Kobe Bryant Shot Selection

Ethan Goldbeck, Yuxiang Wang, Wenbin Yang, Aldo Peter

Abstract

Kobe Bryant is one of the greatest basketball players to have stepped onto a basketball court. He had God gifted basketball talents but nevertheless he believed in hard work and perfection. His talent and drive for greatness led him to a successful NBA career. He is one of the few players to have come straight from highschool into the NBA. Bryant played his entire 20-season career in the National Basketball Association (NBA) with the Los Angeles Lakers. Bryant is an 18-time All-Star, 15-time All-NBA Team, and 12-time All-Defensive team. In his tenure with the Los Angeles Lakers he won 5 NBA championships. Kobe Bryant died in a Helicopter Accident on January 26th, 2020. His sudden death broke so many hearts. As a group and individually, Kobe has played an indirect role in our lives. Kobe is a superstar. As a tribute to Kobe we wanted to utilize this project as an opportunity to showcase his greatness and what he meant to the basketball world.

Description

His presence, impact and influence on and off the court contributed greatly to the revenue streams of both the NBA and LA. With that in mind, we decided to explore his shots selections and his shooting percentages and the impact of his shot selection on his ability to make shots. We will use data on all of his shots to determine which of his shots he made the most. Our target variable will be shot made, with potential features being shot type - shot distance - position on court- time left in-game. Using this model, we will be able to predict the best shot types that a player can take to make his shooting more efficient. This will help us make recommendations for shot types and efficiency levels a player will have to perform in order to increase their salary potentially through a max contract.

Data

Our Kobe Braynt data comes from Kaggle with 30697 rows and 25 columns. This data contains the location and circumstances of every field goal attempted by Kobe Bryant during his 20-year career. This data already removed 5000 of the shot_made_flags (represented as missing values in the csv file). These are the test set shots for which you must submit a prediction. Therefore the data did not require rigorous cleaning. This dataset contains information like playoffs, shot_distance, shot_type.etc which are pretty interesting.

	action_type	combined_shot_type	game_event_id	game_id	lat	loc_x	loc_y	lon	minutes_remaining	period	playoffs	season	seconds_remaining	shot_distance
2	Jump Shot	Jump Shot	12	20000012	34.0443	-157	0	-118.4268	10	1	0	2000-01	22	15
3	Jump Shot	Jump Shot	35	20000012	33.9093	-101	135	-118.3708	7	1	0	2000-01	45	16
4	Jump Shot	Jump Shot	43	20000012	33.8693	138	175	-118.1318	6	1	0	2000-01	52	22
5	Driving Dunk Shot	Dunk	155	20000012	34.0443	0	0	-118.2698	6	2	0	2000-01	19	0

shot_made_flag	shot_type	shot_zone_area	shot_zone_basic	shot_zone_range	team_id	team_name	game_date	matchup	opponent	shot_id
NA	2PT Field Goal	Right Side(R)	Mid-Range	16-24 ft.	1610612747	Los Angeles Lakers	2000-10-31	LAL @ POR	POR	1
0	2PT Field Goal	Left Side(L)	Mid-Range	8-16 ft.	1610612747	Los Angeles Lakers	2000-10-31	LAL @ POR	POR	2
1	2PT Field Goal	Left Side Center(LC)	Mid-Range	16-24 ft.	1610612747	Los Angeles Lakers	2000-10-31	LAL @ POR	POR	3
0	2PT Field Goal	Right Side Center(RC)	Mid-Range	16-24 ft.	1610612747	Los Angeles Lakers	2000-10-31	LAL @ POR	POR	4
1	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	1610612747	Los Angeles Lakers	2000-10-31	LAL @ POR	POR	5

Descriptive statistics

	vars <dbl>	n <dbl>	mean <dbl>	sd <dbl>	median <dbl>	trimmed <dbl>	mad <dbl>	min <dbl>	max <dbl>
action_type*	1	30697	27.85	9.59	27.00	27.03	0.00	1.00	57.00
combined_shot_type*	2	30697	4.09	0.64	4.00	4.10	0.00	1.00	6.00
game_event_id	3	30697	249.19	150.00	253.00	246.92	189.77	2.00	659.00
game_id	4	30697	24764065.87	7755174.89	20900354.00	23103974.32	741638.03	20000012.00	49900088.00
lat	5	30697	33.95	0.09	33.97	33.96	0.11	33.25	34.09
loc_x	6	30697	7.11	110.12	0.00	8.41	126.02	-250.00	248.00
loc_y	7	30697	91.11	87.79	74.00	83.15	109.71	-44.00	791.00
lon	8	30697	-118.26	0.11	-118.27	-118.26	0.13	-118.52	-118.02
minutes_remaining	9	30697	4.89	3.45	5.00	4.78	4.45	0.00	11.00
period	10	30697	2.52	1.15	3.00	2.51	1.48	1.00	7.00

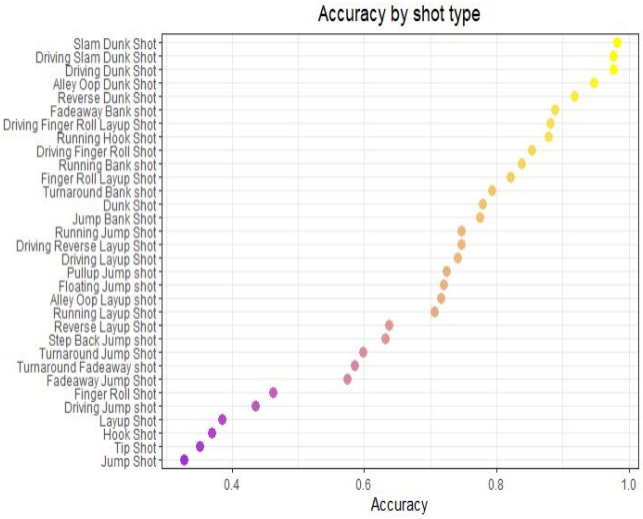
Segments

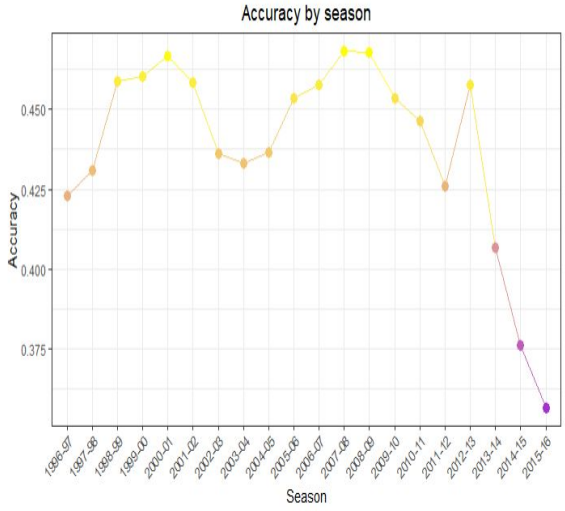
We want to be able to visualize our data better, so we decided to use graphs about Kobe's shot selection as our first step.

Accuracy by season and type

Kobe's shot accuracy for most of his career has been around 45%. As a 18 year old in his first year in the NBA in 1996, Kobe did not play much and was working on perfecting his craft. By 1998, Kobe transitioned into a starting role in the team and his shooting percentage reflected his improved play. His

shot accuracy stayed relatively consistent despite injuries, however over the course of a season, a few percentage increase or decrease in accuracy of your star player could be the difference in making it to the playoffs/championship. In the graph we can see that Kobe shots accuracy began to decrease drastically in the 2013-14 season, which could have been contributed to his old age and lingering injuries from the past. He could have retired then, but since he was still getting paid and the city of LA and the organization loved him, they eased his way into retirement instead of rushing him out by giving him a few 'victory lap' seasons. As for accuracy by shot type, we found dunks have the highest accuracy which makes sense. The thing that surprised us is Kobe's fadeaway bank shot has pretty high accuracy, which is his most famous skill.

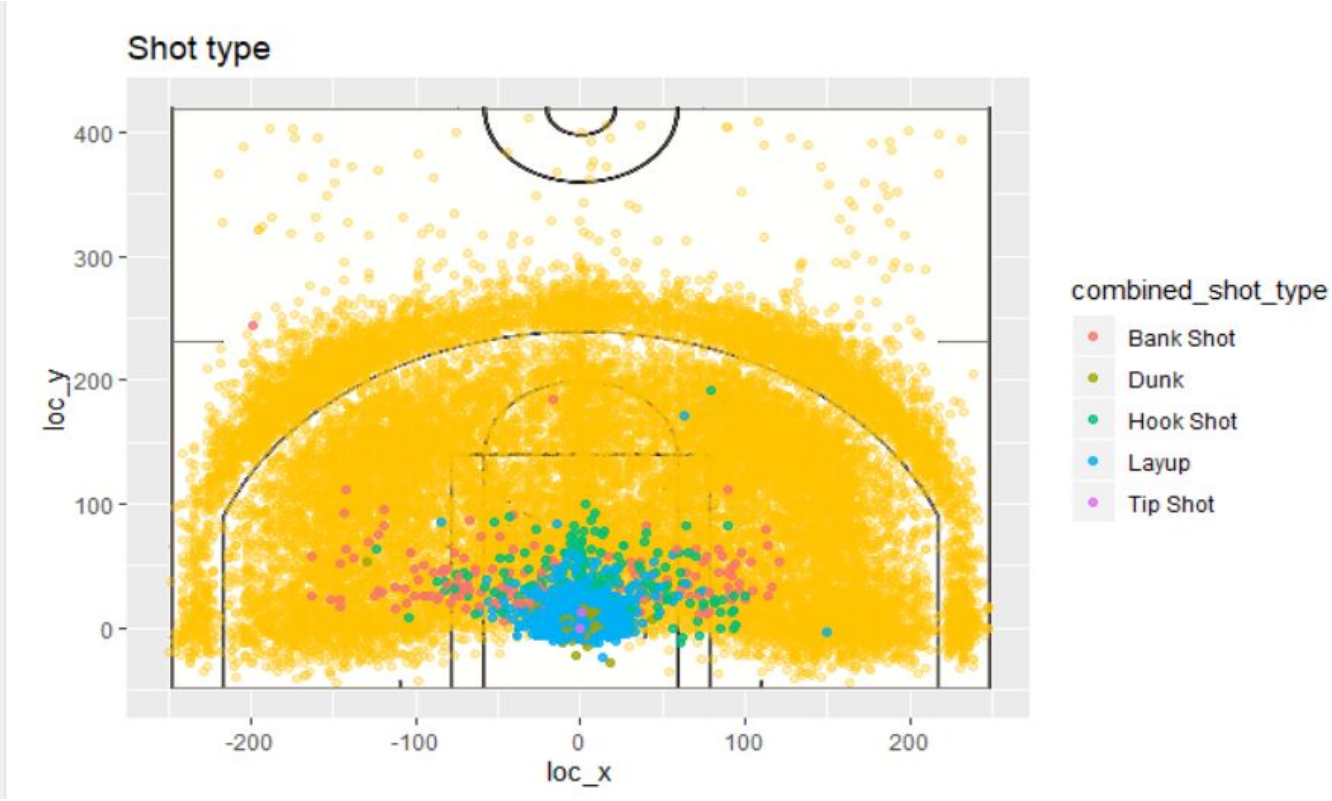




Shot Type

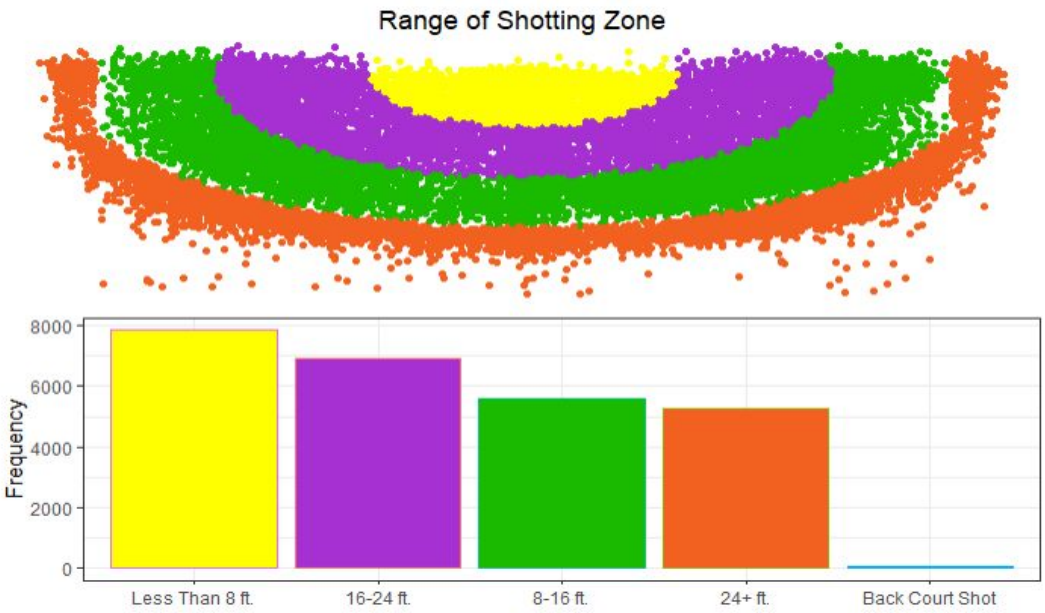
The main NBA shots types include jump shots, hook shot, dunk, layup, bank shot, etc. Kobe took a lot of 2-point jump shots, 3-point jump shots and layups. We found jump shots are one of the main ways Kobe Bryant scored. An interesting data point is the accuracy of his fadeaway jump shot. This is a shot where he is moving away from the basket he is shooting at, while he is shooting. This should be a more difficult shot than a regular jump shot, but his shooting percentage is much higher. This could be partially due to how some shots are coded compared to others.

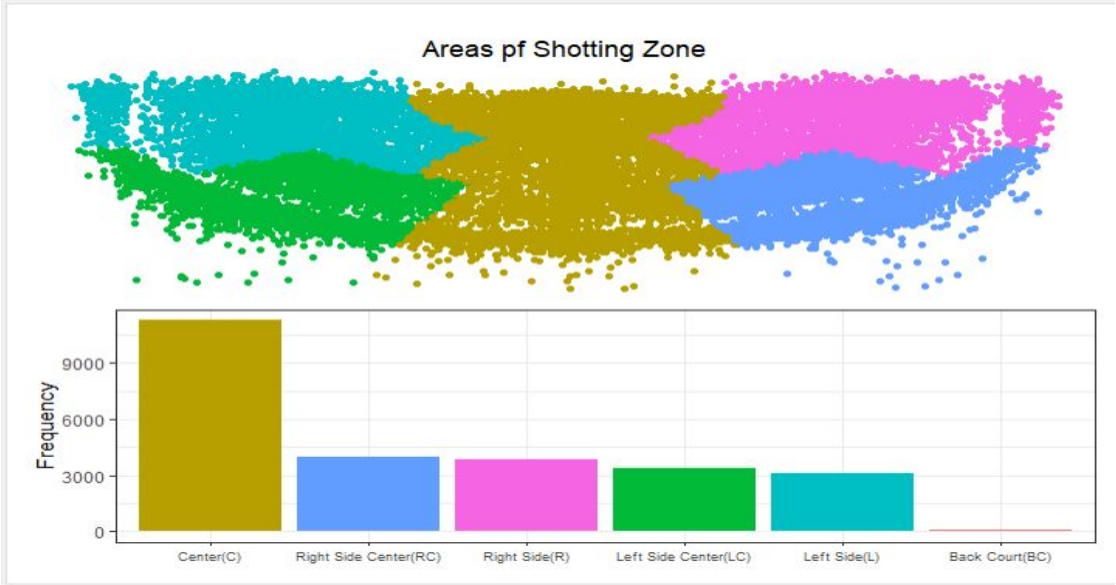
Recommendation: If you want to stop Kobe, try to force him to shoot regular jump shots and hook shots, as he shoots these poorly. If you want Kobe to perform well, highlight dunks and fadeaway jumpers as his main shooting type.



Shot zone range and area

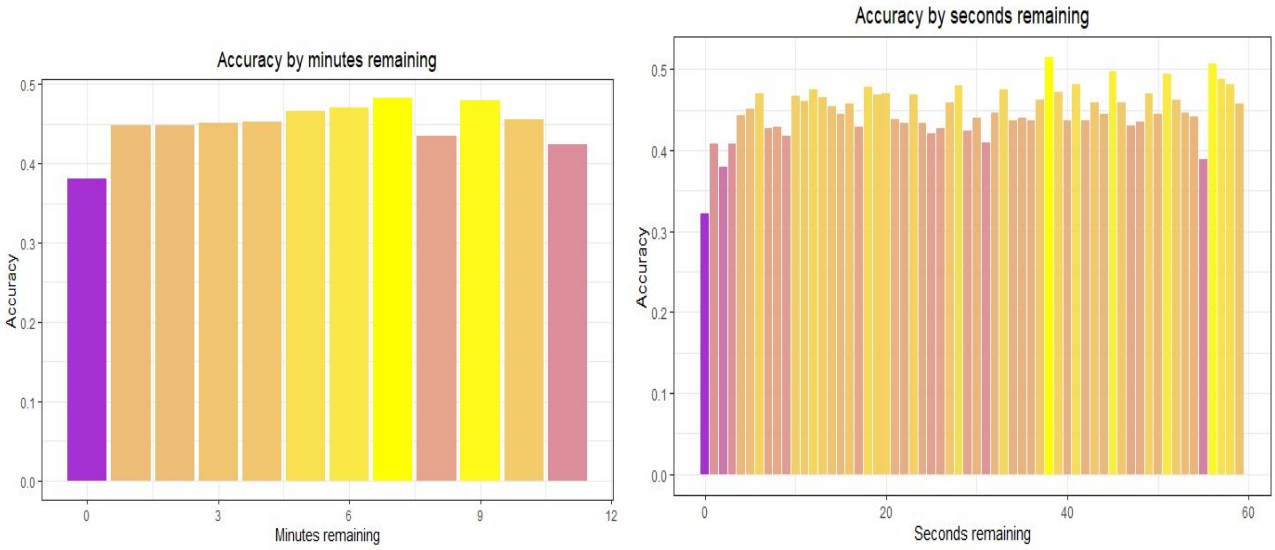
Before the modern NBA mentality of shooting numerous 3-point shots a game. The NBA was dominated by 2-point shots, especially layups and dunks. As we can see Kobe clearly prefered to play through defenders to get to the hoop. We found that Kobe always gets a score in less than 8ft and center of the court, which means Kobe is always driving to the basket and preferred a shot range shot.





Clutch Evaluation

Clutch situations are critical periods in a game or season that have an impact on the overall outcome. In order to evaluate Kobe's performance in clutch situations, we had to observe how Kobe performs in these situations.



Upon review of Kobe's shot performance later in each period, it seems that his shot accuracy actually decreases as the game gets towards the more critical part of each period. Using GLM models to estimate the marginal effects of different clutch factors on Kobe's game we reveal some more interesting aspects of his clutch evaluation. To further evaluate how Kobe performed later in games, we took a look at the marginal effect of being in the fourth quarter on Kobe's shot accuracy. Being in the fourth quarter has an estimated -4.23% marginal effect on shot accuracy for Kobe. This could potentially show us that Kobe is not clutch, seeing as he performs worse the later in the game he plays, but let's review some more attributes about his game.

We also took a look at the marginal effect of being in the playoffs, an inherently clutch situation because winning any game in the playoffs has a high impact on a team's ability to advance to and possibly win the championship, which is the overall goal of every NBA season.

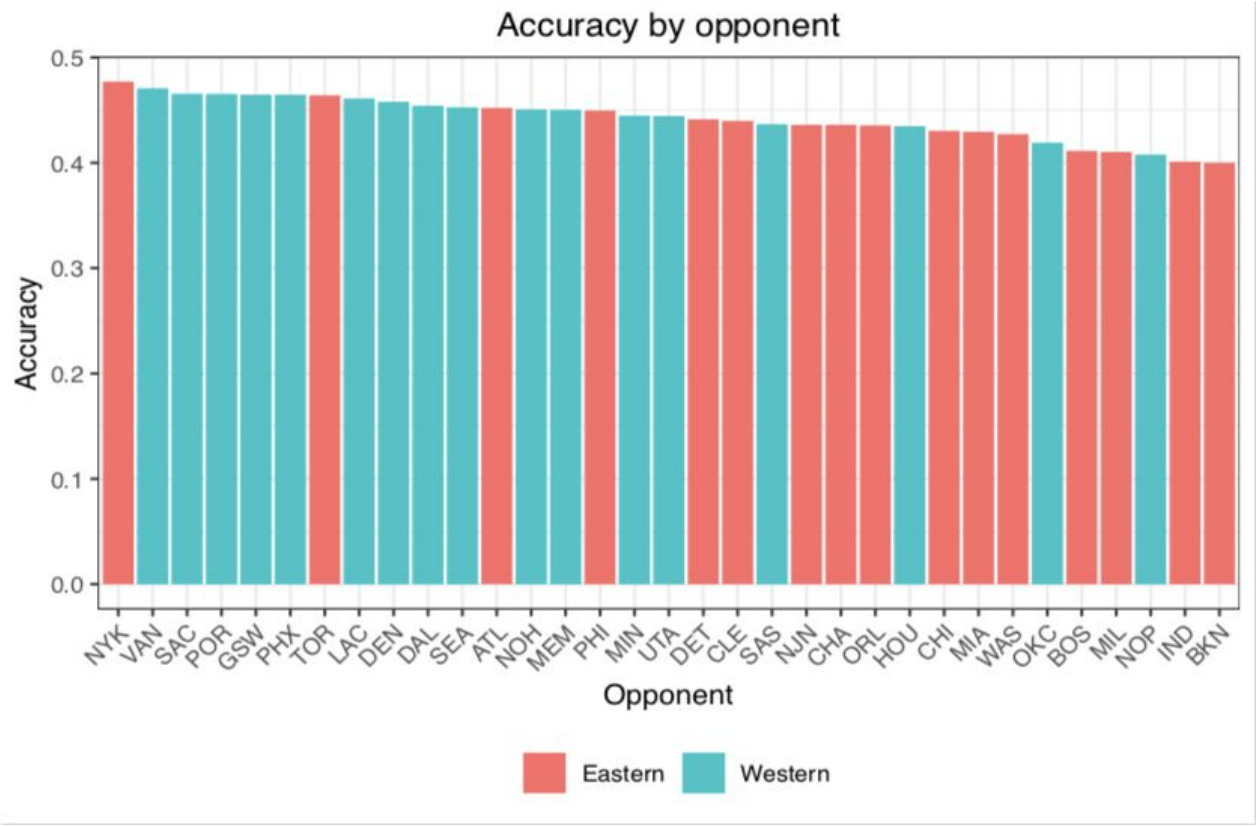
Being in the playoffs has a -0.18% on Kobe's shot accuracy, meaning he shoots worse in the playoffs than normal. However, the competition in the playoffs is harder than the regular season because it is not a random selection of teams, it is only the best teams. When we apply team fixed effects to our model, the marginal effect of being in the playoffs is .01% meaning that Kobe actually slightly ups his game during the playoffs.

Finally, we measured the marginal effect of being in a championship season. We want to see if, when the Lakers ended up winning a championship, did Kobe play better. As it turns out, during his championship seasons (5 of them), there was an .86% marginal effect on his shot accuracy. When the Lakers won their championships, Kobe was playing better.

Can we determine if Kobe is clutch? If anything, we have mixed results. During the ends of periods/games, when a game could be on the line, Kobe's performance decreased. However, this is a look at his performance across all games, not just important games that could be decided in the final few minutes. If the Lakers were already well ahead of their opponent late in a game, this would not be a clutch situation. Also, on the positive side, when Kobe was in the playoffs or on a championship team, the teammates around him were possibly better compared to other years, positively affecting Kobe's game without there being an impact on his clutch factor.

Since there are so many factors that we could not determine with our data, such as the importance of each game, the score at different points of the game, the Lakers team record at the time, the opponent guarding Kobe, the overall shot difficulty, and the positioning of the person guarding him, we don't think we can make a decision as to whether or not Kobe is clutch. However, due to his increase in shot accuracy during the playoffs and championship seasons, we want to lean towards Kobe coming through in the clutch, but due to our own biases and that there is not enough clutch specific data, we cannot\ positively make that statement.

Performance vs. Opponents



Above we have graphed Kobe's shooting accuracy against each team he played during his career. This graph can tell us about not only Kobe's game, but also a bit about the evolution of the NBA over the past 20 years. First, we see that Kobe performs better against Western conference teams than Eastern conference teams. Since the Lakers are also in the Western conference, this could mean that Kobe ups his game when he plays against team more often, or when he plays against a rival.

However, we do notice one Eastern conference team all the way at the top. The New York Knicks. This tells us two things. One, the Knicks were a notoriously bad team for most of the seasons Kobe played against them. However, they were rarely the worst team. This could be explained by the Knicks playing in Madison Square Garden and having a big fan base. Many great basketball players have loved playing at MSG and performing their best on the big stage, because MSG is in downtown Manhattan so there is always high attendance. In fact, Kobe Bryant broke Michael Jordan's record of 59 points scored by an opponent at MSG with 61 points. A record he currently still holds with James Harden. We can

see through the graph above that Kobe was like MJ and other big name players, he liked to perform well at MSG against the Knicks.

We can also see a few teams in this graph that no longer exist. For example, Kobe performed very well against the Vancouver Grizzlies (now known as the Memphis Grizzlies). Even though the Vancouver team was only around during Kobe's initial seasons, he performed so well against them because the Grizzlies, when in Vancouver, were a bad team and never won above 30% of their games. On the other end of the accuracy chart, we see some new teams, such as the New Orleans Pelicans and the Brooklyn Nets, the team Kobe shot the worst against in his career. The reason for his poor performance against these teams was probably not because they defended him well, but because these teams only existed during the later portion of Kobe's career. Most of the games he played against these teams were during his final two seasons when his shots accuracy had declined across the board.

One team that should be highlighted as a 'Kobe Kryptonite' is the Indiana Pacers. A career 44.7% shooter, Kobe only made 40.1% of his shots against the Pacers. The Pacers team had also been a constant in the NBA throughout his career, so the sample of his games against the Pacers spans his entire career. So how did the pacers stop Kobe? The answer is tough, but most likely it was speed and length. Early in his career, Kobe would have been guarded by Reggie Miller or Ron Artest of the Pacers. Later on, the Pacers would have had Jermaine O'Neil to stop Kobe's driving shots, and Paul George, an elite perimeter defender, to follow Kobe step by step. If you want to stop Kobe, a player like Paul George is necessary. Tall, long arms, and incredibly fast. That's the only way to contain him. We can see from the accuracy graph the Kobe dominated teams that do not have this type of rare player.

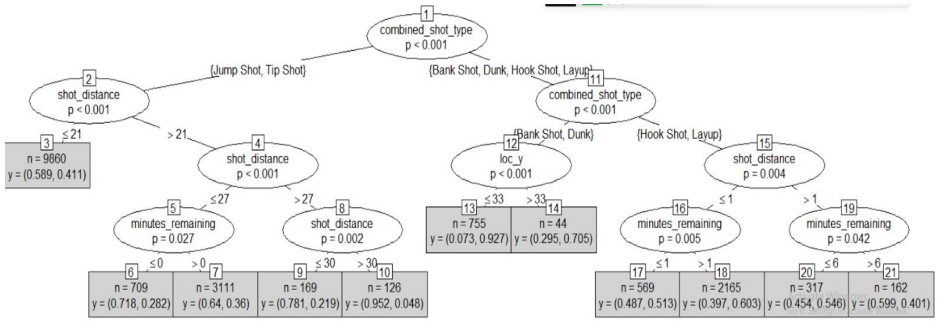
Recommendation: If you want to stop Kobe, try to emulate the Pacers style of defense. Also, try to get a player on your team that is an elite perimeter defender, like Ron Artest or Paul George. If you want Kobe to perform well, try to get him to emulate his play against the Knicks.

Models:

The purpose of our models part is figuring out which mode is better to predict the accuracy of Kobe's shooting. At the beginning of building models, we did model preprocessing as our first step. We want to predict "shot_made_flag", this column is a dummy variable which contains 0 and 1. 0 means missed shot and 1 means made shot. However, this variable also have some NAs which means kobe got foul when he was shooting. So we removed the rows which "shot_made_flag" = NA. Then split our data into 70% as training data and 30% as testing data. In our data, some columns have duplicate meaning, so we decided to combine them as one column , for example, there are remaining_minutes and remaining_sceonds, so we combined them together as remaining_minuses *60 + remaining_second. Also, we selected shot _distance > 40 as 40 because most shooting over 40ft are missed. We also transferred all character variables to factors , this way made our data working better for our models. Finally, we use the package "caret" to find the most important features for our model. Like the graph showing, the combined_shot_type, period and opponent are more important.According to these result, we build a formula : "as.factor(shot_made_flag) ~ combined_shot_type + period + opponent + season + loc_x + loc_y + shot_distance + time_remaining"

	Overall
combined_shot_typeDunk	4.17822498
combined_shot_typeHook Shot	3.18304625
combined_shot_typeJump Shot	5.51106310
combined_shot_typeLayup	4.35670821
combined_shot_typeTip Shot	6.01839584
period	3.46452861
opponentBKN	0.62080401
opponentBOS	1.08473452

RandomForest:

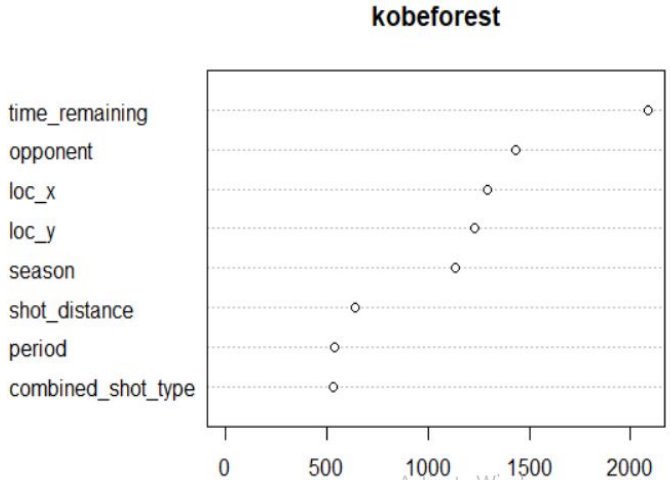


First model we choose Random forest because our target variable is binomial, so classification models really fit our data, also , Random Forest has pretty good performance for classification or regression problems. The predictive perf ormance can compete with the best supervised learning algorithm and offer efficient estimates of the test error without incurring the cost of repeated model training associated.The graphs shows the first tree in my model, we setnode = 1 and mytry = 10 which is equal to our features.

Evaluation:

```
          Reference
Prediction    0    1
         0 2998 1275
         1 1970 1467

               Accuracy : 0.5791
                 95% CI : (0.568, 0.5902)
    No Information Rate : 0.6444
```

```
            Sensitivity : 0.7016
            Specificity : 0.4268
         Pos Pred Value : 0.6035
         Neg Pred Value : 0.5350
             Prevalence : 0.5542
         Detection Rate : 0.3888
   Detection Prevalence : 0.6444
      Balanced Accuracy : 0.5642

       'Positive' Class : 0
```



We found the variable “time_remaining” has the most importance. We built a confusion matrix to evaluate our model, we found the accuracy is 58% which is not too high, Precision is only 54% and recall is 43%, both are pretty low, I think the reason is our dataset doesn;t contain too much factors, the columns combine_shot_type has the most categories which is only 5. This is also the disadvantages of RandomForest.

GLM

The second model we choose the logistic regression , we choose this model because our independent variable is binomial and nonlinear. And output for logistic regression is also binomial , so we set ’family = binomial” to run our logistic model. Then we set the mean of prediction as threshold which is 0.5.

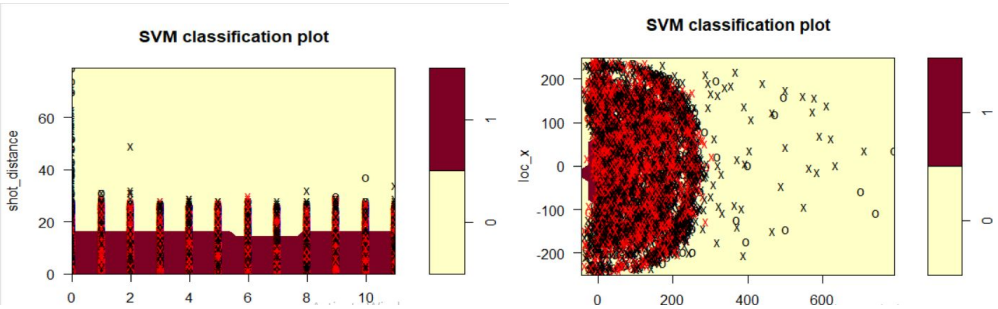
Evaluation:

```
          Reference
Prediction    0    1
         0 3639 2310
         1  634 1127

               Accuracy : 0.6182
                 95% CI : (0.6072, 0.629)
    No Information Rate : 0.5542
    P-Value [Acc > NIR] : < 2.2e-16
```

```
            Sensitivity : 0.8516
            Specificity : 0.3279
         Pos Pred Value : 0.6117
         Neg Pred Value : 0.6400
             Prevalence : 0.5542
         Detection Rate : 0.4720
   Detection Prevalence : 0.7716
      Balanced Accuracy : 0.5898

       'Positive' Class : 0
```

We found the accuracy is 62%, recall is 33% and precision is 64% which is much better than RandomForest.

SVM



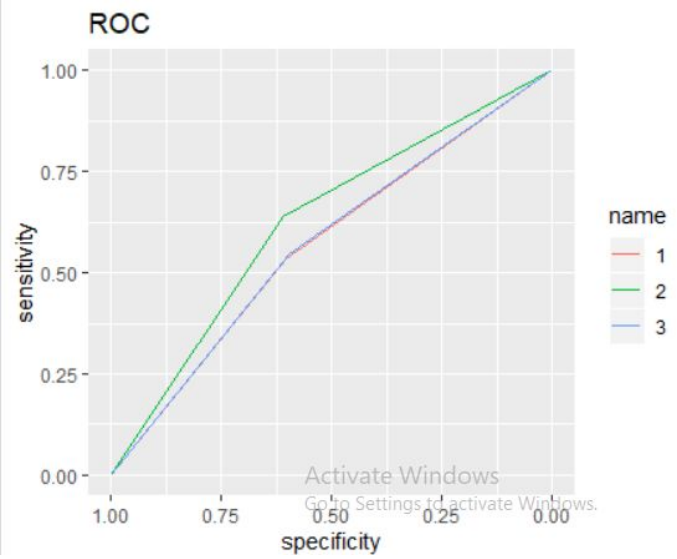
The third model we decided to use SVM. SVM can for linear or nonlinear data. Svm works well when there is clear margin of separation between classes and is relatively memory efficient. So we setup the weights between each class is (1,1). Also set kernel=”radial” which means classification. From the plot form SVM classification, we know this model predicts more positive if the shot_distance is small. Thats pretty make sense.

Evaluation:

```
          Reference
Prediction    0    1
         0 3258 2208
         1 1015 1229

               Accuracy : 0.582
                 95% CI : (0.5709, 0.593)
    No Information Rate : 0.5542
    P-Value [Acc > NIR] : 4.729e-07

                  Kappa : 0.1243

 Mcnemar's Test P-Value : < 2.2e-16
```

```
            Sensitivity : 0.7625
            Specificity : 0.3576
         Pos Pred Value : 0.5960
         Neg Pred Value : 0.5477
             Prevalence : 0.5542
         Detection Rate : 0.4226
   Detection Prevalence : 0.7089
      Balanced Accuracy : 0.5600

       'Positive' Class : 0
```

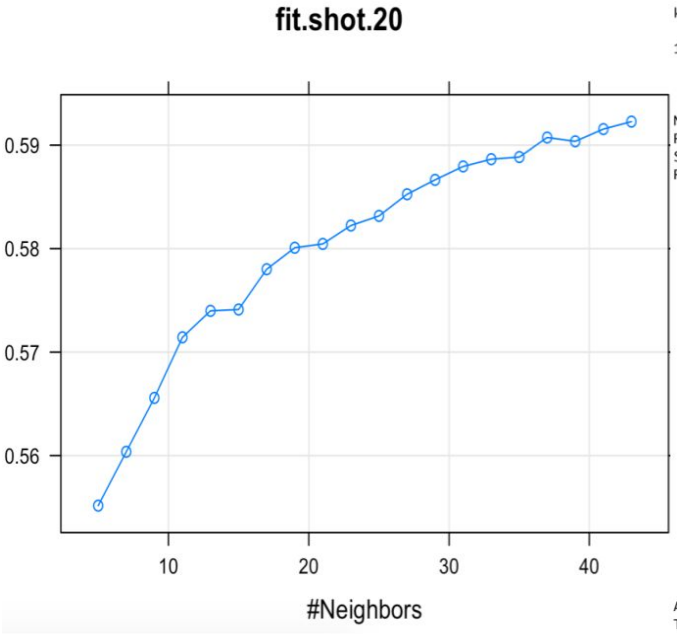
We found the accuracy is 58%, recall is 36% and precision is 55%.The reason might be our data has too many integers and levels of factor columns is too small.Then we created a ROC plot for these three models, we found logistic regression has the highest ROC and accuracy.



Model 4

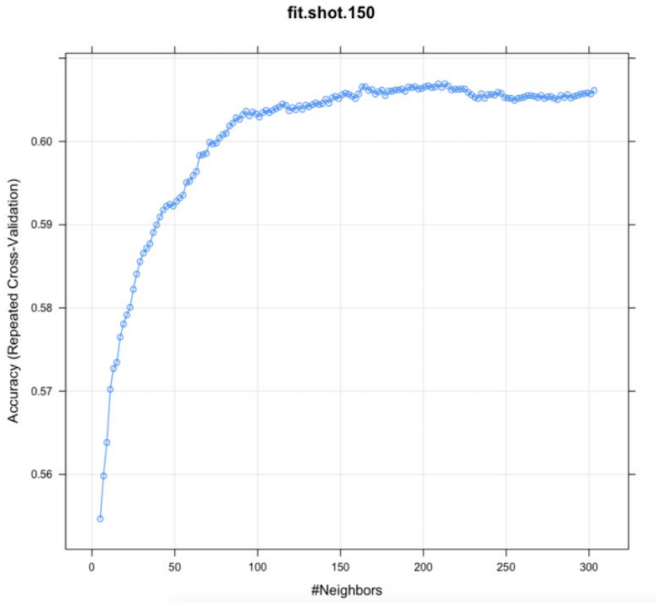
K-nearest neighbors(KNN)

We also used the same training data and test data above for our k-nearest neighbors(KNN) model. To build the KNN model, we first needed to determine the K value. So we tried two ways to build this model: one is using train function and making sure the method equals knn, the other is using kknn function. For the train function at first we made the tuneLength equal to 20, then we plotted it as followed:




```
k-Nearest Neighbors

17987 samples
    5 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 16188, 16189, 16189, 16188, 16188, 16189, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.5551596  0.09599042
   7  0.5603674  0.10253622
   9  0.5655751  0.11038676
  11  0.5714308  0.11989268
  13  0.5739887  0.12307317
  15  0.5741188  0.12115163
  17  0.5780104  0.12794217
  19  0.5800863  0.13074387
  21  0.5804569  0.13024544
  23  0.5822359  0.13265216
  25  0.5831625  0.13318039
  27  0.5852564  0.13633121
  29  0.5866464  0.13813552
  31  0.5879438  0.13975711
  33  0.5886473  0.14045609
  35  0.5888513  0.13992670
  37  0.5907419  0.14329132
  39  0.5903710  0.14186407
  41  0.5915570  0.14367671
  43  0.5922794  0.14475903

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 43.
```

We can see from the figure that the larger the neighbor value, the higher the accuracy. Obviously this graph cannot predict where the best accuracy is going to be, so we tried tuneLength equal to 50, 80 and so on. Finally, we chose 150 as tuneLength. Then we found when k is 213, the accuracy is highest.




```
k-Nearest Neighbors

17987 samples
    5 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 16188, 16187, 16188, 16189, 16189, 16189, ...
Resampling results across tuning parameters:

  k    Accuracy   Kappa
  5  0.5546610  0.09451026
  7  0.5598125  0.10156663
  9  0.5638341  0.10681922

  207  0.6065493  0.16626660
  209  0.6069198  0.16702493
  211  0.6064937  0.16611184
  213  0.6069385  0.16704567
  215  0.6066605  0.16643868
  217  0.6061788  0.16535768

  291  0.6054190  0.16303167
  293  0.6055302  0.16332925
  295  0.6056970  0.16359670
  297  0.6057340  0.16361933
  299  0.6058267  0.16386722
  301  0.6057154  0.16363114
  303  0.6061233  0.16448385

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 213.
```

The confusion matrix and Accuracy as follows:

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 8436 5492
         1 1519 2540

               Accuracy : 0.6102
                 95% CI : (0.603, 0.6174)
    No Information Rate : 0.5535
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.1719
```

Model 5

Second KNN model

After determining the value of k, we can use the KKNN function to make our second KNN model, which accuracy is 60.71%, very similar with the model which was created by train function.

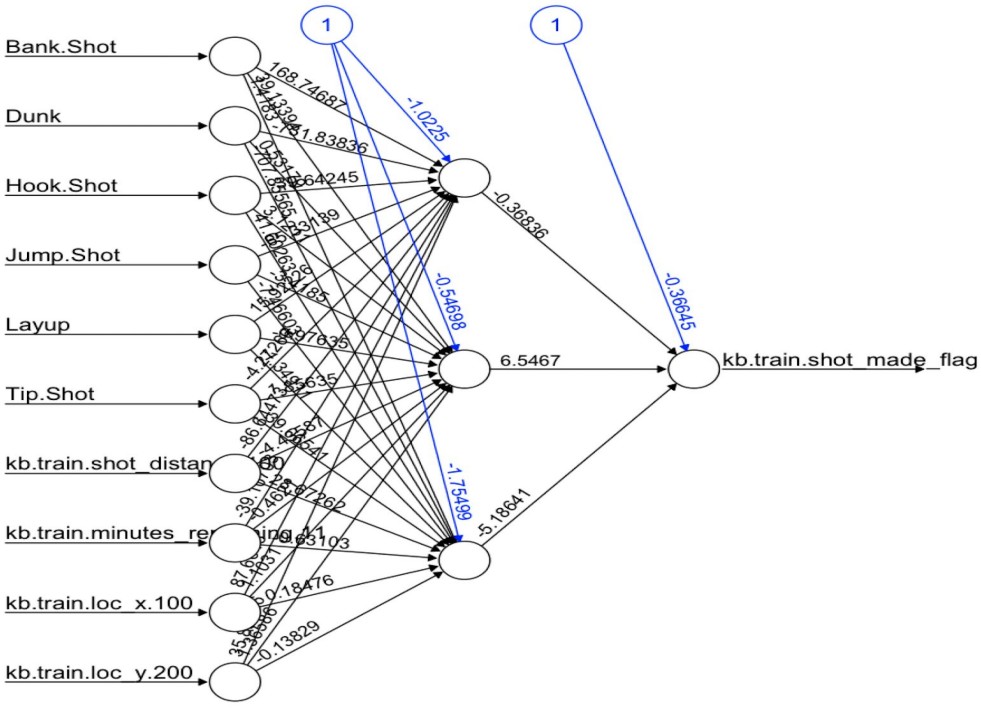
Confusion matrix

Prediction	0	1
1	8584	5455
0	1371	2577

Model 6

Neural Networks

Before building the NN model, we picked categorical variables. We used the ‘class.ind’ function in the ‘nnet’ package to dummy code it. We found that some values in some columns were big, such as loc_x and loc_y, so, we divided them by 100 and 200 to limit the variance of our data.



Conclusion/Deployment

Looking through the Kobe data, we can see that Kobe had the opportunity to take as many shots as he would like. Perhaps that's why his efficiency levels are not as high as we expected. From our models we can conclude that Logistics Regression is the best model we have when predicting Kobe’s shot outcome

but nevertheless our other models do a relatively similar job.In our data, "shot_made-flagged" variable also have some NAs which means kobe got foul when he was shooting. After comparing models, we also splitted data as shot_made_fagged = NA/noNA by using glm to calculate the shooting accuracy if kobe didn't got foul at that time. We got 42.6% which is pretty close to his career. Although we have not tested it on other players' data, we believe our model or at least a similar model could be used to predict any players' shot outcome. Using this information, coaches and players could scheme to maximize the type of shots that are predicted to go in and minimize the type of shots that are predicted to miss. This would hopefully lead to more efficient scoring for any player/team that utilizes this model. The data preprocessing for models also can develop, the main reason impacting our accuracy is that the amount features are too small, so we might split some columns into more categories.