MACHINE LEARNING
LAB REPORT

Submitted By:

Name – Aaryan Sood, Sachin Sushil Singh

Roll number -102103574,102103575

Batch - 3COE21

Submitted To:

Mrs. Suchita Sharma

July 2023 – December 2023

Report: ODI Win Predictor

1. Introduction

Cricket is a sport that involves various strategies, and predicting the outcome of a match is an intriguing challenge. This report explores the development of an ODI (One Day International) cricket match win predictor using machine learning techniques.

2. Dataset Description main files: **ODI_Match_info.csv** and **ODI_Match_Data.csv**. These datasets contain comprehensive information about ODI matches, including details about teams, playe

The dataset used in this project consists of two rs, venues, and match outcomes.

ODI Match Data: This section contains detailed information about each delivery in One Day International (ODI) cricket matches. Here are the fields in this section:

match_id: Unique identifier for each match

season: Season of the match

start_date: Date of the match

venue: Venue where the match took place

innings: Inning number

ball: Ball number in the over

over: Over number

ballofover: Combined representation of over and ball

batting_team: Team batting during the delivery

bowling_team: Team bowling during the delivery

striker: Batsman facing the delivery

non_striker: Batsman at the non-striker end

bowler: Bowler delivering the ball

runs_off_bat: Runs scored by the batsman

extras: Extra runs (wides, no-balls, etc.)

total_runs: Total runs scored including extras

wides, noballs, byes, legbyes, penalty: Details of different types of extras

wicket_type: Type of dismissal (if a wicket falls)

player_dismissed: Batsman dismissed

other_wicket_type: Additional information about dismissal

other_player_dismissed: Additional player involved in a dismissal

cricsheet_id: Identifier related to the cricket scorecard

ODI Match Info: This section contains general information about ODI matches:

id: Unique identifier for the match

season: Season of the match

city: City where the match was played

date: Date of the match

team1, team2: Teams involved in the match

toss_winner: Team winning the toss

toss_decision: Decision taken by the toss-winning team (bat/field)

result: Result of the match

dl_applied: Whether the Duckworth-Lewis method was applied

winner: Winning team

win_by_runs, win_by_wickets: Margin of victory (runs/wickets)

player_of_match: Player of the match

venue: Venue of the match

umpire1, umpire2, umpire3: Umpires for the match

This dataset seems to provide a comprehensive overview of ODI cricket matches, including detailed delivery-level data along with match-specific information. It can be used for various analyses such as player performance, team strategies, match outcomes, and more.

The key steps in data preprocessing include:

- Calculating the total score for each innings in the match.
- Removing Teams that are no longer actively playing cricket internationally.
- Manipulating the dataset to train on 2nd innings data only.
- Merging columns to create new features that reduce redundancy in data.
- Removing irrelevant columns that do not contribute to prediction of the final result.
- Merging match information with total scores to create a consolidated dataset.
- Handling missing values and filtering out matches with DL (Duckworth-Lewis) method applied.

3. Used Methodology

In our machine learning project, we conducted a comprehensive analysis to identify the most suitable algorithm for predicting ODI cricket match outcomes. The methodology can be summarized as follows:

3.1. Dataset Selection:

We utilized two main datasets, namely `ODI_Match_info.csv` and `ODI_Match_Data.csv`, containing detailed information about ODI matches.

3.2. Feature Selection:

We considered a set of relevant features for prediction, including details about the batting team, bowling team, host city, runs left, balls left, wickets left, total runs, current run rate (CRR), and required run rate (RRR).

3.3. Data Preprocessing:

- The data underwent a crucial step of splitting into training and testing sets to assess the model's performance accurately.

- Categorical features were one-hot encoded to facilitate their inclusion in the model.

3.4. Algorithm Evaluation:

- We explored the efficacy of different algorithms, specifically testing all solvers available for logistic regression and utilizing the Random Forest algorithm.

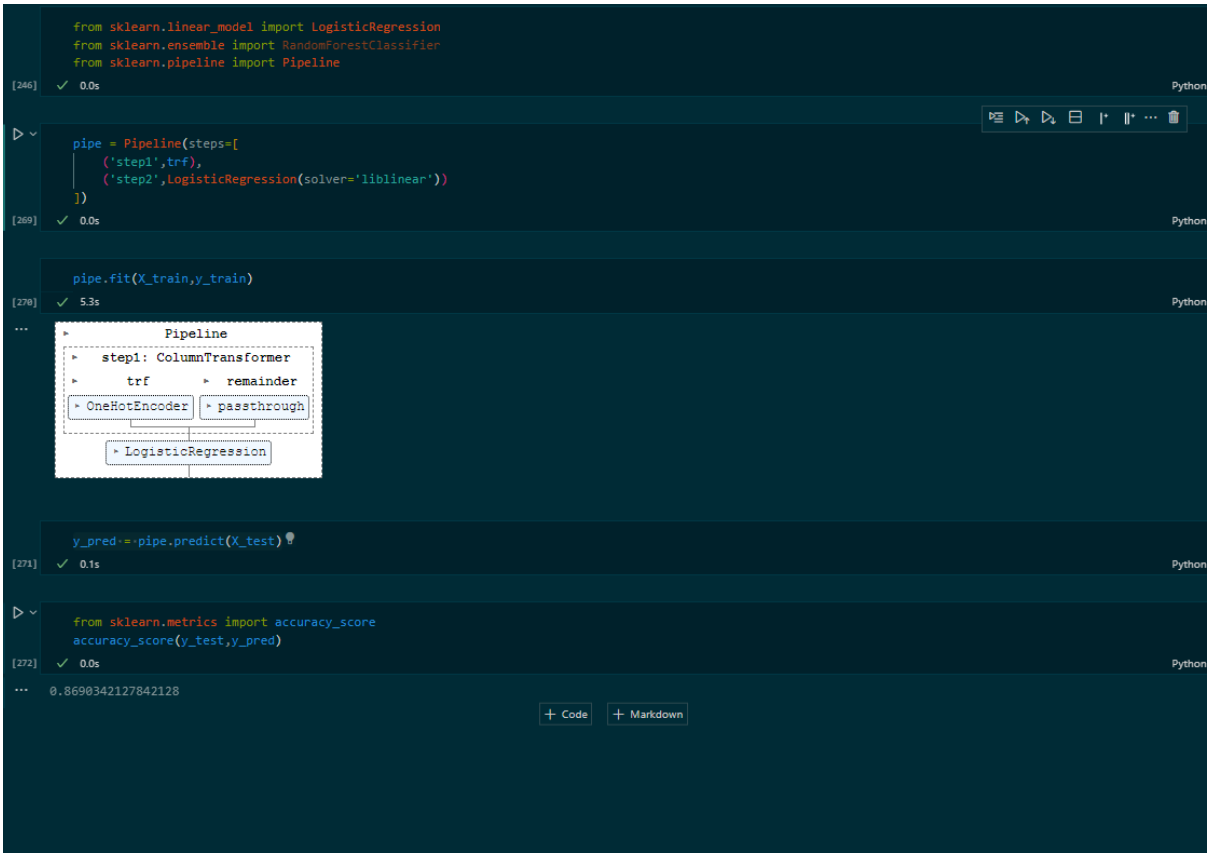
- The solvers for logistic regression were assessed to determine their impact on predictive accuracy.

Logistic regression is a classification algorithm used for predicting binary outcomes. In scikit-learn, the logistic regression model provides different solvers to optimize the model's parameters. Each solver uses a different optimization algorithm to find the weights that maximize the likelihood of the observed data. Here are insights into the common solvers available in scikit-learn's logistic regression and considerations for their use in the context of predicting ODI cricket match outcomes:

3.4.1 Liblinear:

- Insight: Suitable for small to medium-sized datasets, particularly when the number of features is significant compared to the number of samples.

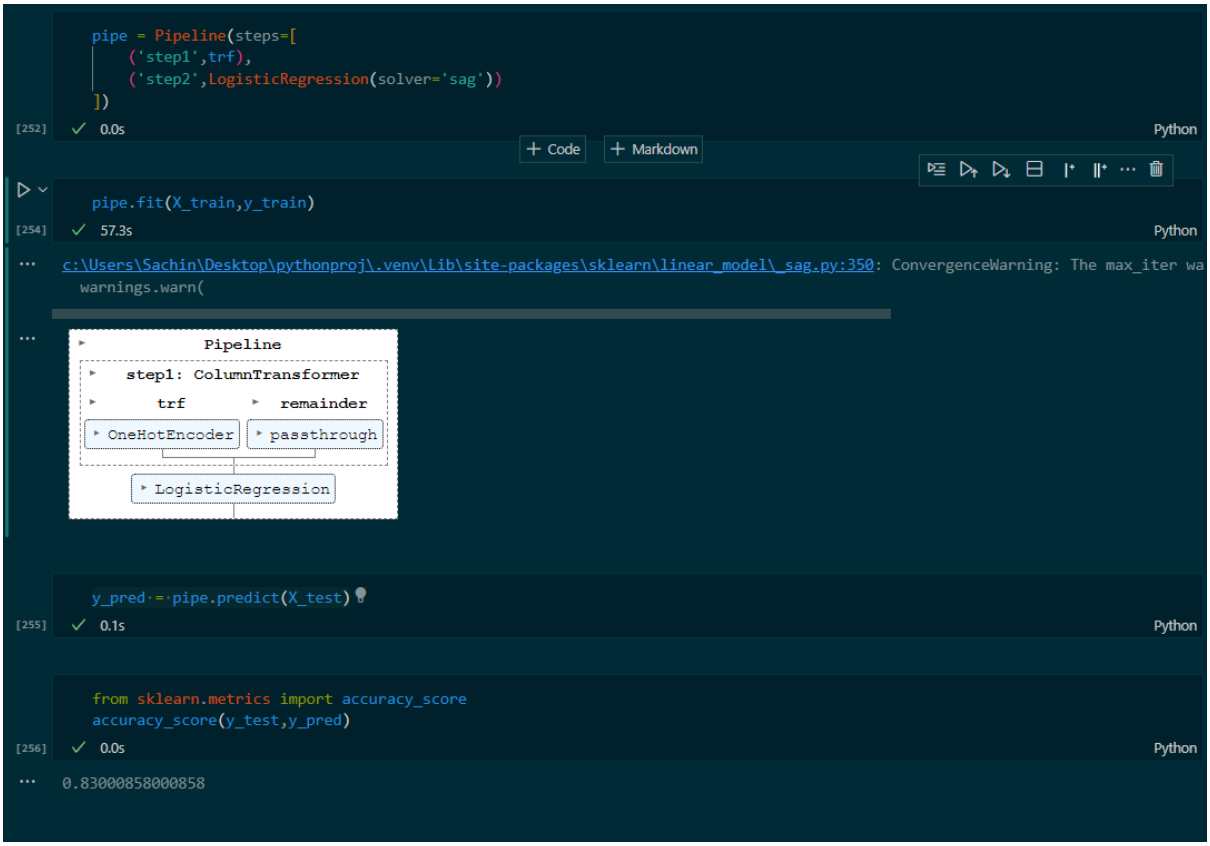
- Considerations: May not scale well for large datasets, and it is not the best choice when dealing with multicollinearity.



3.4.2 Newton-CG:

- Insight: Well-suited for datasets with a large number of features. It uses a Newton method to optimize the weights.

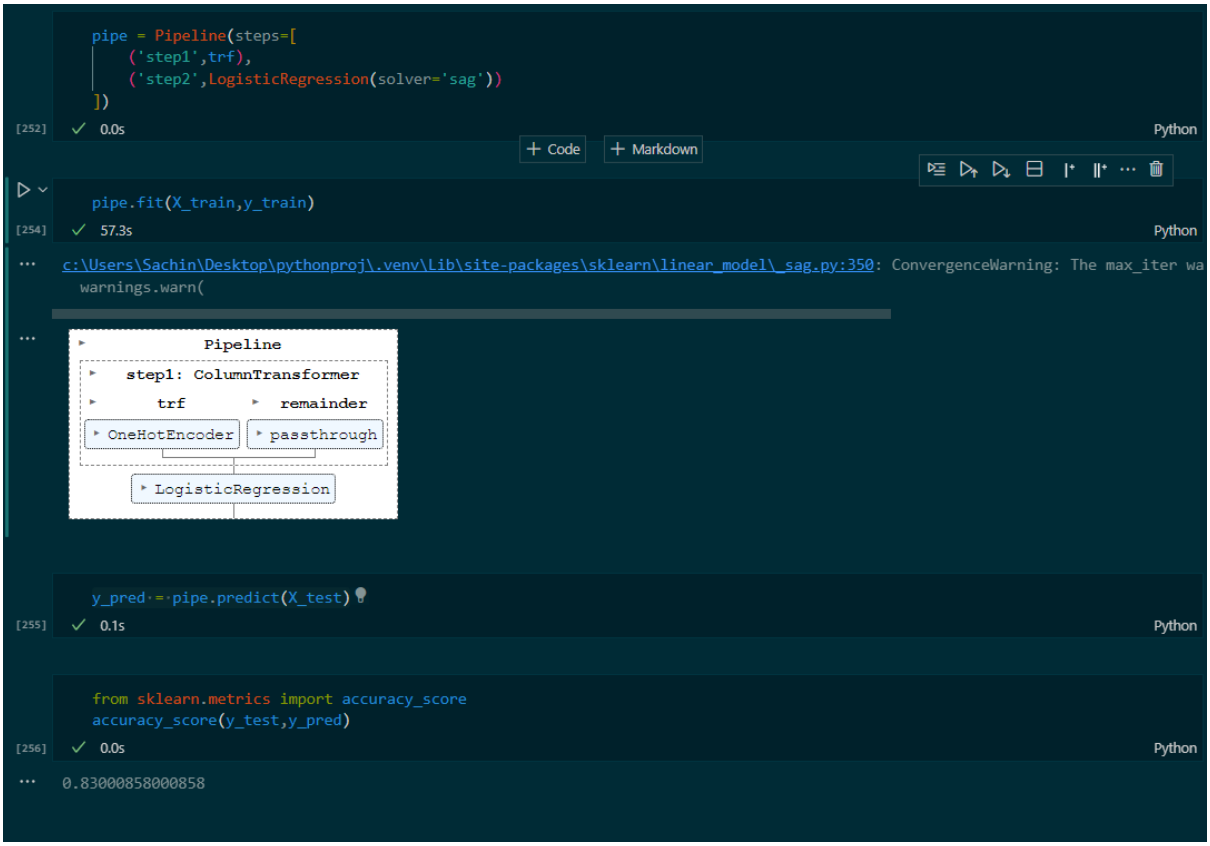
- Considerations: Can be computationally expensive for very large datasets. It also requires the computation of the Hessian matrix, which can be memory-intensive.



3.4.3 Sag (Stochastic Average Gradient):

- Insight: Designed for large datasets and can be more efficient than other solvers, especially when the dataset is sparse.

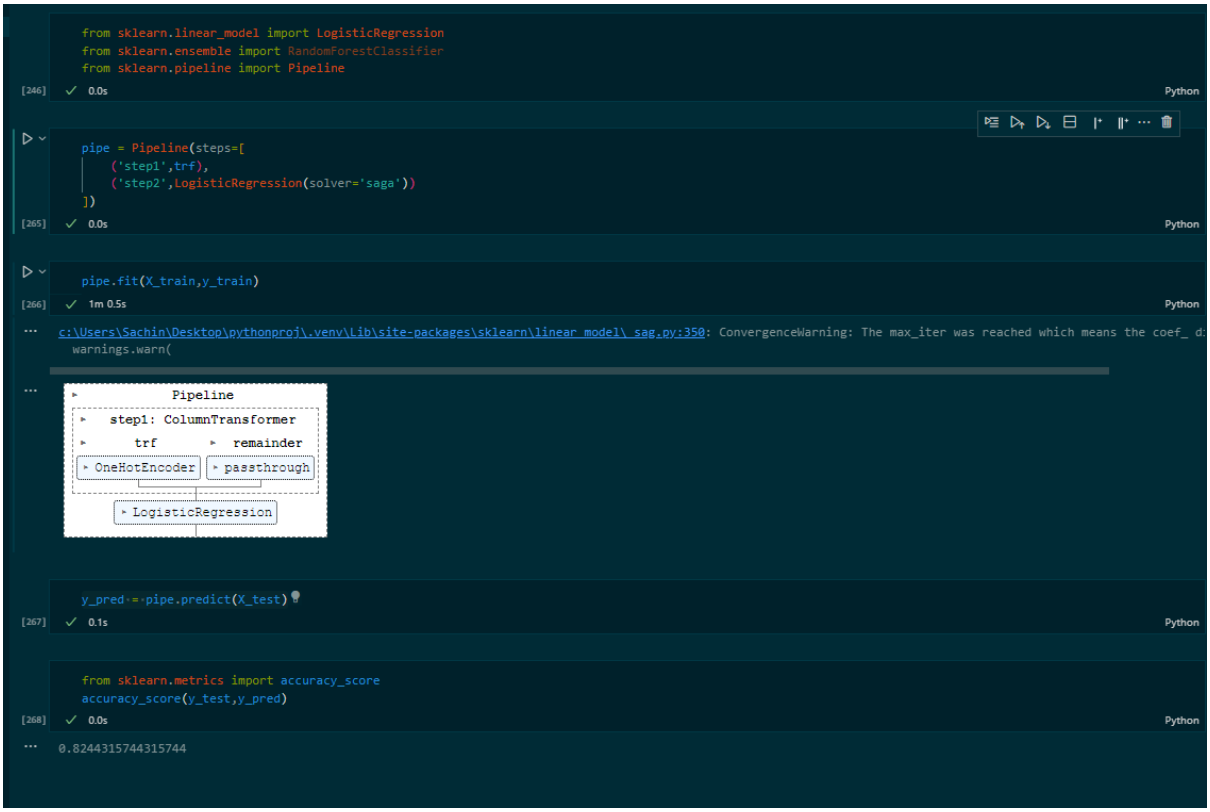
- Considerations: Appropriate for large datasets with a high number of samples and features. However, it may converge more slowly on smaller datasets.



3.4.4 Saga (Stochastic Average Gradient Descent):

- Insight: An extension of Sag that also supports L1 regularization.

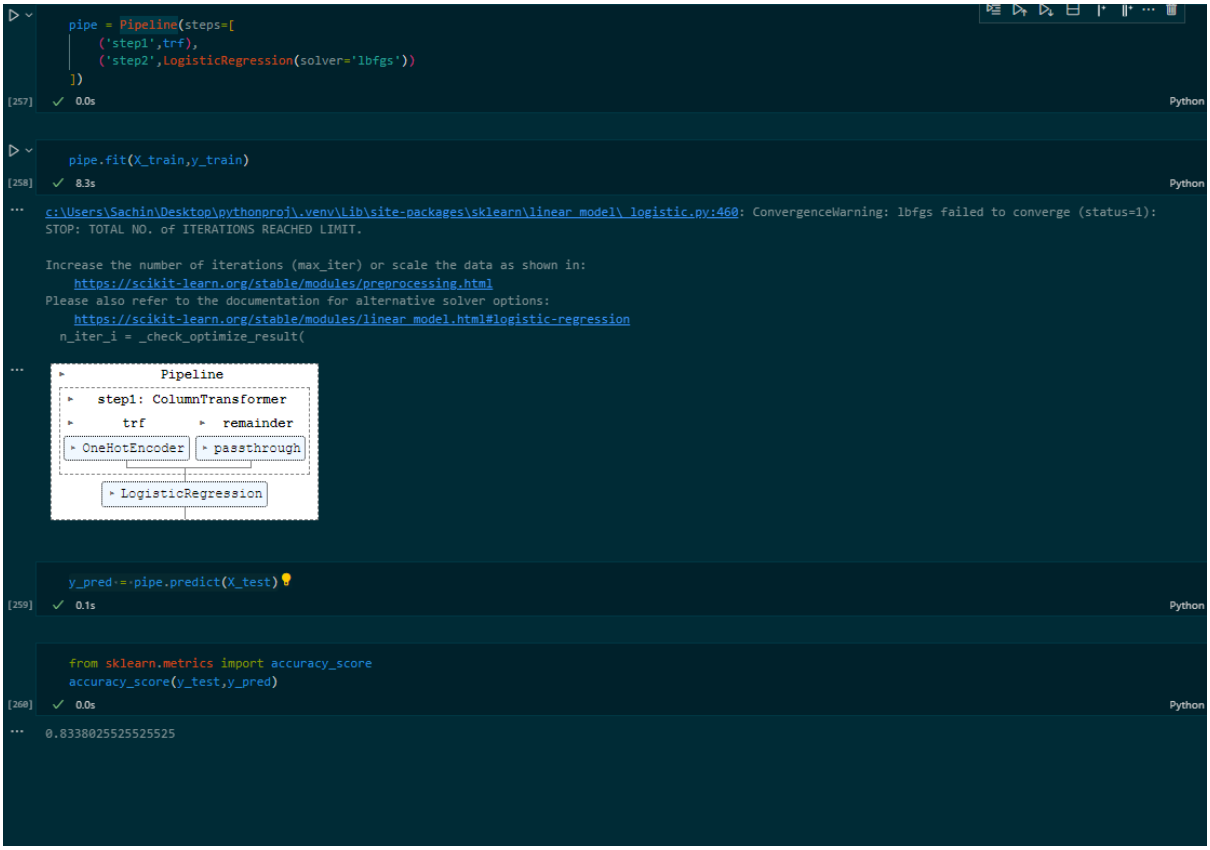
- Considerations: Useful when feature selection or sparsity is important. Can be slower to converge compared to other solvers.



3.4.5 Lbfgs (Limited-memory Broyden–Fletcher–Goldfarb–Shanno):

- Insight: A quasi-Newton method that works well for moderate-sized datasets.

- Considerations: Efficient for problems with a moderate number of samples. May not be the best choice for high-dimensional data.



It's essential to experiment with different solvers, monitor convergence, and choose the one that balances computational efficiency and predictive accuracy for the specific characteristics of your dataset.

While Random Forest is a powerful algorithm for classification and regression tasks, there are specific considerations that might make Logistic Regression a more suitable choice, especially when prediction is done based on probabilities. Here are detailed reasons why Random Forest might not be the optimal choice and the advantages that Logistic Regression offers in this context:

Random Forest Considerations:

1. Lack of Probability Interpretability:

- Issue: Random Forest produces predictions by averaging the results of multiple decision trees. While it's excellent for making accurate predictions, it doesn't provide easily interpretable probabilities.

- Importance: In scenarios where understanding the probability of a certain outcome is crucial (such as predicting match outcomes in cricket), having a clear interpretation of predicted probabilities is valuable.

2. Complexity and Overfitting:

- Issue: Random Forest tends to be a more complex model, consisting of multiple decision trees. This complexity can lead to overfitting, especially when dealing with smaller datasets.

- Importance: Overfitting might result in less reliable probability estimates, particularly when the model is trying to capture noise in the data rather than true underlying patterns.

While Random Forest is a versatile and powerful algorithm, Logistic Regression's interpretability, probabilistic output, and regularization properties make it particularly well-suited for scenarios where predictions are based on probabilities, and the interpretability of the model's predictions is crucial.

3.5 Model Training:

- Logistic Regression Pipeline: Constructed a logistic regression pipeline to streamline the training process.

- Random Forest: Trained the Random Forest algorithm to evaluate its performance in comparison to logistic regression.

6. Performance Assessment:

- Evaluated the performance of each algorithm using appropriate metrics, such as accuracy, to determine their effectiveness in predicting ODI match outcomes.

By systematically comparing various solvers for logistic regression and employing the Random Forest algorithm, our methodology aimed to identify the most suitable algorithm for achieving accurate predictions in the context of ODI cricket matches. The results of this analysis provide valuable insights into the strengths and limitations of different algorithms for this specific prediction task.

4. Results

The trained model demonstrates promising predictive capabilities. The Streamlit web application allows users to input match details and obtain predictions for the probability of winning or losing for the selected teams.

5. Individual Performance : -

1.Sachin Sushil Singh: -

Appropriate Dataset Selection, Data Cleaning and Feature Construction

2. Aaryan Sood: -

Model Building, Testing Analysis, GUI Formation

Sample Output

For example, given a specific scenario:

- Batting Team: India
- Bowling Team: Australia
- Host City: Melbourne
- Target: 300
- Current Score: 220
- Overs Completed: 35
- Wickets Out: 4

India - 83% Probability of Winning

Australia - 17% Probability of Losing