC (Receiver Operating Characteristic) curve is a graphical representation of the performance of a binary classification model. The curve is created by plotting the true positive rate (sensitivity) against the false positive rate (1-specificity) at different classification thresholds.

In data mining, the ROC curve is used to evaluate the performance of a binary classifier by measuring how well it can distinguish between positive and negative samples. A perfect classifier would have a ROC curve that hugs the top left corner of the graph, where both the true positive rate and false positive rate are close to 1.0.



RECOMMENDATION

Driver performance analysis: Data mining techniques could be used to analyze data on driver performance, including lap times, race finishes, and driver behavior during races. This could help teams identify areas for improvement and develop more effective training programs.

Car performance analysis: Data mining could also be used to analyze data on car performance, including engine and chassis data, tire wear, and fuel consumption. This could help teams optimize their car setups for different tracks and race conditions.

Predictive modeling: Data mining techniques could be used to develop predictive models for various aspects of Formula 1 racing, such as predicting race outcomes or forecasting future trends in car and driver performance.

FUTURE SCOPE

Real-time analysis: With the increasing availability of real-time data during races, data mining techniques could be used to analyze this data in real-time to provide teams with insights and recommendations during races.

Machine learning: Machine learning techniques could be used to develop more sophisticated predictive models and decision support systems for teams.

Big data: With the increasing volume and variety of data available in Formula 1 racing, big data technologies and techniques could be used to store, process, and analyze this data more effectively.

CONCLUSION

In this Machine Learning project, the goal is to predict whether a Formula 1 racer would win the championship or not, utilizing various classification ML models. The team employed a multi-step approach, starting with data preprocessing to ensure that the data is in a suitable format for the models. They then addressed the issue of class imbalance in the data by using the Synthetic Minority Oversampling Technique (SMOTE) to generate synthetic data for the minority class. After this, the team applied four different algorithms, including Logistic Regression, KNN, Naive Bayes, and Neural Network, and evaluated their performance using various evaluation metrics.

Through thorough analysis, the team determined that the Neural Network algorithm is the most effective with an accuracy rate of 98.32%. The team further fine-tuned the Neural Network model by adjusting various hyperparameters such as the number of layers, neurons per layer, activation function, and learning rate, to achieve the best possible accuracy rate. This model can be incredibly useful for stakeholders such as racing teams, sponsors, and fans to make informed decisions and predictions about Formula 1 races. By accurately predicting the potential winner, this model can save time, money, and resources, ultimately leading to better decision-making and improved outcomes.