Clustering and Time Series Modelling of Sports Gambling Odds Movement

Aaron Foote

1 Abstract

In recent years, sports gambling has exploded in popularity in the United States. Since the reversal of the Professional and Amateur Sports Protection Act in 2018 that prohibited sports gambling on collegiate and professional sports outside of Nevada, over two-thirds of the states in the US have legalized gambling, leading to millions of dollars in monthly tax revenue for states. The goal of this project is to study odds movement to help bettors extract maximum value from the bets they plan on placing. By clustering the odds offerings and modeling each cluster using time series analysis, a bettor can identify when to place their bet so that they know they are receiving the best odds possible for that event.

2 Introduction

At its simplest, sports gambling involves predicting outcomes of sporting events and staking money on these predictions. Sportsbooks serve to standardize this process. Traditionally, a casino has a sportsbook section where a gambler can place wagers on odds that the casino offers. Nowadays, online sportsbooks such as Fanduel and DraftKings instead provide an app that serves as an online hub where a bettor can place their bets. Thus, rather than two friends making a wager between each other, all bettors using a specific sportsbook will be offered the same odds (those odds may change over time!). Typically, a bettor will have their own opinion on the actual probability and bet according to the difference. What makes the betting process complex is that sportsbooks have a bevy of strategies to tilt the scales in their

favor, ensuring that it is virtually impossible to make a profit in the long run. This paper will outline methodologies that inform betting strategies, using clustering algorithms and time series analysis to do so.

The most basic strategy for this is the inclusion of "juice" in all odds that a sportsbook posts. A simple coin flip perfectly illustrates this concept. The probability of either outcome is 0.5. If two friends were to wager on the outcomes of repeated flips, putting fifty cents on their chosen possibility, in the long run neither would have a net gain. A sportsbook, on the other hand, would force the bettor to bet each outcome as if they each had a probability of .52 or .53. Then, to make a dollar of profit, one would need to bet more than fifty cents, leading to a loss in the long run *regardless of choice*. The only scenario in which one would want to play this game is if they had insider information and knew they weren't actually getting losing odds. Staying ahead of a multi-billion dollar corporation full of employees that set odds for a living is a losing strategy.

Thus, the only hope to win consistently is to find systematic errors or patterns in the sports betting lines. Note that this departs from the traditional strategy of predicting outcomes and instead focuses instead on analyzing the process of sportsbook line setting itself. One clever strategy that has been employed to consistent success is arbitrage betting, where specific differences in odds for the same event on different allow a bettor to bet both sides of an event to create a zero-risk bet.

In the pursuit of finding consistently winning sports gambling strategies, Kreiner et al. study how outlier sportsbooks might be able to be exploited to create a consistent profit. To find the accepted probability of an event, the odds of a given event were gathered across various sportsbooks and aggregated to obtain a consensus. From there, the books that deviated the most from that consensus could be used to get the maximum value out of placed bets. Their research proved their strategy to be successful over simulations that lasted multiple years. Note how the process was *not* to predict outcomes, but instead to leverage the remarkable predictive power of the sportsbooks to find where errors are systematically made.

This project aims to identify patterns in line movement (changes in odds over time) to allow bettors to know the ideal time to place a bet. The dataset for this project is a collection of time series of odds movement over time for soccer matches. Consider Argentina playing France in the 2022 World Cup Final as an example of a match that might have a series in the dataset. For three days leading up to the match, both of the teams in the match have been

known. Thus, bettors have been able to place wagers on who they believe will win the match given the odds. However, over time the odds may shift, creating a time series ripe for forecasting and analysis. A bettor interested in this match would like to know when to place their bet to get the maximum value, and this project aims to allow them to answer that exact question by forecasting how the odds will change up until the match.

To accomplish this, a collection of line movement time series were analyzed to find characteristics that led to similar movement, and then various models were fit for different types of behavior. The process of finding time series that had similar movements was accomplished through clustering. With clusters in hand, models can be fit for each cluster. With effective models, the bettor's question is now reduced to the task of identifying which cluster a given match belongs to. Some basic demographics such as the distribution of home vs. away of the clusters are studied, as well as how the most popular leagues distribute across the clusters.

3 Methods

3.1 Data

The dataset used in this project was compiled by Elad Silvas and Pini Gabai and published to Kaggle [Sil]. Compiled from December 2016 through May 2018, the dataset includes over 30,000 soccer matches, and for each, there are a series of observations of the odds of the home team winning, the away team winning, and tie odds over time. While these odds changes are the most useful variables in the dataset, each match includes the name of the home and away teams, the name of the league/competition in which the match occurred, a timestamp for each observation, and the time at which the match started. The match odds are taken from matches in all kinds of leagues, ranging from high-profile European leagues to obscure amateur leagues all over the world. Below is a table describing each variable in the original dataset.

Variable	Description
match id	Unique integer identifier for each match
date start	Time at which the match starts
competition name	Name of the league in which the match is contested
date created	Exact time (to the minute) that the bookmaker changed the odds for this observation
home team name	Name of home team
away team name	Name of away team
home team odd	Home team's odds for winning at a specific time (decimal odds)
away team odd	Away team's odds for winning at a specific time (decimal odds)
tie odd	Tie odds at a specific time (decimal odds)

The goals of the project were the main guiding forces when considering how to format the data. To start, attributes that could categorize the series were explored. As predicting the maximum and minimum of an odds series is a goal of the project, the series were split into the following four types:

1. Maximum and Minimum are endpoints (type 1)
2. Maximum is within series, minimum is an endpoint (type 2)
3. Minimum is within series, maximum is an endpoint (type 3)
4. Maximum and minimum are within the series (type 4)

For the original dataset, each observation is one-time step in one match, meaning that the matches are stacked on top of one another with the relevant time series as variables. To cluster, however, each time series must be an observation (each series becomes a row), with all of the series stacked in a dataframe. One difficulty when formatting the dataframe for clustering was that each time series is of a different length. To handle this, NAs were appended to the end of each series until they were all the same length and thus could all be bound.

3.2 Clustering

Before clustering can occur, one must calculate the pairwise distances between the objects to be clustered. However, a wrench thrown into the clustering was the unequal length of the odds time series. With some series

having close to 200 observations and some with less than ten, it was necessary to filter out time series without enough observations. The dataset that was eventually clustered included as many of the longest time series that could be included without the memory of the computer being overwhelmed later on during the computationally expensive portions of the code. Then, with the remaining series, NAs were patched onto the end of each series until they were all of the same length.

Since the goal of the project is to train models on time series and use those models, it is necessary to have the centers of the clusters be objects in the dataset rather than just the average of the members of the cluster. Thus, k-means clustering is not a reasonable option and k-medoids clustering is the choice of algorithm. More specifically, the PAM algorithm was used to identify the clusters.

3.2.1 Gower Distance

The Gower distance metric is different from traditional similarity measures in that it does not break down when applied to vectors of non-numeric features. Since the time series in this project are of unequal length, NA values were added on to the end of the series until they were all the same length. Thus, when calculating the similarity measure between two series, the algorithm must be able to handle NA values. When comparing two time series i and j, the formula for their similarity is: $\frac{\sum_{k=1}^{p} s_{ijk} \cdot \delta_{ijk}}{\sum_{k=1}^{p} \delta_{ijk}}$. For a feature k, if the series cannot be compared at this feature (if that feature is NA in one or both series), δ_{ijk} is set to zero. Thus, that feature provides zero weight to the similarity of i and j. Otherwise, $\delta_{ijk} = 1$, and since the only type of variable considered in this case are numeric ones (similarity of NAs not considered), $s_{ijk} = 1 - \frac{|x_{ik} - x_{jk}|}{R_k}$. R_k is the range of the feature k in the sample [Gow71].

3.2.2 PAM Algorithm

The partition around medoids (PAM) algorithm is a two-phase algorithm widely used to perform k-medoids clustering. The goal of the algorithm is to identify k objects (time series in this case) that are centrally located in clusters in the overall dataset. In a sentence, it does this by minimizing the average dissimilarity (calculated distance) of objects to their closest selected objects.

The first phase of the algorithm is the building phase, which is done in such a way as to ensure a relatively high-quality initial set of medoids [Hel]. In this phase, initially, the object that minimizes the sum of distances to all other objects is added to the medoid set M. This is the most central object in the whole dataset. Then, M is built up iteratively. For each object that is not a medoid, the sum of distances from all other points is calculated, and the one with the smallest cost is added to the set.

Once M has been built up to k elements, the swap phase occurs. In this phase, a medoid is considered to be switched with every object on in M. For each considered swapping, the total cost to all other points is calculated, and if any of those total costs are less than the medoid total cost, a swap occurs. This process continues until no more improvements can be made.

3.3 Methodology

The time series to which models were fit in this project were the medoids of the clusters found by the PAM algorithm. Before clustering can occur, it is necessary to determine how many clusters are present. A silhouette plot was used to visually report how similar points are within their cluster relative to their similarity to points in other clusters. The number of clusters used is the point at which this is maximized.

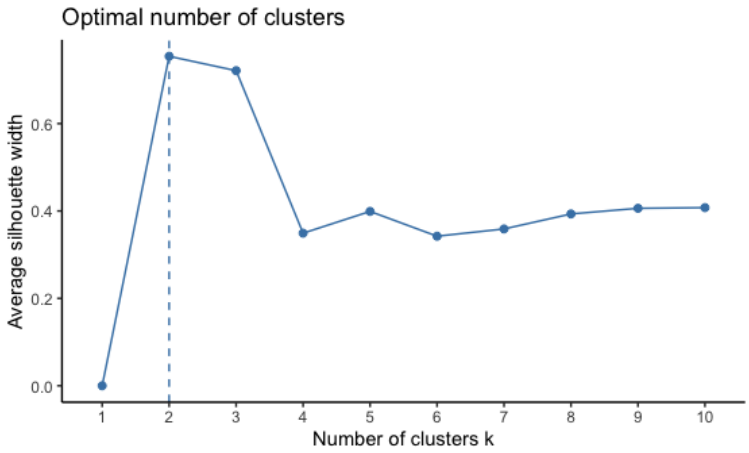


Figure 1: Silhouette Plot

From the plot, it is clear (and even automatically labeled) that the best

number of clusters is $k = 2$. Additionally, the PAM algorithm was run with $k = 4$ to test whether the types were reflected in the clustering.

With the clusters found and medoids extracted, models were then able to be fit. As a first step, naive, mean, and drift models were fit to each medoid to establish baseline performance to which the more complex models could be compared. For each medoid, ARIMA, ARFIMA, and NNETAR models were fit.

3.3.1 ARIMA

The ARIMA model is defined by parameters p, d, and q. The parameter p is the number of autoregressive (lagged values) to include in the model, d is the degree of differencing, and q is the number of moving average (error) terms to include in the model. Thus, an ARIMA$(p, 1, q)$ model for variable y would have the following equation:

$$Y_t = c + \varepsilon_t + \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + Y_{t-1} + \theta_1 \varepsilon_{t-1} + \cdots + \theta q \varepsilon_{t-q}$$

The AIC can determine the values of p and q that lead to the ideal model. Choosing values for p and q is also done by examining the ACF and PACF plots of the data and those plots for the residuals after fitting models.

3.3.2 ARFIMA

The ARFIMA model is quite similar to the ARIMA model and is defined by parameters p, d, and q.The parameter p is the number of autoregressive (lagged values) to include in the model and q is the number of moving average (error) terms to include in the model. The difference between the ARIMA and ARFIMA model is that d is a real number between 0 and 0.5 rather than one or two for an ARIMA. This change makes the ARFIMA more of a long memory model that can account for a trend that tapers off gradually (ACF plot will decrease slowly rather than decaying quickly). Thus, an ARFIMA(p, d, q) model for variable y would have the following equation:

$$Y_t = c + \varepsilon_t + \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + d \cdot Y_{t-1} + \theta_1 \varepsilon_{t-1} + \cdots + \theta q \varepsilon_{t-q}$$

The AIC can determine the values of p and q that lead to the ideal model.

3.3.3 NNETAR

The NNETAR model is defined by parameters p and k. Just as with ARIMA models, p is the number of autoregressive terms to include, but in this case that will determine the number of nodes in the input layer of the network. The parameter k is the number of nodes in the hidden layer. Additionally, there is by default one seasonal AR term (assign parameter P to change). Since the model does not produce an identical outcome each time, even with the same parameters, the model is by default trained at least 20 times, and results are aggregated to determine the weights of the nodes that minimize AIC.

With the best model within each class determined by AIC, the RMSE of each model is calculated on test data to find which of the classes produces the best models. To test the model for each medoid, two other series are selected from the cluster to which the medoid belongs, and RMSE is calculated for the models on those two series. One of the series chosen is the one that is the farthest from the medoid that is still in the cluster, and the other is a series that has a small distance from the medoid relative to the other cluster members.

To summarize the workflow, first, the series are clustered using the PAM algorithm and Gower distance. Then, the medoids are identified and a slew of models is fit. Once the models are fit, they are tested on two other members of the cluster to evaluate the quality of fit. From there, the best model for each cluster can be determined. All that is left to do is explore what leads to a series being put in a particular cluster. The distribution of home and away series across clusters is examined, as is how the most popular leagues in the dataset distribute across the clusters.

4 Results

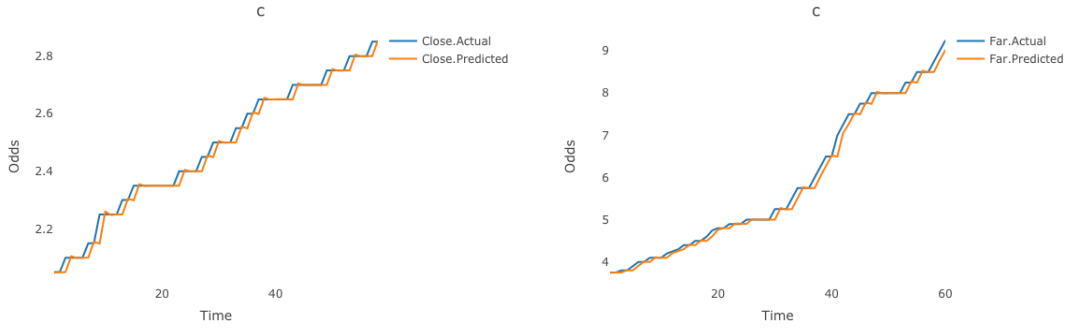
Refer to the Appendix for a full table with the parameters and RMSE for every model fit to every cluster on the far and close testing time series.

4.1 Cluster 1 of k = 2

When the data is split into two clusters, the best model for the first cluster is an ARIMA model with $p = 0$, $d = 1$, and $q = 2$. The functional form of this model is:

$$Y_t = c + \varepsilon_t + Y_{t-1} + 0.099\varepsilon_{t-1} - 0.009\varepsilon_{t-2}$$

Plotted below are the close and far cluster members along with the values forecasted by the model above.

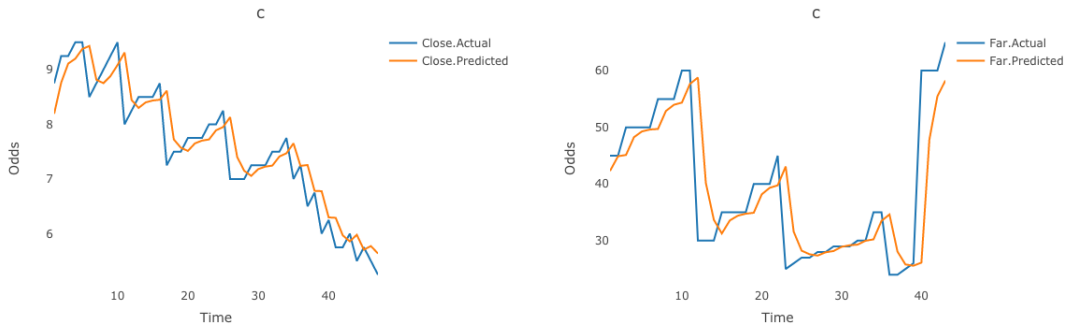


4.2 Cluster 2 of k = 2

For the second cluster, an ARFIMA model with $p = 1$, $q = 1$, and $d = 0.121$ performed best. The functional form of this model is:

$$Y_t = c + \varepsilon_t + 0.961Y_{t-1} + 0.121Y_{t-1} + 0.449\varepsilon_{t-1}$$

Plotted below are the close and far cluster members and the values forecasted by the above model.

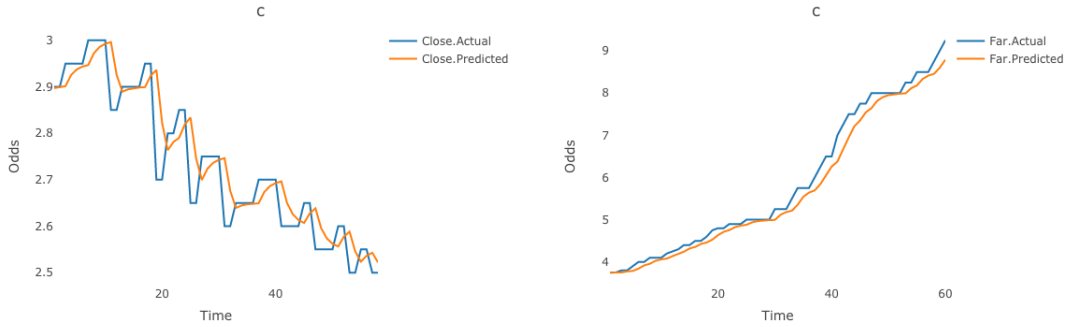


4.3 Cluster 1 of k = 4

For the first cluster of the four cluster splitting, an ARIMA model with $p = 0$, $q = 1$, and $d = 1$ performed best. The functional form of this model is:

$$Y_t = c + \varepsilon_t + Y_{t-1} - 0.519\varepsilon_{t-1}$$

Plotted below are the close and far cluster members and the values forecasted by the above model.

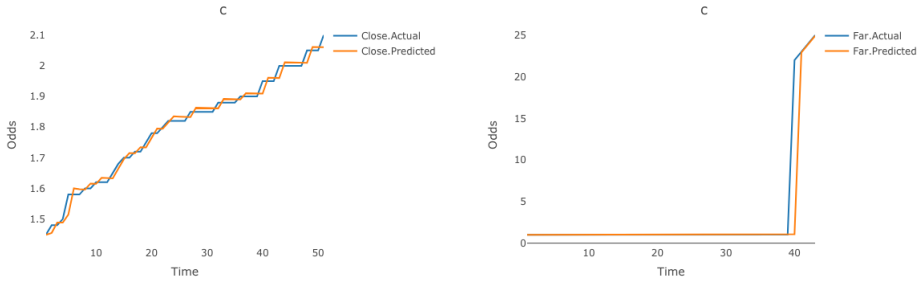


4.4 Cluster 2 of k = 4

For the second cluster, an ARIMA model with $p = 1$, $q = 1$, and $d = 1$ performed best. The functional form of this model is:

$$Y_t = c + \varepsilon_t + 0.9979Y_{t-1} + Y_{t-1} - 0.957\varepsilon_{t-1}$$

Plotted below are the close and far cluster members and the values forecasted by the above model.

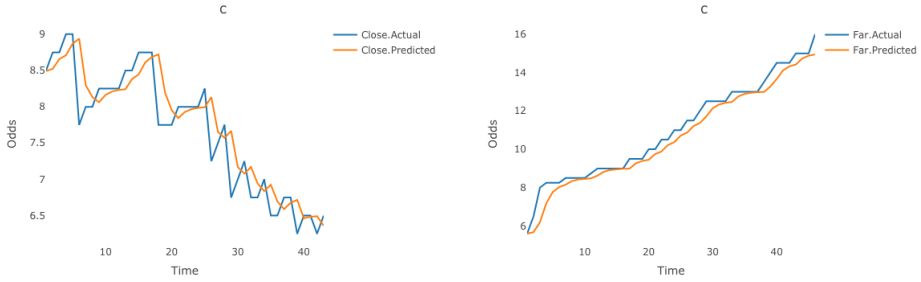


4.5 Cluster 3 of k = 4

For the third cluster, an ARIMA model with $p = 0$, $q = 1$, and $d = 1$ performed best. The functional form of this model is:

$$Y_t = c + \varepsilon_t + Y_{t-1} - 0.458\varepsilon_{t-1}$$

Plotted below are the close and far cluster members and the values forecasted by the above model.

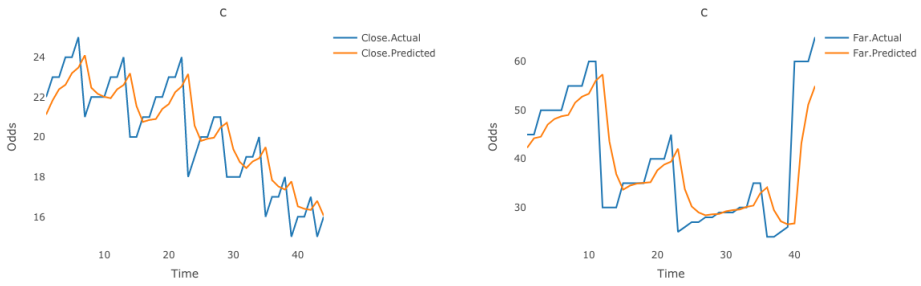


4.6 Cluster 4 of k = 4

For the fourth cluster, an ARFIMA model with $p = 1$, $q = 1$, and $d = 0.00005$ performed best. The functional form of this model is:

$$Y_t = c + \varepsilon_t + 0.964Y_{t-1} + 0.00005Y_{t-1} + 0.484\varepsilon_{t-1}$$

Plotted below are the close and far cluster members along with the values forecasted by the above model.



4.7 Demographics

Below is the distribution of home and away teams across the two clusters, along with how matches played in the Europa League (the most frequent league in the clustered data) is distributed across the clusters. The number in parentheses is the proportion of time series that falls into the given group. For reference, 88% of the time series fall into the first cluster and 12% fall into the second.

	Cluster 1	Cluster 2
Home	376 (0.945)	22 (0.055)
Away	325 (0.817)	73 (0.183)

Europa League	
Cluster 1	Cluster 2
40 (0.952)	2 (0.048)

Just as for the k=2 case, below is the distribution of home and away teams across the four clusters, along with how matches played in the Europa League (the most frequent league in the clustered data) is distributed across the clusters. The number in parentheses is the proportion of the time series that falls into the given group. For reference, 53% of the time series fall into the first cluster, 35% into the second, 10% into the third, and the final 2% into the fourth cluster.

	Cluster 1	Cluster 2
Home	229 (0.575)	97 (0.244)
Away	193 (0.485)	183 (0.460)
	Cluster 3	Cluster 4
Home	59 (0.148)	13 (0.033)
Away	19 (0.048)	3 (0.008)

Europa League	
Cluster 1	Cluster 2
26 (0.62)	14 (0.33)
Cluster 3	Cluster 4
2 (0.048)	0 (0.0)

On the next page are two tables comparing the types distribution across the four clusters and two clusters.

Table 1: Four Clusters

	C1	C2	C3	C4
Type 1	456	144	144	40
Type 2	508	428	52	12
Type 3	280	196	52	4
Type 4	444	352	64	8

Table 2: Two Clusters

	Cluster 1	Cluster 2
Type 1	600	184
Type 2	936	64
Type 3	472	60
Type 4	796	72

5 Discussion

One limitation of the project was the sheer computing power available to the systems used for data management. With over 30,000 matches played across more than 500 leagues, there was incredible potential for a robust and wide-reaching model. However, to prevent R from crashing, less than 5,000 were able to be used.

The two most necessary next steps are strongly connected. First, to fully answer the bettor's question from the introduction, the clusters must be analyzed further. To accomplish this, additional demographic information will be collected. One particularly intriguing piece of information is the popularity of the league in which the match is played. The less popular leagues that get less betting action might have simpler underlying patterns that can be leveraged as compared to the more popular leagues where those simpler patterns have already been taken advantage of by savvy bettors. After collecting various demographics on the matches, the next step would be to evaluate if there is a significant difference in how that demographic distributes among the clusters as compared to the typical distribution. This sort of analysis screams hypothesis testing.

From a cursory examination of the demographics collected this far, it appears that home teams might fall into cluster 1 of k=2 at a higher rate than cluster 2, and the opposite for away teams. Further, it appears that the types created during the data management phase are not meaningful. When their distribution across two clusters, it is clear that all types fall predominantly into the first cluster, with a handful of each type falling into the second.

Although in no way would I feel comfortable advising a prospective gambler on what picks to make using this model, some insightful and applicable steps have certainly been taken. A non-obvious use of this model would be to determine whether a bet would even be worth considering. For instance,

the RMSE of every model for the final cluster of the four and two cluster methods perform worse than the other clusters. Thus, if considering placing a bet that is determined to fall into the fourth cluster, it would be worthwhile to consider a different bet, since the other clusters have much more success modeling the behavior of bets in their cluster. Lastly, it should be noted that the model is still useful even if it is not always correct. To consistently make a profit in sports gambling, one only needs to be correct at a slightly higher rate than the sportsbooks.

6 Appendix

Model	Parameters	RMSE Close	RMSE Far
Mean	NA	0.250	3.901
Naive	NA	0.317	3.621
Drift	NA	0.604	3.165
ARIMA	p=0,d=1,q=2	0.028	0.143
ARFIMA	p=0 ,d=0.490,q=2	0.054	0.279
NNETAR	p=3,k=2	0.025	0.162

Table 3: Cluster 1 (k = 2) Models

Model	Parameters	RMSE Close	RMSE Far
Mean	NA	1.181	34.062
Naive	NA	1.598	35.157
Drift	NA	2.308	36.089
ARIMA	p=0,d=1,q=0	0.481	8.046
ARFIMA	p=1,d=0.121,q=1	0.463	8.358
NNETAR	p=2,k=2	0.500	8.439

Table 4: Cluster 2 (k = 2) Models

Model	Parameters	RMSE Close	RMSE Far
Mean	NA	0.169	3.534
Naive	NA	0.206	3.709
Drift	NA	0.385	4.000
ARIMA	p=0,d=1,q=1	0.063	0.238
ARFIMA	p=1,d=.00004,q=1	0.065	0.334
NNETAR	p=3,k=2	0.072	0.221

Table 5: Cluster 1 (k = 4) Models

Model	Parameters	RMSE Close	RMSE Far
Mean	NA	0.170	6.674
Naive	NA	0.226	6.641
Drift	NA	0.401	6.549
ARIMA	p=1,d=1,q=1	0.019	3.192
ARFIMA	p=0,d=0.496,q=1	0.051	3.695
NNETAR	p=3,k=2	0.019	3.282

Table 6: Cluster 2 (k = 4) Models

Model	Parameters	RMSE Close	RMSE Far
Mean	NA	0.866	4.361
Naive	NA	2.099	5.833
Drift	NA	3.225	7.395
ARIMA	p=0,d=1,q=1	0.365	0.524
ARFIMA	p=1,d=0.00004,q=1	0.366	0.558
NNETAR	p=1,k=1	0.377	0.802

Table 7: Cluster 3 (k = 4) Models

Model	Parameters	RMSE Close	RMSE Far
Mean	NA	3.334	21.346
Naive	NA	3.516	24.754
Drift	NA	5.471	27.232
ARIMA	p=0,d=1,q=3	3.097	9.859
ARFIMA	p=1,d=0.00005,q=1	1.556	8.768
NNETAR	p=2,k=2	1.587	8.353

Table 8: Cluster 4 (k = 4) Models

References

[Gow71] J.C. Gower. "A General Coefficient of Similarity and Some of Its Properties". In: *Biometrics* 27.4 (1971), pp. 857–871. DOI: https://doi.org/10.2307/2528823.

[Hel] Martin Helm. *A Deep Dive into Partitioning Around Medoids*. URL: https://towardsdatascience.com/a-deep-dive-into-partitioning-around-medoids-a77d9b888881. (accessed: 11.20.2022).

[Sil] Elad Silvas. *Football Matches Odds*. URL: https://www.kaggle.com/datasets/eladsil/football-games-odds?select=Matches_Odds.csv. (accessed: 10.04.2022).