Abstract

Data analytics have changed the sport industry and NBA is not exception. NBA teams are hiring statisticians in order they help to select players using Money ball techniques. We will try on a role of NBA data analyst aiming to predict the outcomes of NBA regular season games by building a binary classification models.In addition to this, we will try to predict the shots of Lebron James and game results of Lebron teams by considering only his performance.

1 Main research aim and framework

We have 3 main goals in this Project; Predicting the result of NBA games. Predicting the outcome of Field goal attempts of Lebron James. Predicting game results of Lebron teams by considering only his performance. Statistical tools and approaches are widely used in any sports because they provide accurate and useful information. Especially for NBA, where there are lots of independent variables that can effect game results, or even the whole season. NBA has a lot predictable and unpredictable aspects. The season itself is dynamic during the year in terms of transfers, injuries, coach changes,changing performance of the players etc. Some games are easy to predict, some are impossible. For that reason, it is very interesting work on NBA data.

2 Game Result Prediction

2.1 Data representation

We have collected the seasons data from 2012-13 season to 2018-19 season. Each season consist of 1230 games. After some discussion, we decided to represent each game twice: one time for home team and second time for away team. Example: we have features which represent the id of teams and id of the games: Team-id-x,Team-id-y,Game-id. Lets take a look at one example, suppose we have two Teams-id:1,2 and Game-id 3 then in our data it will be represented like Team-id-x=1,Team-id-y=2,Game-id=3,other features which describes Team 1 the next row will be Team-id-x=2,Team-id-y=1,game-id=3,other features which describes Team 2. In this representation we are predicting the win probability for Team_x. A such way makes our data balanced so we have 50% labels of class 0 and the same of class 1, thus we can use accuracy as a metric.

2.2 Feature engineering

2.2.1 Players statistics

One of our main goal was to aggregate information contained in the table of players. This table consists of statistics about players such as number of blocks, number of 2-point goals etc. gathered for each player in one particular game. There are more than 20 such statistics and we took 16 among which is the

number of minutes played by player one part. The number of minutes played was used to normalize statistics by dividing each of them. We did it because there might be an occasion when a player played few minutes and made a few successful shots, so his field goal percentage (FG_PCT) become high.

Since there is data for each player for each game we have to somehow to aggregate it. We tried 4 different ways to aggregate the data considering the data as time series of player performance: averaging player performance of N last games, time series statistics for N consecutive games, k-means clustering dimensionality reduction, local linear embeddings as player representation. These 4 different datasets were concatenated with other features described below. A few models were tested on them and major vote were performed which we will portray later.

2.2.2 Averaging statistics of players

The simplest idea of aggregating the data is to just take an average of player's statistics of N last games. In order to find an optimal N, we trained and performed prediction for N = [10 ... 70]. It turned out that N=60 is a good choice for our task. Thus, for each player, we calculated averages for each performance statistics of the last 60 games. After that, players grouped by a team and, once again, averages were calculated, this time, for teams.

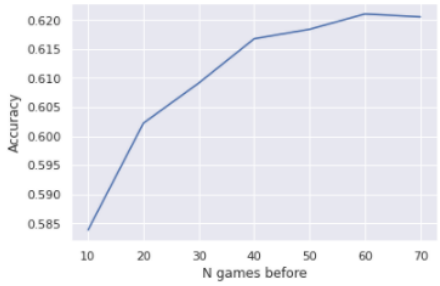

Figure 1: Optimal number of time series

2.2.3 Time series statistics of 60 last game

Besides taking a mean of the last 60 games, we can calculate other statistics such as standard deviation, skewness, median, minimum, maximum, slope. For each players we got 7 * 16 = 112 features which, then, were averaged by teams to get the same features for teams.

2.2.4 Local linear embeddings

Local linear embedding (LLE) is method of manifold learning - non-linear dimensionality reduction [1]. We assumed that performance by each player over

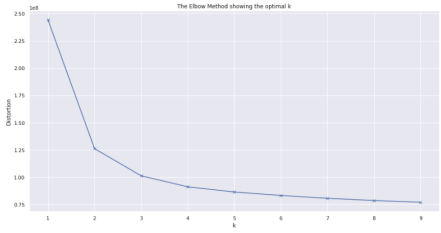

Figure 2: Optimal number of clusters

time can be represented as vectors which made by LLE method since it preserves distances within local neighborhoods [1]. Our assumption was if data is lying on some manifold, it can be unfold to lower space (via LLE) where we can work with it as linear vectors. Thus, we might get representation of a team as sum of players vectors. Thus, in order to represent team performance we can just add those vectors. In this case we didn't take the last 60 games but we made linear embeddings for each player based on previous season (82 games). We represent each player as time series of 82 points for each statistics (BLK, FG3, etc.) and made dimensionality reduction via Local Linear Embedding. In a such way, we obtained recuded vector for each player which we, then, summed up to get representation of a team as a vector. This approach has a few drawbacks since new players can come to the league and, also, players performance varies over seasons.

2.2.5 K-means dimensionality reduction

Instead of averaging directly players performance as in case of 2.2.2, we can, first, reduce dimension and, then, find a mean of reduced vectors. From [4], we know that players can be clusterized using k-means. Based on that and, in order to reduce dimensionality, we can learn clusters on the first season and then apply transformation to cluster-distance space, where in the new space, each dimension is the distance to the cluster centers [2]. Using elbow method Fig. 2, we found out that 3 is a good choice for number of clusters which can represent 3 main positions: Center, Forward, Guard. Players can play mixed position for example Power Forward, Small Forward [3] which, in our case, will be represented in different distances to centroids and, when we trained models it turned out that 5 clusters are slightly better interact with other features. Hence, we have varying measure of players position on a court which can be averaged to get a measure of team as whole.

2.2.6 Inactive score feature

"inactive_score" is the feature we created which is a measure of injury effect on players. The score is calculated in the following way: $Score = D_i/D_r$. where - D_i number of days of recovering (severity of an injury could be expressed

in terms of how many days a player spent to recover), - D_r number of days from complete recovery. Since it changes over time we don't need this feature participate in any aggregation described above, thus it is just averaged by players in a team.

2.2.7 Team features

From NBA api, we could extract also some other features such as number of games team win/loose before. These features were showed to be very effective in predicting NBA results [4]. We normalized these features by dividing on number of games that occurred.

One more feature for a team is staff_changes_score which corresponds how the team affected by changes in team staff on particular date of a game. If changes in team coaches or managers were made long ago the score will be lesser than if it was a recent change. The formula is: $Score = W/D_c$, where W - is weight of a change, for example, Head coach = 1, Assitant coach = 0.7, General manager = 0.8. D_c - number of days from a change. The score accumulates during the season, so if a few changes were made scores are added.

2.2.8 Teams traveling

We know that each match in NBA is playing in the town of one of the teams, so we decided to collect pairwise distances between all towns of teams. Then with this information we decided assign zero for teams which were playing at home city and for teams who came to the town to play we assigned the distances between their town and town where the game take place. We also decided to add home lable feature which indicate of team are playing at home of away (not home town). Let's take a look at the histogram of the Dist_away feature. It is easy to see that there exists dependency of the winning the game and distance, basically we can see that teams who is playing at home wins more often, and we knew before this information, but the interesting part of the histogram is that it looks stable for all seasons and right tail have some distinguish between $WIN = 1$ and $WIN = 0$, that is why we decided to move on with this feature and make different statistics from this feature.

We also added the HOME_LABLE which shows which teams are playing at home city, but there are some teams from the same cities and in this case the HOME_LABLE indicates on which stadium the game took place. On figure 4 we can see the matches between the teams which are from the same cities hence as we can see for TEAM_ID_x equal to 1610612746 the Dist_away is always equal to zero but HOME_LABLE vary between zero and one which indicates on which stadium the game took place

To create more features we have found logical to collect the statistics of movements between cities for last 2/3/5/10/15/all games. As a statistics of such time series we chose to use mean, median,skewness,minimum,maximum,std. Actually it was discovered that for the models more informative statistics for 5/10/15/all

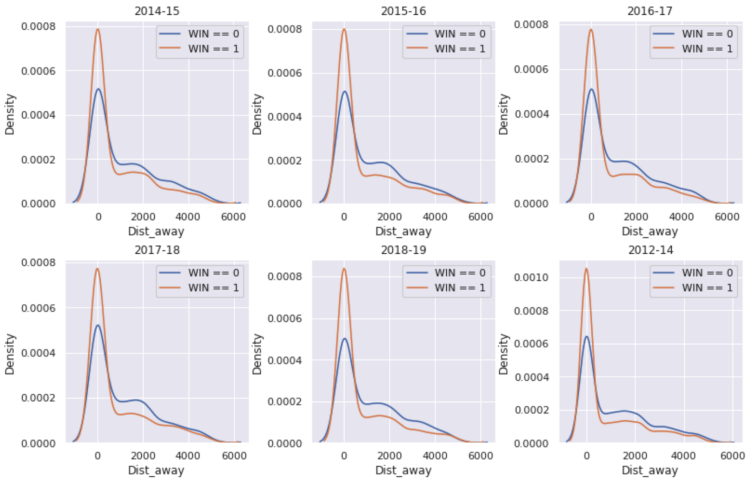

Figure 3: Histogram of the Dist away feature

	GAME_ID	TEAM_ID_x	TEAM_ID_y	Dist_away	HOME_LABLE
44	21200025	1610612746	1610612747	0.0	0.0
910	21200488	1610612746	1610612747	0.0	1.0
1468	21200788	1610612746	1610612747	0.0	0.0
2138	21201146	1610612746	1610612747	0.0	1.0
2298	21300003	1610612746	1610612747	0.0	0.0
3376	21300542	1610612746	1610612747	0.0	1.0
4124	21300916	1610612746	1610612747	0.0	0.0
4592	21301150	1610612746	1610612747	0.0	1.0

Figure 4: The game between teams from same city

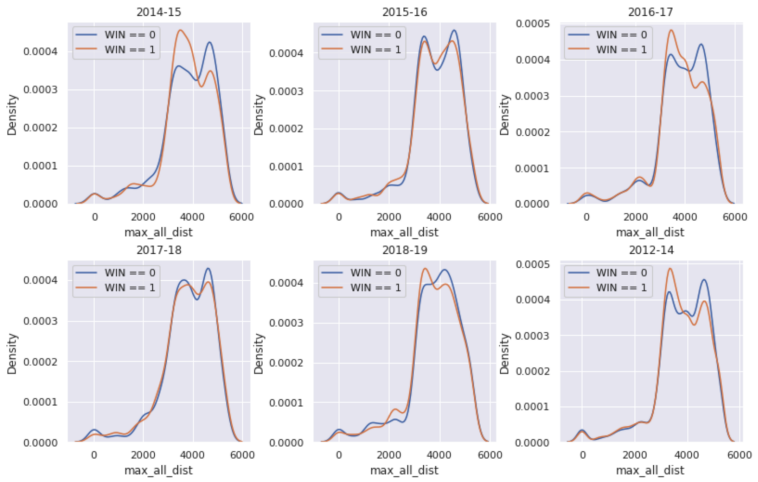

Figure 5: Histogram of the max_all_dist feature

games while 2/3 doesn't give any contribution (with respect to random forest feature importance selection).

The distribution of one interesting feature which we derived from distance away is maximum distance which were traveled by the team before and within the current game. On the figure 5 can be seen that team which traveled less before the game wins more often and this trend preserved for most seasons which we were predicted and also for training data for 2012-14 seasons.

We have found that such features gives increase in the accuracy and decided to proceed to work in such way and collect at this time same time series, but for days of the games before.The idea is pretty simple: for each team look back and see how many days ago team was playing the game and took the same statistics for last 2/3/5/10/15/all games. When we combined distance and days features we got around 0.6 accuracy which were already much better than the baseline model, also we have discovered that days features gives negligible increase in accuracy while features which were created from distance statistics gave and exact distance away feature gave much more.

We also tried to use kernelPCA with different kernels (rbf,sigmoid, linear,polynomial) but unfortunately this approach leverage the score, so we decided to move on with "uncompressed" feature space and perform feature selection later.

We also tried to collect the statistics for time between the cities, but we have discovered that it gave worse result then distance features and it is highly correlated with distance features, which is pretty obvious that is why we decided do not proceed with such features.

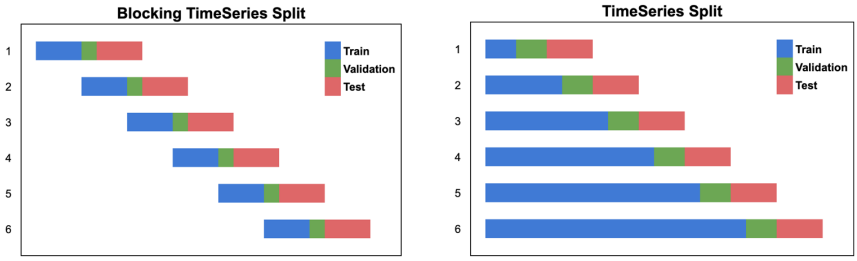

Figure 6: Blocking and Time Series splitting

2.3 Training of the model

There are a few ways of training and predicting that were explored by us. We tried to train using time series splits cross-validation (CV), blocked CV by one season so prediction and training are done in one season. Then, we explored a different approach when training is performed on one season and prediction on the another.

2.3.1 Training and prediction within one season

When applying blocked CV approach, we separate each season on N equal splits where each piece divided on 3 section train set, validation set and test set. In our solution, splits overlap so next train set is started from validation set Fig. 6. In addition , we did time series splitting when training set always starts from the beginning and validation moves N times further which, in turn, moves test set as well. From Fig. 7 and 8 we can clearly see that Time Series splitting strategy gives better results then Blocked splitting. It might be because we have 35 features and there is not enough data for blocked splitting in comparison with time series splitting. Moreover, it can be seen that linear regression in particular Ridge works much better than other methods. It worth to mention that , overall, linear methods (Ridge, LinearSVM, LDA etc.) perform better then non-linear. In addition, the dataset with the highest Ridge score consists of k-means features which might because of the size of features or that k-means in general works good here. The worst dataset is LLE features which is not surprising since we wrote that there are drawbacks in this approach.

2.3.2 Different season for training and prediction

In order to not use time series splitting or blocked splitting, one can take one season for training and the following season for prediction. In such way, we can use just regular cross-validation when training set is shuffled. Moreover, we can explore the idea of times series splitting and make training set increase in size by merging previous seasons. We tried both approaches individual season for training and multiple. Fig. 9 shows comparison between them. Just like in case

Averaging

Time Series splitting

Season	Ridge	LinearSVM	LogisticR	DecTree	Forest	NaiveB	QDA	LDA	SVM(rbf)
2014-15	69.60%	67.90%	68.20%	59.70%	66.90%	57.10%	66.10%	66.70%	65.00%
2015-16	70.40%	68.30%	68.70%	60.80%	68.20%	58.50%	65.90%	67.70%	64.20%
2016-17	64.80%	61.30%	61.80%	57.50%	62.80%	58.00%	62.60%	61.00%	58.60%
2017-18	66.50%	62.70%	64.90%	59.20%	64.70%	58.40%	63.10%	62.30%	58.10%
2018-19	66.70%	65.00%	64.90%	61.40%	64.40%	59.40%	64.20%	64.10%	59.30%
Total	**67.60%**	**65.00%**	**65.70%**	**59.70%**	**65.40%**	**58.30%**	**64.40%**	**64.40%**	**61.00%**

Blocked splitting

Season	Ridge	LinearSVM	LogisticR	DecTree	Forest	NaiveB	QDA	LDA	SVM(rbf)
2014-15	66.80%	63.70%	65.00%	57.80%	62.30%	58.10%	63.70%	60.70%	60.80%
2015-16	67.10%	63.80%	65.00%	56.40%	62.70%	58.70%	63.00%	59.80%	59.60%
2016-17	62.20%	58.60%	59.30%	55.20%	58.60%	57.90%	59.00%	56.40%	57.60%
2017-18	66.50%	63.40%	64.40%	58.90%	63.40%	57.20%	63.50%	59.50%	62.40%
2018-19	63.00%	60.60%	61.30%	54.60%	58.70%	57.90%	60.50%	58.10%	58.40%
Total	**65.10%**	**62.00%**	**63.00%**	**56.60%**	**61.20%**	**58.00%**	**61.90%**	**58.90%**	**59.70%**

K-means

Time Series splitting

Season	Ridge	LinearSVM	LogisticR	DecTree	Forest	NaiveB	QDA	LDA	SVM(rbf)
2014-15	69.20%	66.40%	66.50%	60.30%	66.90%	57.10%	63.70%	66.40%	63.50%
2015-16	70.80%	69.40%	69.20%	61.60%	68.90%	58.50%	63.70%	69.70%	64.70%
2016-17	64.20%	61.40%	61.80%	57.90%	60.90%	58.00%	61.20%	62.70%	59.80%
2017-18	68.10%	64.90%	66.00%	61.30%	65.10%	58.40%	61.30%	64.90%	61.00%
2018-19	67.80%	65.70%	66.10%	59.30%	64.30%	59.30%	62.50%	65.10%	62.40%
Total	**68.00%**	**65.50%**	**65.90%**	**60.10%**	**65.20%**	**58.30%**	**62.50%**	**65.80%**	**62.30%**

Blocked splitting

Season	Ridge	LinearSVM	LogisticR	DecTree	Forest	NaiveB	QDA	LDA	SVM(rbf)
2014-15	68.30%	65.90%	66.60%	58.60%	65.00%	58.30%	59.50%	65.30%	61.80%
2015-16	68.00%	65.60%	66.20%	56.50%	64.00%	58.70%	58.70%	63.60%	61.30%
2016-17	64.30%	61.70%	62.00%	53.70%	57.80%	58.00%	57.10%	60.40%	57.60%
2017-18	65.80%	62.90%	63.30%	57.20%	63.50%	57.20%	59.30%	61.90%	59.30%
2018-19	64.50%	61.70%	62.20%	55.80%	59.70%	58.00%	58.80%	59.40%	59.90%
Total	**66.20%**	**63.60%**	**64.10%**	**56.40%**	**62.00%**	**58.00%**	**58.70%**	**62.10%**	**60.00%**

Figure 7: Time series and blocked splitting comparison 1

Time Series statistics

Time Series splitting

Season	Ridge	LinearSVM	LogisticR	DecTree	Forest	NaiveB	QDA	LDA	SVM(rbf)
2014-15	70.20%	66.80%	67.30%	61.30%	66.90%	61.80%	65.00%	65.20%	60.00%
2015-16	69.10%	67.30%	67.50%	60.80%	67.10%	58.10%	64.00%	66.90%	64.20%
2016-17	65.00%	62.30%	63.10%	57.90%	63.80%	60.80%	62.60%	62.60%	59.70%
2017-18	67.80%	63.10%	64.70%	59.90%	64.80%	60.10%	61.90%	61.80%	59.20%
2018-19	67.10%	64.70%	65.90%	59.40%	64.60%	60.20%	62.20%	63.50%	59.30%
Total	**67.80%**	**64.80%**	**65.70%**	**59.90%**	**65.40%**	**60.20%**	**63.20%**	**64.00%**	**60.50%**

Blocked splitting

Season	Ridge	LinearSVM	LogisticR	DecTree	Forest	NaiveB	QDA	LDA	SVM(rbf)
2014-15	68.00%	65.00%	66.70%	56.60%	65.30%	60.20%	60.60%	60.80%	62.40%
2015-16	67.10%	64.10%	64.80%	57.00%	65.00%	58.70%	61.10%	59.30%	62.60%
2016-17	64.30%	61.30%	62.40%	57.00%	59.50%	59.80%	59.90%	57.90%	58.20%
2017-18	66.30%	63.50%	65.00%	55.70%	63.60%	59.00%	61.20%	59.90%	61.50%
2018-19	63.80%	61.40%	62.00%	56.70%	59.70%	57.50%	59.20%	58.00%	59.90%
Total	**65.90%**	**63.00%**	**64.20%**	**56.60%**	**62.60%**	**59.00%**	**60.40%**	**59.20%**	**60.90%**

Local Linear Embeddings

Time Series splitting

Season	Ridge	LinearSVM	LogisticR	DecTree	Forest	NaiveB	QDA	LDA	SVM(rbf)
2014-15	67.20%	64.40%	65.40%	59.80%	64.80%	62.20%	62.80%	64.40%	62.40%
2015-16	70.80%	68.80%	69.30%	59.70%	68.70%	64.00%	65.50%	68.20%	62.50%
2016-17	64.00%	61.00%	62.00%	57.00%	61.70%	58.90%	61.10%	60.70%	58.70%
2017-18	65.90%	62.40%	63.50%	59.90%	64.40%	60.00%	61.70%	62.20%	58.60%
2018-19	66.50%	63.00%	64.00%	61.00%	63.80%	62.00%	64.10%	62.80%	61.70%
Total	**66.90%**	**63.90%**	**64.90%**	**59.50%**	**64.60%**	**61.40%**	**63.10%**	**63.70%**	**60.80%**

Blocked splitting

Season	Ridge	LinearSVM	LogisticR	DecTree	Forest	NaiveB	QDA	LDA	SVM(rbf)
2014-15	65.80%	63.20%	64.40%	56.30%	61.80%	59.90%	63.50%	60.50%	59.40%
2015-16	66.60%	63.70%	65.30%	56.30%	62.00%	58.00%	63.00%	60.70%	61.60%
2016-17	62.10%	58.40%	58.90%	53.90%	58.20%	56.00%	59.80%	56.40%	55.40%
2017-18	63.90%	60.40%	61.40%	57.40%	61.20%	58.90%	61.00%	59.20%	59.40%
2018-19	64.80%	61.90%	62.40%	55.50%	61.20%	59.10%	62.60%	58.00%	60.80%
Total	**64.60%**	**61.50%**	**62.50%**	**55.90%**	**60.90%**	**58.40%**	**62.00%**	**59.00%**	**59.30%**

Figure 8: Time series and blocked splitting comparison 2

of prediction within one season, the best method with respect to the accuracy metric is Ridge. The results of training on consecutive seasons is slightly better. The same situation with comparison with linear and non-linear, we can see that Ridge outperformed other methods. Surprisingly, the best dataset in this case is "Time Series statistics" and, once again, the worst is one with LLE features but this time the difference is not so big.

2.4 Feature selection

In order to select features, we used Random Forest feature importance. We tried to select via Lasso coefficients but features selected by Random Forest gives better results. For all datasets, except one that contains k-means features, we took 35 top features. Since k-means is supposed to reduce dimensionality of our data, for k-means case, we took the top 20 features.

2.4.1 Majority vote

Majority vote can be effective for this type of data even with simple classifier [4]. Since we stored a lot of predictions from each dataset and each model, we might potentially use the majority vote technique to increase our accuracy. We tried to do it on all prediction vectors, but it doesn't work as we expected. After, some experiments, we decided to took models with the highest score. To be mentioned, we have different number of predictions in case of Time Series and blocked splitting and in case of prediction by seasons, thus we checked all cases and it turned out that time series splitting is the best for majority vote - it also has the best scores. There are 3 predictions of Ridge that gave us $>$ 0.67 accuracy so we took them. The correlation between them is less then 0.6 which is very good for this task Fig. 10. The predictions were averaged and threshold of 0.5 was chosen, since the values can be 0, 0.33, 0.66 or 1 and we need majority which corresponds to 0.66 or 1. In a such way, we obtained the accuracy of 69,08%.

2.5 Feature analysis

Here, we will show features only for the dataset with averaging of N previous games players statistics since it's more describable in terms of players information. From Fig. 11, it can be clearly seen that two dominant features are rate of wins of team 1 and team 2. They outperformed other features which is not surprising since it indicates the strength of team. Moreover, there is a correlation between the number of games that a team won and the number of games a team will win [4]. The next dominant feature is travelling distance which can be considered as indicator home/away but with more information. The tendency to win at home is around 60% [6]. The following features are aggregated players statistics and it turned out that our results are not matched with [4] since these top features represent offensive statistics FG_PCT - field goal percentage, FGM - field goal made, AST - assistance.

One season | **Consecutive seasons**

Averaging

Season	Ridge	LinearSVM	DecTree	Forest
2014-15	69.50%	67.90%	64.30%	68.30%
2015-16	68.00%	67.20%	58.40%	67.70%
2016-17	63.70%	63.40%	58.40%	64.50%
2017-18	66.50%	66.30%	63.10%	66.00%
2018-19	65.90%	64.00%	64.30%	65.30%
Total	**66.70%**	**65.80%**	**61.70%**	**66.40%**

Season	Ridge	LinearSVM	DecTree	Forest
2014-15	69.50%	67.90%	64.30%	68.50%
2015-16	68.90%	69.20%	64.10%	67.40%
2016-17	64.30%	63.40%	62.00%	64.20%
2017-18	66.90%	66.90%	65.20%	66.30%
2018-19	65.70%	65.90%	65.10%	64.80%
Total	67.00%	66.70%	64.20%	66.20%

TS Statistics

Season	Ridge	LinearSVM	DecTree	Forest
2014-15	70.00%	69.90%	62.90%	69.30%
2015-16	68.80%	68.40%	63.90%	66.70%
2016-17	64.80%	64.60%	60.20%	65.20%
2017-18	66.00%	66.30%	60.40%	64.70%
2018-19	66.30%	66.20%	64.60%	65.10%
Total	**67.20%**	**67.10%**	**62.40%**	**66.20%**

Season	Ridge	LinearSVM	DecTree	Forest
2014-15	70.10%	69.90%	62.90%	69.30%
2015-16	69.60%	69.70%	65.00%	67.60%
2016-17	64.60%	64.60%	61.70%	64.70%
2017-18	66.70%	66.30%	65.40%	66.10%
2018-19	66.30%	65.90%	64.90%	64.60%
Total	**67.50%**	**67.30%**	**64.00%**	**66.50%**

Local Linear embeddings

Season	Ridge	LinearSVM	DecTree	Forest
2014-15	67.50%	67.10%	61.00%	65.40%
2015-16	69.20%	68.30%	65.70%	67.60%
2016-17	63.20%	63.10%	62.20%	62.90%
2017-18	65.70%	63.70%	60.90%	64.80%
2018-19	65.40%	62.80%	62.00%	66.20%
Total	**66.20%**	**65.00%**	**62.30%**	**65.40%**

Season	Ridge	LinearSVM	DecTree	Forest
2014-15	68.20%	67.70%	63.60%	65.50%
2015-16	68.80%	68.20%	65.00%	68.30%
2016-17	63.20%	62.60%	61.70%	63.00%
2017-18	65.20%	64.50%	65.40%	66.50%
2018-19	65.30%	65.70%	64.90%	65.30%
Total	**66.10%**	**65.70%**	**64.10%**	**65.70%**

K-means

Season	Ridge	LinearSVM	DecTree	Forest
2014-15	69.50%	68.50%	65.00%	68.40%
2015-16	69.10%	67.40%	63.90%	67.90%
2016-17	63.00%	62.60%	56.70%	63.60%
2017-18	66.00%	65.20%	63.60%	66.10%
2018-19	65.10%	64.40%	61.70%	65.60%
Total	**66.50%**	**65.60%**	**62.20%**	**66.30%**

Season	Ridge	LinearSVM	DecTree	Forest
2014-15	69.50%	68.50%	65.00%	68.40%
2015-16	69.40%	68.20%	64.00%	67.10%
2016-17	63.00%	62.60%	62.10%	62.90%
2017-18	66.60%	66.40%	65.20%	66.50%
2018-19	65.40%	65.30%	65.30%	64.80%
Total	**66.80%**	**66.20%**	**64.30%**	**65.90%**

Figure 9: Comparison of training on one season and consecutive seasons

	Averaging	TS Stat	K-means
Averaging	1.000000	0.576971	0.578759
TS Stat	0.576971	1.000000	0.587996
K-means	0.578759	0.587996	1.000000

Figure 10: Correlation of Ridge predictions for Time Series splitting

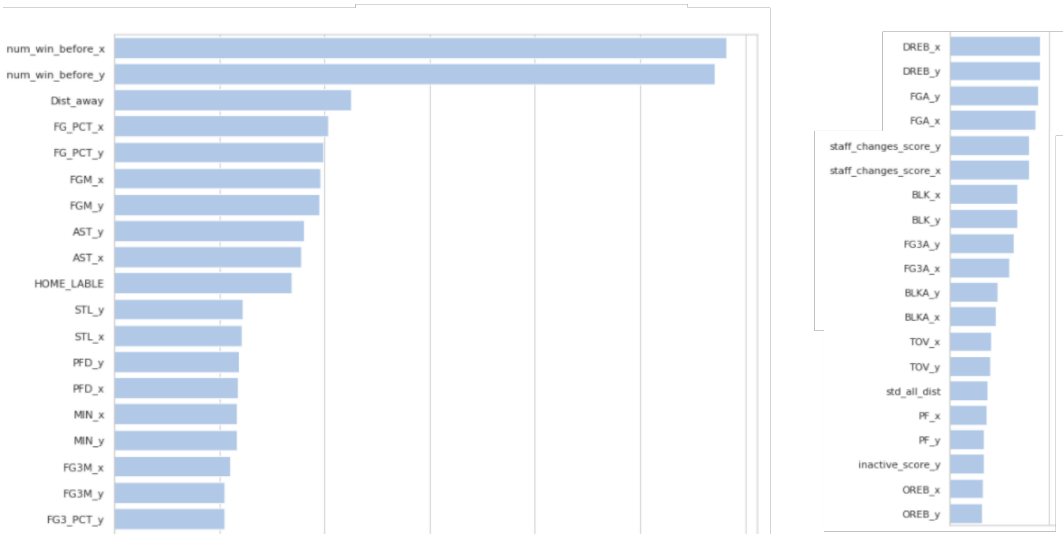

Figure 11: Feature importance

It should be mentioned of our devised features such as inactive_score and staff_changes_score. Fig.11 shows that inactive_score is not good feature. staff_changes_score is also somewhere in the bottom of features list but it interacts with other features which we can see from Fig.12.

2.6 Performance

As we can see at the figures 7-9 the season 2016-17 has the worst results (usually $2-4\%$ worse). To understand such behaviour we have checked all histograms of the important features by season, and also the histogram on which the model was trained to predict the 2016-17 season (particularly seasons 2012-16). We will show and compare just 3 features, because they seems for us more interesting and they are at the top of the feature importance table with respect to random forest.

The figure 13 shows the histograms of some main features: num_win_before_y, FG3M_x. The histograms represented in such order 2014-15,2015-16,2016-17,2017-18,2018-19 and the histogram of the seasons 2012-16 on which the models were trained to predict 2016-17.

As we can see the at all top features exists some problems. The histogram for feature num_win_before_y for 2016-17 season have different shape when WIN=0 the histogram have two hills and one hill when WIN=1, while the histogram for 2012-16 has two hills when WIN=1 and one when WIN=0. The next feature FG3M_x have different problem, for 2016-17 season and 2012-16 seems to be mirrored with respect to the lable WIN. When we considered to train the next season only on season before we have seen the same problems for train season

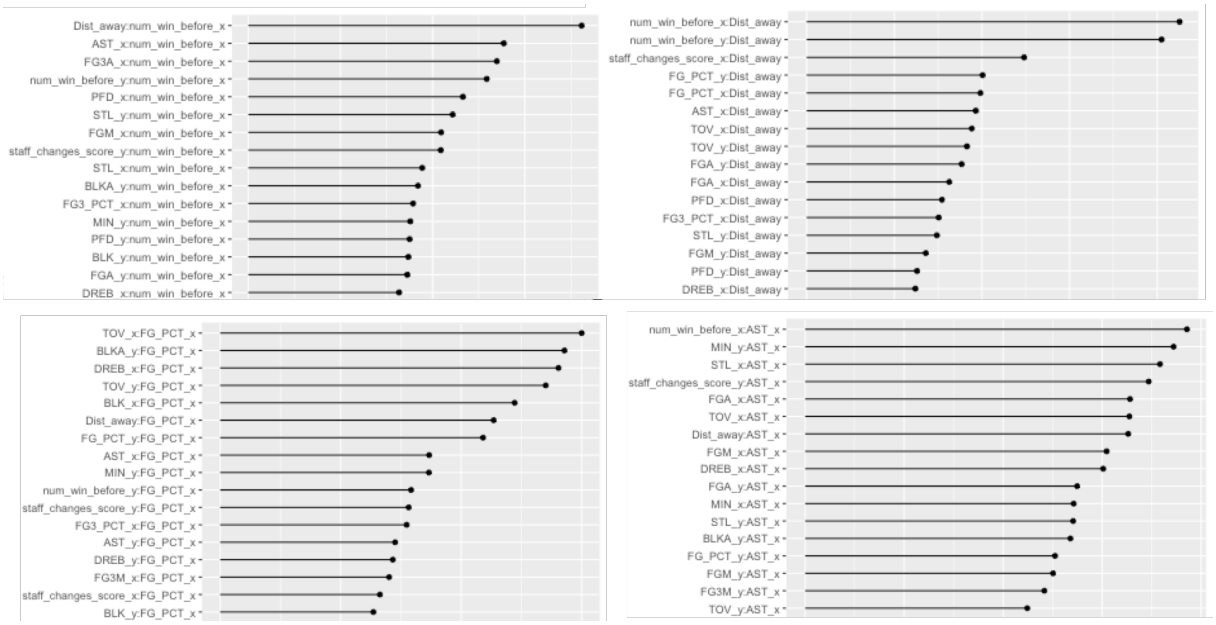

Figure 12: Feature interaction

2015-16 and test season 2016-17, while when we trained on the season 2014-15 and predict the 2015-16 season or trained on 2017-18 season and predicted 2018-19 season we have seen that there is not such problems and consequently the result accuracy was much better on $2-4\%$ depending which season we were predicting.

2.7 Comparison with other projects

We have found two projects with the same goal: to predict NBA games results. The first one is the project from Stanford University [5] which employ simple solution: prediction on previous game scores and home/away advantage. The second one [4] is from Carnegie Mellon University which is more advanced since they used performance statistics from previous matches. We can see comparison in Fig. 14. It's not surprising that we outperformed Stanford project since we used more sophisticated approach. However, we have almost achieved the performance of linear regression of Carnegie University, but with help of majority vote.

2.8 Conclusion

As a conclusion we want to emphasise our experience: linear methods works better for this type of data, we have tested a lot of different approaches to predict the future outcome of the games with different type of splitting on

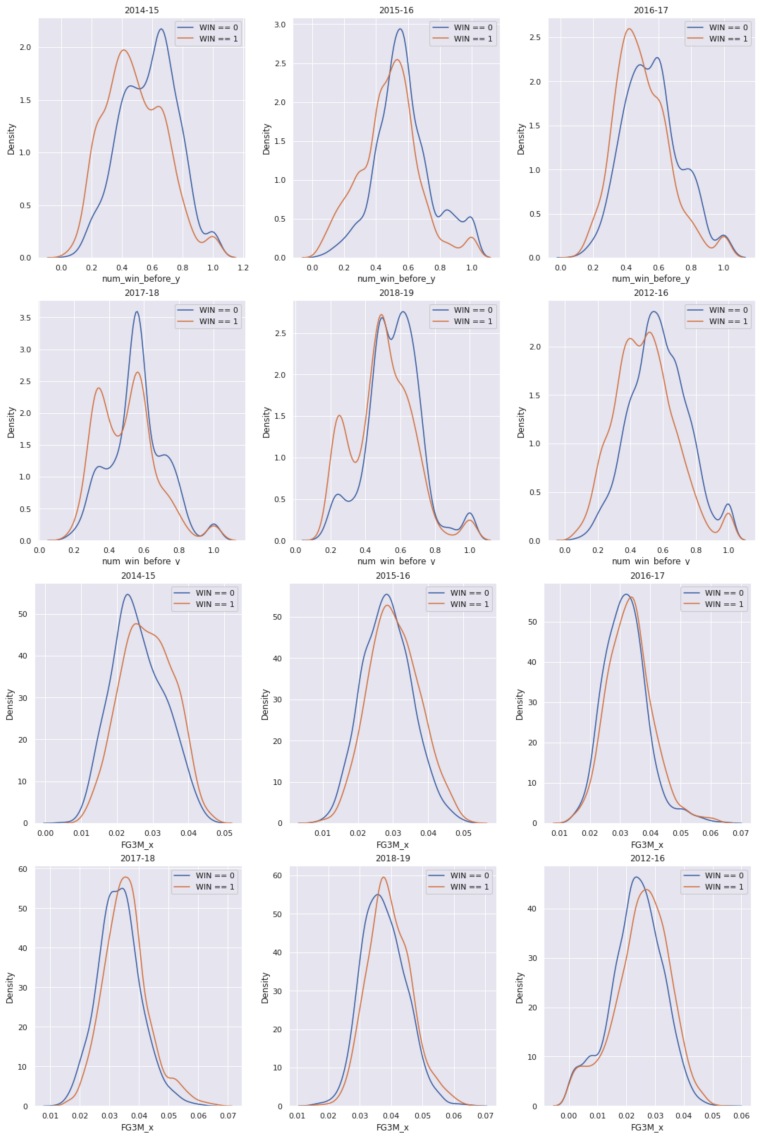

Figure 13: Histograms of the top features

Carnegie Mellon University [1]

GAME OUTCOME PREDICTION ACCURACY RESULTS

	Linear		Logistic		SVM		ANN	
	P	P+C	P	P+C	P	P+C	P	P+C
1992	0.6945	0.6955	0.6736	0.6800	0.6445	0.6527	0.6473	0.6309
1993	0.7110	0.7070	0.6910	0.7000	0.6680	0.6980	0.6601	0.6620
1994	0.6709	0.6836	0.6527	0.6736	0.6527	0.6655	0.6236	0.6400
1995	0.6882	0.6982	0.6773	0.6891	0.6500	0.6864	0.6415	0.6754
1996	0.7309	0.7200	0.6773	0.6955	0.6827	0.6927	0.6664	0.6595
Mean	0.6991	0.7009	0.6744	0.6876	0.6596	0.6791	0.6478	0.6536

Stanford University [2]

	Model 1:SVM	Model 2: LR	Model 3: NNR
TEST: Average Accuracy	62.07%	63.75%	**64.95%**
TEST: Home Score Distance	N/A	**9.8203**	11.7816
TEST: Away Score Distance	N/A	**9.9472**	12.1915
TRAIN: Average Accuracy	**70.99%**	66.85%	70.34%
TRAIN: Home Score Distance	N/A	8.7656	8.8918
TRAIN: Away Score Distance	N/A	8.8426	8.8631

Our best: 69.08 %

Figure 14: Projects comparison

train test and validation sets and we have found that prediction within one season with time series splitting works better then other approaches.In general we have found that time series splitting is slightly better than blocking splitting approach.We also have discovered that consecutive training is more effective then training on one season individually which make sense because more data better performance. Moreover, it turned out that on time series splits k-means features worked better then other which might because of fewer features.

We have compared different techniques of "blending": stacking and majority vote and we have found that in this task majority vote worked better then stacking, but also we have to take into consideration that we used four datasets to have less correlation between the models which were participants in the majority voting. Overall impression after struggling on this task is that: it seems that in general this task has such complex dependencies that to achieve better accuracy than we have achieved have to find some information which well describe each team, players individually right before the game and maybe during the game.

3 Shot Prediction

In this part of the project, we tried to predict outcome (in or out) of shots taken by LeBron James in his whole career from 2003-2020. The raw data is collected from the official website of NBA. The task is divided in to 2 parts. In the first part, we approached the problem as a classic supervised classification problem. First we did some exploratory data analysis,than we performed feature engineering and made predictions. In the second part of the task, we approached the same problem as a time series classification problem. More specifically, We tried to predict the outcome of a shot by using the result of previous shots. By this way, we can also see how, effective the "hot hand" psychology on a players shooting. Here we also introduce the idea of wasserstein time series kernel. This is an approach based on optimal transport theory that simultaneously captures local and global characteristics of time series.

Figure 15: LeBron shooting performance

3.1 Data description and EDA

The data has 29658 rows and 24 columns. These 25 columns contains information about the games and the shots.The columns are; GAME ID, GAME EVENT ID, PLAYER ID, PLAYER NAME, TEAM ID, TEAM NAME, PERIOD, MINUTES REMAINING, SECONDS REMAINING, EVENT TYPE, ACTION TYPE, SHOT TYPE, SHOT ZONE BASIC, SHOT ZONE AREA, SHOT ZONE RANGE, SHOT DISTANCE, LOC X, LOC Y, SHOT ATTEMPTED FLAG, SHOT MADE FLAG, GAME DATE, HTM, VTM.

3.1.1 Shot Result Column(shot made flag)

The "SHOT MADE FLAG" feature is our target variable and shows the output of the shots taken by LeBron. As we can see from figure 15, the data is balanced and the percentage of shot made and missed are allmost equal.

3.1.2 Action Type Column

The action type column shows the actions taken by LeBron for scoring. As seen from figure 16, LeBron is more likely to attempt, Jump shots and Layup shots compared to other action types.

3.1.3 shot Zone Column

The shot zone column shows the shooting zones of LeBron for scoring. As seen from figure **??**, LeBron is more likely to finish in the paint rather than corners or back court.

3.1.4 Shot Distance Column

The shot distance column shows the shooting distance of LeBron for scoring. As seen from figure **??**, LeBron is more likely to attempt, finish close to the rim rather than mid range shots or also we can say some times he is attempting 3 point shots.

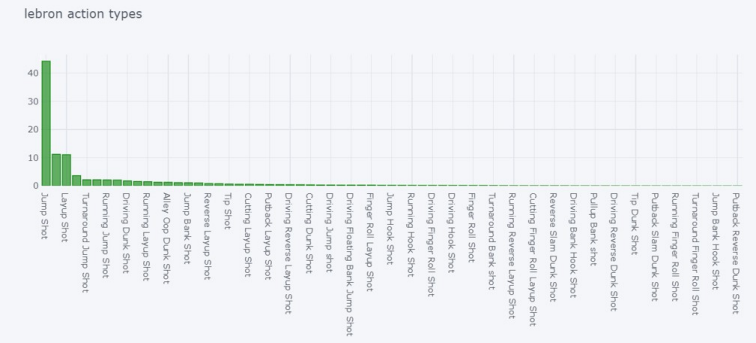

Figure 16: LeBron action type

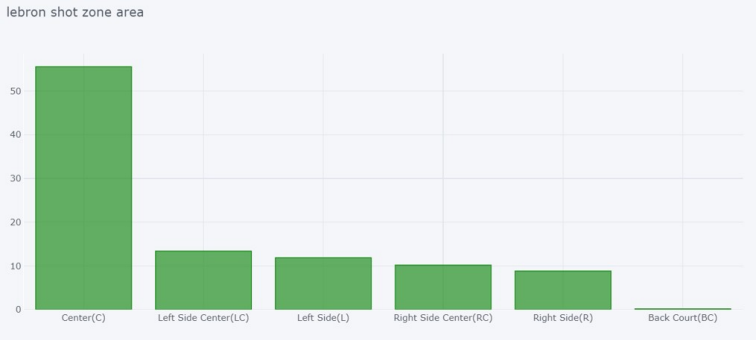

Figure 17: LeBron shot zones

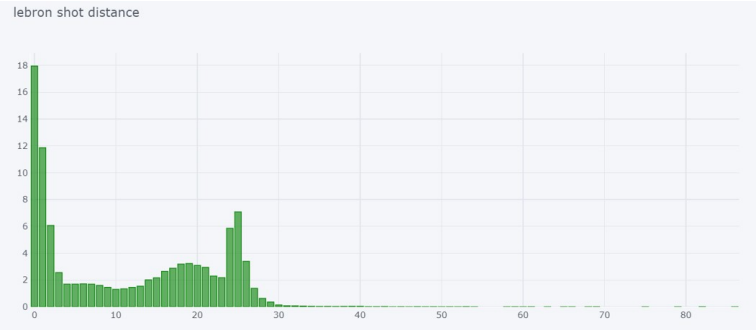

Figure 18: LeBron shot distance

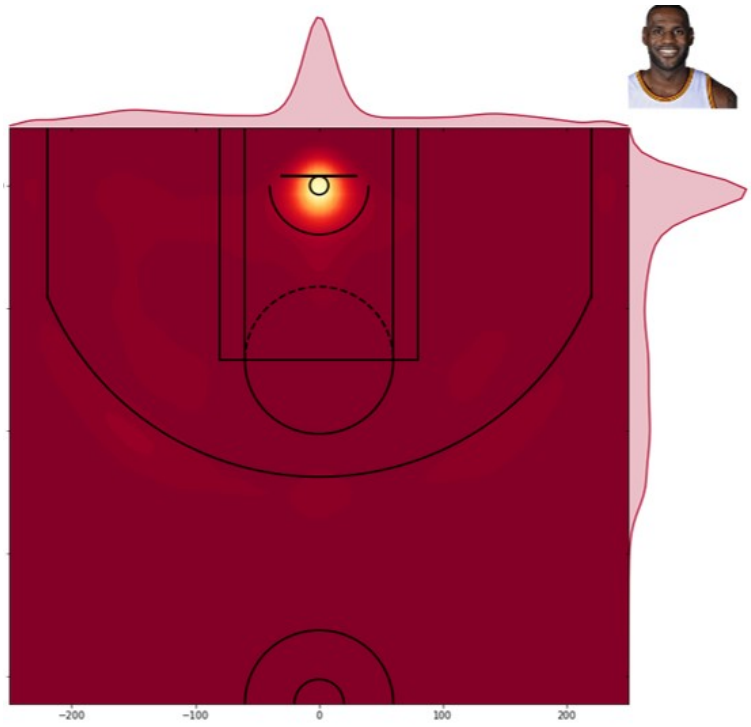

Figure 19: LeBron shot chart

3.1.5 Shot Chart

The figure 19 shows the distribution of shots taken by LeBron. As seen from the figure, Lebron is more likely to attempt shooting close to the rim. When he is attempting mid range shots or 3 point shots, he is more likely to choose left side of the court rather than side.

3.2 Feature Engineering and Feature Importance

3.2.1 Feature Engineering

The Feature engineering performed on this data consists of removing redundant columns, adding new columns and re-formatting some columns. To this end, we removed the columns which are same for all rows. Also we removed the id columns which are different for all rows but non-informative. After that, we added some new features which are; clutch moment, home court advantage and playoff game. The clutch moment feature is a binary features shows if the shot is taken in the last 10 seconds of the game. The home court advantage feature is also a binary feature showing if the lebron team is home team or away team. The playoff feature is also a binary feature showing if the game is a playoff game

	CART	RANDOM FOREST	XGBOOST
1	SHOT_DISTANCE	time_remaining_seconds	ACTION_TYPE_Layup Shot
2	time_remaining_seconds	GAME_EVENT_ID	SHOT_DISTANCE
3	GAME_EVENT_ID	LOC_Y	ACTION_TYPE_Jump Shot
4	LOC_X	LOC_X	ACTION_TYPE_Reverse Layup Shot
5	LOC_Y	SHOT_DISTANCE	ACTION_TYPE_Tip Shot

Figure 20: feature importance

	LIGHTGBM	RANDOM FOREST	XGBOOST
LOG LOSS	0.5849	0.6145	0.5552
ROC_AUC_SCORE	0.7408	0.7309	0.7940

Figure 21: performance of predictive models

or not. Finally we re-formatted the time feature which was previously showing the remaining time for each period. Now it is showing the remaining time for the whole game.

3.2.2 Feature Importance

We measured the importance of each feature by using 3 algorithms. These 3 algorithms are CART, Random Forest and XGboost. The figure 20 shows the top 3 important features for the 3 algorithms. As seen for CART and Random Forest, the features are the same but just the order is differet. For XGboost, action type seems to be more important. For all there methods, shot distance is in top 5 important features.

3.3 Predictive Models and Performances

For the predictions, we compared the performance of 3 methods. LightGBM, Random Forest and XGboost under Log loss and Roc Auc score. As seen from figure 21, The best performing model is XGboost.

3.4 Time series data

In this part of the project, we will explore *hot hand effect*, now we will consider the shooting performance of Lebron James as a time series, and try to predict the outcome of his shots based on past balls.the *hot hand effect* is the purported phenomenon that a person who experiences a successful outcome has a greater chance of success in further attempts. The concept is often applied to sports and skill-based tasks in general and originates from basketball, whereas a shooter is allegedly more likely to score if their previous attempts were successful.

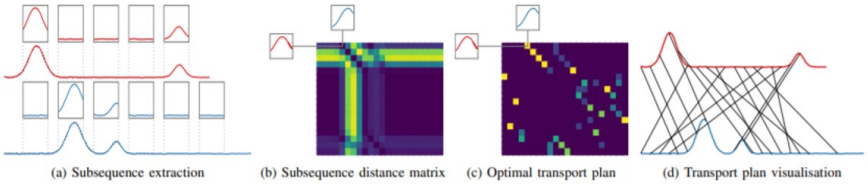

Figure 22: WTK intuition

3.5 Wasserstein Time Series kernel

Here we also introduce the idea of wasserstein time series kernel [7]. In this work, the authors propose a kernel approach based on optimal transport theory that simultaneously captures local and global characteristics of time series.

To measure the dissimilarity between two time series 22, the method proceeds in several steps. (a) First, all sub sequences of the two time series are extracted based on a sliding window approach (here, not all sub sequences are shown because their windows overlap). (b) Second, the pairwise distance matrix between all sub sequences is calculated. Yellow indicates large distances, whereas blue indicates small distances. On its own, this matrix is not sufficient to assess the dissimilarity between the two time series, because it is not clear which sub sequences correspond to each other. (c) Calculating the optimal transport plan makes correspondences more readily visible. Yellow indicates a large fraction of transported mass (high similarity), whereas blue indicates a small fraction (low similarity). For example, the two highlighted sub sequences are matched with each other in the plan. Since the two time series have different lengths, some rows of the transport plan contain fractional matching, making it possible to detect fine-grained differences in the distributions of sub sequences. (d) A visualisation of the transport plan. Each line indicates a match between two sub sequences, with the line being anchored at the respective beginning of the corresponding sub sequences. The thickness of the line indicates the transport value. For clarity, only the largest transport values are shown.

If we want to have a kernel that belongs to an RKHS, we must prove that it is positive definite.This is equivalent to stating that for any data set, the symmetric matrix D with $D_{i_j} = W_1(T_i, T_j)$ is (conditionally) negative definite, which implies that it has at most one positive eigenvalue. The empirical results indicate in the work [7], that for some data sets and some configurations, the authors observe more than one positive eigenvalue in D; the kernel matrix K is therefore not positive definite. To cope with this, a Krein SVM algorihm is also used in addition to the regular SVM.

3.6 SVM kernel comparison

To test *hot hand effect*, we will compare the performance of this wasserstein kernel with default kernels of available in sklearn linear polynomial, rbf and sigmoid by using a SVM classifier. The figure 23 shows the prediction results for

	Linear	RBF	polynomial	sigmoid	Wasserstein
5-fold CV score(accuracy)	0.5575	0.5056	0.4945	0.4678	0.5170
Test score (accuracy)	0.5196	0.5208	0.4926	0.5233	0.5196

	Linear	RBF	polynomial	sigmoid	Wasserstein
5-fold CV score(accuracy)	0.5363	0.5133	0.5325	0.5854	0.5588
Test score (accuracy)	0.4872	0.4834	0.4719	0.4974	0.4821

Figure 23: prediction accuracy for 10 previous and 50 previous balls

5 different SVM kernels with 10 previous and 50 previous balls of Lebron. The scores are accuracy scores for 5 fold cross validation in train set and a test set score. Here we can see that there is significant advantage of using Wasserstein kernel for this problem. This may have some several reasons. First the time series is binary and we don't test if there is really a temporal dependency between time series. Second, wasserstein kernels may not be positive definite. Although we introduce the idea of Krein svm, we didn't use in this set up. Finally, at the end, we are looking for the "hot hand" effect. (This is a pretty psychological thing, so it is not a surprise to have accuracy around 0.5). Let's give one more chance to Wasserstein kernel with a different data to see if it really works or not.

4 LeBron effect on game Results

4.1 Data Description

To better test our Wasserstein distance kernel, we generate a new data again from official NBA site. LeBron is one of the greatest players in NBA history. But he really can change the result of a game with his individual performance? With the same approach, now we will try to predict the if LeBron teams can win their games depending on LeBron's past game performances. As a performance metric, we will use the NBA fantasy points of Lebron from 2009-2020.

4.2 SVM kernel comparison

now once again, we will compare wasserstein kernel with default kernels of available in sklearn linear polynomial, rbf and sigmoid by using a SVM classifier. The figure 23 shows the prediction results for 5 different SVM kernels with 5 previous, 10 previous and 15 previous games of Lebron. The scores are accuracy scores for 5 fold cross validation in train set and a test set score. Also the krein SVM scores for the test set are present. for 5 games lagged, wasserstein kernel is sligthly best scoring kernel but it is not the best in test test. However, the krein svm score is far better than any other kernels. For both 10 and 15 games lagged cases, wasserstein kernel has the highest accuracy among all the other kernels.

As we saw, Wasserstein kernel performed better in continuous time series

	Linear	RBF	polynomial	sigmoid	Wasserstein
5-fold CV score(accuracy)	0.5379	0.5203	0.4659	0.5423	0.5431
Test score (accuracy)	0.6173	0.5804	0.5138	0.5175	0.6099

Krein SVM grid search test score: 71.16

	Linear	RBF	polynomial	sigmoid	Wasserstein
5-fold CV score(accuracy)	0.6206	0.6094	0.5307	0.6533	0.6719
Test score (accuracy)	0.6797	0.6722	0.3445	0.6499	0.6834

Krein SVM grid search test score: 69.09

	Linear	RBF	polynomial	sigmoid	Wasserstein
5-fold CV score(accuracy)	0.6773	0.6832	0.5310	0.6953	0.7296
Test score (accuracy)	0.6741	0.6610	0.4176	0.6516	0.6760

Krein SVM grid search test score: 68.35

Figure 24: prediction accuracy for 5, 10 and 15 previous games

then binary time series and outperformed all other kernels in 10 and 15 step a head forecasts. Krein SVM performed better even though in this problem all kernel matrices are positive definite. Is this because of it is better tuned or the stabilization term of krein svm contributes to learning? For the Future work, we can embed this approach to our very first goal (predicting the game result) by predicting the game outcome fir the best player of each team.

References

[1] https://scikit-learn.org/stable/modules/manifold.html

[2] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[3] https://jr.nba.com/basketball-positions/

[4] https://www.mbeckler.org/coursework/2008-2009/10701_report.pdf

[5] http://cs229.stanford.edu/proj2017/final-reports/5231214.pdf

[6] https://bleacherreport.com/articles/1520496-how-important-is-home-court-advantage-in-the-nba: :text=In%20the%20first%20round%2C%20home,back%20up%20to%2080%20percent

[7] https://bastian.rieck.me/research/ICDM2019_WTK.pdf/