FORMULA 1 PREDICTION

HAMNA NOOR
ABUBAKAR

ABSTRACT

The report explores F1 racing predictions. How different parameters like location, circuit and constructor affect the result of the race. Randomforest model is used for training, testing and deployment.

BACKGROUND

Formula one (also known as Formula 1 or F1) is the highest class of international single-seater auto racing owned by the Formula One Group. The word "formula" in the name refers to the set of rules to which all participants' cars must conform. A Formula One season consists of a series of races, known as Grands Prix (French for "grand prizes" or “great prizes"), which take place worldwide on purpose-built circuits and on public roads.

This project aims towards understanding the data of past Formula 1 races and predict the outcome of the race for user inputs by using Machine Learning Algorithms.

EXPERIMENTAL SETUP

The experimental setup is divided in three steps. Different datasets are merged into one file and different preprocessing steps are performed to get clean data. Linear Regression model is implemented on the set and deployment.

PREPROCESSING

Pre-processing involves cleaning the data before feeding it to the machine learning model. If there are any duplicates or missing values. They need to be handled accordingly to avoid erroneous results.

The dataset consists of multiple files. Each containing specific information. All the data files are merged into one file based on common attributes and \and that file is further used for pre-processing.
Column names are renamed for clarity and additional columns like drive_dnf(driver did not finish) added to facilitate production analysis.

Feature Engineering:

driver_confidence and constructor_relaiblity are new features added to quantify the confidence of drivers and reliability of constructors.

Active Status Identification:

The code creates binary indicators (active_driver and active_constructor) to identify whether drivers and constructors are currently active in the sport.

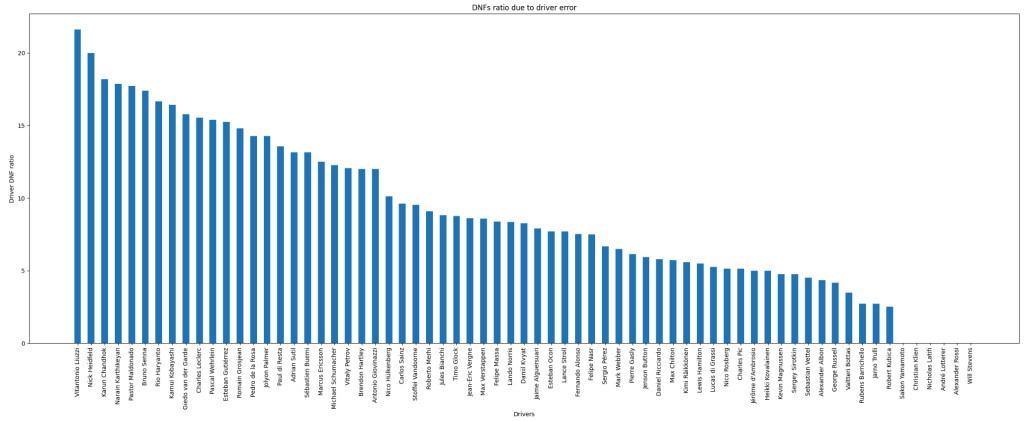
Data Filtering:

This setup can be useful for later filtering the dataset to focus only on active participants or for analyzing the performance of active drivers and constructors separately from those who have retired.

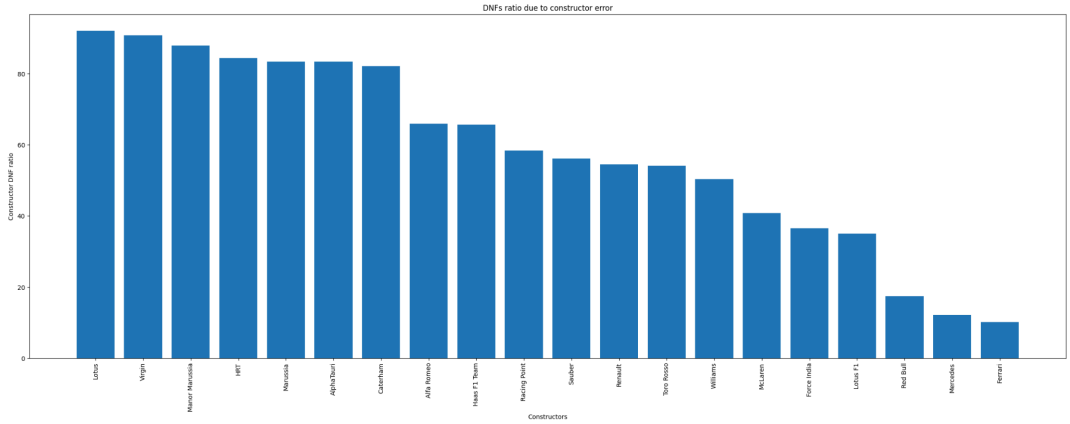
Exploratory Data Analysis

The ratio of driver did not finish, constructor did not finish and races won in hometown for driver and constructor are analyzed since they greatly influence the results.

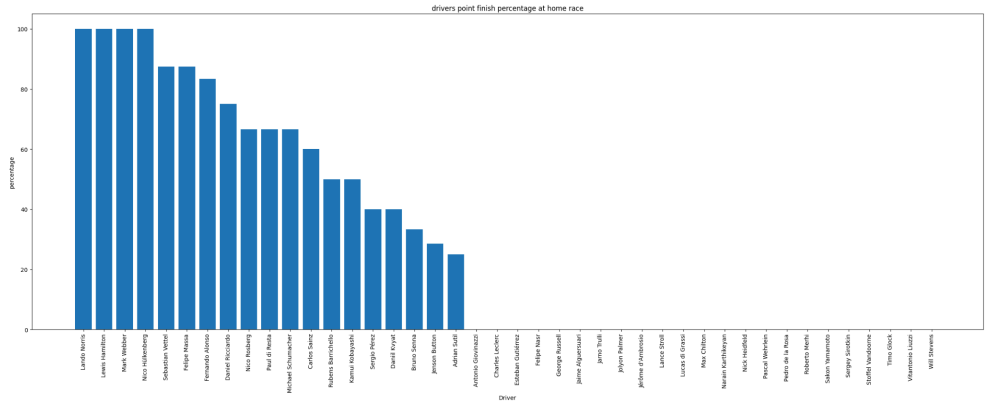
"Did not finish" ratio for each driver is represented in each bar graph. Calculated by dividing the total number of DNFs by the number of races entered.



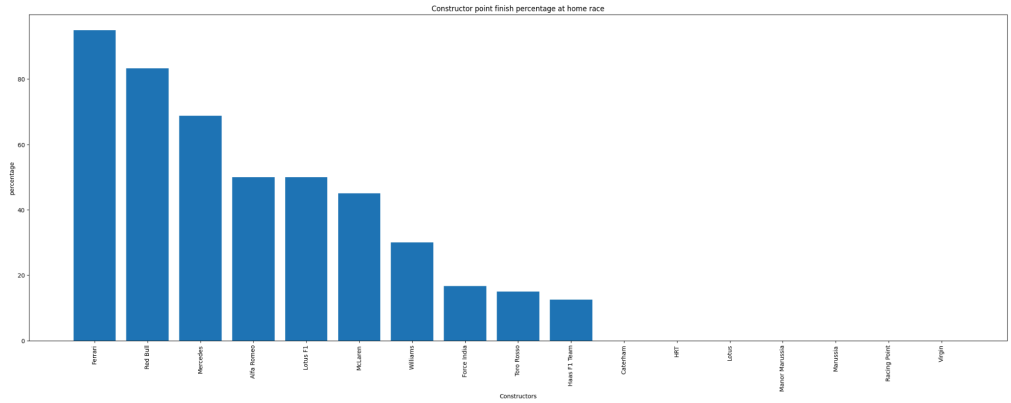
The bar graph below shows the DNF ratio for each constructor, with the constructors sorted in descending order based on their DNF ratios.



The following bar graph shows the percentage of times driver ended in top 10 for races in how country.



Similarly the same ratio is calculated with respect to constructor.

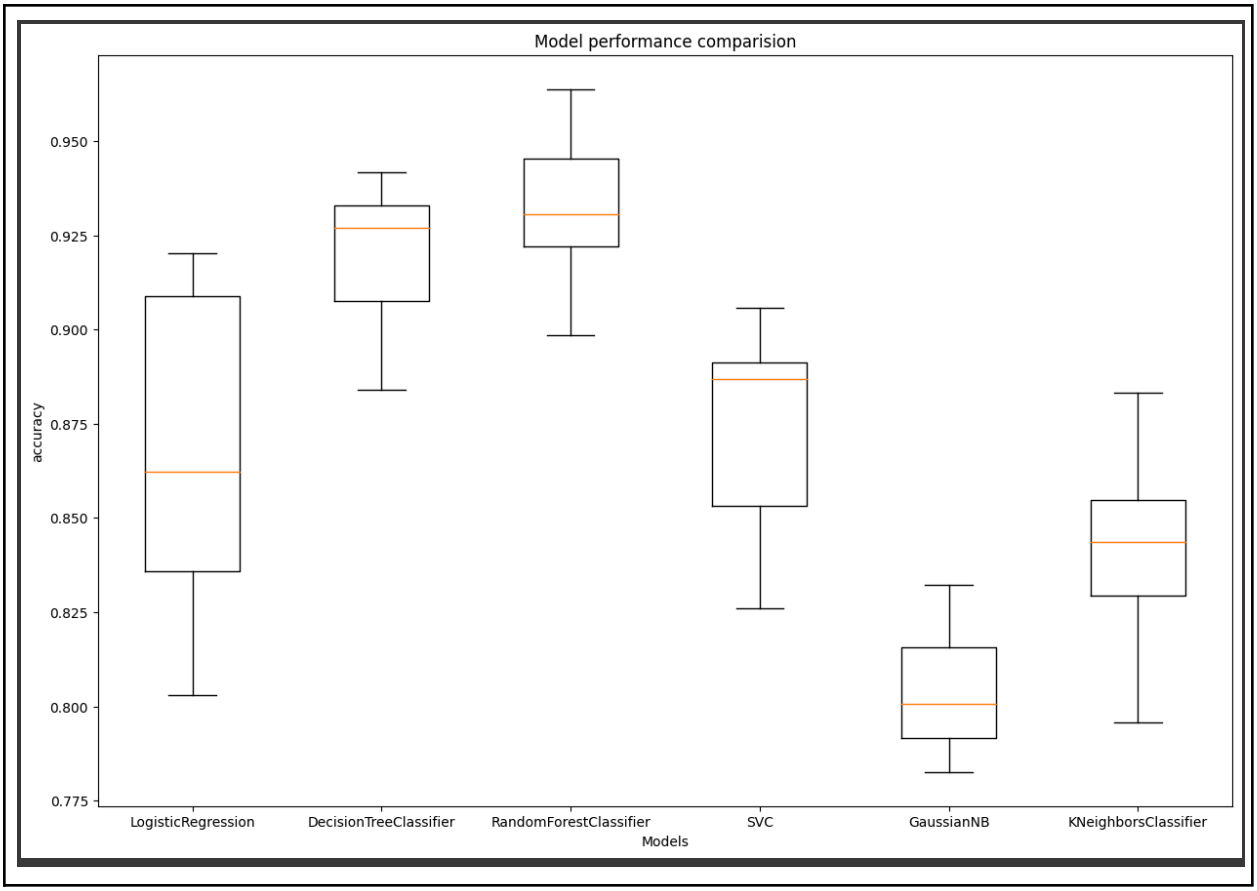


MACHINE LEARNING MODEL

Cross Validation for Models:

Cross validation is done for various models like 'LogisticRegression', 'DecisionTreeClassifier', 'RandomForestClassifier',' SVC', 'GaussianNB' and KNeighborsClassifier. It is done using the StratifiedKFold technique to keep the proportion of output class in the dataset.
The following is the output for cross validation.

```
LogisticRegression : 0.8661536020311011
DecisionTreeClassifier : 0.9193060404104518
RandomForestClassifier : 0.931656616947001
SVC : 0.874933883423252
GaussianNB : 0.804400719348355
KNeighborsClassifier : 0.8400137522479636
```



It shows the Random Forest Classifier has the best performance among all.
Finally the dataset is trained and tested using the random forest classifier. The following [parameters are set for RFC.

```
random_parms = {
    'n_estimators': [int(x) for x in np.linspace(start=200, stop=2000, num=10)], #no of trees
    'max_features': ['auto', 'sqrt'], # the number of features to consider when looking for the best split at each node.
    'max_depth': [int(x) for x in np.linspace(10, 110, num=11)], #depth for each tree
    'min_samples_split': [2,5,8,10,15,20], #the minimum number of samples required to split an internal node.
    'min_samples_leaf' : [1,2,4,6,8,10], #minimum number of samples that must be present in a leaf node.
    'bootstrap': [True, False] #T-> bootstrap samples are used F->entire dataset is used to build tree
}
```

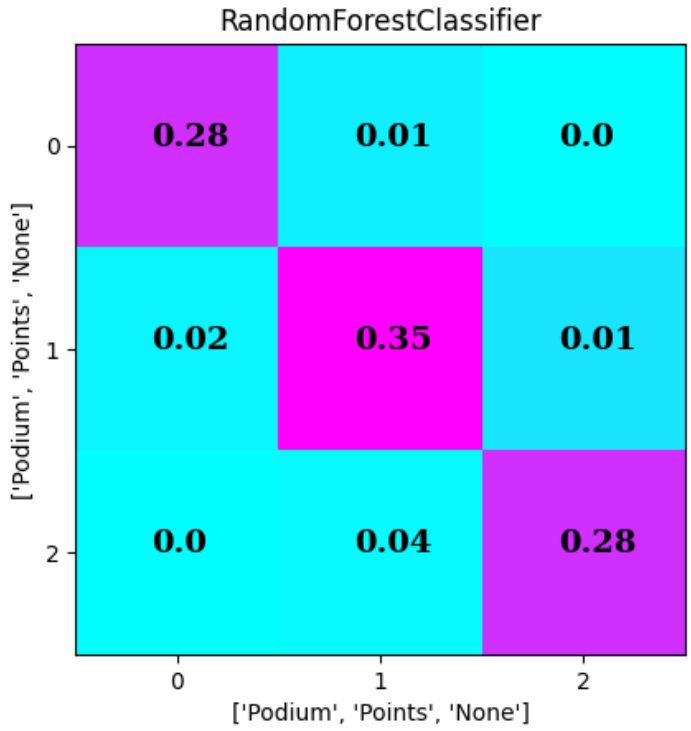
MODEL EVALUATION

A comprehensive examination of the model's performance was carried out using a variety of evaluation standards, including the F1-score, recall, and precision.

RESULT

The results of model performance are as follows.

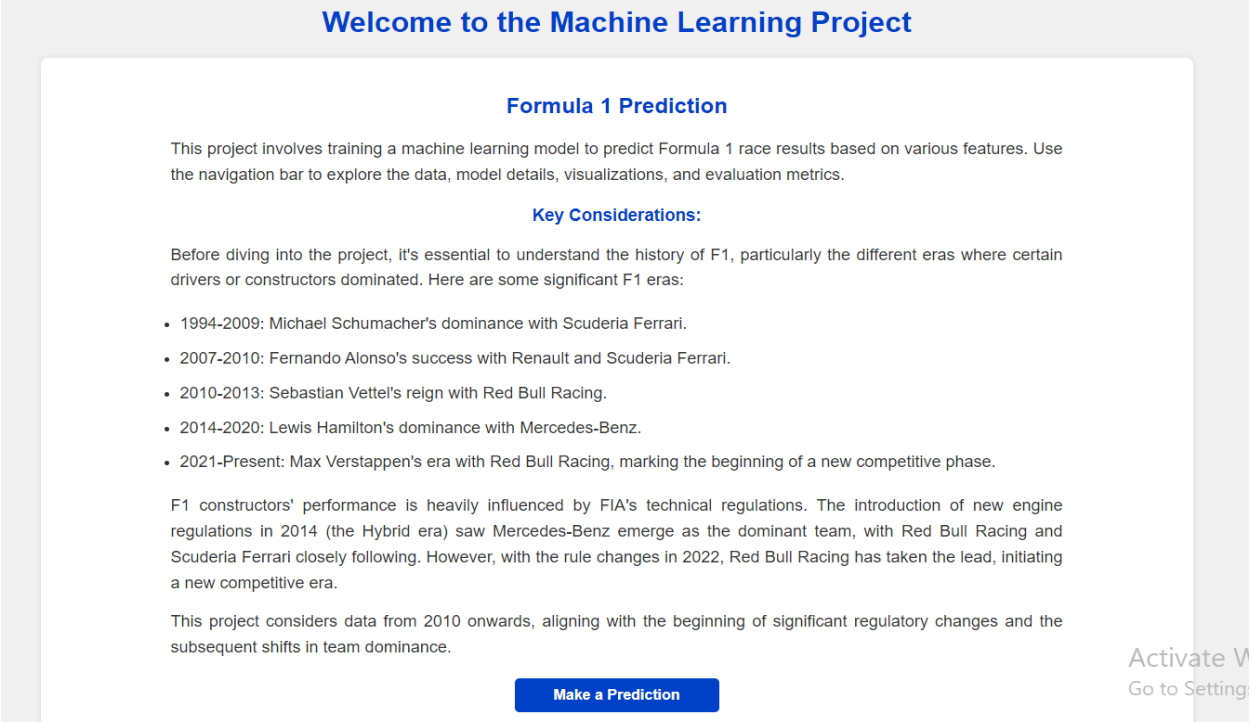
```
rf_pression : 0.9202200238785605
rf_f1 : 0.9218422421327498
rf_recall : 0.9254329004329005
```



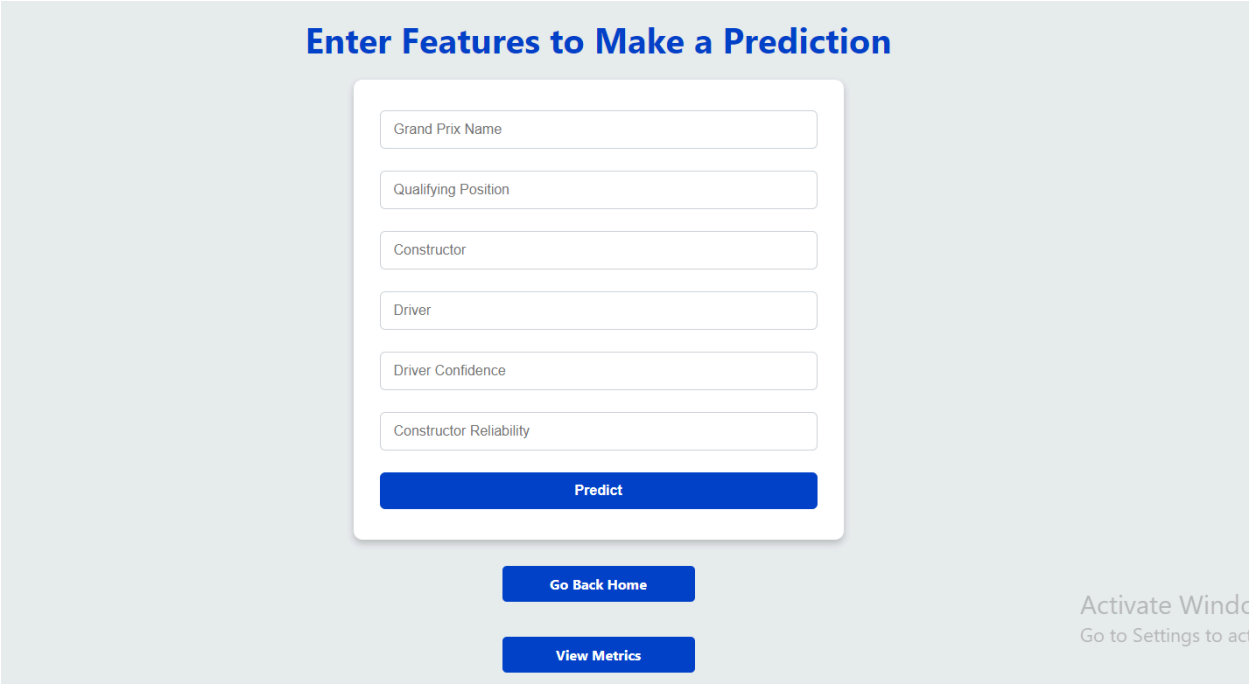
DEPLOYMENT

This project implements Python's Flask framework to create an application of “F1 Prediction” Project.

Home Page:



Prediction Page:



Result Page:

Prediction Result

Grand Prix Name: Bahrain International Circuit

Driver: Lewis Hamilton

Constructor: McLaren

Qualifying Position: 4

Driver Confidence: -4.472636815920398

Constructor Reliability: -39.852130325814535

Predicted Race Position: 2

Class Probabilities:

0.044629065002557894

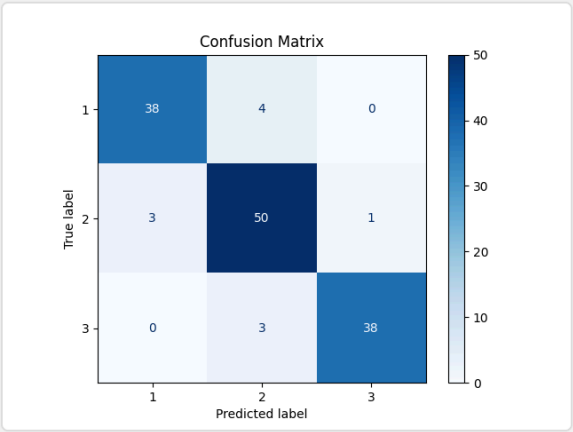
0.8602562274701115

0.09511470752733159

Make Another Prediction

Performance Matrix

Model Performance Metrics



Precision: 0.9261

Recall: 0.9192

F1-Score: 0.9222

Back to Home

FUTURE WORK

A deeper analysis of the dataset can be carried out to evaluate the performance of the model.

CONCLUSION

This experiment covers a detailed analysis of the different models on the F1 data set. This analysis can be very useful for placing bets and analyzing performance of different teams.