*The purpose of the provided code is to split the dataset into training and testing sets for a machine learning task. Here's a breakdown of the code :*

***(feature engineering and prediction)***

1. `X <- match\_df[, c('average\_rank', 'rank\_difference', 'point\_difference', 'is\_stake', 'is\_worldcup')]`

- This line selects specific columns from the `match\_df` DataFrame and assigns them to the variable `X`. These columns include 'average\_rank', 'rank\_difference', 'point\_difference', 'is\_stake', and 'is\_worldcup'. These columns likely represent the features or independent variables used for training the machine learning models.

2. `y <- match\_df$is\_won`

- This line assigns the 'is\_won' column from `match\_df` to the variable `y`. This column likely represents the target variable or the dependent variable that the machine learning models aim to predict.

3. `set.seed(42)`

- This line sets a random seed to ensure reproducibility. By setting the seed to the same value, you can obtain the same train-test split every time you run the code, which helps with result reproducibility and consistency.

4. `train\_indices <- caret::createDataPartition(y, p = 0.8, list = FALSE)`

- This line uses the `createDataPartition` function from the `caret` package to create indices for splitting the data into training and testing sets. The function takes the target variable `y` and splits the data into two sets with a ratio of 80% for training data and 20% for testing data. The resulting indices are stored in the `train\_indices` variable.

5. `X\_train <- X[train\_indices, ]`, `X\_test <- X[-train\_indices, ]`, `y\_train <- y[train\_indices]`, `y\_test <- y[-train\_indices]`

- These lines use the `train\_indices` to split the feature matrix `X` and the target variable `y` into separate training and testing sets. The training set consists of the rows indexed by `train\_indices`, while the testing set contains the remaining rows. The resulting subsets are stored in `X\_train`, `X\_test`, `y\_train`, and `y\_test`, respectively.

By splitting the data into training and testing sets, you can use the `X\_train` and `y\_train` subsets to train your machine learning models, and then evaluate their performance on the unseen data using the `X\_test` and `y\_test` subsets. This separation helps assess the model's ability to generalize to new, unseen data and avoid overfitting to the training data.

*The purpose of trying different machine learning models, such as Logistic Regression, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Gaussian Naive Bayes, Decision Trees, and Random Forest, in a notebook is to explore and compare their performance on a specific task or dataset. Here are the main reasons for trying multiple models:*

Accuracy is a common evaluation metric used in classification tasks. It measures the proportion of correct predictions made by the model out of the total number of predictions. In this case, the logistic regression model correctly predicted the outcome (whether a match is won or not) approximately 68.02% of the time on the test set

**1. Model Comparison:** Different machine learning models have different strengths and weaknesses. By trying multiple models, you can compare their performance metrics, such as accuracy, precision, recall, F1 score, or area under the ROC curve. This comparison helps identify which models are more suitable for the given task or dataset.

**2. Model Selection:** Based on the comparison, you can select the best-performing model or a combination of models to use for your specific problem. Model selection involves considering factors such as model complexity, interpretability, training and prediction time, and the trade-off between bias and variance.

**3. Hyperparameter Tuning:** Each machine learning model has various hyperparameters that control its behavior and performance. By trying different models, you can also experiment with tuning their hyperparameters to optimize the model's performance. This process involves adjusting parameters through techniques like grid search or random search to find the best combination for your specific problem.

**4. Ensembling Techniques:** Trying different models allows you to explore ensembling techniques such as combining the predictions of multiple models to improve overall performance. Techniques like voting (where predictions are made based on majority voting) or stacking (where predictions become inputs for another model) can help leverage the strengths of individual models and mitigate their weaknesses.

**5. Generalization:** Different machine learning models make different assumptions about the data and have different capabilities to handle various types of problems. By trying different models, you can assess the generalization ability of each model, i.e., how well it can perform on unseen data, and choose the most robust model that performs well on both the training and test/validation datasets.

*Bias and variance are two important concepts in machine learning that relate to the predictive performance of a model.*

Bias refers to the error introduced by approximating a real-world problem with a simplified model. It captures how closely the model's predictions align with the true values. A high bias indicates that the model is making strong assumptions about the data, leading to oversimplification and potentially underfitting the training data. In other words, the model is unable to capture the underlying patterns and tends to consistently underestimate or overestimate the true values. High bias can result in low accuracy and poor generalization to unseen data.

Variance, on the other hand, refers to the variability or sensitivity of the model's predictions to fluctuations in the training data. It measures how much the model's predictions change when trained on different subsets of the data. A high variance indicates that the model is too complex and sensitive to noise or random fluctuations in the training data, resulting in overfitting. Overfitting occurs when the model learns the training data too well but fails to generalize to new, unseen data. High variance can lead to poor performance on test data.

In summary:

- Bias captures the model's tendency to underfit the training data and make strong assumptions, leading to systematic errors.

- Variance captures the model's sensitivity to noise and its ability to fit the training data too closely, resulting in high variability and poor generalization.

The goal is to find a balance between bias and variance, known as the bias-variance tradeoff. It involves selecting a model that is neither too simple (high bias) nor too complex (high variance), but rather has an optimal level of complexity to achieve good performance on both training and test data. Various techniques such as regularization, cross-validation, and ensemble methods can be employed to manage bias and variance and improve model performance.

Group stage prediction

\_\_\_Starting group F:\_\_\_

Morocco vs. Croatia: Croatia wins with 0.60

Morocco vs. Belgium: Belgium wins with 0.66

Morocco vs. Canada: Morocco wins with 0.55

Croatia vs. Belgium: Belgium wins with 0.61

Croatia vs. Canada: Croatia wins with 0.60

Belgium vs. Canada: Belgium wins with 0.65

\_\_\_Starting group C:\_\_\_

Argentina vs. Saudi Arabia: Argentina wins with 0.69

Argentina vs. Mexico: Draw

Argentina vs. Poland: Argentina wins with 0.57

Saudi Arabia vs. Mexico: Mexico wins with 0.73

Saudi Arabia vs. Poland: Poland wins with 0.67

Mexico vs. Poland: Draw

\_\_\_Starting group A:\_\_\_

Senegal vs. Qatar: Senegal wins with 0.62

Senegal vs. Netherlands: Netherlands wins with 0.61

Senegal vs. Ecuador: Senegal wins with 0.59

Qatar vs. Netherlands: Netherlands wins with 0.74

Qatar vs. Ecuador: Ecuador wins with 0.57

Netherlands vs. Ecuador: Netherlands wins with 0.64

\_\_\_Starting group E:\_\_\_

Germany vs. Japan: Draw

Germany vs. Spain: Spain wins with 0.58

Germany vs. Costa Rica: Germany wins with 0.55

Japan vs. Spain: Spain wins with 0.64

Japan vs. Costa Rica: Draw

Spain vs. Costa Rica: Spain wins with 0.58

\_\_\_Starting group H:\_\_\_

Uruguay vs. South Korea: Draw

Uruguay vs. Portugal: Portugal wins with 0.58

Uruguay vs. Ghana: Uruguay wins with 0.69

South Korea vs. Portugal: Portugal wins with 0.65

South Korea vs. Ghana: South Korea wins with 0.62

Portugal vs. Ghana: Portugal wins with 0.71

\_\_\_Starting group B:\_\_\_

Iran vs. England: England wins with 0.63

Iran vs. USA: USA wins with 0.57

Iran vs. Wales: Wales wins with 0.56

England vs. USA: Draw

England vs. Wales: Draw

USA vs. Wales: Draw

\_\_\_Starting group G:\_\_\_

Switzerland vs. Cameroon: Switzerland wins with 0.60

Switzerland vs. Brazil: Brazil wins with 0.63

Switzerland vs. Serbia: Draw

Cameroon vs. Brazil: Brazil wins with 0.75

Cameroon vs. Serbia: Serbia wins with 0.65

Brazil vs. Serbia: Brazil wins with 0.56

\_\_\_Starting group D:\_\_\_

Denmark vs. Tunisia: Denmark wins with 0.55

Denmark vs. France: France wins with 0.59

Denmark vs. Australia: Denmark wins with 0.60

Tunisia vs. France: France wins with 0.68

Tunisia vs. Australia: Draw

France vs. Australia: France wins with 0.63

Expected Points

Starting group F:\_\_\_

Morocco : 1.6526449812661026

Croatia : 3.6146314412505074

Belgium : 5.76783915561168

Canada : 0.0

\_\_\_Starting group C:\_\_\_

Argentina : 4.286353942719144

Saudi Arabia : 0.0

Mexico : 3.196319202031786

Poland : 2.4812713690338812

\_\_\_Starting group A:\_\_\_

Senegal : 3.6079965458122643

Qatar : 0.0

Netherlands : 5.959668575161054

Ecuador : 1.7127215169764838

\_\_\_Starting group E:\_\_\_

Germany : 2.178141394621764

Japan : 0.972807670163196

Spain : 5.380953594716832

Costa Rica : 0.511359888856254

\_\_\_Starting group H:\_\_\_

Uruguay : 2.5899996526460924

South Korea : 2.3520097476957256

Portugal : 5.8162960019154815

Ghana : 0.0

\_\_\_Starting group B:\_\_\_

Iran : 0.0

England : 2.922749501335205

USA : 2.6840236954486825

Wales : 2.6832559153963644

\_\_\_Starting group G:\_\_\_

Switzerland : 2.2706012878512096

Cameroon : 0.0

Brazil : 5.810936160477053

Serbia : 2.482411222170502

\_\_\_Starting group D:\_\_\_

Denmark : 3.4485978500610264

Tunisia : 0.494587799222057

France : 5.696886533016041

Australia : 0.505412200777943

Group stage Survival probability

Senegal : 0.5238123555510896

Qatar : 0.28657994414763255

Netherlands : 0.7207123136586894

Ecuador : 0.4107822279220528

Iran : 0.36831897628115345

England : 0.5691708021782123

USA : 0.5126970678459446

Wales : 0.5323556070273117

Argentina : 0.6086463678095058

Saudi Arabia : 0.22215428639701668

Mexico : 0.5868558522665447

Poland : 0.5128567082594242

Denmark : 0.5151762204409138

Tunisia : 0.3721040502951258

France : 0.6893043301262857

Australia : 0.3877831877546305

Germany : 0.4893859548936812

Japan : 0.4109739077103775

Spain : 0.6419635779920858

Costa Rica : 0.43841533837914887

Morocco : 0.3819955705681707

Croatia : 0.5217642297484982

Belgium : 0.7041053684174787

Canada : 0.3482241700140098

Switzerland : 0.45510860867744996

Cameroon : 0.25700333565836253

Brazil : 0.6878946534259283

Serbia : 0.5383228703970194

Uruguay : 0.5313857116348

South Korea : 0.45081127452719727

Portugal : 0.7025008859381351

Ghana : 0.24790543284071997

**CROSS\_VALIDATION for Random forest**

Cross-validation is a resampling technique used in machine learning and statistical modeling to assess the performance and generalization ability of a predictive model.

The basic idea behind cross-validation is to split the available data into multiple subsets, typically referred to as "folds." The model is trained on a portion of the data (known as the training set) and evaluated on the remaining data (known as the validation or test set). This process is repeated multiple times, with different subsets of the data serving as the validation set each time.

The most common type of cross-validation is k-fold cross-validation, where the data is divided into k equally sized folds. The model is trained on k-1 folds and evaluated on the remaining fold. This process is repeated k times, each time using a different fold as the validation set. The performance metrics obtained from each iteration are then averaged to give an overall estimation of the model's performance.

The benefits of cross-validation include:

1. More reliable performance estimation: Cross-validation provides a more robust estimate of a model's performance compared to a single train-test split. It helps to reduce the impact of data variability and provides a better understanding of how the model generalizes to unseen data.

2. Efficient use of data: By using all available data for training and validation, cross-validation allows for more efficient utilization of the dataset, which can be particularly useful when the dataset is small.

3. Model selection and hyperparameter tuning: Cross-validation is often used to compare different models or tune hyperparameters. By evaluating models on multiple validation sets, it helps in selecting the best-performing model or finding the optimal combination of hyperparameters.

If I run cat("Bias:", bias, "\n") I get Bias: -66.76265  
this means : A negative value for bias suggests that the cross-validated model is performing worse than the model trained without cross-validation. This is unexpected since cross-validation is typically expected to improve or provide a similar estimate of the model's performance.

In this case, the negative bias value of -66.76265 indicates that the cross-validated random forest model has significantly lower accuracy compared to the random forest model trained without cross-validation (which had an accuracy of 67.41% as per your earlier result).

The negative bias value could be a result of various factors, such as issues with the data, model selection, or the cross-validation process itself. Here are a few possibilities to consider:

1. Data issues: It's possible that the data used for cross-validation differs in some way from the data used to train the original random forest model. For example, if there are missing values or differences in data preprocessing between the two cases, it could lead to differences in performance.

2. Model selection: It's important to ensure that the same hyperparameters and settings are used for both the cross-validated model and the original model. Any differences in model configuration can lead to different performance results.

3. Cross-validation process: The choice of the number of folds (`k`) or the random splitting of the data during cross-validation can affect the results. You might try different values of `k` or experiment with different random seeds to see if the bias value changes.

If I now run : cat("Variance:", variance, "\n") I get

Variance: 0.0003822415

This means : A variance value of 0.0003822415 indicates very low variability in the accuracy values obtained from the cross-validation process. This suggests that the model's performance is relatively consistent across different folds of the cross-validation.

A low variance in the accuracy values is generally desirable as it indicates that the model is stable and not overly sensitive to the specific training and validation subsets used during cross-validation. It suggests that the model's performance is reliable and can be generalized to unseen data.

In your case, the low variance value suggests that the random forest model's accuracy is consistent across the different folds of the cross-validation process. This is a positive outcome, as it indicates that the model is not heavily influenced by the particular training and validation subsets used during cross-validation.

It's important to note that the interpretation of the variance value should be considered in conjunction with the bias value. While a low variance is desirable, the negative bias you obtained suggests that the cross-validated model's overall performance is worse than the model trained without cross-validation.

**CROSS\_VALIDATION for logistic regression**

cat("Bias:", bias, "\n")

Bias: -67.34309

This means : A bias value of -67.34309 suggests that the cross-validated logistic regression model performs significantly worse than the logistic regression model trained without cross-validation. The negative bias indicates that the cross-validated model's accuracy is much lower than the accuracy obtained from the non-cross-validated model (which had an accuracy of acc\_log). A negative bias is typically unexpected and suggests that there may be issues with the cross-validated model's performance.

**Logistic Regression ROC curve**

The Receiver Operating Characteristic (ROC) curve is a graphical representation of the performance of a binary classification model. It illustrates the trade-off between the true positive rate (TPR) and the false positive rate (FPR) at different classification thresholds.

The ROC curve is created by plotting the TPR on the y-axis against the FPR on the x-axis. Each point on the curve represents a different threshold used for classifying the positive and negative classes. By varying the threshold, you can adjust the balance between correctly identifying positive instances (sensitivity) and incorrectly classifying negative instances (1 - specificity).

The diagonal line from (0,0) to (1,1) in the ROC plot represents a random classifier with no predictive power. The goal is for the ROC curve to be as close as possible to the top-left corner of the plot, which indicates high sensitivity and low FPR. A perfect classifier would have an ROC curve that coincides with the top-left corner (TPR=1, FPR=0).

The Area Under the ROC Curve (AUC) is a single metric that quantifies the performance of the classifier. The AUC ranges from 0 to 1, with a higher value indicating better discrimination between the positive and negative classes. An AUC of 0.5 suggests random guessing, while an AUC of 1 represents a perfect classifier.

In summary, the ROC curve provides a visual representation of the performance of a binary classification model, showcasing the trade-off between true positive rate and false positive rate. The AUC serves as a summary measure of the classifier's discriminatory power, with higher values indicating better performance.

**print(roc\_obj)**

An Area Under the Curve (AUC) value of 0.7541 indicates the performance of the classifier in terms of its ability to discriminate between the positive and negative classes. The AUC ranges from 0 to 1, with higher values indicating better discrimination.

in your case, an AUC of 0.7541 suggests that the logistic regression model has moderate discriminatory power. It performs better than a random classifier (AUC = 0.5) but may have room for improvement to achieve higher accuracy.

The AUC value is a useful metric for comparing different models or assessing the overall performance of a classifier. It provides a summary measure of the ROC curve, considering the classifier's performance across all possible classification thresholds.

Remember that the interpretation of AUC values can vary depending on the specific problem domain and the desired level of performance. It is advisable to consider other evaluation metrics and domain-specific considerations to gain a comprehensive understanding of the model's performance.