Project Overview:

This repository analyzes a number of machine learning models designed to predict whether a cricket player (batter and bowler) will exceed a specific performance threshold in upcoming matches. The project leverages open source historical data on players' performances in each individual game and extracts the most relevant features including runs scored, strike rates, recent performance in the last 'N' games and match conditions, to train various predictive models such as logistic regression, Random Forest, GBM, KNN and SVM.

The analysis is focused around a) finding the right feature combinations for each classifier and b) coming up with a ranking system to compare the performance of the classifiers based on Accuracy, TPR and TNR.

Feature Engineering:

1) Batting in ODI format:

The project looks at 2 different formats of the game: T20 and ODI. For the sake of simplicity, let's just focus on the ODI format first. In each format there are two distinct playing roles: batter and bowler (identical to baseball, bowler is the pitcher equivalent in cricket). A separate model is built for each playing role.

Let's look at the **batting role** first. These are the initial features that I was able to extract from the raw data.

1. "Opposition": opponent in the upcoming game (categorical)
2. "Ground": name of the ground where the game will be played (categorical)
3. "Country": the native country of the player (categorical)
4. "Avg_runs": runs scored per game over the entire dataset for a particular player (numeric)
5. "Recent_avg": runs scored per game over the last 5 games for a particular player (numeric). To gauge the current form of the player.

6. "Avg_sr": how quickly did a particular player score his runs over the entire dataset (numeric)

7. "Recent_avg_sr": how quickly did he score in the last 5 games (numeric). To gauge the current form of the player.

I applied **all of these features** to each of my classifiers and recorded the TPR, TNR and Accuracy scores. After that, I wrote a program that will perform a brute force search over the entire feature set and find the best feature combination for each classifier. For example, with 7 features the search space will be 2^7=128 different combinations. I compared the results for before and after for each classifier in order to try to extract some insights about each classifier. The summary of the experiment is below:

<u>ODI Batting Table:</u>

Classifier	Metric	All Features	Optimal Features	Drop/Increase	My observation
LOGISTIC REGRESSION	TPR	66.41	67.8 (TPR)	1.39	
	TNR	65.72	82.37 (TNR)	16.65	
	Accuracy	66.22	66.4 (Accuracy)	0.18	Negligible
KNN	TPR	64.7	100.0 (TPR)	35.3	
	TNR	58.96	78.27 (TNR)	19.31	
	Accuracy	63.1	72.21 (Accuracy)	9.11	Significant
GRADIENT BOOSTING	TPR	55.23	55.23 (TPR)	0	
	TNR	79.28	100.0 (TNR)	20.72	
	Accuracy	61.91	61.91 (Accuracy)	0	Negligible, (GBM Managed to extract the most important feature on its own ?)
RANDOM FOREST	TPR	50.2	58.23 (TPR)	8.03	
	TNR	83.06	89.06 (TNR)	5	
	Accuracy	59.33	61.34 (Accuracy)	2.01	small difference

My observation from the above analysis was only KNN can get some significant help from a curated set of features. GBM showed no increase in accuracy which made me wonder if it was inherently able to extract the best features on its own (spoiler: this hypothesis was later proven wrong when I looked at the results for the bowler role, however, it held true for batting in T20 format!). Logistic regression and Random Forest showed pretty stable results overall.

If you are curious about which features KNN found to be the most useful, here they are: *Best Accuracy combination:* ***('country',)*** *with Accuracy = 72.21, with TPR = 100.0, with TNR = 0.0.* This looks rather odd to me because it doesn't use any numerical features like the players' overall performance or recent performance. Also, this result is very limiting as it fails to get any TN correctly. When all features were used, the TPR/TNR/Accuracy had a much more balanced result.

Based on this analysis, my conclusion was it's safe to use all of the features for predicting batting performance and using a very narrow subset of features can lead to a lopsided performance like the one we see with KNN.

---

2) **Batting in T20 format:**

Now, I wanted to test this hypothesis on the second format of the game: T20. This is a shorter format and as a result the dynamics of the game are substantially different from ODI. But the features I have used are very fundamental to the game and therefore I thought it's a reasonable experiment to compare the impacts of these features across these two formats of the game to see if my conclusion above holds true.

<u>T20 Batting Table:</u>

Classifier	Metric	All Features	Optimal Features	Drop/Increase	Observation
LOGISTIC REGRESSION	TPR	59.57	68.61 (TPR)	9.04	
	TNR	70.9	73.78 (TNR)	2.88	
	Accuracy	63.48	64.33 (Accuracy)	0.85	negligible increase
KNN	TPR	63.69	90.10 (TPR)	26.41	
	TNR	58.13	75.50 (TNR)	17.37	
	Accuracy	61.78	64.14 (Accuracy)	2.36	marginal increase
GRADIENT BOOSTING	TPR	55.85	84.97 (TPR)	29.12	
	TNR	75.7	81.61 (TNR)	5.91	
	Accuracy	62.7	62.70 (Accuracy)	0	Negligible, (GBM Managed to extract the most important feature on its own ?)
RANDOM FOREST	TPR	48.3	62.75 (TPR)	14.45	
	TNR	80.51	90.25 (TNR)	9.74	
	Accuracy	59.41	60.66 (Accuracy)	1.25	marginal increase

I was pleased to see the same trends in T20 format data- this validates the experiment is producing consistent results. We see a small increase in KNN but the other classifiers are showing almost similar performance.

Let's look at the optimal KNN features: *Best Accuracy combination: ('opposition', 'country', 'avg_runs', 'avg_sr', 'recent_avg_sr') with Accuracy = 64.14, with TPR = 68.5, with TNR = 55.87.* Although the performance is not lopsided, the features are very different from what we saw with ODI.

This shows that curating features specific to the classifier will lead to overfitting and will not generalize well to unseen data. Therefore, I am happy to stick to my original hypothesis that for predicting batting performance- it's safe to use all of the features initially extracted from the dataset to prevent overfitting and generalizing to unseen data.

---

3) **Bowling in ODI format:**

Now, let's look at the other aspect of the game: bowling (or pitching in baseball) and the importance of feature selection on the performance of the classifiers. I ran the same experiment as above: at first I tried all the features that I was able to extract from the dataset and then I tried the brute force search to find the optimal feature combinations of each feature. For the sake of brevity, I won't talk about what each feature does. Let's jump to the results:

ODI Bowling Table:

Metric	All Features	Optimal Features	Drop/Increase	Observation
TPR	52.2	58.34 (TPR)	6.14	
TNR	77.31	85.36 (TNR)	8.05	
Accuracy	60.35	61.62 (Accuracy)	1.27	small increase
TPR	54.58	95.31 (TPR)	40.73	
TNR	62.67	83.26 (TNR)	20.59	
Accuracy	57.2	67.97 (Accuracy)	10.77	significant increase
TPR	46.7	66.98 (TPR)	20.28	
TNR	82.02	87.10 (TNR)	5.08	
Accuracy	58.17	62.79 (Accuracy)	4.62	considerable increase (My hypothesis from the batting table analysis is proven wrong here)
TPR	43.47	84.43 (TPR)	40.96	
TNR	83.81	89.25 (TNR)	5.44	
Accuracy	56.57	66.43 (Accuracy)	9.86	significant increase

We see that logistic regression once again showed a pretty stable performance (spoiler: logistic regression will remain as the most stable performer when talking about performance ranking across formats and roles, this simple classifier is in fact our star performer). Just like before, we see a significant difference with KNN. GBM and Random Forest also show considerable increases.

Let's look at the optimal features selection for each of these classifiers that saw a significant increase and try to spot if the performance is lopsided with respect to TPR and TNR:

- KNN: Best Accuracy combination: ('career_wickets_per_game',) with Accuracy = 67.97, with TPR = 90.92, with TNR = 20.27
    - Pretty poor TNR, indicative of lopsided performance.

- GBM: Best Accuracy combination: ('recent_wickets_per_game', 'recent_strike_rate') with Accuracy = 62.79, with TPR = 66.98, with TNR = 54.07
    - Balanced TPR/TNR and a 4.6% increase in overall accuracy. GBM makes a strong case for curated feature engineering for bowling.

- Random Forest: Best Accuracy combination: ('recent_wickets_per_game',) with Accuracy = 66.43, with TPR = 84.43, with TNR = 29.0
    - Pretty poor TNR, indicative of lopsided performance.

We start to see some evidence of feature engineering for the bowling aspect of the game, GBM making the strongest case for a curated set of features. Looking at the features GBM picked: it emphasizes on the two features that tracks the players most recent performance in the last 5 games which intuitively makes sense.

4) **Bowling in T20 format:**

Let's look at a similar table for T20 format:

T20 Bowling Data:

Classifier	Metric	All Features	Optimal Features	Drop/Increase	Observation
Classifier	Metric	All Features	Optimal Features	Drop/Increase	
LOGISTIC REGRESSION	TPR	55.9	58.90 (TPR)	3	
	TNR	72.5	89.00 (TNR)	16.5	
	Accuracy	60.65	62.06 (Accuracy)	1.41	Small increase
KNN	TPR	57.58	100.00 (TPR)	42.42	
	TNR	54.9	77.67 (TNR)	22.77	

	Accuracy	56.81	71.56 (Accuracy)	14.75	significant increase
GRADIENT BOOSTING	TPR	54.85	56.61 (TPR)	1.76	
	TNR	73.49	78.44 (TNR)	4.95	
	Accuracy	60.18	61.06 (Accuracy)	0.88	negligible increase
RANDOM FOREST	TPR	50.13	84.76 (TPR)	34.63	
	TNR	76.79	87.90 (TNR)	11.11	
	Accuracy	57.75	68.67 (Accuracy)	10.92	significant increase

We see almost a similar pattern. Logistic regression showing stable performance once again. GBM took a different turn this time whereas Random Forest saw a big increase.
Let's look at the optimal features for KNN and Random Forest:

- KNN: Best Accuracy combination: ('recent_wickets_per_game',) with Accuracy = 71.56, with TPR = 98.9, with TNR = 3.3
  - Very poor TNR, indicative of lopsided performance

- Random Forest: Best Accuracy combination: ('recent_wickets_per_game',) with Accuracy = 68.67, with TPR = 84.76, with TNR = 28.49
  - Poor TNR, indicative of lopsided performance

Conclusions:

- ***No clear evidence supporting that a curated feature set will lead to a consistently better result for any of the classifiers***

- ***Logistic regression showed impressively high consistency and stability across the board***

Performance Ranking:

Here is a quick summary of the accuracy of each classifier across the four different dimensions of the game, before I talk about my overall observations (Refer to the github repository's test artifacts section for more comprehensive results):

Classifier Performances by Accuracy:

ODI Batting

1) Logistic Regression: 66.14%
2) Gradient Boosting: 63.11%
3) KNN: 62.60%
4) Random Forest: 59.62%

ODI Bowling

1) Logistic Regression: 60.35%
2) Gradient Boosting: 58.17%
3) KNN: 57.20%
4) Random Forest: 56.57%

T20 Batting

1) Logistic Regression: 63.48%
2) Gradient Boosting: 62.70%
3) KNN: 61.78%
4) Random Forest: 59.41%

T20 Bowling

1) Logistic Regression: 60.65%
2) Gradient Boosting: 60.18%
3) KNN: 56.81%
4) Random Forest: 57.75%

Calculating Average Accuracy:

Now, let's calculate the average accuracy for each classifier across all four scenarios:

1. Logistic Regression: 62.66%
2. Gradient Boosting: 61.04%
3. KNN: 59.60%
4. Random Forest: 58.34%

The final ranking was determined by calculating the average accuracy of each classifier across all four cricketing scenarios. ***Logistic Regression consistently showed the highest accuracy*** in both ODI and T20 formats, whether batting or bowling, leading it to rank the highest. Gradient Boosting also performed well, particularly in T20 formats, securing the second rank. KNN, while strong in individual categories, especially for TPR as noted earlier, fell behind in overall accuracy, placing third. Random Forest, despite its ability to handle diverse data and environments, ranked last, likely due to its conservative nature and possibly the settings or feature interactions not being fully optimized for these specific tasks.

This methodological approach ensures a fair and comprehensive comparison, balancing performance across different formats and roles to identify the ***most consistently effective classifier: Logistic Regression. This conclusion coincides with my earlier analysis of feature engineering- where we saw an impressively stable and consistent performance of logistic regression across the board.***