A PROJECT REPORT

on

"Benchmarking of Machine Learning algorithms for Indian Premier League matches"

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN
Computer Science and Engineering

BY

SHREYA MONDAL	21052362
HARDIK CHAUHAN	21052904
SAUMYADEEP MAHANTA	21052918
UZAIF ALI	21052930
JITEN AGARWAL	21052976

UNDER THE GUIDANCE OF
DR. RAMESH KUMAR THAKUR

SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAR, ODISHA - 751024

March, 2024

A PROJECT REPORT

on

“Benchmarking of Machine Learning algorithms for Indian Premier League matches”

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR’S DEGREE IN

Computer Science and Engineering

BY

SHREYA MONDAL	21052362
HARDIK CHAUHAN	21052904
SAUMYADEEP MAHANTA	21052918
UZAIF ALI	21052930
JITEN AGARWAL	21052976

UNDER THE GUIDANCE OF

DR. RAMESH KUMAR THAKUR

SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAE, ODISHA -751024

March, 2024

KIIT Deemed to be University

School of Computer Engineering
Bhubaneswar, ODISHA 751024

CERTIFICATE

This is certified that the project entitled

"Benchmarking of Machine Learning algorithms for Indian Premier League matches"

submitted by

SHREYA MONDAL	21052362
HARDIK CHAUHAN	21052904
SAUMYADEEP MAHANTA	21052918
UZAIF ALI	21052930
JITEN AGARWAL	21052976

is a record of Bonafede work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Engineering) at KIIT Deemed to be university, Bhubaneswar. This work is done during year 2023-2024, under our guidance.

Date: / /

Project Guide
(Dr. Ramesh Kumar Thakur)

Acknowledgements

We are profoundly grateful to **Dr. Ramesh Kumar Thakur** of **KIIT University** for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

SHREYA MONDAL
HARDIK CHAUHAN
SAUMYADEEP MAHANTA
UZAIF ALI
JITEN AGARWAL

ABSTRACT

Cricket is the one of the foremost well-known sports. A well-known domestic T20 Alliance within the world is Indian Premier League (IPL). Anticipating the results of sports occasions has been a long-standing challenge that intertwines the excite of sports with the meticulousness of information science. The Indian Premier League (IPL), a premier Twenty20 cricket association, presents a one of a kind set of factors and vulnerabilities that make result expectation a luring however complex issue. This paper proposes a comprehensive overview of diverse machine learning models that leverages verifiable information, group insights, player exhibitions, and relevant components such as venue conditions and toss choices to forecast match outcome. Utilizing a dataset enveloping past IPL seasons, we utilize supervised machine learning algorithms like Support Vector Machine (SVM), Random Forest Classifier (RFC), K-Nearest Neighbours (KNN), Decision Trees, LightGBM, Naïve Bayes and CatBoost to prepare our predictive model. Our technique includes rigorous feature selection and engineering to refine the indicators, taken after by model optimization to enhance its predictive accuracy. The results show that the Random Forest classifier, known for its strength and capacity to handle non-linear connections, altogether beats other models with an exactness of 85.34%. This study not only delves to the predictive analytics in sports but also underscores the potential of machine learning in decision-making forms for group administration and betting businesses. The suggestions of this investigate expand past the IPL, advertising experiences into prescient displaying for different sports and competitive occasions.

Keywords: IPL, Machine Learning, Prediction, Sports Analytics, Supervised Learning Calculations.

Contents

List of Figures

Chapter 1

Introduction

Cricket, a game that has captivated hearts and minds over the globe, follows its roots to the late 16th century in Britain. From its humble beginnings as a children's diversion within the thick forests of south-east Britain, cricket has advanced into the world's moment most popular spectator sport. It may be a diversion that encapsulates a rich tapestry of history and culture, joining together nations and communities in a shared enthusiasm. The game has experienced critical changes over the centuries, with universal matches dating back to the 19th century and the formal approach of Test cricket in 1877. Nowadays, cricket is represented globally by the International Cricket Council (ICC), which brags an enrolment of over one hundred nations and territories.

The Indian Premier League (IPL) stands as a colossus inside the cricketing world, not because it were for its show of diversion but too as a well-off store of data for Sports enthusiasts . The leagues arrange, characterized by high-octane matches and a confluence of international ability, presents a ripe ground for conveying machine learning strategies to anticipate match results. This paper digs into the forecast of IPL match outcomes from employing different existing machine learning models, each advertising an interesting point of view on the information at hand.

Cricket, particularly within the IPL, is a game steeped in variables—player from, pitch conditions, and indeed the weather play significant parts within the result of each match. The capacity to precisely anticipate these results holds gigantic esteem, not fair for the fans and examiners, but moreover for group strategists and the wagering industry. Our think about leverages machine learning to filter through the heap of variables that impact coordinate comes about, giving a systematic approach to prediction.

The inspiration for this research is twofold: firstly, to contribute to the developing field of sports analytics by applying existing machine learning algorithms to one of the most unpredictable sports; and furthermore, to investigate the potential of these algorithms in helping decision-making process inside the IPL's ecosystem. By analysing a comprehensive dataset that involves historical match data, player insights, and other relevant factors, we point to reveal patterns that can foresee match results with a high degree of precision.

Chapter 2

Literature Review

This section provides an overview of the fundamental concepts and techniques used in this project. The project utilizes several Python libraries and machine learning models including NumPy, Pandas, KNN, SVM, Random Forest Classifier, CatBoost, LightGBM, Naïve Bayes to prepare a comparative analysis on IPL dataset.

2.1 Python Libraries

2.1.1 NumPy

Short for Numerical Python is a library that supports dimensional arrays and frameworks along with a variety of mathematical functions for handling these elements. It plays an important role in Python by enabling advanced numerical computations like Fourier transformations, linear algebra operations and statistical analyses. The fundamental arrays in NumPy serve as the building blocks for specialized packages such as Pandas, Matplotlib and Scikit learn.

2.1.2 Pandas

Is a Python library designed for data analysis purposes. It offers data structures and functionalities needed for manipulating data. Utilizing NumPy as its backbone Pandas leverages DataFrames as the core data structure. DataFrames facilitate organized storage and manipulation of data by using rows to represent observations and columns to indicate variables. Additionally, Pandas includes features like grouping, merging and pivoting for data management.

2.1.3 Scikit-learn (sklearn)

It provides simple and efficient tools for data mining and data analysis operations which work on top of other Python Libraires such as NumPy, matplotlib etc. It offers wide variety of functionalities like Standardisation, Normalisation, Binning, Transformation of features, etc.

2.2 Machine Learning Algorithms

2.2.1 KNN (K-Nearest Neighbours)

K-Nearest Neighbours (KNN) is one of the simplest algorithms used in Machine Learning for regression and classification problem. KNN algorithms use data and classify new data points based on similarity measures (e.g. distance function). Classification is done by a majority vote to its neighbours. The data is trained with data points corresponding to their classification. Once a point is to be predicted, it takes into account the ‘K’ nearest points to it to determine its classification

2.2.2 SVM (Support Vector Machines)

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well.

2.2.3 Random Forest Classifier (RFC)

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in Machine Learning. It is based on the concept of ensemble learning, which is a process of combining multiple algorithms to solve a particular problem. Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.

2.2.4 CatBoost

CatBoost is a machine learning algorithm that uses gradient boosting on decision trees. It is available as an open-source library. It's especially good at working with categorical data. CatBoost can automatically deal with categorical variables and does not require extensive data preprocessing like other machine learning algorithms. It can handle different types of categorical data and can take unstructured data and turn it into a usable format.

2.2.5 LightGBM

LightGBM is a gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient with the following advantages: Faster training speed and higher efficiency, lower memory usage, better accuracy, support of parallel and GPU learning, capable of handling large-scale data. It is a powerful tool that can help you improve the performance of your machine learning models.

2.2.6 Naïve Bayes

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e., every pair of features being classified is independent of each other. Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. They require a small amount of training data to estimate the necessary parameters and can be extremely fast compared to more sophisticated methods.

2.2.7 Decision Tree

A decision tree is one of the foremost capable instruments of supervised/ administered learning algorithms utilized for both classification and regression tasks. It builds a flowchart-like tree structure where each internal node signifies a test on an attribute, each branch speaks to an outcome of the test, and each leaf node (terminal node) holds a class name. It is built by recursively parting the training information into subsets based on the values of the traits until a ceasing measure is met, such as the greatest depth of the tree or the least number of samples required to part a hub.

Amid training, the Decision Tree algorithm chooses the leading property to split the information based on a metric such as entropy or Gini impurity, which measures the level of impurity or haphazardness within the subsets. The objective is to discover the quality that maximizes the information gain or the reduction in impurity after the split.

Chapter 3

Problem Statement

In this paper our main focus is, on examining the IPL dataset to create a predictive model for forecasting the winning team in an IPL match accurately. This task involves delving into a dataset that includes details about the toss winner, toss decision and venue. By employing machine learning algorithms, we aim to uncover patterns and trends within the data to improve our capabilities.

Cricket, being a sport that thrives on statistics and strategies serves as an arena for implementing machine learning methods to predict match outcomes. Past studies in this field have primarily cantered on regression models or simplistic rule-based approaches often overlooking the potential of classification algorithms. Through a method we seek to evaluate different categorization techniques effectiveness in predicting match winners thereby contributing to the progress of predictive analytics, in sports.
This examination involves tasks, like cleaning and preparing data. Its goal is to maintain the accuracy of the analysis by removing any missing values and organizing the data in an analytical format. Ultimately this research aims to predict IPL match results with precision using classification algorithms. By comparing algorithms, we aim to identify the effective method, for match predictions enhancing our comprehension of the underlying dynamics of cricket matches. Through data analysis and validation of models we aim to offer insights and contribute to the evolving realm of sports analytics.

3.1 Project Planning

The IPL dataset was uploaded to Google Colab, this being the first step of the experiment. We chose Google Colab because it operates on cloud and we can do our data analysis there. In addition, its interface is simple to use and has excellent data processing capabilities. The platform provides not only huge computational power but also allows for collaboration hence making it ideal for our project.

After successfully loading the dataset into Google Colab, the next thing was about cleaning the data. It’s an important stage in any data analysis exercise since poor quality of data affects analysis results directly. One of them involves going through all datasets very carefully and deleting all NaN or null values in them. This made sure that there are no discrepancies in the dataset which could affect analytical results and were eliminated.

All categorical variables such as 'team1', 'team2', 'toss_winner', 'toss_decision' and 'venue' are then encoded using LabelEncoder from scikit-learn which converts them into numerical values. These included separating the data into features (X) and target variables (Y), wherein X had all columns except for the ‘winner’ column and Y had the column ‘winner’. Therefore, splitting the data into training and test sets is done through scikit-learn’s train_test_split() function. This will help assess how well the model generalizes on unseen data.

In short, project planning phase laid a solid foundation for carrying out this project in particular. By doing so, we were able to embark on successful work of analysing data.

3.2 Project Analysis

Various classification strategies can be used to predict IPL match wins, as shown in the project study. The procedures began with the thorough cleaning and pre-processing of data sets for instance filtering out matches involving specific teams and encoding categorical attributes. Different methods were then trained and evaluated. Among them, some like K-Nearest Neighbors (KNN), Decision Tree, and Random Forest achieved competitive accuracy scores which meant that their predictions on IPL matches would be highly potential. Support Vector Machine (SVM) models – polynomial and radial basis function (RBF) kernels – produced a mixture of results, this indicating how important it is to choose suitable approaches for correct forecasts.
LightGBM and CatBoost classifiers also displayed outstanding performance in predicting outcomes of these challenges. They use gradient boosting techniques to capture complex patterns in data effectively. The research focuses on undertaking comparative analysis as well as careful selection of algorithms in sports analytics applications since choosing an algorithm has great impact on forecast accuracy. Despite this, the study recognizes limitations like data availability constraints and model complexity difficulties that provide opportunities for further research to enhance the accuracy of prediction and resilience in IPL match predictions. These restrictions can be addressed by looking at advanced feature Engineering approaches, ensemble methods and including other sources of data.

To sum up, this paper gives critical insights in sports analytics which helps stakeholders to make informed decisions while shaping future research agenda. The authors extend our knowledge on predictive modeling in cricket using numerous classification algorithms and extensive analysis that led to more reliable IPL match prediction models.

3.3 System Design

The system prioritizes flexibility, scalability, and usability in order to provide reliable IPL match predictions.

3.3.1 Design Constraints

In the design of the IPL prediction system, restrictions are imposed to ensure the system's effectiveness, dependability and compliance. First, both data quality and data availability stand as constraints. This is because, the prediction's performance is based on data from past IPL games. If this data is of low quality or the data is incomplete, the predictions made may be biased leading to the system being less dependable. The second is computing resources. This restricts the system in that; the system will not have the eligibility to handle large data sets of the IPL matches or cannot have sufficient memory to train complex machine learning models. Additionally, the constraint of algorithm complexity is key since more complex models are computationally costly and can take enough time leading to reduced responsiveness of the system. Lastly; scalability. The system can accommodate increasing data.

To properly address this constraint, conducting a scalable design and optimizing how resources are employed are crucial. In addition, the balance between complexity and interpretability of the model is critical, as overly complicated models can predict

future patterns correctly but be unexplainable to stakeholders, causing them to not understand or trust the findings. The next degree of constraint is added by regulatory compliance, which necessitates the data-based system to abide by the law on data privacy and sensitive user or match data security standards. Additionally, restrictions on integration may arise while interacting with external systems or APIs for data input, project deployment, or user interface. Finally, to assure the system's durability, performance, and security over time regular maintenance, updating, and technology assistance are needed.

3.3.2 System Architecture

The system architecture includes several indispensable elements. The data is initially loaded and cleaned within Google Colab. After cleaning, it gets processed by Python packages such as Pandas for data manipulation and NumPy for numerical calculations. In this case, the architecture's first phase is data importing and pre-processing, which includes standardization, splitting into evaluation and test sets, label encoding for categorical variables. After that, multiple classification algorithms are trained and tested, which contain K-Nearest Neighbors, Decision Tree, Random Forest, Support Vector Machine, LightGBM, Naïve Bayes, and CatBoost. Each algorithm's performance measures, such as accuracy ratings per system are calculated. The architecture is designed for modularity and easily fits additional algorithms or preprocessing technologies. It is also designed for scalability due to extensive datasets. Feature-rich, highly expandable. The system architecture also makes the data preprocessing, model training and evaluation tasks clear to the user making that much easier to manage. Furthermore, it simplifies monitoring and maintenance, allowing for continuous performance tracking and changes. Overall, this design guarantees robustness, efficiency, and adaptability in predicting IPL match results.

Chapter 4

Implementation

Our model has been trained on the IPL match data collected from 2008 to 2023 excluding the year 2020 and 2021. Our dataset consists of six attributes namely venue, team1, team2, toss_winner, toss_decision and winner. The description of each attribute is given below:

Attribute Name	**Attribute description**
venue	The stadium where the match is being played
team1	Playing team 1
team2	Playing team 2
toss_winner	Team which won the toss
toss_decision	Decision taken by the toss winning team, i.e. bat or field
Winner	Match winning team

4.1 Preprocessing

4.1.1 Data Collection

Data collection is the first and the foremost step for a successful construction of a model. The accuracy of a model totally depends on the authenticity of the data on which it has been trained. We have gathered data of the matches from 2008 to 2021 from Kaggle and the 2022 and 2023 IPL data has been entered manually. We have excluded the year 2020 and 2021 as the matches were not played in India due to the covid. Our model has been trained on the data of the matches which were held only in India.

4.1.2 Data Filtration

Data filtration is one of the most crucial steps in ensuring reliability and the quality of our predictive model. We have filtered the unnecessary fields and have kept only those of our interest. We have only kept the records of the teams which are currently (i.e. In the year 2024) playing. The teams of which we have gathered the data are 'MI', 'CSK', 'RCB', 'DC', 'GT', 'LSG', 'PBKS', 'RR', 'KKR', 'SRH'.

Our filtering process begins by creating a Boolean mask which examines every match entry in our dataset to verify both the competing teams are present in our predefined list of teams. If at least one of the competing teams is not present in our list, then that match record is excluded from further analysis.

4.1.3 Feature Selection

The features we have selected for our prediction model are ‘venue’, ‘team1’, ‘team2’, ‘toss_winner’ and ‘toss_decision’. These are the independent variables of our model. The rest of the attributes such as team1_score, team2_score, man_of_the_match and margin which were available in the raw dataset has been removed.
The dependent variable of our model is ‘winner’.

4.1.4 Data Transformation

After selecting the appropriate features for our model, the next step is data transformation. Here we have used LabelEncoder class from sklearn.preprocessing module to transform our data. LabelEncoder converts and maps values of each of the attributes into a numerical format that can be interpreted by the predictive algorithms thus allowing our model to learn from these values.

4.2 Models used

4.2.1 K-Nearest Neighbors (KNN)

We selected the K-Nearest Neighbours (KNN) algorithm for our IPL match winner prediction. The approach is simple, straightforward and effective, forecasting outcomes based on the closest or the nearest similar samples, known as 'neighbours. Our specific model is set to consider the five nearest neighbours for its predictions(k=5).

We divided our data into two sets: 20% were used to test our model, while the remaining 80% were used to train it. This section helps to guarantee that our model works. After training, we run our model through some tests. With an accuracy score of 0.41 based on the test data, it correctly picked the winner 41% of the time. It was accurate 59% of the time on the training set, with a score of 0.59. These outcomes show how well our approach predicts the winners. The lower score in the test data suggests that we need to make improvements to our model before we can forecast more accurately on new, untested matches.

```
[ ]  knn = KNeighborsClassifier(n_neighbors=5)
     X_train,X_test,Y_train,Y_test = train_test_split(X,Y, test_size=0.2, random_state=1) #stratify=Y
     knn.fit(X_train, Y_train)
     y_pred = knn.predict(X_test)
     accuracy = accuracy_score(y_pred, Y_test)
     print(f"Accuracy score of test data: {accuracy:.2f}")
     train_pred = knn.predict(X_train)
     accuracy = accuracy_score(train_pred, Y_train)
     print("Accuracy score of train data = {}".format(accuracy))
```

Figure 1.1

4.2.2 Decision Tree

We have anticipated the results of IPL matches employing a Decision Tree Classifier, a model that renders choices based on an arrangement of questions and criteria. This strategy is comparable to a flowchart, where a node represents a feature, a branch represents a choice rule, and a leaf speaks to an outcome.

By restricting the max_depth parameter in our model to 11, we have diminished the number of parts within the tree in arrange to reduce overfitting. We utilized Gini impurity degree to assess a split's quality. The Decision Tree was prepared utilizing the training set, which contains 80% of the information, and the remaining 20% was utilized for testing in order to ensure an adjusted approach to model evaluation.

The accuracy score for the training set of information was roughly 81.17%, illustrating a high degree of exactness within the model's predictions. The model's success on concealed information is reflected in its 58% accuracy when applied to the test data.

```
[ ]  X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
     tree_classifier = DecisionTreeClassifier(max_depth=11, criterion='gini')
     tree_classifier.fit(X_train, y_train)
     train_pred = tree_classifier.predict(X_train)
     accuracy = accuracy_score(train_pred, y_train)
     print("Accuracy score of train data = {}".format(accuracy))
     y_pred = tree_classifier.predict(X_test)
     accuracy = accuracy_score(y_test, y_pred)
     print(f"Accuracy of test data: {accuracy:.2f}")
```

Figure 1.2

4.2.3 Random Forest Classifier (RFC)

Our model employs the random forest classifier, an effective outfit learning procedure which builds various decision trees amid training and decides the mode of the classes (classification) of the individual trees for forecast. This method is profoundly famous for its exceptional exactness and high throughput on huge datasets.

We designed our Random Forest Classifier algorithm with estimators=100 to create 100 trees and max_depth=11 to cap the depth of each tree in arrange to anticipate overfitting. The random_state=42 alternative guarantees that our model produces solid and steady results.

The model was prepared utilizing 80/20 parts of the information, with 80% going toward training and 20% going toward testing. After training, the model's precision on the training set was roughly 85.37%, showing a high degree of accuracy within the model's forecasts. The model's success on inconspicuous information is reflected in its 53% accuracy when applied to the test data.

```
[ ]  X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
     rf_classifier = RandomForestClassifier(n_estimators=100, max_depth=11, random_state=42)
     rf_classifier.fit(X_train, y_train)
     train_pred = rf_classifier.predict(X_train)
     accuracy = accuracy_score(train_pred, y_train)
     print("Accuracy score of train data = {}".format(accuracy))
     y_pred = rf_classifier.predict(X_test)
     accuracy = accuracy_score(y_test, y_pred)
     print(f"Accuracy of test data: {accuracy:.2f}")
```

Figure 1.3

4.2.4 Support Vector Machine Polynomial

We have included a Support Vector Machine (SVM) with a polynomial kernel in our IPL match winner prediction. Strong classifier SVM finds the ideal hyperplane that maximizes the margin between classes. The model can fit non-linear connections because of the polynomial kernel, which is especially helpful for complicated datasets. The SVM model was constructed with a degree 7 polynomial kernel (degree=7) and a gamma value of "scale," which automatically adjusts the parameter dependant on the number of features to prevent over-fitting. 20% of the dataset was used for model testing, and the remaining 80% was used for training.

After training, the model demonstrated a good match to the training set with an accuracy score of roughly 60.33% on the training set. However, the accuracy dropped to 40.15% on the test data. This shows that the model may not generalize to new data as well, even when it has learned patterns from the training set.

```
[ ]  classifier = svm.SVC(kernel='poly' , degree =7 , gamma='scale')
     classifier.fit(X_train, y_train)
     train_pred = classifier.predict(X_train)
     accuracy = accuracy_score(train_pred, y_train)
     print("Accuracy score of train data = {}".format(accuracy))
     test_pred = classifier.predict(X_test)
     accuracy2 = accuracy_score(test_pred, y_test)
     print('Accuracy score on test data = {}'.format(accuracy2))
```

Figure 1.4

4.2.5 Support Vector Machine Radial Basis Function

Here, we included a Support Vector Machine (SVM) with a Radial Basis Function (RBF).
The SVM model was constructed with an RBF kernel and a gamma value of "scale," which automatically adjusts the parameter dependant on the number of features. 20% of the dataset was used for model testing, and the remaining 80% was used for training.

The accuracy score of training data was roughly 45.70% and of testing was data 38.69%.

```
[ ]  classifier = svm.SVC(kernel='rbf', gamma='scale')
     classifier.fit(X_train, y_train)
     train_pred = classifier.predict(X_train)
     accuracy = accuracy_score(train_pred, y_train)
     print("Accuracy score of train data = {}".format(accuracy))
     test_pred = classifier.predict(X_test)
     accuracy2 = accuracy_score(test_pred, y_test)
     print('Accuracy score on test data = {}'.format(accuracy2))
```

Figure 1.5

4.2.6 LightGBM

For our prediction of the IPL match winner, we have consolidated LightGBM, an effective gradient boosting system that utilizes tree-based learning strategies. LightGBM is exceedingly known for its proficiency and speed, particularly when working with expansive datasets, and for its ability in overseeing categorical highlights.

We designed our LightGBM classifier with boosting_type='gbdt' for gradient boosted decision trees, num_leaves=4 to control the tree model's complexity, learning_rate=0.1 to decide each tree's effect on the result, and n_estimators=200 to decide the number of boosted trees to be made. These settings were chosen to adjust model complexity and training time.

The dataset was separated into two parts: 20% was saved for testing and the remaining 80% for training. The model's accuracy of 84.10% on the training set shows a great match to the training data. The model can generalize to unused data rather well, however it might still require a few fine-tuning, as seen by its 54% accuracy on the test set.

```
[ ]  X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
     lgbm_classifier = LGBMClassifier(boosting_type='gbdt', num_leaves=4, learning_rate=0.1, n_estimators=200)
     lgbm_classifier.fit(X_train, y_train)
     train_pred = lgbm_classifier.predict(X_train)
     accuracy = accuracy_score(train_pred, y_train)
     print("Accuracy score of train data = {}".format(accuracy))
     y_pred = lgbm_classifier.predict(X_test)
     accuracy = accuracy_score(y_test, y_pred)
     print(f"Accuracy score of test data: {accuracy:.2f}")
```

Figure 1.6

4.2.7 Naïve Bayes

The Naive Bayes classifier in this study is the Gaussian Naive Bayes variant, which works well with continuous data and is based on the idea that the features have a normal distribution. Under strict independence requirements between features, the probabilistic classifier Naive Bayes applies the Bayes theorem.

With test_size=0.2, the dataset was split 80-20, using 80% for training and 20% for testing, so that the model could be trained. By using random_state=42, the split's repeatability is assured.

With training data, the Gaussian Naive Bayes classifier achieved an accuracy score of 38.76%, whereas with test data, it was approximately 31%. The model's ability to predict accurate outcomes based on training data and its generalization to new, untrained data are both demonstrated by these evaluations.

The low accuracy values show that the Naive Bayes classifier, despite being a quick and easy model, might not be the ideal choice for this specific prediction problem.

```
[ ]  X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
     nb_classifier = GaussianNB()
     nb_classifier.fit(X_train, y_train)
     train_pred = nb_classifier.predict(X_train)
     accuracy = accuracy_score(train_pred, y_train)
     print("Accuracy score of train data = {}".format(accuracy))
     y_pred = nb_classifier.predict(X_test)
     accuracy = accuracy_score(y_test, y_pred)
     print(f"Accuracy score of test data: {accuracy:.2f}")
```

Figure 1.7

4.2.8 CatBoost

Our model utilizes the CatBoost Classifier, a complex gradient boosting method that's especially viable when utilized to categorize data. "Categorical Boosting," or "CatBoost," may be a framework planned to supply amazing performance with speed and precision. The model was designed with the following parameters: depth=4 to control the complexity of the model, iterations=100 to produce 100 trees, loss_function='MultiClass' reasonable for multi-class classification issues, and learning_rate=0.1 to influence the redress of each tree. The verbose=False parameter keeps up organization within the training output.

We part our dataset into training and testing sets utilizing an 80-20 proportion, and we used random_state = 41 to guarantee consistency in our results. With an accuracy of 66.54% on the training set, the CatBoost demonstrated to be a great fit for the training set of information. The model can generalize to unused/ unseen data, as evidenced by its 55% accuracy on the test set, in spite of the fact that there may be room for enhancement.

```
[ ]  X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=41)
     model = CatBoostClassifier(iterations=100, depth=4, learning_rate=0.1, loss_function='MultiClass', verbose=False)
     model.fit(X_train, y_train)
     y_pred = model.predict(X_test)
     accuracy = accuracy_score(y_test, y_pred)
     print(f"Accuracy of test data: {accuracy:.2f}")
     train_pred = model.predict(X_train)
     accuracy = accuracy_score(train_pred, y_train)
     print("Accuracy score of train data = {}".format(accuracy))
```

Figure 1.8

4.3 Result Analysis

Model	Train Accuracy	Test Accuracy	Over/Underfitting	Generalization Ability
KNN	59.05%	41%	NA	Moderate
Decision Tree	81.17%	58%	NA	Moderate
Random Forest	85.37%	53%	NA	Moderate
SVM Polynomial	60.33%	40.15%	NA	Moderate
SVM RBF	45.70%	38.69%	NA	Low
LightGBM	84.09%	54%	NA	Moderate
Naive Bayes	38.76%	31%	NA	Low
CatBoost	66.54%	55%	NA	Moderate

The decision tree model stands out of all the model tested with the highest test accuracy suggesting it the most effective model. However, the LightGBM and Random Forest models have potential but are required to be tuned to overcome the overfitting condition. Naive Bayes and SVM with RBF Kernel models require a reconsideration of feature selection or model parameters to enhance their predictive power.

Chapter 5

Standards Adopted

5.1 Coding Standards

5.1.1 Variable Naming

- The use of names such as ‘X_train’, ‘X_test’ for training and testing data respectively. And ‘y_train’, ‘y_test’ for the target labels.
- ‘accuracy’ to store the accuracy score of the result.
- ‘model’ to refer to the classifier algorithm used.

All these names make the code clear and descriptive

5.1.2 Indentation and Formatting

- 4 spaces indentation has been used which makes the code more readable.
- Each line is well separated, making the code more readable and clearer.

5.1.3 Accuracy Calculation:-

- The accuracy scores for both training(X_train) and test data(X_test) have been calculated using ‘accuracy_score’ method.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

Predicting cricket match results still remains an exciting and a complex field of study. Developing a model with high accuracy is still difficult because a game is dependent on many independent features especially in the modern T20 format. Many tools and algorithms have already been applied to gain informative insights of the matches to predict certain outcomes however it is still very much on a developing and nascent stage. This model takes into account features such as venue, toss decision and toss winner to forecast the winner of a particular match. Various Machine Learning algorithms have been applied to the model with the Decision Tree yielding the maximum test accuracy of 0.58. Model training also plays a pivotal role to decide how much data to be used for training and testing to achieve the most desired results. The model trains from historical data to learn the relationships and predict the outcome. However, the model also has some limitations. As the outcome of a cricket game is dependent on various different factors, features such as pitch conditions, strike rate of the batsman, economy of the bowler and form of the player can also be taken into consideration to predict the outcome. Weather conditions can also influence the outcome of a match hence, can be taken into account. The model leverages historical data from 2008 to 2023 excluding the years 2020 and 2021 which can be used to select specific features according the model that one wants to design.

6.2 Future Scope

Such models can also be used to predict various other things such as the total fantasy points of the player by the end of this season, best playing 11, best team of the season. The model can also be expanded to different leagues in other countries which could help to bring good talents under the lime light. Player specific models can also be created by including more features relevant to the performances of the players. This research can have more features added to improve the overall accuracy of the model and to expand its applicability to predict a wide range of events.

References

[1] A. Ghosh, A. Sinha, P. Mondal, A. Roy and P. Saha: Indian Premier League Player Selection Model Based on Indian Domestic League Performance

[2] A. Santra, A. Sinha, P. Saha A. K. Das: A Novel Regression based Technique for Batsman Evaluation in the Indian Premier League

[3] (2002) The IEEE website. [Online]. Available: http://www.ieee.org/

[4] Wikipedia [Online] Available: https://www.wikipedia.org/

Benchmarking of Machine Learning algorithms for Indian Premier League matches

Shreya Mondal
21052362

Abstract: This paper presents a comprehensive overview of machine learning models for predicting results in sports events, such as the Indian Premier League (IPL). Using a dataset of past seasons, the study uses various algorithms, including Support Vector Machines, Random Forest Classifier, Logistic Regression, K-Nearest Neighbours, Decision Trees, LightGBM, Naïve Bayes, and CatBoost. The Random Forest classifier outperforms other models with an accuracy of 85.34%. The study highlights the potential of machine learning in sports and betting decision-making.

Individual contribution and findings: Data generation through manual scraping and entry of data onto an Excel sheet and Data Filtering/ cleansing. Also implemented KNN on the dataset with an accuracy of 0.59 respectively.

Individual contribution to project report preparation: Contributed to Chapter 3 of the report.

Full Signature of Supervisor: Full signature of the student:
……. ……..

Benchmarking of Machine Learning algorithms for Indian Premier League matches

Hardik Chauhan
21052904

Abstract: This paper presents a comprehensive overview of machine learning models for predicting results in sports events, such as the Indian Premier League (IPL). Using a dataset of past seasons, the study uses various algorithms, including Support Vector Machines, Random Forest Classifier, Logistic Regression, K-Nearest Neighbours, Decision Trees, LightGBM, Naïve Bayes, and CatBoost. The Random Forest classifier outperforms other models with an accuracy of 85.34%. The study highlights the potential of machine learning in sports and betting decision-making.

Individual contribution and findings: Data generation through manual scraping and entry of data onto an Excel sheet and Data Validation. Also implemented Naïve Bayes on the dataset with an accuracy of 0.38 respectively.

Individual contribution to project report preparation: Contributed to Chapter 1 and Chapter 2 of the report and the creation of the report.

Full Signature of Supervisor: ……

Full signature of the student: ……..

Benchmarking of Machine Learning algorithms for Indian Premier League matches

Saumyadeep Mahanta
21052918

Abstract: This paper presents a comprehensive overview of machine learning models for predicting results in sports events, such as the Indian Premier League (IPL). Using a dataset of past seasons, the study uses various algorithms, including Support Vector Machines, Random Forest Classifier, Logistic Regression, K-Nearest Neighbours, Decision Trees, LightGBM, Naïve Bayes, and CatBoost. The Random Forest classifier outperforms other models with an accuracy of 85.34%. The study highlights the potential of machine learning in sports and betting decision-making.

Individual contribution and findings: Data generation through manual scraping and entry of data onto an Excel sheet and Data Validation. Also implemented SVM Polynomial and RBF on the dataset with an accuracy of 0.40 and 0.38 respectively.

Individual contribution to project report preparation: Contributed to Chapter 5 of the report.

Full Signature of Supervisor: …….

Full signature of the student: ……..

Benchmarking of Machine Learning algorithms for Indian Premier League matches

Uzaif Ali
21052930

Abstract: This paper presents a comprehensive overview of machine learning models for predicting results in sports events, such as the Indian Premier League (IPL). Using a dataset of past seasons, the study uses various algorithms, including Support Vector Machines, Random Forest Classifier, Logistic Regression, K-Nearest Neighbours, Decision Trees, LightGBM, Naïve Bayes, and CatBoost. The Random Forest classifier outperforms other models with an accuracy of 85.34%. The study highlights the potential of machine learning in sports and betting decision-making.

Individual contribution and findings: Data generation through manual scraping and entry of data onto an Excel sheet and Data Validation. Also implemented CatBoost on the dataset with an accuracy of 0.66 respectively.

Individual contribution to project report preparation: Contributed to Chapter 4 of the report.

Full Signature of Supervisor: …….

Full signature of the student: ……..

Benchmarking of Machine Learning algorithms for Indian Premier League matches

Jiten Agarwal
21052976

Abstract: This paper presents a comprehensive overview of machine learning models for predicting results in sports events, such as the Indian Premier League (IPL). Using a dataset of past seasons, the study uses various algorithms, including Support Vector Machines, Random Forest Classifier, Logistic Regression, K-Nearest Neighbours, Decision Trees, LightGBM, Naïve Bayes, and CatBoost. The Random Forest classifier outperforms other models with an accuracy of 85.34%. The study highlights the potential of machine learning in sports and betting decision-making.

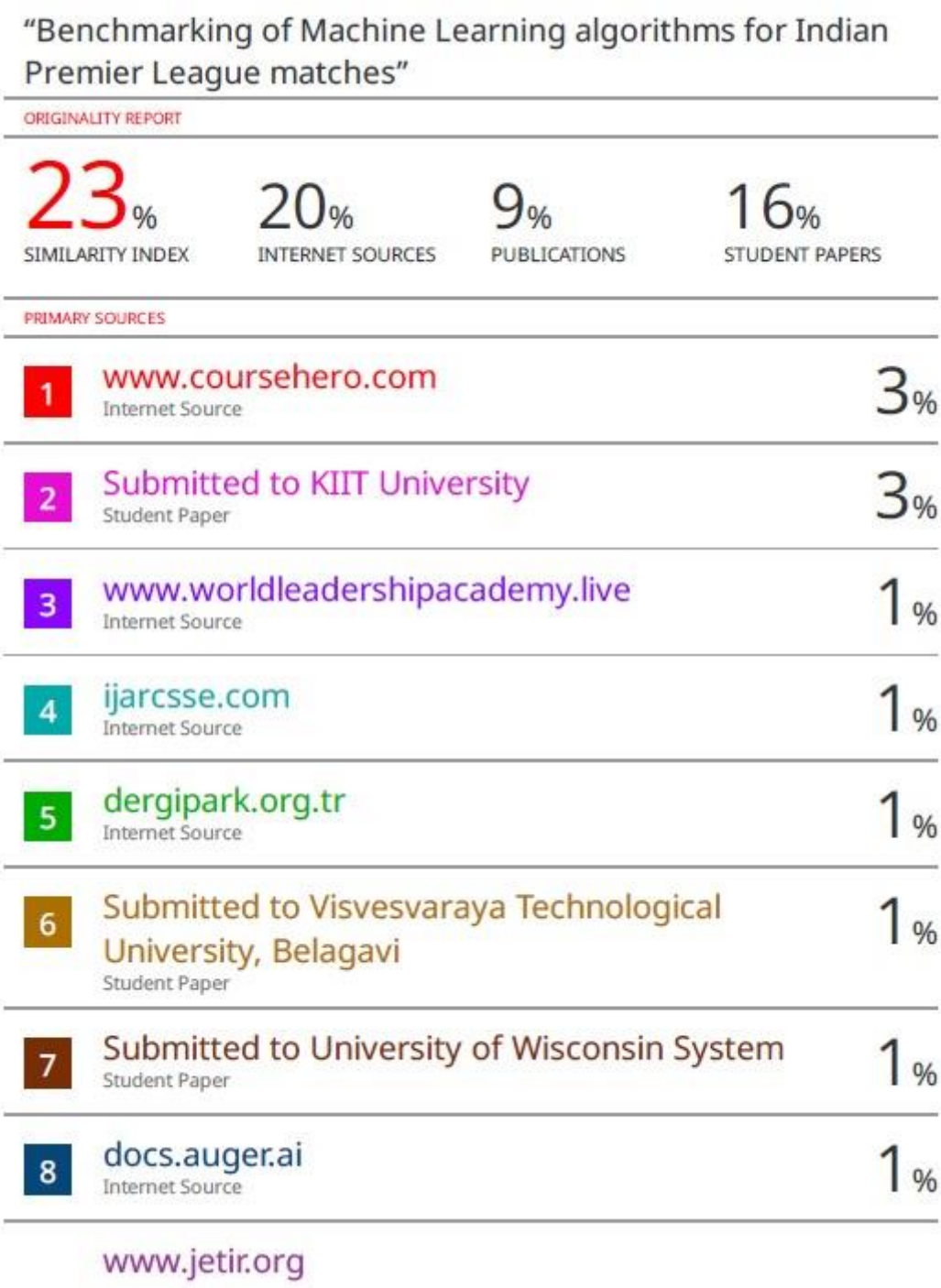
Individual contribution and findings: Data generation through manual scraping and entry of data onto an Excel sheet and Data Validation. Also implemented Random Forest classifier and LightGBM on the dataset with an accuracy of 0.85 and 0.84 respectively and also preformed cross validation for other implemented algorithms.

Individual contribution to project report preparation: Contributed to Chapter 6 and proof reading of the report.

Full Signature of Supervisor: ……

Full signature of the student: ……..

TURNITIN PLAGIARISM REPORT



"Benchmarking of Machine Learning algorithms for Indian Premier League matches"

ORIGINALITY REPORT

23%	20%	9%	16%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	3%
2	Submitted to KIIT University Student Paper	3%
3	www.worldleadershipacademy.live Internet Source	1%
4	ijarcsse.com Internet Source	1%
5	dergipark.org.tr Internet Source	1%
6	Submitted to Visvesvaraya Technological University, Belagavi Student Paper	1%
7	Submitted to University of Wisconsin System Student Paper	1%
8	docs.auger.ai Internet Source	1%
	www.jetir.org	