Machine Learning
Kobe Bryant Shot Prediction

Machine Learning Project

Filip Sotiroski

Mohammad Ismail Tirmizi

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

May, 2022

TABLE OF CONTENTS

Overview

Analyzing and modeling sports data can be very beneficial for the success of sports teams. There are multiple variables that decide if a team is good or if the team will win a certain game.
In our case, we are interested in analyzing and predicting the shot efficiency of NBA's player Kobe Bryant. We have found a dataset that contains all his shots taken in his NBA career in the only club he played in, the LA Lakers. We need to predict the target value, if a shot has entered the hoop or not.
Our main motivation to do this project is because we are interested in using a sports dataset to predict the outcome of a sports event. Big basketball, football teams are expanding their data analytics/science teams in order to become better. This means that it is important to know how to work with sports data and it offers many opportunities in the future.

Kobe Bryant

One of the best basketball players that played in the NBA. He is recognized as the player with the biggest motivation and winning mentality. He was drafted by the Los Angeles Lakers in 1996 in the first round as the 13th pick. He played from 1996 until 2016 for the Lakers.

He was a 5 times NBA champion, from 2000-02 he won 3 titles and 2 titles from 2009-10. Throughout his career he made 26200 shots and scored 33643 points. During his championship years, he won the title with the help of two big-man centers, Shaquille O'Neal and Pau Gasol. He was named as MVP in 2008 and has two Finals MVP in 2009 and 2010. He had 15 consecutive seasons in the playoffs, from 1997 until 2012. His peak was during the 2000's. In 2013 he suffered an injury which made him play 41 out of 162 regular games in the season 2013 and 2014.

The Lakers played very competitively through Kobe's career. Some of their bad seasons were from 2004-07 and 2012-2016, when Kobe was injured and old.

Kobe Bryant played as a shooting guard, but can also play the small forward positions. He was known for creating his own shots and is a competent three-point shooter.
He is a basketball Hall of Fame. Unfortunately he passed away in a tragic accident in 2020.

Dataset

https://www.kaggle.com/competitions/kobe-bryant-shot-selection/overview

The dataset has 25697 values. It has 23 features, and 1 target label. There are no missing values as this is the official NBA statistics. The dataset has a mix of categorical and numerical values, in fact 9 categorical and 14 numerical. Kaggle also provides a test set, but we won't be using that as it has missing target labels. These 5000 missing labels are randomly distributed in the entire dataset. It may be that for many of the games we don't have the entire dataset available. So this may make game-wise or even season-wise analysis very hard as we don't have true labels to see the true picture.

One of the biggest challenges that we will face is the data leakage. In this case, we would have to be careful when picking the right rows to train the date. We can not predict his shots in 2005 with data from 2010. This is also one of the reasons why we chose this dataset as this problem can come up very often.

We were given these following 24 features:

combined_shot_type: The shot type that led to the shot. There are 6 unique values containing the basic shots like jump shot, layup, dunk.
action_type: the more detailed explanation how the shot was made. There are 30+ different action types, all of them derived from the combined_shot_type feature. Meaning, that it carries more details.
game_event_id: unknown
game_id: the identifier for every NBA game.
lat, lon: coordinates on the floor.
loc_x, loc_y: coordinates on the flor where the middle point is the centre of the court.
minutes_remaining: remaining time in minutes until the end of the *quarter.* The MM part from MM:SS.
period: ranges from [1..7]. The values from [1..4] are regular time, [5..7] is overtime.
playoffs: binary value if the shot was made in the playoffs.
season: The season in which the shot was taken.
seconds_remaining: Seconds remaining, complementary to minutes_remaining. The SS part from MM:SS.
shot_distance: the distance from where the shot was taken and the hoop.
shot_made_flag: The label that we want to predict. If the ball entered the hoop or no.
shot_type: 2 point or 3 point shot.
shot_zone_area, shot_zone_basic, shot_zone_range: the area from which the shot was taken.
team_id: The id of the team he was playing for. (LAL)
team_name: The name of the team he was playing for (LAL)
game_date: The date on which the shot happened
matchup: Against whom they were playing, home or away.
opponent: The opponent's team name
shot_id: The unique id related to every shot.

Exploratory Data Analysis

Data Cleaning

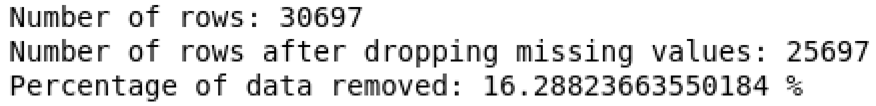
The first step in any machine learning task is to get an understanding of the data and ensure that it lies within reasonability. The understanding part can be as deep or shallow as we want, but it definitely helps us in other phases of the pipeline, so the more the better. We also want to see what patterns are in the data and is it making sense to us. This is because If we find something too unreasonable we can expect the model to have a hard time making sense of that datum as well.

We start with, in the first notebook: “1-EDA”, by doing basic inspection of the dataset. We also used ‘spreadsheet’ and glossed over all the columns and tried to understand what each column is conveying. The `df.describe(include='all')` gave us a statistical view of the dataset as well.

Dealing with Missing Values

First order of operation is to deal with missing values. We checked each column for it. We used the output of `df.describe(include='all')` to tell us if there are missing values. If there are missing values the count of that column would be less than the total rows. This however only works if the missing value is missing and not replaced with some other place holder. Luckily *our dataset didn’t have any missing values in any column except for the target column* ‘shot_made_flag’.

The missing values in the shot_made_flag column are part of the test set made by kaggle. Since they didn’t provide us with the groundtruth for these, we went on to remove these rows. The output is:



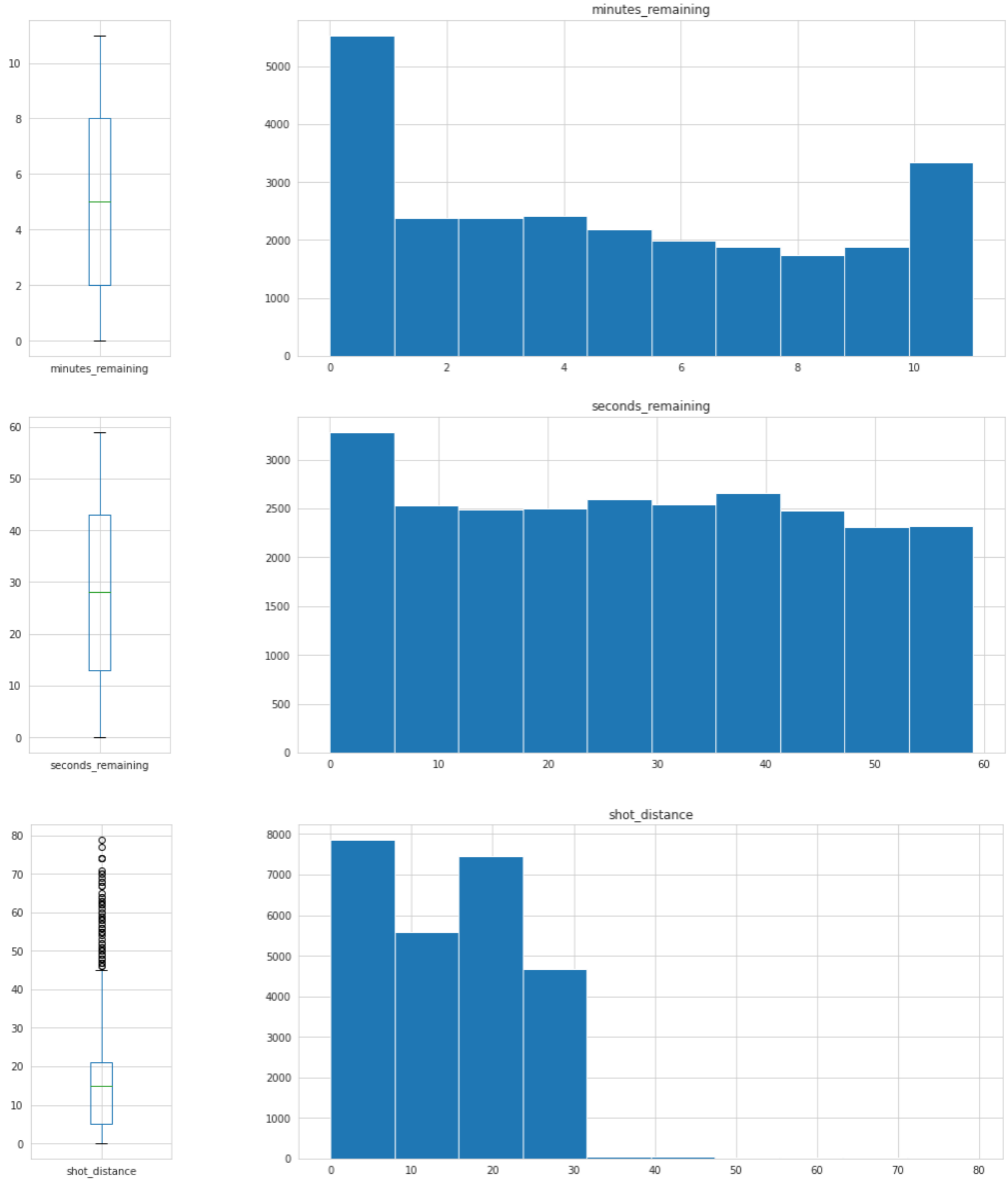
```
Number of rows: 30697
Number of rows after dropping missing values: 25697
Percentage of data removed: 16.28823663550184 %
```

Because we removed 16% of all the values, we are aware that we are not working with all the shots that Kobe has ever done in his career. Therefore, we need to make an assumption that we are working with 85% of the shots in his career. Through the EDA we did not notice any difference in the expected patterns of his performance, for example: injured seasons, championship seasons etc. The distribution of missing rows per game is balanced. For each game, on average 16% of the rows are taken out with a standard deviation of 10%.

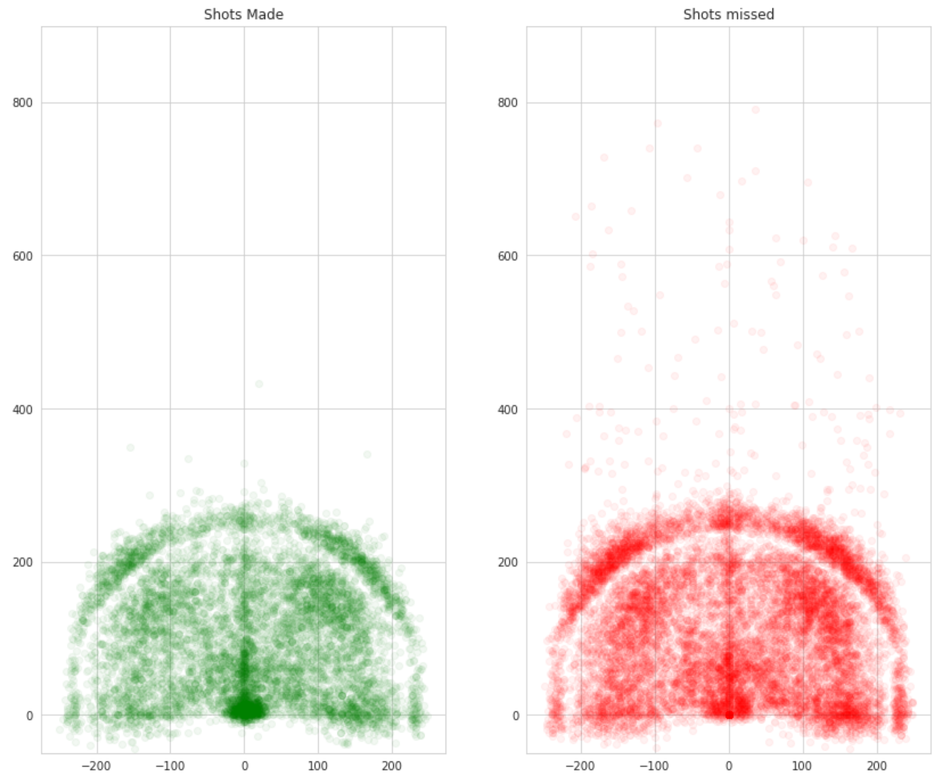
If we had missing values in other columns we would have to impute them with other techniques. E.g. We could have used a mean, median or mood to fill the missing values. Another advanced technique could be to train a model on the column with missing values, then use that model to predict the missing values.

Outlier Detection

Next task we did was check for outliers. Firstly we got some idea from the df.describe() function about the outliers, next for each column we made box plots to visually see the data. This was for a more detailed analysis: E.g. plot



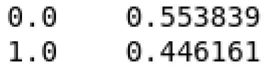
As we can see in shot_distance from first glance it looks like there are outliers, but in this case they are not. We need more context to understand this.



As we can see from the faint dots near the other end of the court, we get to know that more shots were made near the opponent's hoop, but some were made from far away as well. So the shot_distance column may have this skewness but that doesn't mean that those values are outliers. So we didn't do anything to this column.

From the box plot we also got to know that (`lat` & `lon`) and (`loc_x` & `loc_y`) are giving the same information, just with different coordinate systems, so we removed the redundant columns. Note this removal happens towards the end in notebook 3. The shot zones are more useful.

Lastly we checked the dataset's distribution:



```
0.0    0.553839
1.0    0.446161
```

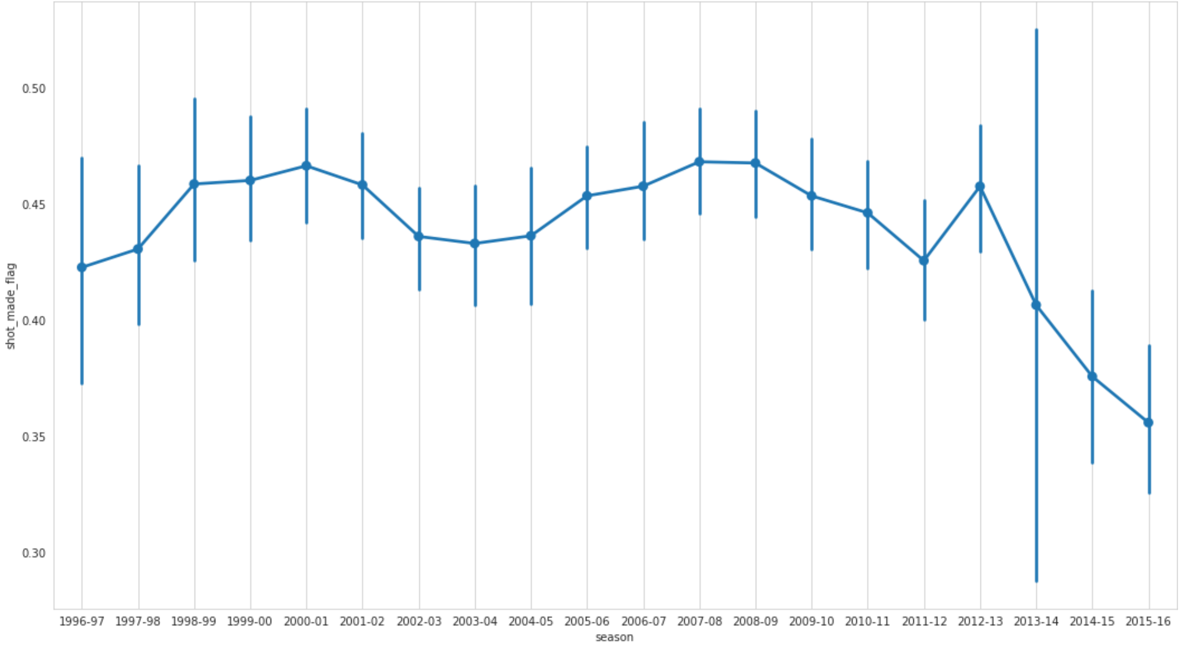
So this looks like a fairly balanced dataset.

Domain Specific Data Analysis

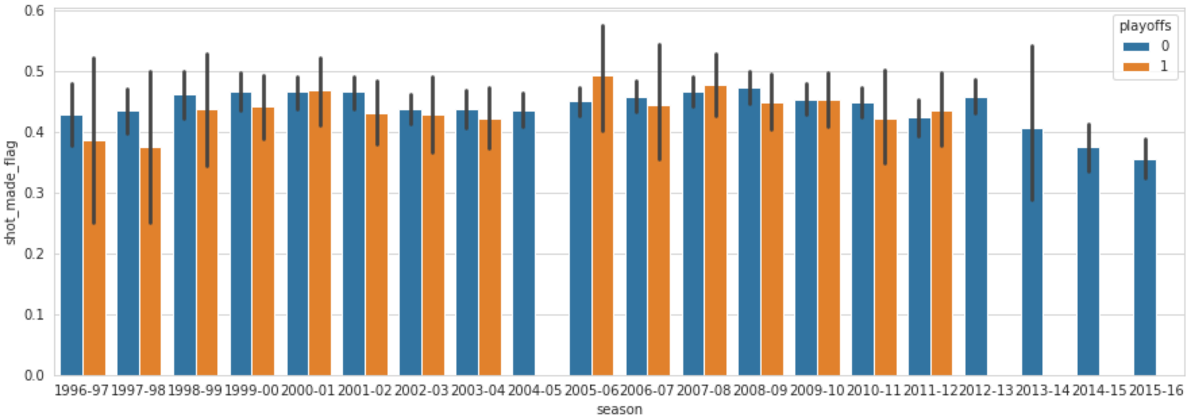
We notice that we have a fair amount of features that can help with predicting the shot's success. Especially features like opponent, season, shot zones.

He played in the 90's, an era with heavy defense, especially during his first years when the game was mostly played through a big-man like Shaq. We can expect a different player behavior during the years. He had many different teammates and tactics during the year.
Another very important factor is the opponent through the years. Teams change and in one year Kobe can have an easier matchup, therefore shooting better, or he could be playing against more defensive players.

We can see his Field Goal attempts during each of his seasons. We see the small hills during the years when he won the championship, but also a decrease after the injury (notice the high variance) in 2012 and the period when he got old.

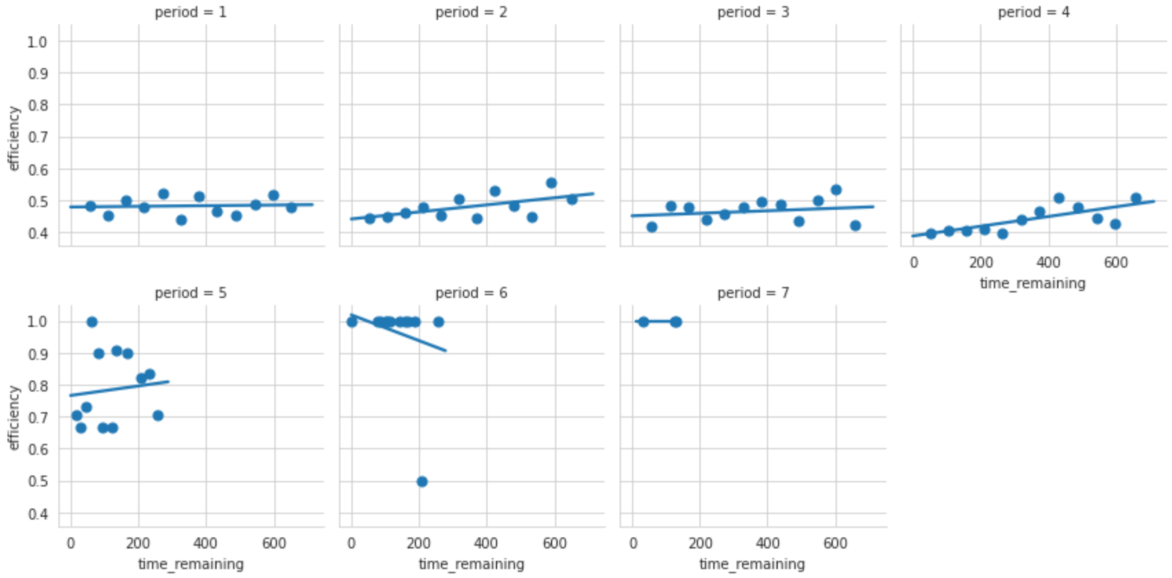


His contribution during the playoffs can also change depending on the year. We would expect that he would have a better FGP (Field Goal Percentage) during the years, but the data shows differently. This is mainly because they play against stronger teams and the defense is on a higher level in the playoffs.

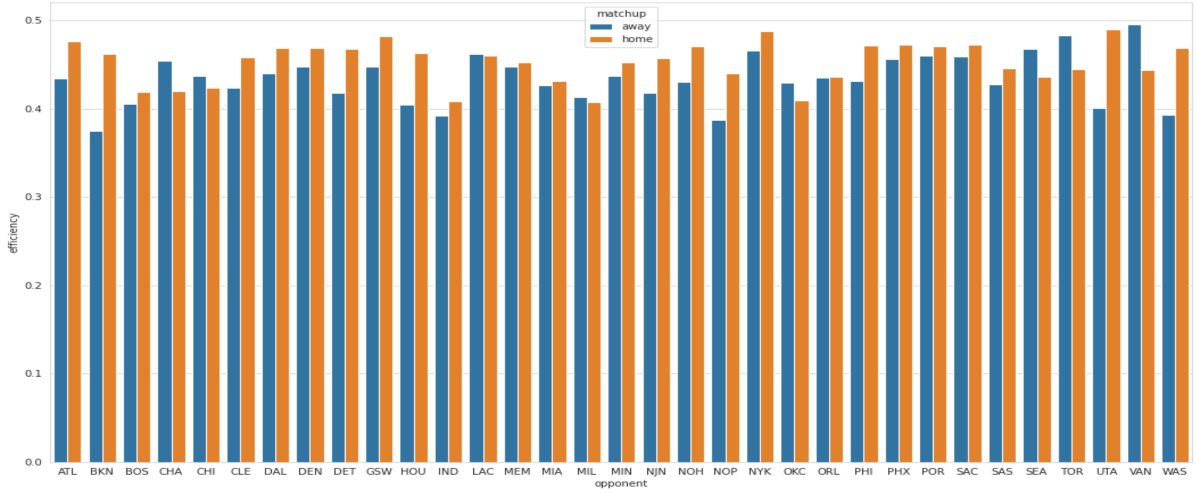


Another extremely important feature is the time left in the game. Each period the player can play differently. We made a regression plot with 12 bins to represent the 12 minutes in each quarter. Note that 0 means there is no time left in the quarter, so the direction is from right to left.

We notice how the shot efficiency decreases, mostly because there is more pressure until the end of the quarter. We see that the slope is much steeper in the fourth period because some games are being decided in the last quarter. Teams play better defense and shots are harder to be taken. For the overtime, there are not a lot of data points, especially for OT2 and OT3, i.e. period 6 and 7.. We notice from the graphs that he has a good shooting percentage in the OT2.

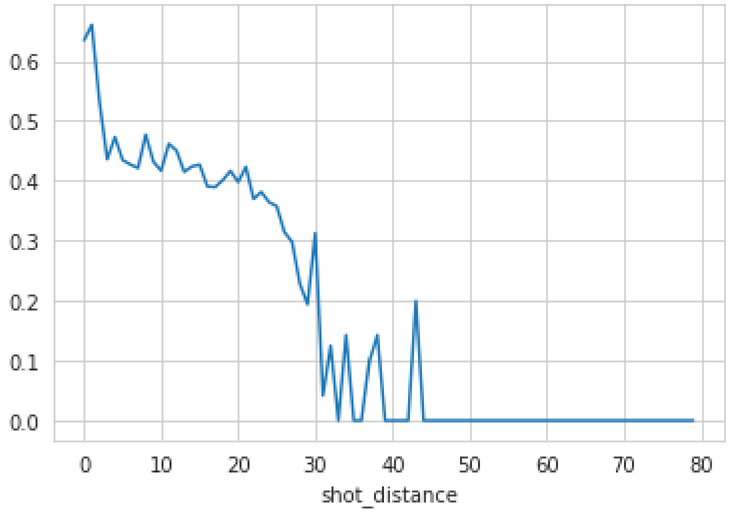


Home field advantage can mean a lot for teams. Here we see how much better they perform when they play home in the Staples center. We see on the graph that there some bars are not on the same height as others. Those are franchises that are new/old like Vancouver (Memphis) Grizzlies, New Jersey (Brooklyn) Nets, Supersonics, Thunder.

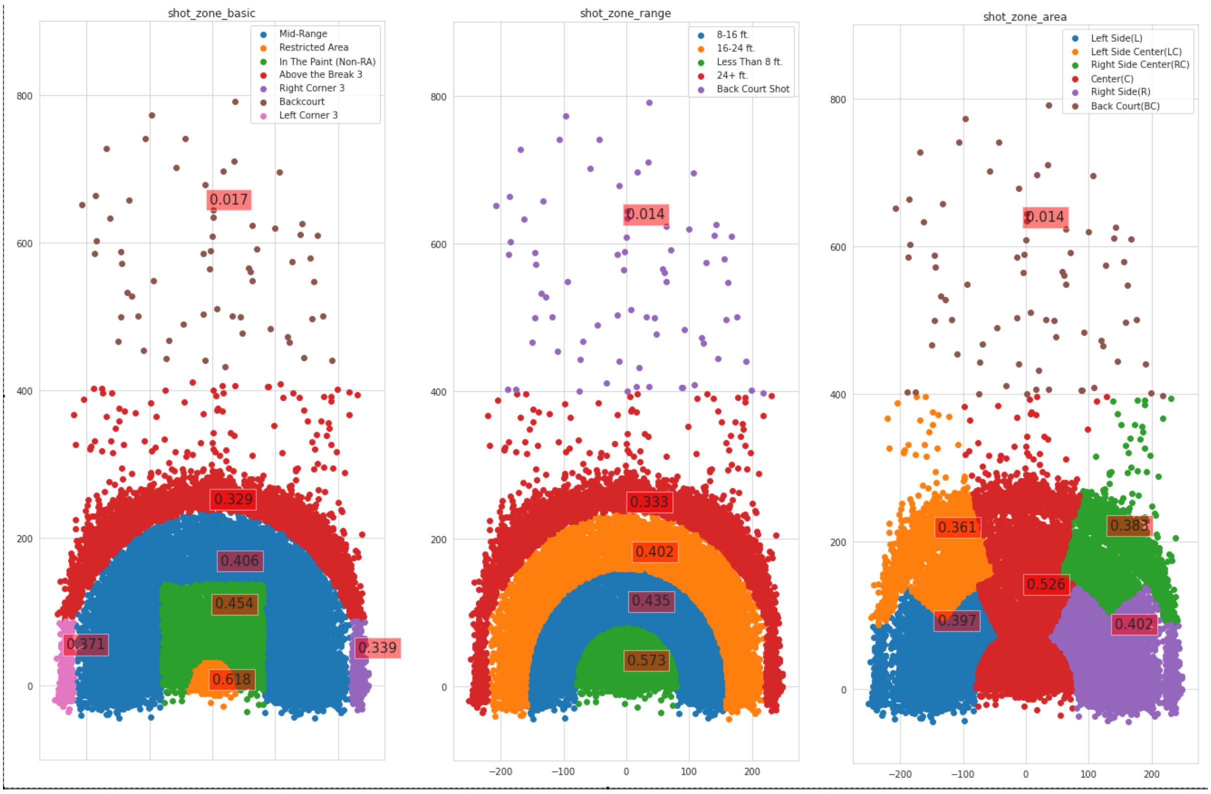


Some players are shooting from some parts of the court much better than the other. For example, Kobe was a great mid-range shooter and had very successful layups and dunks. We could also expect more three point attempts as since 2015, the teams in the league have started playing a wider game with the emphasis on three points.

From analyzing the shot zones, we see it was more effective from the center area. Of course, the closer he was to the hoop, the easier it was to score.



We also observe that he is shooting a little bit better on the left side compared to the right side.



Feature Engineering

Basic feature engineering:
This basic feature engineering is used so that we are able to easily manipulate through the dataset and use the features to do the EDA.
As usual, we first removed the N/A values from our dataset. There were no duplicates in the data.
From the date, we extracted the month and the weekday. Sometimes players have periods where they play bad or they might play better on weekends.
The time management part consists of different columns describing the value of minutes and hours. We therefore combined seconds and minutes remaining into a new feature called *time_remaining*. The previous two columns were dropped.
In the NBA league they use the @ notation when a team plays away. We also had to change that and created a new feature called *matchup* where we show if a team is playing home or away.

Advanced feature engineering:
As we don't have more statistical knowledge for that particular shot, we would need to generate it by ourselves. This is the moment where we have to be very careful in order not to introduce data leakage. We should follow the rules that for every shot (each row), we should not have any data from the future.
For the current game, we would like to measure his shot statistics before taking the shot. We are calculating the field goal percentage, shooting streak and points before the shot. This means that with these features we would be able to say if the player has a 'hot hand' and is in a good shooting streak or if the player is shooting good or bad that night.
We are also adding the statistics for the average FGP from the past 5 games.

Categorical to Numerical

As explained in the lab we first split the data into train and test before doing any conversion. This was done to prevent the data leakage from test to train.

While doing the target encoding, we realized that applying the target encoding function on the whole dataset will result in data leakage. For example, if we did target encoding on the opponent, it would transform the opponent value with the mean of the target variable. The final result is the equivalent of the Field Goal Percentage against the opponent. Therefore, we changed the algorithm in order to prevent the data leakage and we are doing it incrementally on the data that was made before the shot.

Target encoding was one of our options as we got around 150 features after applying one-hot-encoding on the whole dataset. This is too much in regards of the feature vs row ratio. With the encoding choices we have reduced it to around 30 features.

The conversion scheme was as follows:

Column	Encoding	Reason
action_type	Target	Too many values
opponent	Target	Too many values
combined_shot_type	One hot	Few values and one-hot handles categorical data better.
shot_type	One hot	Few values and one-hot handles categorical data better.
shot_zone_area	One hot	Few values and one-hot handles categorical data better.
matchup	One hot	Few values and one-hot handles categorical data better.
shot_zone_basic	Ordinal	Need not to create new column, the information had order
shot_zone_range	Ordinal	Need not to create new column, the information had order

Training vs test vs validation set

In order to prevent data leakage in our dataset when training models we apply the feature engineering to both training and test set *separately*.
The reason we are doing this is because we want to remove the dependence of both the train and test set when doing the feature engineering. Like:

processed_train = feature_engineering(train-set)
processed_test = feature_engineering(test-set)

An exception is when we create the validation set with cross-validation. At that moment both the current test set and validation set have been processed as one with the feature engineering function (train-set at the time). We understand that this means that validation-set-fold has data-leakage from the train-set-folds, this is some “risk” that we accept for our use case as we want to facilitate the validation. Together with our professor we decided not to overwork this part as it does not have an impact of higher magnitude on the final result, as the result that matters at the end is the score of the test set.

Validation set is the end part of the training set. E.g. if training-set has season 1, 2, 3 then Validation-set will have the last 20% rows from season 3.

Models and results

In order to predict the target value, we decided to create two different models.

The first model consists of training the data during the regular season (82 games), and testing on the playoffs for that particular season. The idea is to see how well the model can predict the shots in the playoffs if it was trained during the regular season.
We are training on the shots done in the 82 games of the season. We should keep in mind that they are playing against 30 other teams in the regular season. In the playoffs they can play against at most 4 teams, if they reach the final, and for each team they play 4 to 7 games. We have a lot of variety in the training data, simply because there are many shots done against teams that did not participate in the playoffs.

Some of our expectations/intuitions are:
If we train on season X, we would expect that the test seasons X+1 would have a higher accuracy than the season X+5. This is because in season X+1, the matchups and rosters are almost the same as the season before (X). Compared to season X+5 where many teams have changed, the model might have "false" weight for some opponents which were previously a good matchup for Kobe, but in the present they are a bad matchup. We should notice a decrease in the accuracy as Kobe was performing worse because he got older and injured.

For the sake of simplicity, this model will be called **"season" model**

The second model consists of training the data against one team (A) during one season and testing against the same team (A) but on a later season.
We are curious to see how the model trained during one season will perform in another. As we mentioned before, teams can get better, have different matchups and teammates.
It's important to note that we will train only on the shots that were taken against the same team (A), during a range of seasons. Compared to the season model where we have 30 teams, here we narrowed it down to 1 team.

Some of expectations/intuitions are:
We expect that if we train on the range of seasons when they were playing good, we should get a good score when testing on seasons when the opposition team was bad. Contrary, if the opposition team was good.

It is very interesting to observe if there will be a sharp increase or decrease in the accuracy when we test with seasons that are further away from the trained season. Maybe Kobe Bryant changed his way of shooting?
For the sake of simplicity, this model will be called **"team" model**

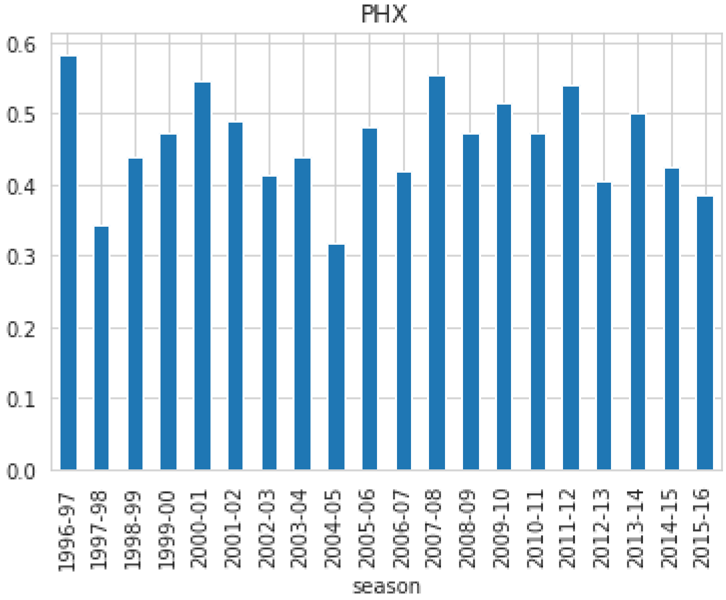
For both models, we will choose the best model depending on the *best validation score*. Once the model is picked, we will apply the test set.

We picked *accuracy* as our main measure for our models as we are dealing with a topic which is only interested in improving the performance of the player. Meaning, that we are only interested if a shot was taken or not.

Results Team model

We have trained different models on the following seasons: ['1999-00', '2000-01', '2001-02', '2002-03', '2003-04']. These were the seasons when they were performing well and won three NBA titles. The team that we chose for the team model is Phoenix Suns (PHX). They are in the Atlantic division together with LAL and they play at least 4 games in one season. They have a chance to meet early in the playoffs as they are from the West.

We see the shot accuracy Kobe had during the years against PHX. In total we are training for 21 games against Phoenix with 332 shots.
We are testing the seasons ‘2005-06’ with 226 rows and ‘2009-10’ with 170 rows. Although season ‘2004-05’ is the successor of the training seasons, we have excluded it because of the very low number of rows.



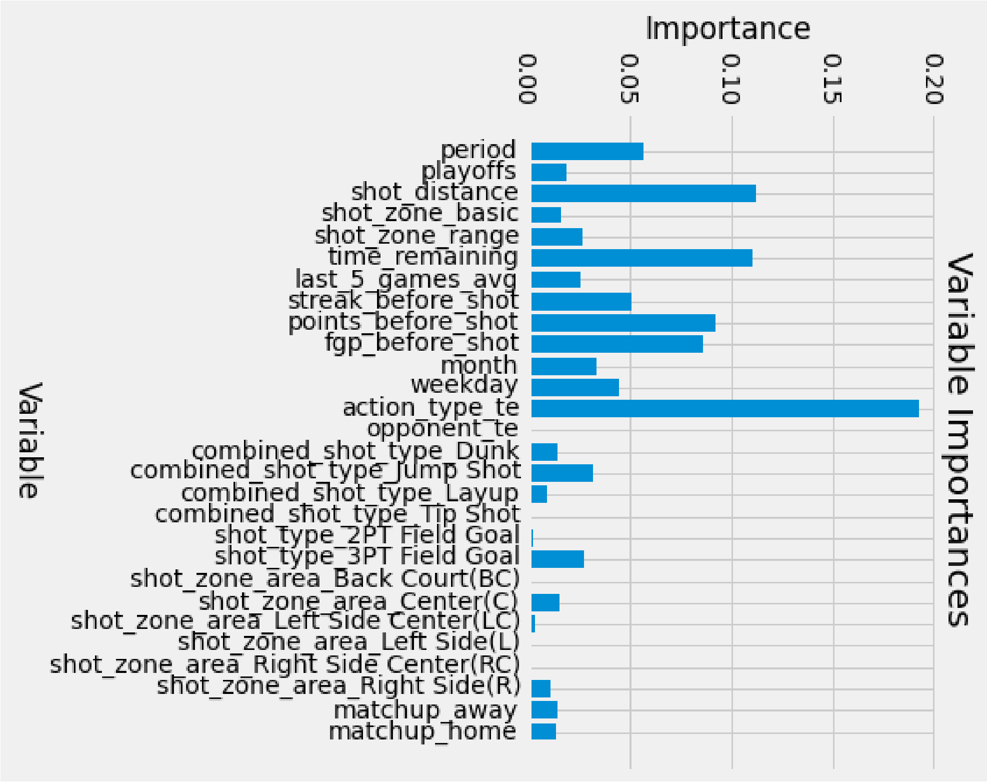
For both splits we use linear regression models, lasso and ridge regression, as well as random forest.
For the validation test set we are using 20% of the training set, which in this case is 67 rows. And the rest is used for the validation training set - 265 rows.

Algorithm	Validation
Linear Regression	0.626
Lasso Regression	0.626
Ridge Regression	0.626
RandomForest Basic	0.552
RandomForest GridSearch	0.582
RandomForest GridSearch + important values	0.641

For the RandomForest we are running the base model (out of the box). Then we use gridsearch to find the best parameters. For the team model we found ot the following are the best parameters to run the RandomForest:

class_weight	balanced
max_depth	25
min_samples_leaf	6
min_samples_split	4
n_estimators	10

Once we find the best parameters we choose the top 6 most important values for the forest. As top features we pick **“action_type_te, shot_distance, time_remaining, points_before_shot, fgp_before_shot, period”.**



We can observe that the best RandomForest parameters have a large depth and a small number of trees. T

Once we have selected the model with the best validation score (random forest + gridsearch + top features), we predict with the test set.

Season	Test accuracy
2005-06	0.6018
2009-10	0.5767

The accuracy is higher when testing on a season that is closer to the last training season.
In 05-06, PHX were a good team, in the regular seasons they won 3:1 against Kobe, but they met in the playoffs where PHX won 4:3. Kobe had a 48% accuracy that season against PHX. Given data of all his shots we are able to predict 60% of his shots from the seasons before.
We get a slightly smaller accuracy in the 09-10 season. Lakers won 3:1 in the regular season and 4:2 in the playoffs. This is one of the seasons that is similar to the training seasons in terms of his high performance, but it is more in the future and there are different rosters.

Results Season model

We have trained in the regular season in 1999-00 and test on the playoffs of the same year. This is a season when they won the championship and had a very good score.
For this season we have 326 rows in the playoff (test set) and 986 rows in regular season (training) from which 198 are used for validation.

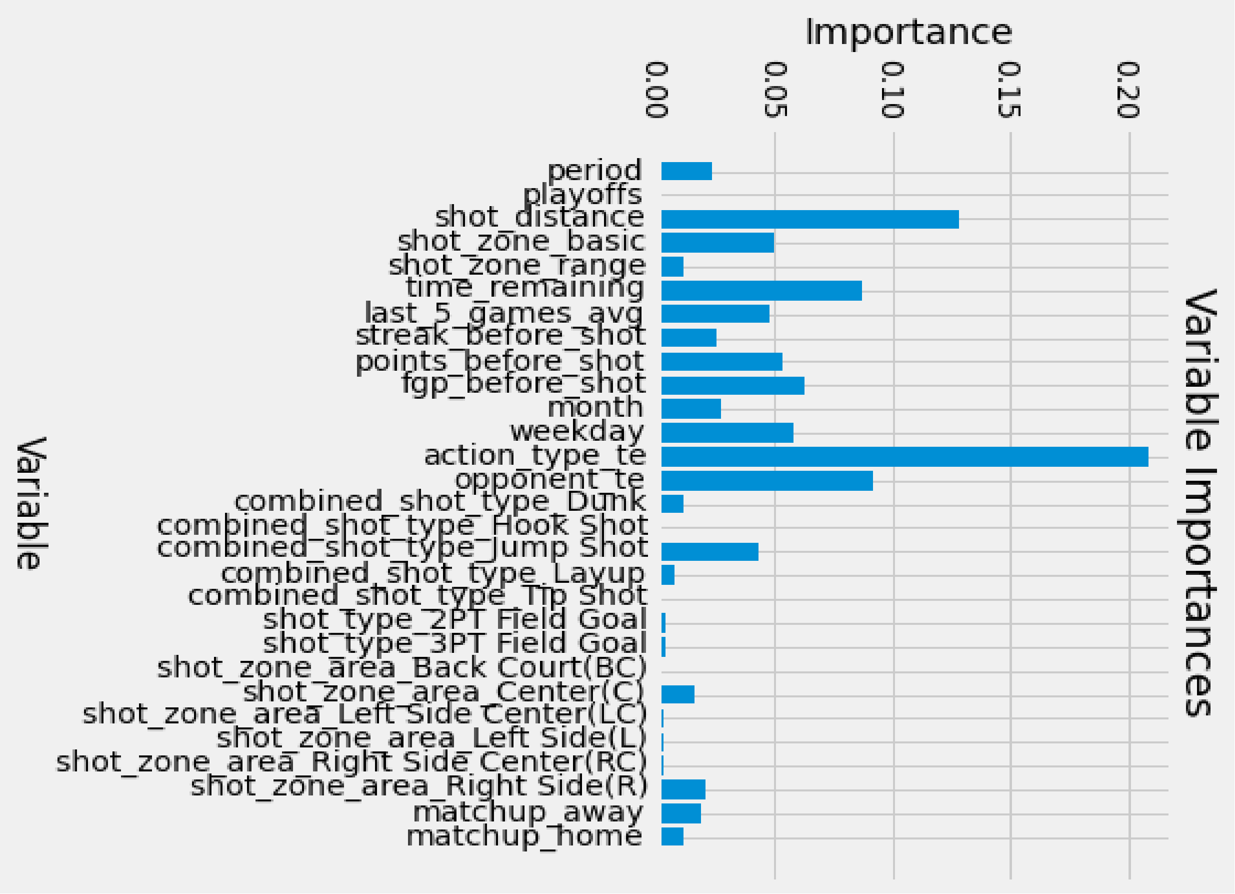
Algorithm	Validation
Linear Regression	0.691
Lasso Regression	0.702
Ridge Regression	0.707
RandomForest Basic	0.651
RandomForest GridSearch	0.626
RandomForest GridSearch + important values	0.691

The max_depth has decreased for this model:

class_weight	balanced
max_depth	20
min_samples_leaf	6
min_samples_split	4
n_estimators	10

The most important values of the RandomForest have changed too. We get the following: **“action_type_te, shot_distance, time_remaining, opponent_te, fgp_before_shot, weekday”.**

As we can see, the opponent feature was not present before because now we are training on 30 teams, and testing on 4 teams. In the previous model we only had 1 team in the train and one in the test. As it is a season, the day on which the player is playing is more important now. This means that they might rest more on some days or perform better during weekends. Action type (layup/jump shot/dunk) is still the top feature for both.



We finally pick Ridge Regression as our best performing model from the validation set and get the following test accuracy:

Season	Test accuracy
1999-00	0.5460

As expected, the performance has changed regarding the regular season. The level of playing is more intensive and harder. The strategies can change in the playoffs which can reflect on the test results. The validation results are quite higher than the test which can show that there is some difference between the performance in regular vs playoffs.

Conclusion

Overall we can say that the dataset is a very unique and challenging. It is a real world example, different from the conventional datasets, oriented towards analytics and pattern mining. We managed to do classification on it. One of the biggest challenges was to decide on what subset we are going to train our data. Finally we chose to train and test for regular and playoffs, as well as training and testing against one team. There were multiple possibilities on splitting the dataset, but finally we believe for this scope and the required work that this is the most optimal way of splitting the dataset.

We believe that if we had more statistical data we could have made better predictions. Somethings that we felt we were missing was: if a game was won/lost, The record of the opposing team, players against whom they were matching up, teammates with whom he was playing, free throws, etc.

We learned a lot about pattern mining and feature extraction. We were introduced to the topic of data leakage. At many points during the project we were noticing different data leakage problems which we had to fix especially when doing the feature engineering

Even though the results in terms of accuracy are not perfect, we followed all the steps that a real world project would do with a lot of cooperation with the professor and general discussion on what we could improve and where we can focus more.