WINNER PREDICTION USING MACHINE LEARNING MODELS

Project submitted to the University of Kerala in partial fulfilment of the requirement for the degree of

MASTER OF SCIENCE

IN

STATISTICS

Exam code: 62521401

Course code: ST 546

July 2023

Contents

Chapter 1

INTRODUCTION

Cricket is being played in many countries around the world. There are a lot of domestic and international cricket tournaments being held in many countries. The cricket game has various forms such as Test Matches, Twenty20(T20) Internationals, etc. Indian Premier League(IPL) is also one of them and has great popularity among them. It's a T20 cricket game league played to inspire young and talented players in India. The league was conducted annually in March, April, or May and has a huge fan base in India. There are eight teams that represent eight cities that are chosen from an auction. These teams compete against each other for the trophy.

On the other hand, Cricket is known as the game of chance. Fans and followers are also concerned with predicting the winner of a tournament or match. The outcome of the game is determined by the team's luck, the performance of the players, and the variety of other factor consideration. The match that takes place before the day will also affect the prediction. Because of the huge popularity and presence of the company, the stakeholders benefit significantly more. The size of the data we use for analysis determine the accuracy of the data as well as the records that are kept in order to predict the outcome.

Many natural factors affecting the sport, enormous media scope, and a huge betting market have given strong reason to model and train the sport from various aspects. Although the complex and different rules determine the sport, the power

of players and their performances on a given day, and many other parameters play an important role in influencing the best result of a match. Because the technology is improving at a faster speed and also the large market in betting and huge demand for cricket has influenced the common people to use Machine learning (ML) calculations to predict the results of cricket matches. Use of machine learning and data science makes life easier in every aspect, using ML and predicting the outcomes before the match will help the players and coaches to analyze the weak areas. We adopt several ML concepts and algorithms so as to predict the winner of the cricket match. And machine learning is booming and firmly identified with (and frequently covers with) computational insights, which also focuses on prediction-making through the utilization of technology. Its solid connections to numerical improvement, which conveys strategies, hypothesis and application areas to the sector. ML is a few of the time conflated with data processing where the latter subfield concentrates more on exploratory information analysis and is understood as supervised learning.

With the progression in innovation and additionally in sports, predicting the results of a match has evolved to be so basic. Cricket is one among the most popular and most watched team games in the world. We are predicting the results of a T20 cricket matches using ML concepts like supervised learning to predict the champions of the matches. We utilize career statistics and also the team performances like batting and bowling performances so as to train the models. However, the unpredictable rules governing the sport, the capacity of players and different parameters play a necessary part in influencing the results of the match. Therefore we are using supervised learning algorithms to predict the end result of the sport and it will help the coaches of the team to understand and analyze where actually the team is going wrong.

1.1 Objective of the study

Predicting a winner in a sports such as cricket is especially challenging and involves very complex processes. But with the introduction of machine learning, this can be made much easier and simpler. The first objective of this study is to predict the match winner of IPL using historical data of IPL from season 2008 to 2022. For a strategic game, players need to predict the next step or the next few steps of the maximum probability of action. Then players decide how to deal with so that they have the greatest chance to win. Hence another objective of the study is to improve player accuracy and performance and create better strategies for the remaining game. In statistical analysis model fitting is a measure of how well a machine learning model generalizes to similar data to that on which it was trained. A model that is well-fitted produces more accurate outcomes. Hence in the last objective of the study is to fit models for winner prediction and select which one is best for the future prediction with good accuracy.

1.2 Organisation of project

This project is organized into six chapters. Besides this introductory section, in Chapter 2 we describe the existing system related to this project. Chapter 3 gives a detailed discussion on the system architecture and working of each module. In Chapter 4, we discuss the brief description on the dataset used and its source. Chapter 5 provide the data pre-processing and model building. Finally, some concluding remarks are made in Chapter 6.

Chapter 2

REVIEW OF LITERATURE

In order to get required knowledge about various concepts related to the present application, existing recent literature where studied. Sinha (2020) suggested a model for predicting the result of the match supported historic data. During the extraction of features various features has been involved but most vital features has been taken during prediction. The author also made a team structure in terms of slots which defines most vital slots contributing to match winning and a ranking system for the players through their performance statistics. He used K-means to cluster all players consistent with their performance and K-nearest neighbor (KNN) is employed to seek out interchangeable player to a specific player. Support Vector Machine (SVM) model was trained using linear, polynomial and Radial Basis function (RBF). Thus, he preferred SVM with RBF Kernel for prediction. Lokhande (2018) consider problem of interactive models of predictions where a user predicts the results of each game in order to be rewarded which would further help him strengthen his Fantasy squad. His aim is not only to attract more users to this game that is Fantasy Cricket, but also aims at improving the general attraction to the Premier League. This happens because in a predictive model, a user makes a prediction on every game, and ends up watching that game to check if his prediction is going right. Lamsal and Choudhary (2020) discussed the varied factors that influence the result of an IPL matches were identified. The seven factors which significantly influence the results of an IPL match include

the house team, the away team, the toss winner, toss decision, the stadium, and therefore the respective teams' weight. Hence designing machine learning model for predicting the match outcome of an auction based 2020 format premier league with the accuracy of 72.66% and F1 score of 0.72 is very satisfactory at this stage. Goel(2021) considered 692 matches of all the seasons (2008–2019) to train their model and all the matches of season (2019) to test. In future we can start right from the beginning of the first inning. Although study is done on IPL T20 matches only, however similar approach could be applied to predict outcome in other versions of Cricket matches as 336 well i.e., test cricket and ODI matches. Vistro et al. (2019) discussed to predict the match winner of IPL using historical data of IPL from season 2008 to 2017. Sample, Explore, Modify, Model and Assess(SEMMA) methodology has been selected for conducting the analysis of IPL T20 match winner dataset. Preprocessing has been done on the dataset to form it consistent by removing missing value, encoding variables into numerical format. Best features were selected by visualizing attributes of knowledge with target variable. On selected features several machine learning models has been applied on the to predict the winner and therefore the results were outstanding. Decision Tree model was applied which predicted the match winner with good accuracy 76.9%. Srikantaiah et al. (2021) trying to find out the match winner of an IPL match based on the stadium they are choosing and the toss decision using machine learning techniques like SVM, Random Forest, Logistic Regression etc. Dhonge et al. (2021) discussed the model that has two methods the first one is prediction of score and the second one is team winning prediction. In these the score prediction includes linear regression, lasso regression and ridge regression whereas in winning prediction SVC classifier, decision tree classifier and random forest classifier are used. The model used the supervised machine learning algorithm to predict the winning. Random Forest Classifier used for good accuracy and the stable accuracy so that desired predicted output is accurate. Dayalbagh et al. (2016) suggested a predictor model for predicting the outcomes of the matches in IPL being played in April – May, 2016. The model has three components which are based on

multifarious considerations emerging out of a deeper analysis of T20 cricket. The models are created using Data Analytics methods from machine learning domain. Kumar and Roy (2018) discussed a model for predicting the result of the match supported historic data. For predictive model Linear Regression have been used and for classification KNN, SVM and Naive Bayes have been used. Abhishek et al. (2019) they considered various machine learning models were trained and used to perform within the time lapse between the toss and initiation of the match, to predict the winner. The performance of the model developed are evaluated with various classification techniques where Random Forest and Decision Tree have given good results.

Chapter 3

METHODOLOGY

In this chapter, we present the details of various machine learning techniques used in the subsequent chapters for predicting the winner of IPL.

3.1 Machine Learning

ML is a data analytic technique that teaches computers to do what comes naturally to human and animals-to learn from experiences. It is a scientific study of algorithms and statistical models that system use to perform a specific task without being explicitly programmed. Machine learning algorithms are used in various fields like data mining, image processing, predictive analytics etc.

Since the evolution of the humans have been using many types of tools to perform tasks in a simpler way. This led to the invention of different machines which made the human life easy by enabling people to meet various life needs, including travelling, industries and computing. Machine learning is the one among them.

According to Arthur Samuel, ML can be defined as the field of study that gives computers the ability to learn without being explicitly programmed. Arthur Samuel was famous for his checkers playing program. ML is used to teach machines to handle the data more efficiently, it is very useful in situations where we find it difficult to interpret the extract information from the data. The demand for ML is in rise with the abundance of datasets available. Many industries has

been using ML techniques to extract relevant data. The purpose of ML is to learn from the data. Many mathematicians and programmers apply several approaches to find the solution of problem which are having huge data sets. The main focus area of ML can be organized around three primary research areas given below.

• Task oriented studies: The development and analysis of learning systems are oriented towards solving a predetermined set of tasks.

• Cognitive simulation: This is a cognitive modelling approach where focus is investigation and computer simulation of human learning processes.

• Theoretical analysis: The theoretical exploration of the space of possible learning methods is in focus.

ML relies on different algorithms to solve data problems. They mainly consist

• Supervised Learning

• Unsupervised Learning

• Reinforcement Learning

3.1.1 Supervised Learning

Supervised learning is a machine learning technique of learning a function that maps an input to an output based on example input-output pairs. That is, Dataset contains both input and output. Based on this machine learning system build a model, so that when given new observation, it will try to find out corresponding output. It infers a function from labelled training data consisting of a set of training examples.

The supervised learning algorithms are those algorithms which needs external assistance. In simple words we can say that in supervised learning approaches task is performed using a set of human prepared example. The input dataset divided into train and test dataset. The train dataset has output variable which needs to be predicted or classified. Supervised learning mostly leaves the probability for inputs undefined.

Let $X_1, X_2, \ldots, X_n$ be a set of input features and Y be a target function. Let a set of training examples where the values of the input features and the target

features are given for each example and a new example, where only values for the input features are given. In supervised learning, we have to predict the values for the target features for the new examples.

3.1.2 Unsupervised Learning

In unsupervised learning, the computer learns how to do something that we didn't teach how to work out. Models are not supervised using training dataset. Model itself find pattern from given data. Unsupervised learning is a machine learning where manual labels of input are not used.

3.1.3 Reinforcement Learning

The algorithm learns a policy of how to act given an observation of the world. Every action has some impact in the environment, and the environment provide feedback that guides that guides the learning algorithm. Thus it is a feedback based ML technique where agents learn automatically using feedback. Since there is no labelled data, the agent is bound to learn by its experience only. Reinforcement Learning is used in operations research, information theory, game theory, control theory etc.

3.2 Classification and Regression Algorithms

Let $X_1, X_2, \ldots, X_n$ be a set of input features and Y be a target function. When Y is a categorical feature, it is said to be a classification problem and it is a regression problem when Y is a continuous feature.

Classification: In classification problems, test data is assigned into different categories. Classification problems are of two type – binary or multiclass. The target attributes can take only two possible values in binary classification. In multiclass classification, the target attribute can have more than two values. The classification is a classical method which is used for categorization of objects into given discrete number of classes. Some common classification algorithms are decision

trees, random forest, support vector machine, naive bayes, gradient boosting classifier etc.

Regression: Regression tells about the relationship between independent variables or features and a dependent variable. It is a predictive modelling method in machine learning, where an algorithm is used to predict continuous outcomes. Algorithms are trained to understand the relationship between independent and dependent variable. The model can then be leveraged to predict the outcome of new and unseen input data, or to fill a gap in missing data. Some common regression algorithms are linear regression, Poisson regression, gradient booster regression etc.

3.3 Machine Learning Models

The implementation of three machine learning algorithms is shown in this study. They are Naive Bayes, Logistic regression, and Random forest algorithm.

3.3.1 Naive Bayes Classifiers

In statistics, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features. They are among the simplest Bayesian network models, but coupled with kernel density estimation, they can achieve high accuracy levels.

Naive Bayers classifiers are highly scalable, requiring a number of parameters linear in the number of variables (feautures/predictors) in a learning problem. Maximum likelihood training can be done by evaluating a closed-form expession, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers

In the existing literature, naive Bayes models are known under a variety of names, including simple Bayes and independence Bayes. All these names reference the use of Bayes' theorem in the classifier's decision rule, but naive Bayes is not (necessarily) a Bayesian method.

Naive Bayes is a simple technique for constructing classifiers models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle. All naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the colour, roundness, and diameter features.

In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.

Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, an analysis of the Bayesian classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of naive Bayes classifiers. Still, a comprehensive comparison with other classification algorithms in 2006 showed that Bayes classification is outperformed by other approaches, such as boosted trees or random forests.

An advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification.

Probabilistic model

Abstractly, naive Bayes is a conditional probability model that assigns probabilities $p\left(C_k|x_1, x_2, \ldots, x_n\right)$ for each of the k possible outcomes or classes C_k given a problem instance to be classified, represented by a vector x= $(x_1, x_2, \ldots, x_n)$ encoding some n features (independent variables).

The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model

on probability tables is infeasible. The model must therefore be reformulated to make it more tractable. Using Bayes' theorem, the conditional probability can be expressed as:

$$p(C_k|x) = \frac{p\,(C_k)\;\; p\,(x|C_k)}{p\,(x)}. \tag{3.1}$$

In Bayesian probability terminology, (3.1) can be written as

$$posterior = \frac{prior \;\; \times likelihood}{evidence}. \tag{3.2}$$

In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on and the values of the features are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model $p(C_k, x_1, x_2, \ldots, x_n)$ which can be rewritten as follows. Using the chain rule for repeated applications of the definition of conditional probability, we can write

$$\begin{aligned} p(C_k, x_1, x_2, \ldots, x_n) &= p(x_1, x_2, \ldots, x_n, C_k) \\ &= p\,(x_1|x_2, \ldots, x_n, C_k)\, p(x_2, \ldots, x_n, C_k) \\ &= \ldots \\ &= p\,(x_1|x_2, \ldots, x_n, C_k)\;\; p\,(x_2|x_3, \ldots, x_n, C_k) \end{aligned} \tag{3.3}$$

Now the "naive" conditional independence assumptions come into play. Assume that all features in x are mutually independent, conditional on the category C_k under this assumption,

$$p\,(x_i|x_{i+1}, \ldots, x_n, C_k) = p\,(x_i|C_k)\,. \tag{3.4}$$

Thus, the joint model can be expressed as

$$p\,(C_k|x_1, x_2, \ldots, x_n) \propto p(C_k, x_1, x_2, \ldots, x_n) \propto p\,(C_k) \prod_{i=1}^{n} p\,(x_i|C_k). \tag{3.5}$$

This means that under the above independence assumptions, the conditional dis-

tribution over the class variable C is

$$p\left(C_k|x_1, x_2, \ldots, x_n\right) = \frac{1}{Z} p\left(C_k\right) \prod_{i=1}^{n} p\left(x_i|C_k\right). \tag{3.6}$$

where, the evidence

$$Z = p\left(x\right) = \sum_{k} p\left(C_k\right) p\left(x|C_k\right)$$

is a scaling factor dependent only on $(x_1, x_2, \ldots, x_n)$ that is, a constant if the values of the feature variables are known.

Types of Naive Bayes Model

There are three types of Naive Bayes Model, which are given below:

1) Gaussian: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.

2) Multinomial: The Multinomial Naïve Bayes classifier is used when the data is came from a multinomial distribution. It is primarily used for document classification problems, it means a particular document belong to which category such as politics, education etc. The classifier uses the frequency of words for the predictors.

3) Bernoulli: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks.

3.3.2 Logistic Regression Algorithm

In logistic regression algorithm, the response variable takes only two values 0 and 1. It's a multivariable method for modelling the relationship between multiple variables and a categorical variable.

Suppose the model be

$$y_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_k x_{ik} + \epsilon_i, \quad i = 1, 2, 3, \ldots, n, \tag{3.7}$$

where $\boldsymbol{x_i'} = [x_{i1}, x_{i2}, \ldots, x_{ik}]$ and $\boldsymbol{\beta'} = [\beta_0, \beta_1, \ldots, \beta_k]$. In matrix notation, (3.7) can be expressed as

$$y_i = \boldsymbol{x_i'\beta} + \epsilon_i, \quad i = 1, 2, 3, \ldots, n. \tag{3.8}$$

The response variable y_i takes only two values, 0 and 1 and it is assumed to follow a Bernoulli distribution with parameter π_i. Its probability distribution is as follows

$$g(y_i) = \pi_i^{y_i} (1 - \pi_i)^{1-y_i}, \quad y_i = 0, 1. \tag{3.9}$$

Since $E(\epsilon_i) = 0$, the expected value of the response variable is

$$E(y_i) = \pi_i. \tag{3.10}$$

From (3.8) it follows that

$$E(y_i) = x_i'\beta = \pi_i \tag{3.11}$$

This means that the expected response given by the response function

$$E(y_i) = x_i'\beta \tag{3.12}$$

It is just the probability that the response variable takes on the value 1.
A natural choice of $E(y)$ would be the cumulative distribution function of a random variable. In particular, the logistic distribution, whose cdf is the simplified logistic function yields a good link is given by

$$E(y) = \frac{1}{1 + \exp(-x'\beta)}$$

The linear predictor is given by

$$\eta = \sum_j \beta_j x_{ij} = x_i'\beta, i = 1, 2, \ldots, n \; j = 1, 2, \ldots, n \tag{3.13}$$

The linear predictor η is defined by the transformation $\eta = \log \frac{\pi}{1-\pi}$. This transformation is called logistic transformation of the probability π and the ratio $\frac{\pi}{1-\pi}$ is called log odds.

3.3.3 Gradient Boosting Algorithm

Gradient boosting is a sophisticated machine techniques which recursively fit a weak learner to the residual to improve the model performance with a gradually increasing number of iterations. This algorithm automatically discover complex data structure, including non-linearity and high order interactions, even in the context of hundreds,thousands, or ten thousands of potential predictors.

The idea of boosting came out of the idea of whether a weak learner can be modified to become better. Michael Kearns, an American computer scientist articulated its goal as the Hypothesis 'Boosting Problem' stating the goal from a practical standpoint as:

An effcient algorithm for converting relatively poor hypotheses into very good hypotheses.

AdaBoost the First Boosting Algorithm

To understand gradient boosting algorithm efficiently we must have some knowledge about AdaBoost (Adaptive Boosting) Algorithm which is also a boosting method. This algorithm starts by building a decision stump and then assigning equal weights to all the data points. AdaBoost was the first realization of boosting that saw great success in application. Decision trees with single split are the weak learners in AdaBoost works by weighting the observation, putting more weight on those that are difficult to handle and less on those that are handled well. New weak learners are added sequentially that focus their training on the more difficult patterns. Thus the samples that are difficult to classify receive increasing larger weights until the algorithm identifies a model that correctly classifies these samples. Predictions are made by majority vote of the weak learners predictions, weighted by their individual accuracy.

Generalization of AdaBoost as Gradient Boosting

AdaBoost and related algorithms were recast in a statistical framework first by Leo Breiman and called them ARCing algorithms. ARCing is an acronym for Adaptive Re-weighting and combining where each step consists of a weighted minimizing followed by a re-computation of the classifiers and weighted input. This framework

was then further developed by Friedman and called Gradient Boosting Machines. It was called gradient boosting or gradient tree boosting later. The statistical framework cast boosting as a numerical optimization problem where the objective is to minimize the loss of the model by adding weak learners using a gradient descent like procedure. This class of algorithms were describe as a stage wise addictive model. This is because one new weak learner is added at a time and existing weak learners n the model are frozen and left unchanged. The generalization allowed an arbitrary differentiable loss function to be used and expanding the technique beyond binary classification problems to support regression, multi-class classification and more.

Working of Gradient Boosting

Gradient Boosting for classification has 3 components.

• Loss Function: The role of the loss function is to estimate how good the model is at making predictions with the given data. The loss functions used depends upon the type of problem being solved. An advantage of the gradient boosting framework is that a new boosting algorithm does not have to be derived for each loss function that may want to be used. The Algorithm is a generic enough framework that any different loss function can be used.

• Weaker Learner: A weak learner is one that classifies our data but does so poorly. Decision trees are used as the weak learner in gradient boosting. Specifically regression tree are used that output real values for splits and whose output can be added and correct the residuals in the predictions. Trees are constructed in a greedy manner, that the best split points based on purity score are chosen such that loss is minimum. Initially, very short decision trees where used that only had a single split, called a decision stump. Larger trees are used generally with 4 to 8 levels. Weak learners are constructed in a specific ways, such as a maximum number of layers, nodes , splits or leaf nodes. This is to ensure that though the learners remain weak, it can still be constructed in a gredy manner.

• Additive Model: Trees are added one at a time, and existing trees in the model are not changed. A gradient descent procedure is used to minimize the loss when

adding trees. Gradient descent is usually used to minimize a set of parameters, such as the coefficients in a regression equation, weights in a neural network etc. The weights are updated to minimize error after calculating error or loss. Instead of parameters , we use weak learner sub-models or more specifically decision trees. After calculating the loss, we must add a tree to the model that reduces the loss in order to perform the gradient descent procedure. This is achieved by parameterizing the tree, then modifying the parameter of the tree and by moving in right direction by reducing the residual loss. Generally this approach is called functional gradient descent or gradient descent with functions.

The output for the new tree is then added to the output of the existing sequence of trees in an effort to correct or improve the final output of the model. A fixed number of trees are added or training stops once loss reaches an acceptable level or no longer improves on an external validation dataset.

3.4 Matrix to Assess the Quality of classifiers

Matrix to Assess the Quality of classifiers consist of the following components,

True positive: Those instances where predicted class is equal to the actual class. A true positive can also be considered as an outcome where the model correctly predicts the positive class.

True negative: Those instances where predicted class and actual class are both negative. A true negative is an outcome where the model correctly predicts the negative class.

False positive: Those instances where predicted class or answer is positive, but actually the instances are negative. i.e, a false positive is an outcome where the model incorrectly predicts the positive class.

False negative: Those instances where predicted class is negative, but actually the instances are positive or a false negative can be said as an outcome where the model incorrectly predicts the negative class.

Precision: It is frequency with which a model was correct when predicting the

positive class. It is defined as given below:

$$Precision\,(p) = \frac{True\ Positive}{True\ Positive + False\ Positive}. \tag{3.14}$$

or

$$Precision\,(p) = \frac{True\ Positive}{All\ Positive\ Predicted}. \tag{3.15}$$

Precision gives an idea about if we predict positive then how often was it really positive instance

Recall: Recall identifies out the possible positive labels, how many did the model correctly identify?. i.e, it refers to what percentage of actual instances we are able to find. It is defined as:

$$Recall\,(r) = \frac{True\ Positive}{True\ Positive + False\ Negative}. \tag{3.16}$$

or

$$Recall\,(r) = \frac{True\ Positive}{All\ actual\ postive\ instances} \tag{3.17}$$

F-Measure: F-Measure is also known as F-1 score. It is a measure of accuracy. It considers both the precision p and the recall r of the test to compute the score. Here, p is the number of correct positive results divided by the number of all positive results returned by the classifier, and r is the number of correct positive results divided by the number of all relevant samples. Relevant sample means all samples that should have been identified as positive. The F-1 score reaches its best value at 1 which means it has perfect precision and recall, and its worst at 0. The F-measure is defined as

$$F = 2\left(\frac{precision.recall}{precision + recall}\right). \tag{3.18}$$

Chapter 4

DATA SOURCE AND DATA DESCRIPTION

4.1 Data Source

For the present study, the data released by *Cricsheet* is used. The dataset is taken from kaggle, a secondary source of data.

4.2 Data Description

The IPL matches dataset consist of two separate Comma Seperated Value(CSV) files: matches and deliveries. These files contains the information of each match summary and ball by ball details respectively.

The first dataset name is *matches.csv* (IPL Matches data from 2008 to 2022) whose size is 461 kb and it is taken from the Kaggle Repository. The number of attributes is 20 and total number of records is 950. The Attributes of the dataset are ID, City, Date, Season, MatchNumber, Team1, Team2, Venue, TossWinner, Toss-Decision, SuperOver, WinningTeam, WonBy, Margin, method, Player-of-Match, Team1Players, Team2Players, Umpire1, Umpire2.

The second dataset name is *deliveries.csv* (IPL Matches data from 2008 to 2022) whose size is 19483 kb and it is taken from the Kaggle Repository. The number

of attributes is 17 and total number of records is 225954 .The Attributes of the dataset are ID, innings, overs, ballnumber, batter, bowler, non_striker, extra_type, batsman_run, extras_run, total_run, non_boundary, isWicketDelivery, player_out, kind, fielders_involved, BattingTeam.

4.3 Software

Python is the statistical software used to do this analysis. Python is currently the fastest growing programming language in the world. It is an object oriented programming and structured programming. Python was conceived in the late 1980s by Guido van Rossum. There are many Python library for descriptive statistics. To do statistical analysis on IPL match data, certain libraries have to be imported. The various libraries imported are:

• Pandas: Pandas is an open source library that is mainly for working with relational or labelled data both easily and intuitively. It provided various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library.

• NumPy: NumPy is a python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and can use it freely.

• Matplotlib: Matplotlib is a third party library for data visualization. Matplotlib is a python library used to create 2D graphs and plot by using python scripts. It has a module named pyplot which make things easy for plotting by providing features to control line styles, fond properties, formatting axes etc. It works well in combination with NumPy, SciPy and Pandas.

• Seaborn: Seaborn is an open source python built on top of Matplotlib. It is used for data visualization and exploratory data analysis. Seaborn works easily with dataframes and the panda library. The graphs created also be customized easily.

• Sklearn: Scikit-learn (Sklearn) is the most useful and robust library for machine learning in python. It provides a selection of efficient tools for machine learning

and modelling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

Chapter 5

DATA ANALYSIS

5.1 Data preprocessing

There are some significant steps in data preprocessing in Machine Learning:
1) **Acquire the dataset**: Acquiring the dataset is the first step in data preprocessing in machine learning. This dataset will be comprised of data gathered from multiple and disparate sources which are then combined in a proper format to form a dataset. There are several online sources from where we can download datasets, we have downloaded data set from *kaggle.com*.
2) **Import all the crucial libraries**: Importing all the crucial libraries is the second step in data preprocessing in machine learning. The three core Python libraries used for this data preprocessing in Machine Learning are: NumPy, Pandas, Matplotlib.
3) **Import the dataset**: In this step, we need to import the datasets that we have gathered.

5.1.1 Data cleaning

No quality data, no quality results. This is the basic line behind the success of any algorithm. Because before using any algorithm we have to process the data in advance if we expect the best result from it. There are many potential problems with data such as incorrect values, inconsistencies, imperfections etc. There are

many pre-processing steps to deal with issues such as data cleaning and feature selection. The most crucial libraries for data analysis are Numpy and Pandas. Now we eliminate the raw with missing values and noises are smoothened. Other attributes that can be found using simple python programs includes runs left, balls left, wickets left, total runs, current run rate and required run rate. Following that there are some other changes that need to be made, such as in the initial years, there were different teams playing, so we need to remove those matches, and some other teams change there name so we want to rename them. Then we eliminate the matches where D/L is applied due to rain or other natural calamities.

5.1.2 Feature selection

After data cleaning the key-features extraction is considered as the important part of the feature selection. Only key features that are quite natural for predicting the cricket match score are extracted and understand. Here are some of the main features used that are listed as follows:

1. Batting team
2. Bowling team
3. City
4. runs_left
5. balls_left
6. wicket_left
7. current_run_rate
8. requird_run_rate
9. target

ID	Batting Team	Bowling Team	City	runs_left	balls_left	wickets_left	current_run_rate	required_run_rate	target	result
63224	Chennai super kings	Punjab Kings	Chandigarh	36	12	6	9	18	198	0
66046	Delhi Capitals	Royal Challengers Bangalore	Delhi	95	66	8	7.44	8.63	162	0
94090	Punjab Kings	Kolkata Knight Riders	Kolkata	125	89	9	7.54	8.426	164	0
86130	Chennai super kings	Sunrisers Hyderabad	Hyderabad	80	46	7	9.162	10.43	193	0
128624	Kolkata Knight Riders	Punjab Kings	Chandigarh	64	68	9	7.038	5.64	125	1

Table 5.1: The first five row of the data

features shown in the Table 5.1, comprises of our dependent and independent variables. Here "result" is our target categorical dependent variable.

5.1.3 OneHot Encoding

Since the dataset is a mix of both categorical and numerical data and Since machine learning algorithms only undestand data of numerical nature, the categorical data are encoded into numeric ones using one hot encoder One hot encoder is a technique that will convert categorical data into numeric data. It takes values in ascending order and converts into numeric data from 0 to $n - 1$.

5.2 Data Visualization

Visualizations are important part of any research to understand the business and behavior of data in a way that how different attributes are relating to target variable and what attribute should be the point of focus. Visualizations of data give valuable meaning insights. By the visualizations every end user can easily represent the data into understandable interactive graphic. Cubes will be generated related to different aspects of data. There are various visual analytic tools to create visualizations but as this research has been done in python so visualizations will also be made in python programming using mat plot lib libraries. As the topic of this research is to predict the winner of match so all the cubes will be related about how different attributes of data are interacting with match winner variable. A heatmap which shows correlation between numerical attributes are plotted. The strength of the linear relation between any two features will be determined here.

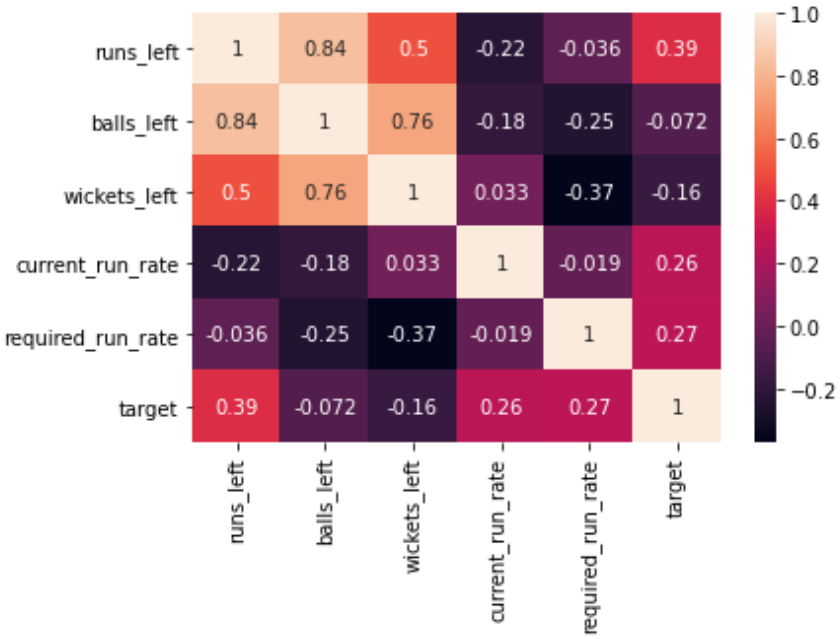

Figure 5.1: Heatmap

From the Figure 5.1 we can say that as the color changes from darker shade to lighter shade , the correlation increases. Also by analysing the heatmap, it is understood that none of the features are highly correlated with each other. Thus, each feature might have their individual contribution towards winner prediction. A Line graph uses lines to connect individual data points . Here we plot a line graph of City against its count

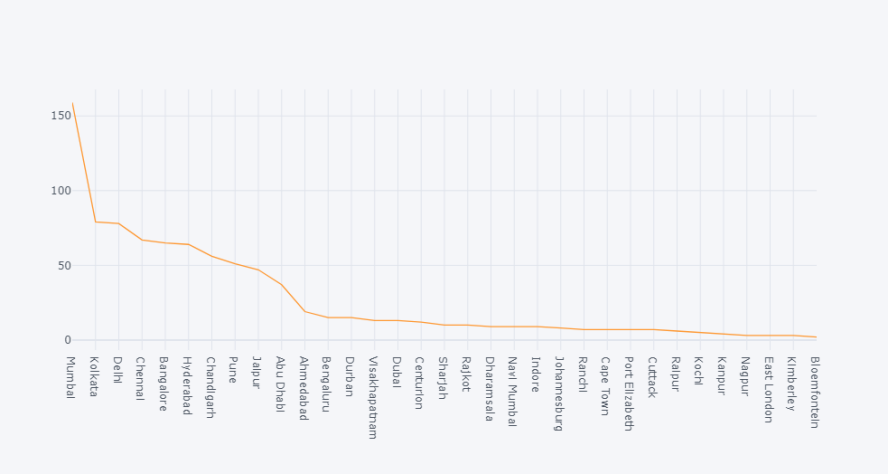

Figure 5.2: Game hosted in each city

From the Figure 5.2 we can say that Mumbai hosted more games while Bloemfontteln hosted the least.

The below bar graph shows the count of wins that receive per team.

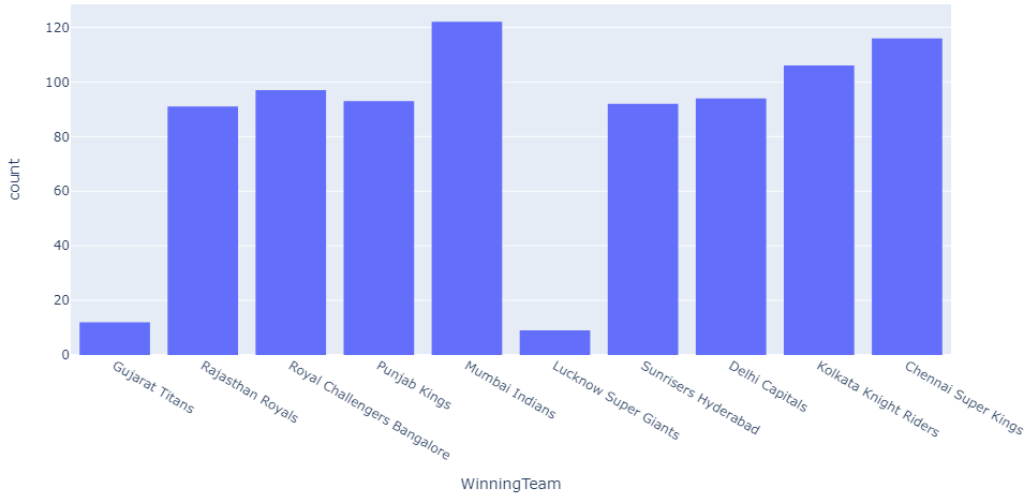

Figure 5.3: Number of matches won by each team

From the Figure 5.3 we can observe that Mumbai Indians have highest number of wins and Lucknow Super Giants have the lowest wins.

5.3 Model Evaluation

5.3.1 Evalute Models With Train and Test Sets(Train-Test)

Before building the model, Train-Test split evaluation has to be done. It's a techniques for evaluating the performance of the ML algorithm. It uses different training and testing datasets. Train-Test evaluation is performed by splitting the original datasets into two parts- Train datasets and Test dataset.

Train Dataset: It is used to fit the ML model.

Test Dataset: It is used to evaluate the fit ML model.

In Train-Test split we train the algorithm on the first part, then make predictions on the second part and evaluate the predictions aganist the expected results. The size of the split can depend on the size and specifies of the dataset. But it is common to use 67% of the data for training and the remaining 33% for testing. Thus here we choose 67% of the rows to go inside train data and the 33% to go the test data.

After Train-Test evaluation is done, we fit different models and assess their model accuracy.

The fitted models are,

- Naive Bayers
- Logistic Regression
- Random forest

5.3.2 Naive Bayes

We build a Naive Bayes model using GaussianNB() function from scikit learn library of python. We create an object of the class NaiveBayers and store the variable. We can asses the object using variable name. After building the model, the accuracy, precision, Recall value and F-1 score is calculated using accuracy-score, precision-score, recall-score, f1-score respectively. These functions are imported from the sklearn.metrics library of python.

The accuracy of the model was found to be 71.71%, precision value is 78.16%,re-

call value is 70.19% and F-1 score is 73.96%.

The confusion matrix of the test dataset is given below

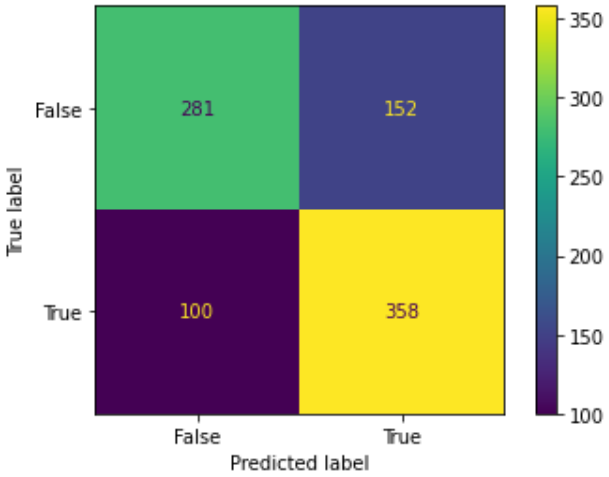

Figure 5.4: Confusion matrix by Naive Bayers prediction model

In the confusion matrix it can be observed that 40.17% of the test data indicates True positive, 31.53% indicates True negative, 17.05% indicates False positive and 11.22% indicates False negative. Thus we can say that if we can predict the winner using the fitted model Naive Bayes method we can obtain the prediction with accuracy about 71.71%.

5.3.3 Logistic Regression

The logistic regression model is fitted using LogisticRegression() function from scikit learn library of python. LogisticRegression class is imported from the sklearn library of python. Like in case of Naive Bayers, here also we create an object of the class and store in a variable and access the object using variable name.

The accuracy, precision, Recall value and F-1 score is calculated. The accuracy of the model was found to be 80.13% , precision value is 81.65%,recall value is 80.08% and F-1 score is 80.86%.

The confusion matrix of the test dataset is given below

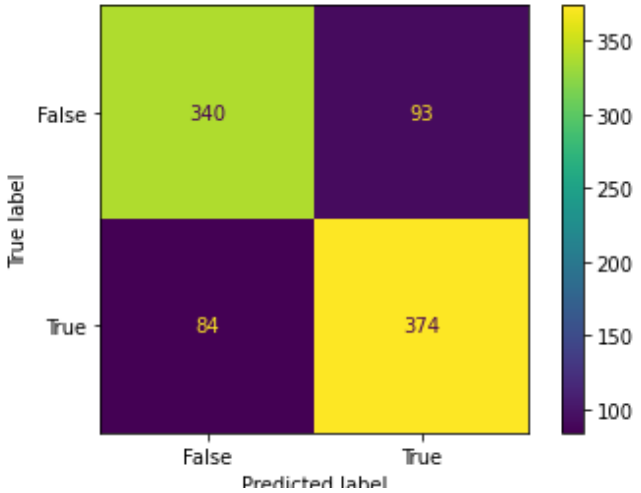

Figure 5.5: Confusion matrix by Logistic Regression prediction model

In the confusion matrix it can be observed that 41.97% of the test data indicates True positive, 38.15% indicates True negative, 10.43% indicates False positive and 9.42% indicates False negative. Thus we can say that if we can predict the winner using the fitted model Logistic Regression method we can obtain the prediction with accuracy about 80.08%.

5.3.4 Gradient Boosting Classification

The gradient boosting classification fitted using GradientBoostingClassifier function. GradientBoostingClassifier class is imported from the sklearn.ensemble library of python. Like in case of logistic regression, here also we create an object of the class and store it in a variable and access the object using the variable name. The accuracy, precision, Recall value and F-1 score is obtained. The accuracy of the model was fount to be 83.50%. Precision value is found to be 86.02% and the recall value is 82.59% . The F-1 score is 84.27%.

The confusion matrix of the test dataset is given below

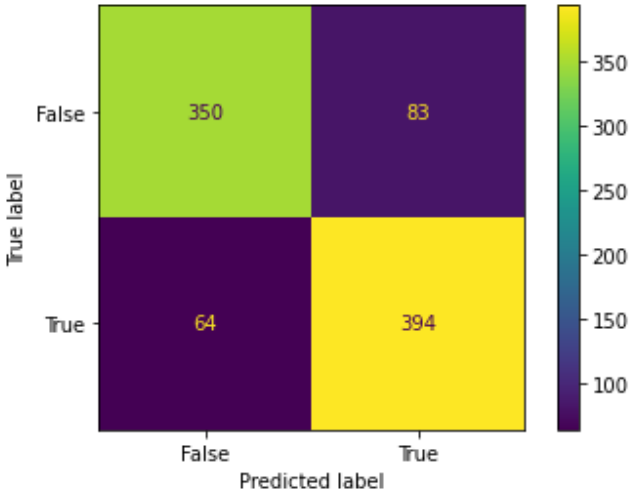

Figure 5.6: Confusion matrix by Gradient Boosting prediction model

In the confusion matrix it can be observed that 44.21% of the test data indicates True positive, 39.28% indicates True negative, 9.31% indicates False positive and 7.18% indicates False negative. Thus we can say that if we can predict the winner using the fitted model Gradient Boosting method we can obtain the prediction with accuracy about 83.50%.

5.3.5 Quality matrix

Thus we used three different models to fit our dataset. Different models gives different accuracy, precision, recall value and F-1 score. We compare these metrics to access quality of classifiers and decide which among the three models is best to predict the winner of the match. The following table gives the quality metrics of different machine learning model considered above.

Models	Accuracy	F-1 score	Precision	Recall Value
Naive Bayers	71.71%	73.96%	78.16%	70.19%
Logistic Regression	80.13%	80.86%	81.65%	80.08%
Gradient boosting	83.5%	84.27%	86.02%	82.59%

Table 5.2: Quality matrix of different models

As we compare each model, it is understood Gradient Booster Classification model has the best accuracy with highest F-1 score and recall value along with good precision. So it is considered as the best model to predict the winner among the three models.

Chapter 6

CONCLUSION

In this study we predict the match winner of IPL using historical data of IPL from season 2008 to 2022. To conduct the analysis and predicting the winner of IPL various branches of Data Science has been converged including Pre-Processing of data, Visualizations of data, preparation of data, feature selection and implementing different machine learning models for the predictions. Preprocessing has been done on the dataset to make it consistent by removing missing value, encoding variables into numerical format. Best features were selected by visualizing attributes of data with target variable. On selected features several machine learning models has been applied on the to predict the winner and the results were outstanding. First of all, Naive Bayers model was applied which predicted the match winner with accuracy 71.71%. Then we applied Logistic Regression on the selected features and the predicted the winner with 80.13% accuracy. In the last Gradient Boosting machine learning model was applied, and results were good. The accuracy achieved by Gradient boosting was 83.15% without tuning of parameters. In this study this accuracy is more efficient because we are predicting match winning percentage in the second inning. In cricketing field, to achieve the full convergence into data science world, it would require a lot of additional data to meet the full picture of analysis, i.e. every computation needs to perform very well, data needs to meet all the business problems and business systems. The prediction of winner produced through this project required a lot of domain information and expertise

for observations and their relations to the winning team.

The T20 format of cricket carries a lot of randomness, because a single over can completely change the ongoing pace of the game. Given the scale of the betting industry worldwide, there are obviously monetary gains for anyone with access to superior prediction techniques, whether through working with betting companies, selling predictions to professional gamblers or personal betting. This study can benefit cricket club managers, sports data analysts and scholars interested in sports analytics, among others.

Bibliography

[1] Abhishek, S., Patil, K., Yuktha, P., & Meghana, S. (2019)*" Predictive Analysis of IPL Match Winner using Machine Learning Techniques"*, International Journal of Innovative Technology and Exploring Engineering, Vol. 9, No. 1, pp. 430-435.

[2] Dayalbagh, D., Patvardhan, C., & Lakshmi, V.(2016) *"Data Analytics based Deep Mayo Predictor for IPL-9"*, International Journal of Computer Applications, Vol. 152, No. 6, pp. 6-11.

[3] Dhonge, N., Dhole, S., & Wavre, N. (2021) *"IPL Cricket Score and Winning prediction using machine learning techniques"*, IRJMETS.

[4] Goel, R.(2021) *"Dynamic Cricket match outcome prediction"*, Journal of Sports Analytics.

[5] Kumar, S., & Roy, S. (2018) *"Score Prediction and Player Classification Model in the Game of Cricket using Machine Learning"*, International Journal of Scientific and Engineering Research, Vol. 9, No. 2, pp. 237-242.

[6] Lamsal, R., & Choudhary, A. (2020) *"Predicting Outcome of Indian Premier League (IPL) Matches Using Machine Learning"*, ResearchGate.

[7] Lokhande, R.(2018) *"Prediction of live cricket score and winning"* , IJRTD, Vol. 4, No. 5.

[8] Sinha, S.(2020) *"IPL Win Prediction System To Improve Team Performance using SVM"*, IJFGCN Vol. 13.

[9] Srikantaiah., Khetan1, A., Kumar, B., Tolani, D., & Patel, H. (2021) *"Prediction of IPL Match Outcome Using Machine Learning Techniques."* Intelligent Computing Communication and Security (ICIIC).

[10] Vistro, D., Rasheed, F., & David, L. (2019) *"Cricket winner prediction with application of machine learning and data analytics"* ,IJSTR Volume 8, Issue 09.

[11] https://www.kaggle.com/datasets/vora1011/ipl-2008-to-2021-all-match-dataset

APPENDIX

Imports

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import cufflinks as cf
cf.go_offline()
import plotly.graph_objects as go
import plotly.express as px
from sklearn.pipeline import Pipeline
```

Import data

```
balls = pd.read_csv('ball.csv')
balls.shape
matches = pd.read_csv('matches.csv')
matches.shape
balls.head()
matches.head()
```

Finding total score of the first innings

```
total_score = balls.groupby(['ID', 'innings']).sum()['total_run'].reset_index()
total_score = total_score[total_score['innings']==1]
total_score['target'] = total_score['total_run'] + 1
```

Merge the target with match dataset

match_df = matches.merge(total_score[['ID','target']], on='ID')

Removing old teams and updating teams new names

match_df['Team1'] = match_df['Team1'].str.replace('Delhi Daredevils', 'Delhi Capitals')

match_df['Team2'] = match_df['Team2'].str.replace('Delhi Daredevils', 'Delhi Capitals')

match_df['WinningTeam'] = match_df['WinningTeam'].str.replace('Delhi Daredevils', 'Delhi Capitals')

match_df['Team1'] = match_df['Team1'].str.replace('Kings XI Punjab', 'Punjab Kings')

match_df['Team2'] = match_df['Team2'].str.replace('Kings XI Punjab', 'Punjab Kings') match_df['WinningTeam'] = match_df['WinningTeam'].str.replace('Kings XI Punjab', 'Punjab Kings')

match_df['Team1'] = match_df['Team1'].str.replace('Deccan Chargers', 'Sunrisers Hyderabad')

match_df['Team2'] = match_df['Team2'].str.replace('Deccan Chargers', 'Sunrisers Hyderabad')

match_df['WinningTeam'] = match_df['WinningTeam'].str.replace('Deccan Chargers', 'Sunrisers Hyderabad')

Remove the matches where D/L applied

match_df['method'].unique()

match_df = match_df[match_df['method'].isna()]

match_df = match_df[['ID','City','Team1','Team2','WinningTeam','target']].dropna()

Merge the match_df dataset with balls dataset

balls.columns

balls['BattingTeam'] = balls['BattingTeam'].str.replace('Kings XI Punjab', 'Punjab Kings')

balls['BattingTeam'] = balls['BattingTeam'].str.replace('Delhi Daredevils', 'Delhi Capitals')

balls['BattingTeam'] = balls['BattingTeam'].str.replace('Deccan Chargers', 'Sun-

risers Hyderabad')

balls = balls[balls['BattingTeam'].isin(teams)]

balls_df = match_df.merge(balls, on='ID')

Only select rows where we are in 2nd innings

balls_df = balls_df[balls_df['innings']==2]

Create new column current_score after each ball

balls_df['current_score'] = balls_df.groupby('ID')['total_run'].cumsum()

Create new column runs_left after each ball

balls_df['runs_left'] = np.where(balls_df['target']-balls_df['current_score']≥0,

balls_df['target']-balls_df['current_score'], 0)

Create new column balls_left after each ball

balls_df['balls_left'] = np.where(120 - balls_df['overs']*6 - balls_df['ballnumber']≥0,120

- balls_df['overs']*6 - balls_df['ballnumber'], 0)

Create new column wicket_left after each ball

balls_df['wickets_left'] = 10 - balls_df.groupby('ID')['isWicketDelivery'].cumsum()

Create new column current_run_rate after each ball

balls_df['current_run_rate'] = (balls_df['current_score']*6)/(120-balls_df['balls_left'])

Create new column required_run_rate after each ball

balls_df['required_run_rate'] = np.where(balls_df['balls_left']>0,

balls_df['runs_left']*6/balls_df ['balls_left'], 0)

Create new column result after each ball

def result(row):

return 1 if row['BattingTeam'] == row['WinningTeam'] else 0

balls_df['result'] = balls_df.apply(result, axis=1)

index1 = balls_df[balls_df['Team2']==balls_df['BattingTeam']]['Team1'].index

index2 = balls_df[balls_df['Team1']==balls_df['BattingTeam']]['Team2'].index

balls_df.loc[index1, 'BowlingTeam'] = balls_df.loc[index1, 'Team1']

balls_df.loc[index2, 'BowlingTeam'] = balls_df.loc[index2, 'Team2']

Creating final dataset for prediction

final_df = balls_df[['BattingTeam', 'BowlingTeam','City','runs_left','balls_left','wickets_left',

'current_run_rate','required_run_rate','target','result']]

X = final_df.drop('result', axis=1)

y = final_df['result']

Data Visualization

matches['City'].value_counts().iplot()

fig = px.histogram(match_df, x='WinningTeam')

fig.show()

heatmap= sns.heatmap(X.corr(),annot=True)

One hot enconding

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder trf = ColumnTransformer([

('trf',OneHotEncoder(sparse=False,drop='first'),['BattingTeam','BowlingTeam', 'City'])

], remainder = 'passthrough')

Evalute Models With Train and Test Sets

from sklearn.model_selection import train_test_split

X = final_df.drop('result', axis=1)

y = final_df['result']

X.shape, y.shape

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.01, random_state=42)

Predictive model using Naive Bayers Algorithm

from sklearn.naive_bayes import GaussianNB

naivebayers = Pipeline(steps=[('step1',trf), ('step2',GaussianNB())])

naivebayers.fit(X_train, y_train)

y_pred2 = naivebayers.predict(X_test)

from sklearn.metrics import precision_score

from sklearn.metrics import recall_score

from sklearn.metrics import f1_score

accuracy_score(y_pred2, y_test)

precision_score(y_pred2, y_test)

recall_score(y_pred2, y_test)

f1_score(y_pred2, y_test)

confusion_matrix2 = metrics.confusion_matrix(y_test, y_pred2)

cm_display2 = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix2,

display_labels = [False, True])

import matplotlib.pyplot as plt

cm_display2.plot()

plt.show()

Predictive model using Logistic Regression Algorithm

from sklearn.linear_model import LogisticRegression

logistic = Pipeline(steps=[('step1',trf), ('step2',LogisticRegression(solver="liblinear"))])

logistic.fit(X_train, y_train)

y_pred1 = logistic.predict(X_test)

accuracy_score(y_pred1, y_test)

precision_score(y_pred1, y_test)

recall_score(y_pred1, y_test)

f1_score(y_pred1, y_test)

cm_display1 = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix1,

display_labels = [False, True])

cm_display1.plot()

plt.show()

Predictive model using Gradient Boosting Algorithm

from sklearn.ensemble import GradientBoostingClassifier gbc = Pipeline(steps=[

('step1',trf), ('step2',GradientBoostingClassifier())])

gbc.fit(X_train, y_train)

y_pred3 = gbc.predict(X_test)

accuracy_score(y_pred3, y_test)

precision_score(y_pred3, y_test)

recall_score(y_pred3, y_test)

f1_score(y_pred3, y_test)

confusion_matrix3 = metrics.confusion_matrix(y_test, y_pred3)

```
cm_display3 = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix3,
display_labels = [False, True])
cm_display3.plot()
plt.show()
```