CAPSTONE PROJECT BUSINESS REPORT

Presented by- PUSHPENDRA TAWAR

BATCH- JUNE-C-21

**Problem Statement**

BCCI has contracted an external analytics consulting firm to assist with its data analytics. In this partnership, strategic changes are made to make India win by extracting actionable insights from historical match data. Objective is to build Machine Learning models that correctly predict wins for the Indian Cricket Team. The next step is to extract actionable insights and recommendations from the model.

India will also play the following 10 matches in the next 10 days. It is important to predict the outcome of the matches, and if you get a loss, suggest some changes, and re-run your model until you get a win. The same strategy cannot be used throughout the series, as opponent will become accustomed to it and come up with their own counter strategy. As such, for all the below 5 matches, you must suggest unique strategies to help India win. There should be suggestions that correspond with the variables in the dataset. Be sure to carefully consider whether these suggestions could be implemented. Total no. of matches will be 5, out of that 1 test match with England in England, it would be day match in rainy season,2 T20 and 2 ODI matches with Australia and Shri Lanka in India, all these matches will be day and night matches. It will be winter season at the time of match.

**Case Study Moto**

Main aim is to create Machine Learning models which correctly predicts a win for the Indian Cricket Team. Developing a model to extract and provide actionable insights and recommendation.

**Aim of Study:**

Cricket is the second most popular sports in the world. There are several different cricket format matches are organized every year this purpose and BCCI is one of the riches and organization among all cricket board. There are several series of Day, T20 and Test matches held every year around the globe. This proposed paper is specifically concentrating on enactment and measuring the difference between the models to foretell the captivating team of cricket matches. Data is accessed by the computer programs developed using Machine learning to build models. As of now, data analysis is need for each and every field to examine the sets of data to extract the useful information from it and to draw conclusion and as well make decisions according to the information. The algorithm first analyses the data to create a model, specifically for understanding the patterns in the data. For this case study we are using classification method like Decision Tree, Naive Bayes, K-Nearest Neighbour and Random Forest to predict the win of Indian cricket team so board can strategically analyse and plan according to the given insights.

**Data report:**

Total 2929 matches result for all three format (T20, ODI,Test) have taken. We have total 21 variable present in the dataset in which 3 datatypes are given float64(9), int64(4), object (10). We have ‘Result’ as target variable. Noise is present in the dataset, so we have do some data pre-processing, check for data imbalance present in the target variable dataset shown in Fig1.

**METHODOLOGY**

The proposed method consists of five sub modules, namely, loading the dataset, pre-processing and EDA (univariate and Bivariate).

Dataset head

Graphical user interface

Description automatically generated

**Fig.1**

Data info Missing Value

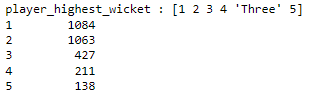
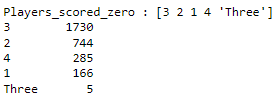
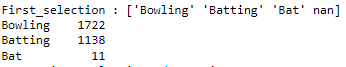
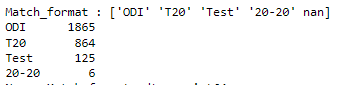
**Fig.2 Fig.3**

**A picture containing graphical user interface

Description automatically generatedText

Description automatically generatedTable

Description automatically generated Fig.4**



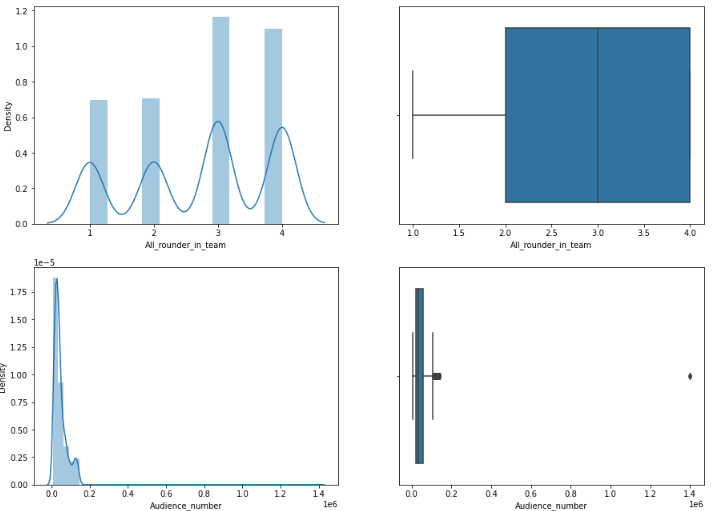
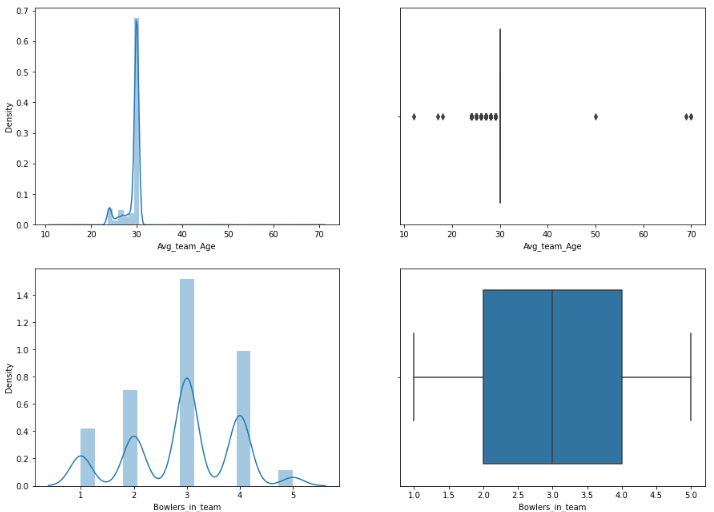
**Fig.5**

* From above data info we can clearly see that there are some missing value present in both numerical and categorical variable that we will be imputing accordingly.
* From descriptive data we can say that 2457 times india has win the matches,we can see that avg\_team\_age minimum age is 12 and max age is 70 that we have to resolve as it can effect to our model.
* Here we are dropping Game\_number and wicket\_keeper\_in\_team as they are not showing any useful insight.
* Also we will be treating all noise present in ‘match\_format,’first\_selection’,’player\_scored\_zero’ and ‘player\_higest\_wicket’ variable.
* After pre-processing we now do EDA (univariate and bivariate analysis)

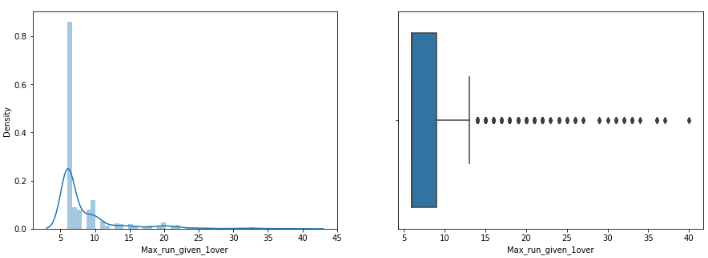
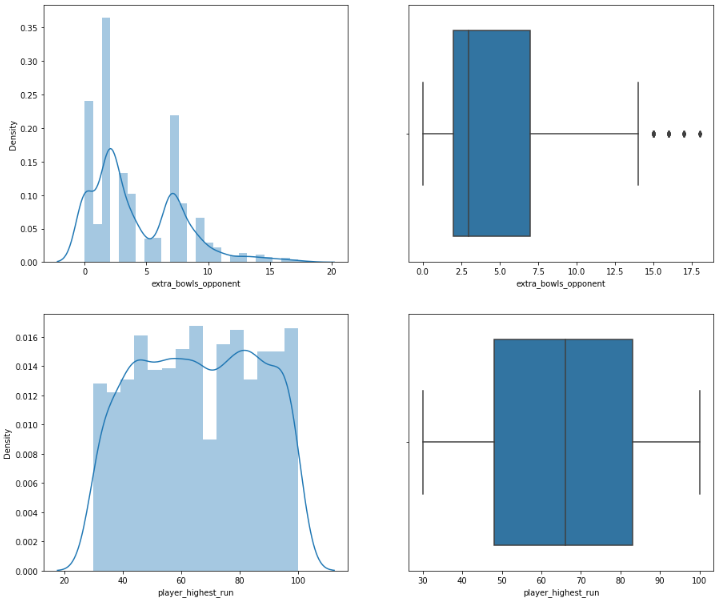
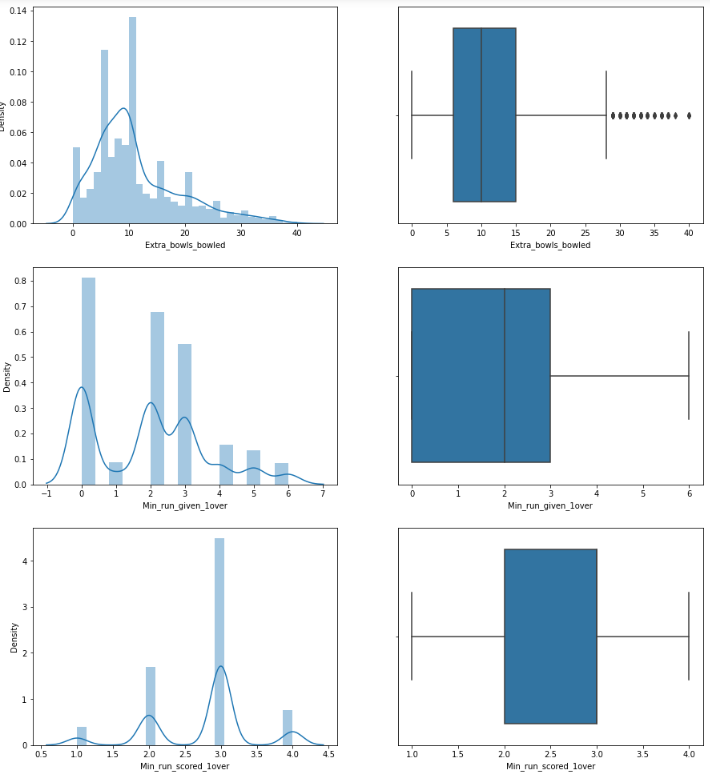
**EDA:**

Univariate analysis

**Numeric variables Histogram**



**Fig.6**

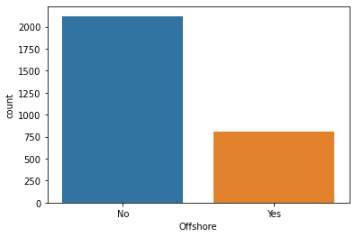
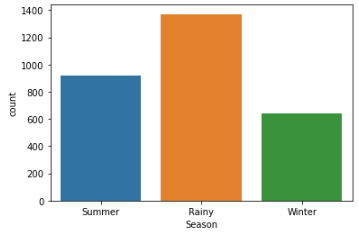
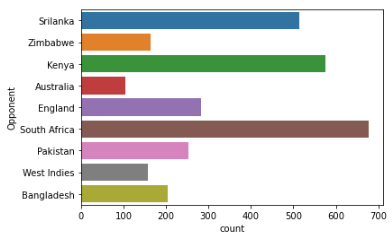
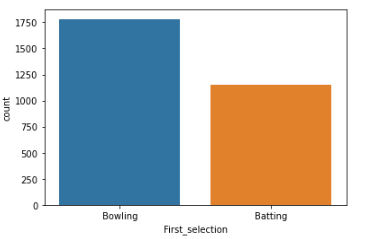
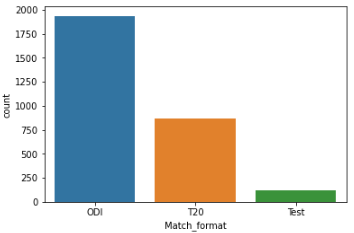
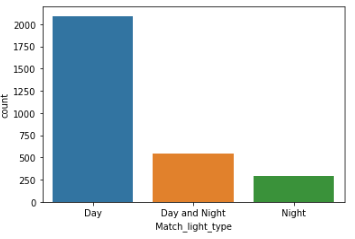
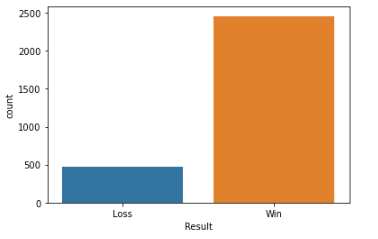


**Fig.6**

**Insight**

* From above histogram we can say that variable is not normally distributed.
* We can see that some outliers present in **‘max\_run\_given\_1over’,’ extra\_bowls\_opponet’, Extra\_bowls\_bowled’,’ Audience\_number’ and ‘Avg\_team\_Age’**.
* Her I am going to treating outlier only for **‘Avg\_team\_Age’** variable as we can see some error entry in the variable.
* For other we are assuming the outlier as a real datapoint.

**Univariate analysis for Categorical variables**

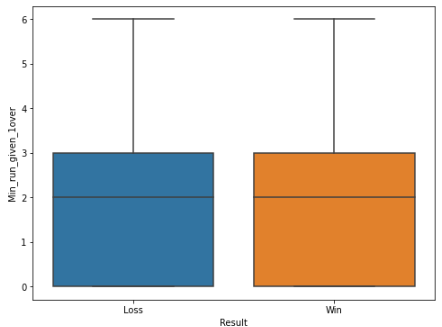
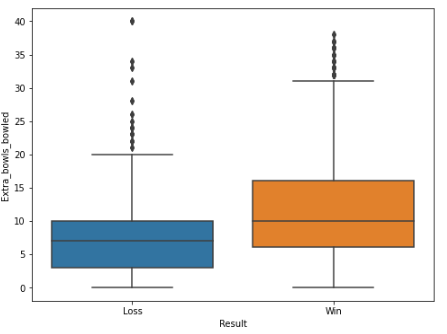
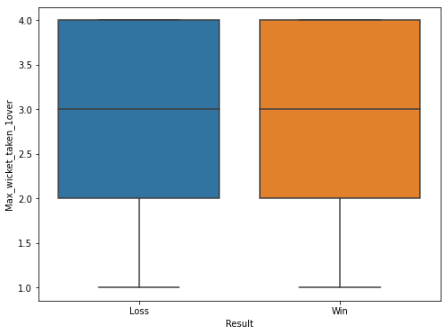
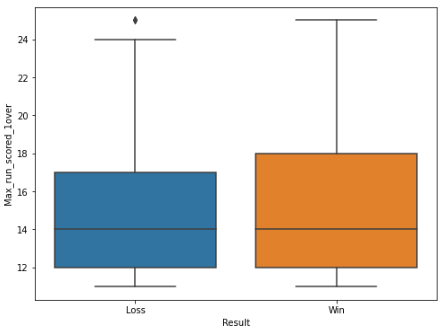
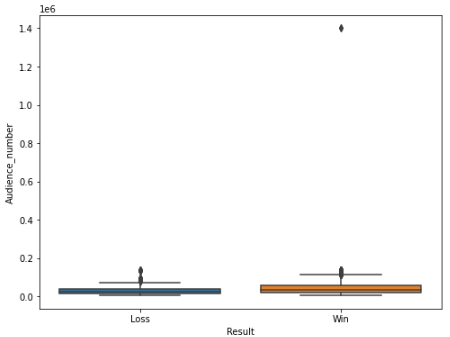
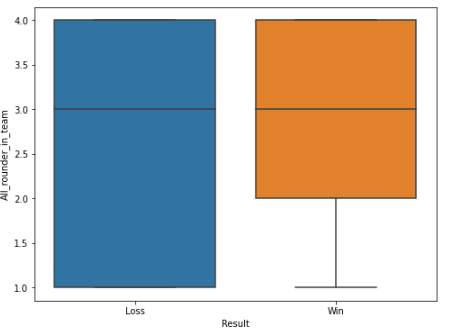
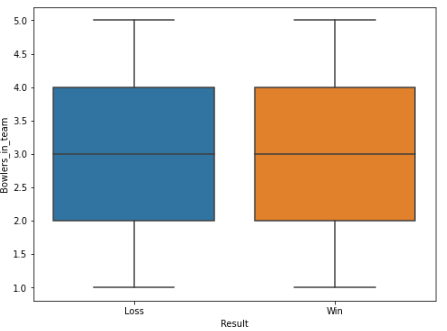
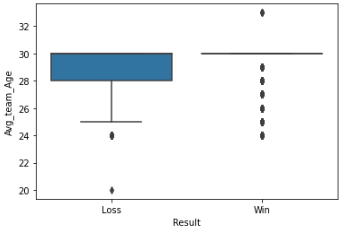
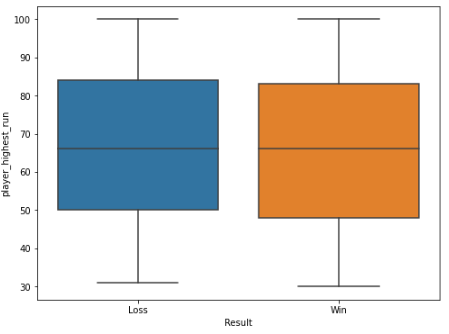
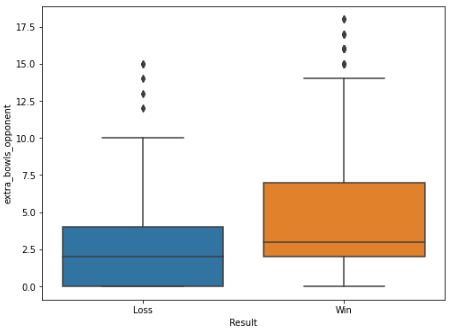
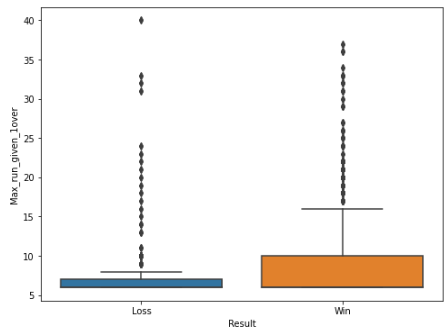
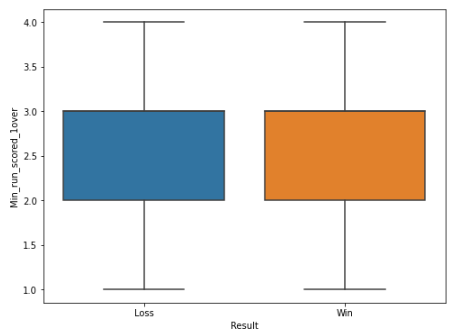


**Fig.7**

**Insight: -**

* Around 72% of matches are played in India and only 28% are played out off India.
* Most of the matches are played in Rainy Season.
* Majority of the matches are played against South Africa (676).
* Around 71% ODI Matches are played in Day light.
* Team have win around 83% of the matches.
* 60% of the time team gets chance to bat first.

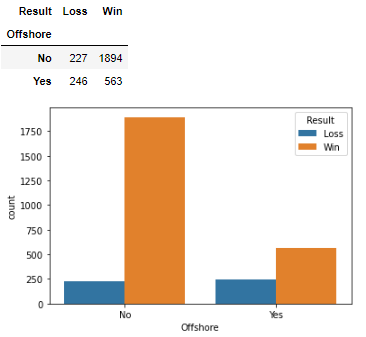
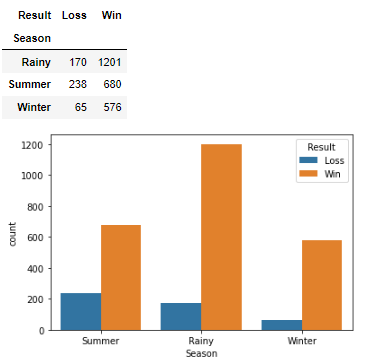
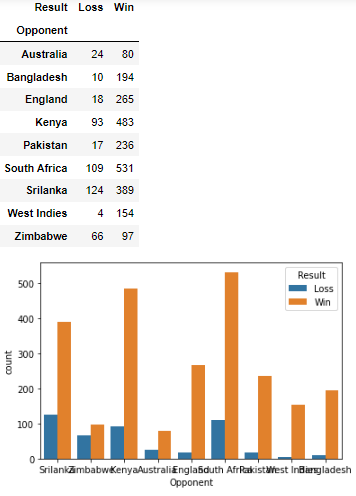
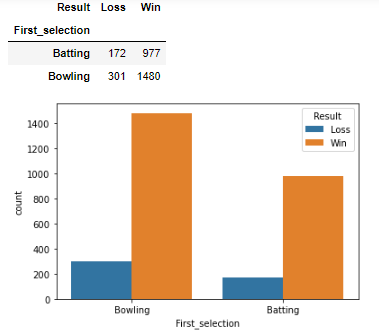
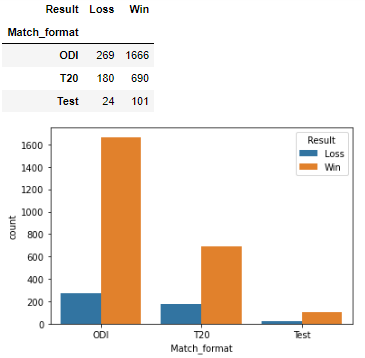
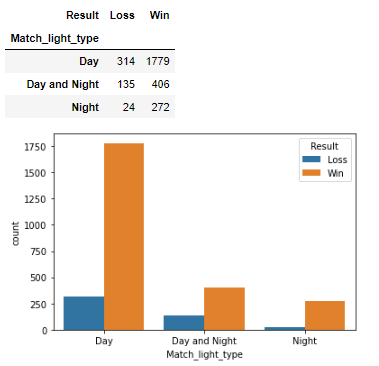
**Bivariate Analysis:** For numerical variable



**Fig.8**

**Bivariate Analysis:**

For Categorical variable



**Fig.9**

**Bivariate insight:**

* Most of the matches are played against south Africa in which 531 times India secure to win.
* 1781-time team have opt to bowl first out of which 1480 have win the match.
* On average, 19% of the time when playing outside the country, and 65% when playing within the country, team manage to win.
* Extra\_bowl\_opponent has help team to win the matches.
* Inexperienced team (young player) has the higher chances to lose the match.
* We can see data is imbalance we have to do smote to resolve this issue.
* During the bowling contest, the team won 51% Matches while Batting 33%.
* Team has won 57% of ODI matches and 24% of T20 matches.

**Encoding the data**

* As many machine learning models cannot work with string values we will encode the categorical variables and convert their datatypes to integer type.
* For variable like ‘Result’,’Offshore’ and ‘First\_selection’ I have used simple categorical conversion technique (pd.categorical). This will convert the values into 0 and 1. As there is no level or order in the subcategory any encoding will give the same result.
* For remaining variable we have used dummies encoding technique.
* We can see the value count of this variable after encoding as below

A picture containing diagram

Description automatically generated A picture containing text

Description automatically generated

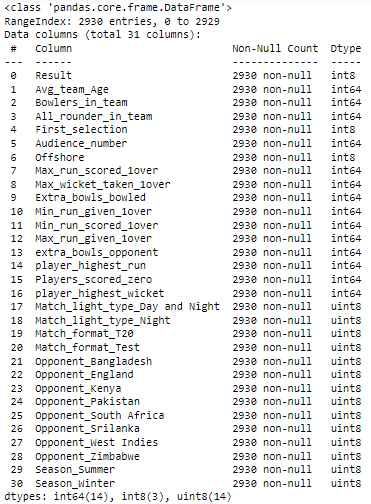
A picture containing logo

Description automatically generated

**Info and head of dataset after encoding-**

A picture containing table

Description automatically generated



**Fig.2**

**Splitting the data into test and train**

* Our target variable is ‘Result. As we can see that there is a data imbalance in the variable.
* So, here I will be using the oversampling technique (ie. SMOTE) and check whether it improves our model performance or not.
* We have stored all the predictor in x and target variable in y.

**Let’s look at the shape of the data after splitting our dataset in (70:30).**

**Text

Description automatically generated**

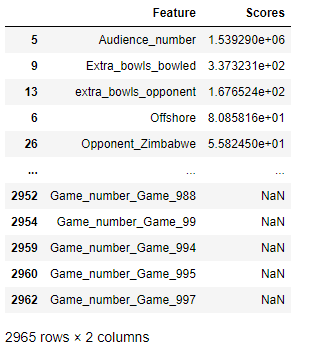
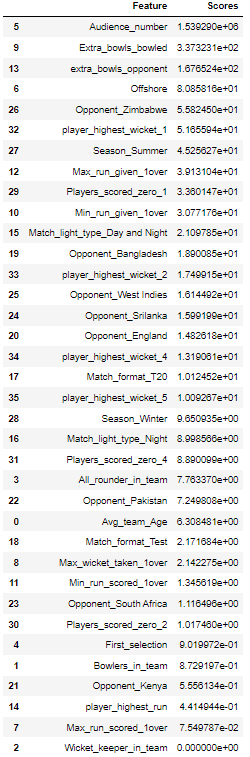
**Fig.3**

* X train - denotes 70% training dataset (except the target column called "Result").
* X test- denotes 30% test dataset (except the target column called "Result").
* y train- denotes the 70% training dataset with only the target column called "Result".
* y test- denotes 30% test dataset with only the target column called "Result".

**Feature Selection**

* Chi-square test is used to determine the relationship between the predictor and target variable.
* In Feature selection, we aim to select the features which are highly dependent on the target variable.
* Higher the chi-square value indicate that the feature is more dependent on the target variable and can be select for model training.
* Chi-square score for Game number is null. So, we eliminate non-significant variable Game number.
* After second iteration we find Wicket keeper as non-significant variable as per chi-square test and same we can in the heat map. So, both the variable have been eliminated to train our model with remaining predictor.

|  |
| --- |
| **Variables** |
| **Result** |
| Avg\_team\_Age |
| Match\_light\_type |
| Match\_format |
| Bowlers\_in\_team |
| All\_rounder\_in\_team |
| First\_selection |
| Opponent |
| Season |
| Audience\_number |
| Offshore |
| Max\_run\_scored\_1over |
| Max\_wicket\_taken\_1over |
| Extra\_bowls\_bowled |
| Min\_run\_given\_1over |
| Min\_run\_scored\_1over |
| Max\_run\_given\_1over |
| extra\_bowls\_opponent |
| player\_highest\_run |
| Players\_scored\_zero |
| player\_highest\_wicket |

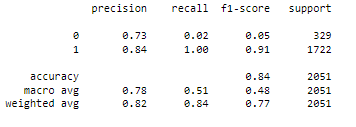
 

**Model Building**

* For this classification problem I am going to build following model
* **Logistic Regression**
* **Linear Discriminant Analysis (LDA)**
* **KNN**
* **Decision Tree (CART)**
* **Random Forest**
* **Naïve Bayes**
* **ANN**

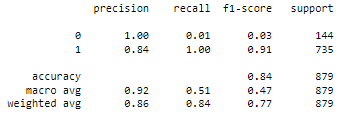
**Logistic Regression**

Performance matrix on training data



**Fig.4**

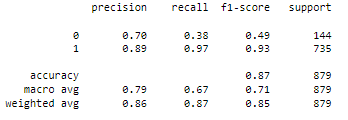
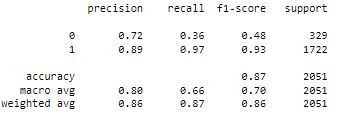
Performance matrix on testing data



**Fig.5**

* We can observe very low recall and F1 score for the zero class.
* For the Training set we got Accuracy= 84 %, Precision= 78 %, recall= 51 %, f1-score= 48 %.
* For the Testing set we got Accuracy= 84 %, Precision= 92 %, recall= 51 %, f1-score= 47 %.

**Linear Discriminate Analysis**



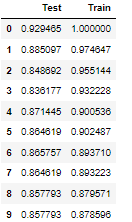
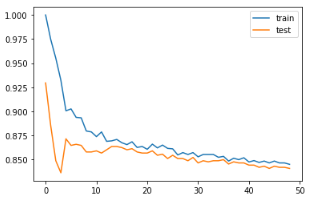
**Fig.6**

* LDA is performing well as compared to Logistic model.
* For the Training set we got Accuracy= 87 %, Precision= 80 %, recall= 66 %, f1-score= 70 %.
* For the Testing set we got Accuracy= 87 %, Precision= 79 %, recall= 67 %, f1-score= 71 %.

**K-Nearest Neighbors Model**

* KNN is a distance based supervised machine learning algorithm that can be use to solve both classification and regression problems. It becomes very slow when we deal with large amount of data.
* For this classifier scaling data is necessary. KNN is a distance base algorithm, so we have scaled our data.
* Here, I have use z-score for scaling our data.

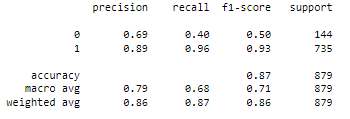
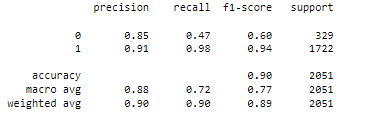
Let’s look at the variance if we increase the numbers of neighbors in train and test both



**Fig.7**

* From above table we can see that after 5th neighbors it’s not changing that much. Variance is not changing that significantly for both Train and Test. So, our model will perform well if we build our model with 5 number of neighbors.

Performance matrix on Scaled Training and Testing dataset



Table

Description automatically generated

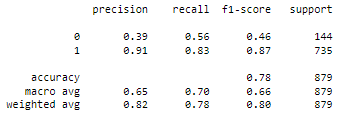
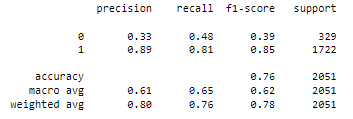
**Fig.8**

* KNN model perform well.
* For the Training set we got Accuracy= 87 %, Precision= 80 %, recall= 66 %, f1-score= 70 %.
* For the Testing set we got Accuracy= 87 %, Precision= 79 %, recall= 67 %, f1-score= 71 %.

**Naïve Baye’s Model**

* Navie Baye’s classifiers is a model based on applying Baye’s theorem with strong independent assumption between the features.
* Here the method that we are going to use is the GaussianNB() method, also know as BernoulliNB().A general assumption in this method is the data is following a normal distribution.

Performance matrix on Training and Testing dataset

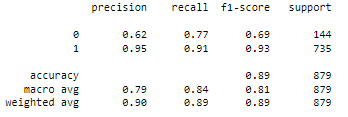
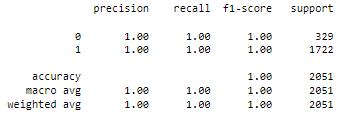


**Fig.9**

* Naïve Bayes’ model performs well.
* For the Training set we got Accuracy= 76 %, Precision= 61 %, recall= 65 %, f1-score= 76 %.
* For the Testing set we got Accuracy= 78 %, Precision= 65 %, recall= 70 %, f1-score= 78 %.
* Surprisingly our recall and precision have increased in the test set.

**CART Model**

* Cart is a classification and regression predictive model machine learning technique.



**Fig.10**

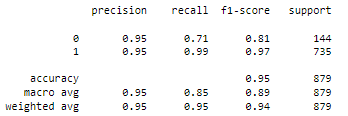
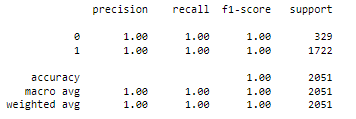
* We can clearly see that our train set is over fitted. We can solve this issue by using pruning technique or by selecting important variable.
* For the Training set we got Accuracy= 100 %, Precision= 100 %, recall= 100 %, f1-score= 100 %.
* For the Testing set we got Accuracy= 89 %, Precision= 79 %, recall= 84 %, f1-score= 81 %.
* Table

  Description automatically generated with medium confidenceImportant features are mentioned below.

**Random Forest Model**

* Random Forest an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time.

Performance matrix on Training and Testing dataset



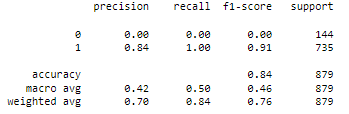
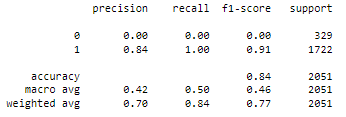
**Fig.11**

* We can clearly see that our train set is over fitted. We can solve this issue by using pruning technique or by selecting important variable that I will do in further model tunning part.
* For the Training set we got Accuracy= 100 %, Precision= 100 %, recall= 100 %, f1-score= 100 %.
* For the Testing set we got Accuracy= 95 %, Precision= 95 %, recall= 85 %, f1-score= 89 %.

**ANN Model**

* Artificial neural networks (ANNs) use learning algorithms that can independently adjust or learn, in a sense as they have receive new input.

Performance matrix on Training and Testing dataset



* ANN is performing very poor as compared to another model.
* For the Training set we got Accuracy= 84 %, Precision= 42 %, recall= 50 %, f1-score= 46 %.
* For the Testing set we got Accuracy= 84 %, Precision= 42 %, recall= 50 %, f1-score= 46 %.

**Model Tunning and Ensemble Method**

* Tuning is the process of maximizing a model's performance without overfitting or creating too high of a variance. In machine learning, this is accomplished by selecting appropriate "hyper-parameters"
* Grid Search is one of the most common methods of optimizing the parameters
* Models such as Bagging, Boosting, Gradient boosting, etc are prone to under or over fitting of data. Overfit means the model work well in train data but work relatively poor in test dataset. Underfit is vice-versa of overfitting model.

**Apply Logistic Regression with hyper parameters**

* Before fitting the model, it is important to know about the hyper parameters that is involved in model building.
* Penalty
* Solver
* Max\_iter
* tol, etc.
* To find the best combination among these parameters we will use the "GridSearchCV" method. This method can perform multiple combinations of these parameters simultaneously and can provide us with the best optimum results.

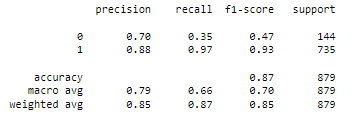
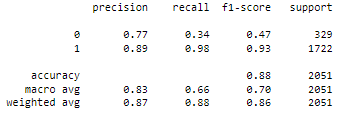
After performing the search, the best parameters came out to be-

**A picture containing graphical user interface

Description automatically generatedText

Description automatically generated with low confidence**

Performance matrix on training and Test data with tunning



**Fig.12**

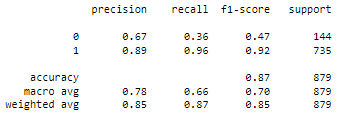
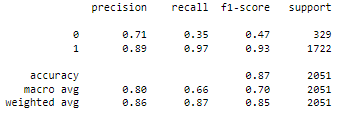
* The model performs well for both train and test set with hyper parameter as compared to without tunning.
* For the Training set we got Accuracy= 88 %, Precision= 83 %, recall= 66 %, f1-score= 70 %.
* For the Testing set we got Accuracy= 87 %, Precision= 79 %, recall= 66 %, f1-score= 46 %.

**Apply LDA with hyper parameters**

* Before fitting the model, it is important to know about the hyper parameters that is involved in LDA model building.
* Solver = ‘auto’, Shrinkage = ‘lsqr’

A picture containing graphical user interface

Description automatically generated



**Fig.13**

* After turning our LDA model we can see that there is no change in the performance of the model.
* For the Training set we got Accuracy= 87 %, Precision= 80 %, recall= 66 %, f1-score= 70 %.
* For the Testing set we got Accuracy= 87 %, Precision= 78 %, recall= 66 %, f1-score= 70 %.

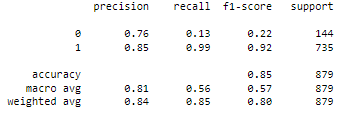
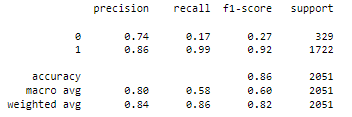
**Apply CART with hyper parameters**

* Before fitting the model, it is important to know about the hyper parameters that is involved in CART model building.
* criterion = ‘gini, max\_depth = ‘3’

**A picture containing website

Description automatically generated**

Performance matrix on training and Test data with tunning



**Fig.14**

* After tunning the CART model, we succeed to come up with overfitting issue.
* For the Training set we got Accuracy= 86 %, Precision= 80 %, recall= 58 %, f1-score= 60 %.
* For the Testing set we got Accuracy= 85 %, Precision= 81 %, recall= 56 %, f1-score= 57 %.

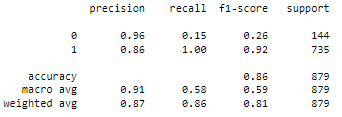
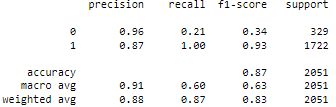
**Apply Random Forest with hyper parameters**

* Before fitting the model, it is important to know about the hyper parameters that is involved in Random Forest model building shown as below.

**Graphical user interface, text

Description automatically generated**

Performance matrix on training and Test data with tunning

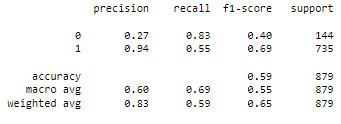
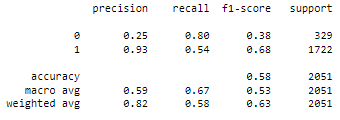


**Fig.15**

* After tunning the Random Forest model, we succeed to come up with overfitting issue.
* For the Training set we got Accuracy= 87 %, Precision= 91 %, recall= 60 %, f1-score= 63 %.
* For the Testing set we got Accuracy= 86 %, Precision= 91 %, recall= 58 %, f1-score= 59 %.

**Naïve Bayes’ on scaled data**

Performance matrix on training and Test data with tunning



**Fig.16**

* After scaling the data our NB model start performing poor.
* For the Training set we got Accuracy= 58 %, Precision= 59 %, recall= 67 %, f1-score= 53 %.
* For the Testing set we got Accuracy= 59 %, Precision= 60 %, recall= 69 %, f1-score= 55 %.

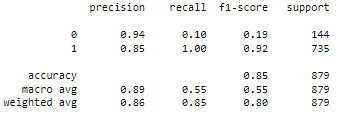
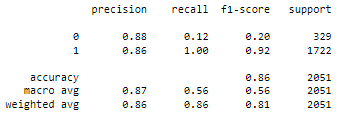
**Bagging Model (Using Random Forest Classifier)**

* Bagging is an ensemble technique. Ensemble techniques are the machine learning techniques that combine several base models to get an optimal model.
* Bagging is designed to improve the performance of existing machine learning algorithms used in statistical classification or regression.
* Here, we will use random forest as the base classifier. We use Hyper-parameters that we have obtain from Grid Search.

**Text

Description automatically generated**

Performance matrix on Training and Testing dataset using Bagging Technique



**Fig.17**

* Bagging performs good but not good as compared to simple random forest model.
* For the Training set we got Accuracy= 86 %, Precision= 87 %, recall= 56 %, f1-score= 56 %.
* For the Testing set we got Accuracy= 85 %, Precision= 89 %, recall= 55 %, f1-score= 55 %.

**Boosting Model**

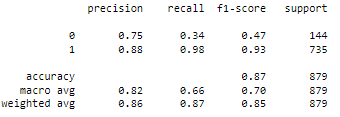
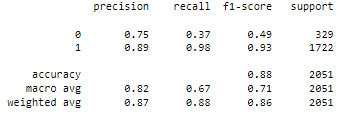
* Boosting is also an ensemble technique. It converts weak learners to strong learners.
* Each time base learning algorithm is applied, it generates a new weak learner prediction rule. This is an iterative process, and the boosting algorithm combines these weak rules into a single strong prediction rule.
* There are many types of Boosting techniques. We are going to use following technique for this case study.

1. ADA Boosting.
2. Gradient Boosting.

**ADA Boosting Model**

* This model is used to increase the efficiency of binary classifiers, but now used to improve multiclass classifiers as well.

Performance matrix on Training and Testing dataset



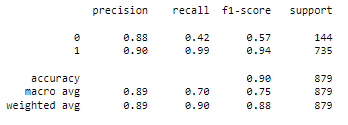
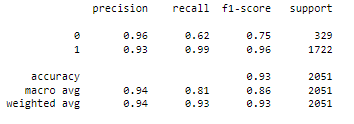
**Fig.18**

* ADA Boosting is performing good as compare to other model.
* For the Training set we got Accuracy= 88 %, Precision= 82 %, recall= 67 %, f1-score= 71 %.
* For the Testing set we got Accuracy= 87 %, Precision= 82 %, recall= 66 %, f1-score= 70 %.

**Gradient Boosting Model**

* This model is just like the ADABoosting model. Gradient Boosting works by sequentially adding the misidentified predictors and under-fitted predictions to the ensemble, ensuring the errors identified previously are corrected.
* This method tries to fit the new predictor to the residual errors made by the previous one.

Performance matrix on Training and Testing dataset



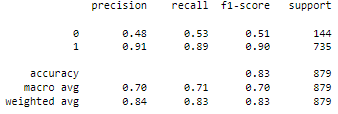
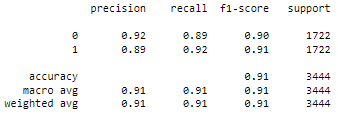
**Fig.19**

* Gradient boosting model is performing good for this classification problem.
* For the Training set we got Accuracy= 93 %, Precision= 94 %, recall= 81 %, f1-score= 86 %.
* For the Testing set we got Accuracy= 90 %, Precision= 89 %, recall= 70 %, f1-score= 75 %.

**SMOTE**

* SMOTE (Synthetic Minority over sampling Technique) is used when we encounter with data imbalance problem.
* We know that we were having data imbalance in our target variable, so we can also

Performance matrix on Training and Testing dataset



**Fig.20**

* From above table using SMOTE technique doesn’t increase the performance of the model.
* Smote technique performing good on training set but underfitting in the test set it can bee eliminated by dropping non important variable.
* For the Training set we got Accuracy= 91 %, Precision= 91 %, recall= 91 %, f1-score= 91 %.
* For the Testing set we got Accuracy= 83 %, Precision= 70 %, recall= 71 %, f1-score= 70 %.

**Compare all Model**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **LR-TRAIN** | **LR-TEST** | **LR(Tune)-TRAIN** | **LR(Tune)-TEST** | **LDA-Train** | **LDA-Test** | **LDA(Tune)-Train** | **LDA(Tune)-Test** | **KNN-Train** | **KNN-Test** | **NB-Train** | **NB-Test** |
| **Accuracy** | 84 | 84 | 88 | 87 | 87 | 87 | 87 | 87 | 90 | 87 | 76 | 78 |
| **F1 Score** | 48 | 47 | 70 | 70 | 70 | 71 | 70 | 70 | 77 | 71 | 62 | 66 |
| **Recall** | 51 | 51 | 66 | 66 | 66 | 67 | 66 | 66 | 72 | 68 | 65 | 70 |
| **Precision** | 78 | 92 | 83 | 79 | 80 | 79 | 80 | 78 | 88 | 79 | 61 | 65 |
| **AUC** | 73 | 73 | 83 | 82 | 82 | 83 | 82 | 83 | 95 | 83 | 76 | 78 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **CART-TR** | **CART-TEST** | **RF-TRAIN** | **RF-TEST** | **ANN-rain** | **ANN-Test** | **Bagging-Train** | **Bagging-Test** | **Ada-train** | **Ada-Test** | **GB-Train** | **GB-Test** | **Smote-rf** | **Smote-rf-test** |
| **Accuracy** | 86 | 85 | 87 | 86 | 84 | 84 | 88 | 87 | 88 | 87 | 93 | 90 | 91 | 83 |
| **F1 Score** | 60 | 57 | 63 | 59 | 46 | 46 | 70 | 70 | 71 | 70 | 85 | 76 | 91 | 70 |
| **Recall** | 58 | 56 | 60 | 58 | 50 | 50 | 66 | 66 | 67 | 66 | 80 | 70 | 91 | 71 |
| **Precision** | 80 | 81 | 91 | 91 | 42 | 42 | 83 | 79 | 82 | 82 | 94 | 89 | 91 | 70 |
| **AUC** | 83 | 82 | 92 | 89 | 76 | 75 | 90 | 87 | 87 | 85 | 95 | 89 | 96 | 83 |

* In this classification problem the most important measurement matrix we see is Recall, precision, accuracy, and F1-Score.
* In this case, precision is the total predicted win and loss. Recall is total Actually win and loss.
* F1- score is the harmonic mean of precision and recall.
* In this case our most important matrix is Recall because we must predict winning for the Indian team and must reduce the false positive rate.
* Comparing all models, I am going with ‘**Gradient Boosting Model’** for this Case study.
* Gradient Boost Model have less False ‘+ve’ and False ‘-ve’ for both win and loss Classes. Compare to other model it has Higher Precision, Recall and Accuracy for both Train and Test.

**Recommendation**

* Try to collect more some more predictor, like total score, bowling style etc. for better Model.
* Try to add more than 3 all-rounders in the team that will improve the team performance.
* If team opt for bowling first with an Avg team age of 30, with 4 bowlers in the team has higher chance to win against England in test match in Rainy season in England
* If team opt for bowling first with an Avg team age of 30, minimum 3 bowlers in the team, scoring average 15 runs per over has higher chance to win against Australia in T20 match in Winter season in India.
* If team opt for Batting first with an Avg team age of 30, with 3 bowlers in the team and at least one player should score century has higher chance to win against Sri Lanka in ODI match in Winter season in India.

END