Assignment 2: Predicting Kobe Bryant's Made Field Goals

[Report]
Vishaal Prasad
University of California, San Diego
A10709125

1. INTRODUCTION

I'm an avid NBA fan, and I grew up a huge Lakers fan. Naturally, that meant I was a huge Kobe Bryant fan as well. After Kobe's career came to an end in April 2016, Kaggle released a dataset charting every single shot he took in his career, as well as associated data: who the opposing team was, type of shot, distance of shot, date, which quarter it was, etc. The Kaggle task, which is a playground task done for fun, is to predict whether a withheld shots went in or not. He took roughly 30,000 attempts total, with 5,000 held out as the test set.

To begin with my analysis, I performed a series of basic analyses to understand certain relation of the data better.

Of the given data, Kobe made 11,465 field goals out of 25,697 field goal attempts = 44.62% FG% (FG% = field goal percentage, the percent of shots Kobe took that he made). This is our mean dataset, and we would use this value if we were building a trivial classifier.

We may suspect that depending on whether the game was in the regular season or the postseason (where defenses are better), Kobe's FG% may vary. We find that his regular season FG% is 44.64% and his postseason FG%: 44.46%. This is not a big gap, but it's explainable. Kobe's final 3 seasons were marked by recovery from an injured Achilles, which is a severe basketball injury. His overall FG% dropped significantly, Relatedly, the Lakers were not good enough to make the playoffs, so his playoffs FG% remained mostly untouched. I actually remember that before he tore his Achilles, his regular season FG% was 45.1%, so we can see that indeed a regular season/ postseason gap did exist.

Speaking of the Achilles injury, we expect there to be a relationship between season number and FG%, particularly his first and last years where his percentages were low (youth/ injury). Indeed, we see in Figure 1 that this is the case.

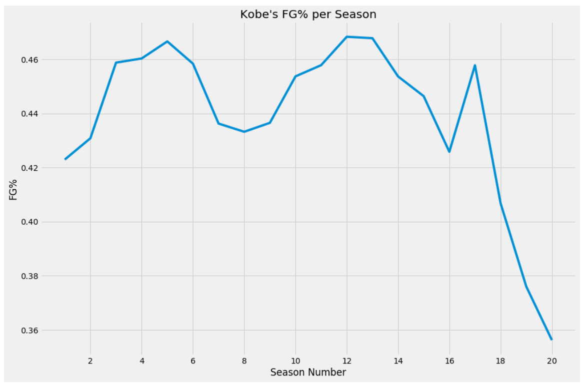

Figure 1: Kobe's FG% per season. We see a precipitous drop in the final 3 seasons. This matches intuition for how Kobe's style evolved over the years.

The next thing I want to examine is how his shooting varied by quarter. Intuitively, we expect that because Kobe often rested in the 2nd and 4th quarters, we should see fewer shots here than in the 1st and 3rd. Figure 2 below validates that hypothesis.

Figure 2 also reveals that his accuracy drops in the 4th quarter significantly, but otherwise strays around 44.5%-46%. This again passes intuition; Kobe had a reputation of a "closer" and took many difficult shots as a result of that, lowering his FG%. Note that the "5th" period encompasses all Overtime periods.

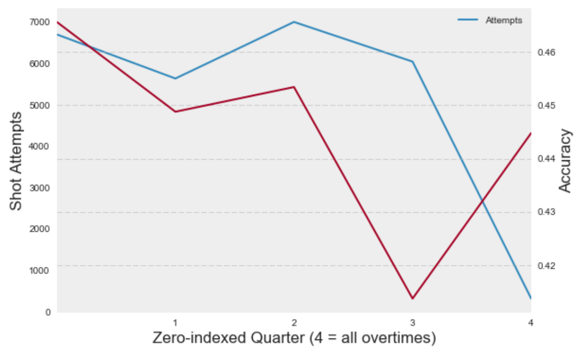

Figure 2: Field Goal Attempts and Accuracy per Quarter, zero-indexed. Index 4 is all overtime periods.

From Figure 3, we find out the vast majority of his shots are jump shots. This isn't a surprise watching him play, but it does mean that we should evaluate accuracy by shot type as well. I debated stacking a plot of accuracy on top of this, but it's already clear that we can reasonably condense this category into a single binary variable: is_jump_shot.

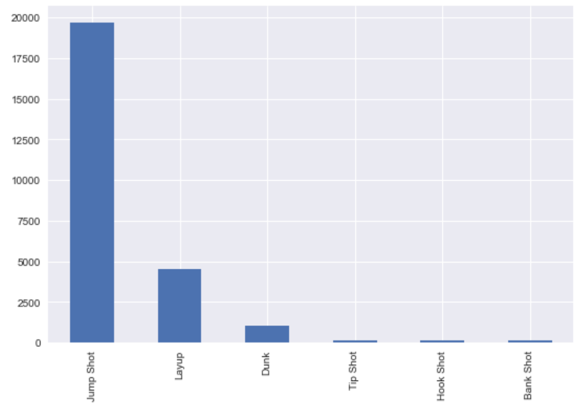

Figure 3: The majority of Kobe's shot attempts were jump shots, followed by layups and dunks.

However, jump shot itself isn't that descriptive. Kobe was known to be an excellent midrange shooter (anywhere from 10-23 feet, roughly), but his 3-point shooting was fairly average (24+ feet). However, it is unclear how exactly we would plot that. We are given categories in the data, but do they convey anything meaningful within the data? Instead, let's create them from the data ourselves. To do this, I use a clustering algorithm (Gaussian Mixture Model) to model shot preferences (e.g. layups vs. baseline fadeaways vs. 3 pointers) as Gaussians. Because this analysis is more sophisticated, I leave it to Section 3.

Finally, one might consider there were characteristic points during a game where Kobe behaved differently than the norm. Prevalently, at the end of quarters and (especially) at the end of games, Kobe had a tendency to shoot difficult shots, both in terms of distance and in terms of defenders guarding him. We would expect his percentages to be lower here. To get this information required data massaging – I am not that familiar with Pandas, and so this code was taken from the kernel cited in [1]. I do not claim that this is my work!

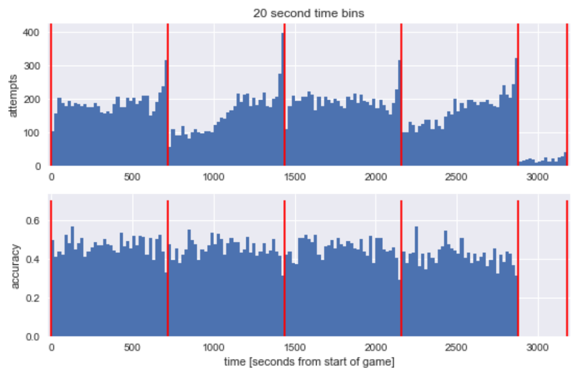

Figure 4: Kobe's shot attempts (top) and accuracy (bottom) during each 20 second bin of the game. Note the decreased accuracy at the end of quarters (red bars).

Indeed, we see this to be the case. Therefore, it may be worthwhile to include this feature.

2. PREDICTIVE TASK

The predictive task, as noted before, is to predict whether some withheld shots from Kobe's career went in or not. This is evaluated via Kaggle by log-loss (*not* classification accuracy). Note their methods ask to predict a shot only by the shots that come before it (to avoid leakage), but for the purposes of this assignment, I'll disregard that. So each event will have the full dataset to work with.

For purpose of evaluating my model, Kaggle has a public leaderboard on which I could evaluate this model. However, unfortunately the leaderboard is closed. Therefore, I will simply create a held out test set of 5000 shot attempts and train on the other attempts. I'll still use log-loss to compare to the methods on the Kaggle leaderboard. The median is at about 0.6100, but I do not expect to compete even with that. I'd like to get to the first quartile (i.e. 25th percentile) which is about 0.665 log-loss. Note that 0.69315 log-loss is random guessing (i.e. always predicting a miss).

We are asked to describe our features here, but I elaborate on 3b.

3. MODEL

a. Predictor

This problem is a pretty classic logistic regression task. There are two categories, a made shot and a missed shot, and we want to predict *the probability* that a given shot goes in. Furthermore, the error function is log-loss, which is the same objective function which logistic regression optimizes for. Therefore, a logistic regression model is well-justified here. Training it will be done via a standard numerical optimization routine (L-FGBS) as we did in class. The specific routine doesn't really matter as logistic regression is convex.

There are other aspects to be considered. In terms of overfitting, I will note in the following subsection that I won't have many features. Mostly the ones that were described in the introduction and a few that will be described in the following subsection. However, I will use L2-regularization to fight overfitting as well.

Logistic regression was always my first choice to use here. It fits the problem description well, it is simple and out-of-the-box, and I have a strong intuition about it. However, for the sake of comparison, I tried a few other methods. In terms of using another simple method, I used Fisher's LDA. As for more complex classifiers, I used random forests.

There are a few weaknesses and strengths of the models. I provided an overview of why I chose logistic regression above. Fisher's LDA makes the assumption that the two classes come from a multivariate normal distribution with the same covariance matrices. This is a strong assumption I am not sure holds up. On the flip side, if the assumption is true, then we would expect to see better results. In contrast, random forests are a powerful and flexible tool. For the most part, I would expect them to work well. However, they are computationally intensive at test time.

I discuss failures and successes in Section 5. However, one thing to note is that as this was tested on log-odds, I tried predicting on the probabilities as well as the class predictions (i.e. rounding the probability to 0 or 1). Unsurprisingly, using class predictions led to a drastically larger log-loss (i.e. 1.8 instead of 0.67)!

b. Feature Space

The features that I chose to utilize were based on the insights that I gained from the introduction section. To briefly recap:

- The season number of the shot: after Kobe tore his Achilles, for example, the shots that he usually made he often missed, and his overall percentages are down. As is discussed in Section 5, I play around with this feature's representation a bit.
- The quarter of the shot. Self-explanatory, again encoded as a single variable. As with the previous feature, I play around with its representation in Section 5.
- Whether the shot was in the last 24 seconds of the quarter.

In addition, I need to incorporate the type of shots taken. The easy thing to do was to have one on layups and another on jump shots. But that still struck me as over simplistic. As mentioned in Section 1, Kobe had his hot spots, for example. He was a noted midrange (16-23

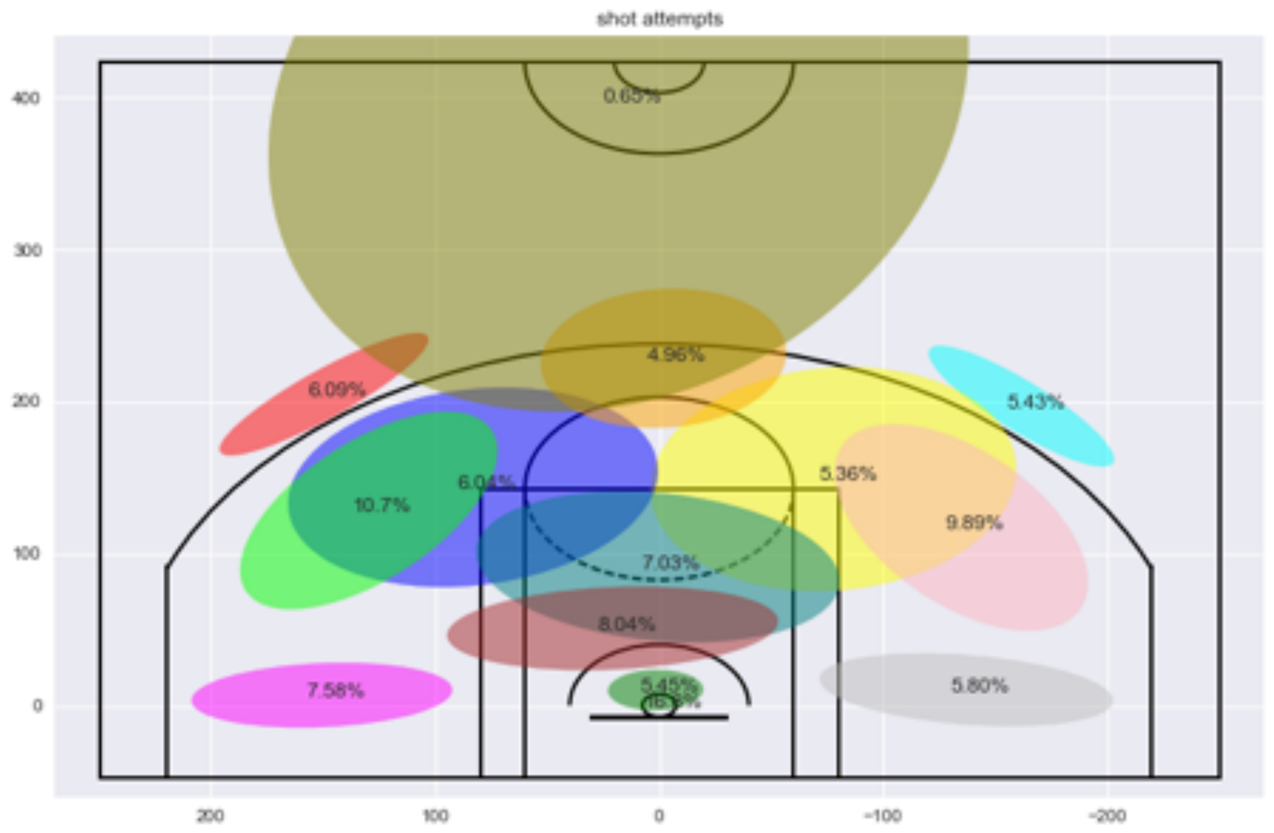

Figure 5: This chart, generated by code in [1] and [2], shows a GMM of where Kobe's shot attempts are produced. These attempts fit our general notions of the type of shots Kobe shoots.

feet) shooter, but not as great from the 3 point line. And even from midrange, he had his hot spots. It would be nice to have a method to automatically pick out related ranges. To that end, I decided to use a clustering algorithm. The best choice was a GMM, since I didn't expect them to be circular like k-means assumes. *Again, the code was implemented in [1] and I simply picked it from there, doing some hyper-parameter tuning based on my intuitions. Note that the great visualization code was taken from [2].*

In Figure 5, we can broadly cluster the shots Kobe tended to take. The largest eclipse at the top represented most of the general heaves he took, for example as a quarter was ending. The red and sky blue eclipses represent "wing 3s" (notice how they are strictly outside the 3-point line). The orange one encompasses shots directly inline from the basket, outside the 3-point line as well as just inside it. The dark blue and yellow circles represent shots "at the elbow." These were locations where Kobe often excelled [3-4]. The overlapping green and light pink jump shots were often "pullup" jump shots on which he had slightly lower effectiveness. The magenta and gray circles were turn around fadeaway jump shots – very difficult shots with likely lower overall FG%. Finally, we see three clusters of shots in the "paint" (teal, maroon, and green).

Figure 6 (next page) shows his actual percentages on each of these clusters. Most of the results fit my previous analysis, though his low percentage at the left elbow (yellow circle) is a surprise. Again, this analysis & plotting come from [1] and [2] respectively. I do not take credit for them.

Consequently, we have one more feature to put in our feature vector: membership to a particular cluster. To contrast this feature, I also will use absolute shot distance, with no context as to where on the court it was.

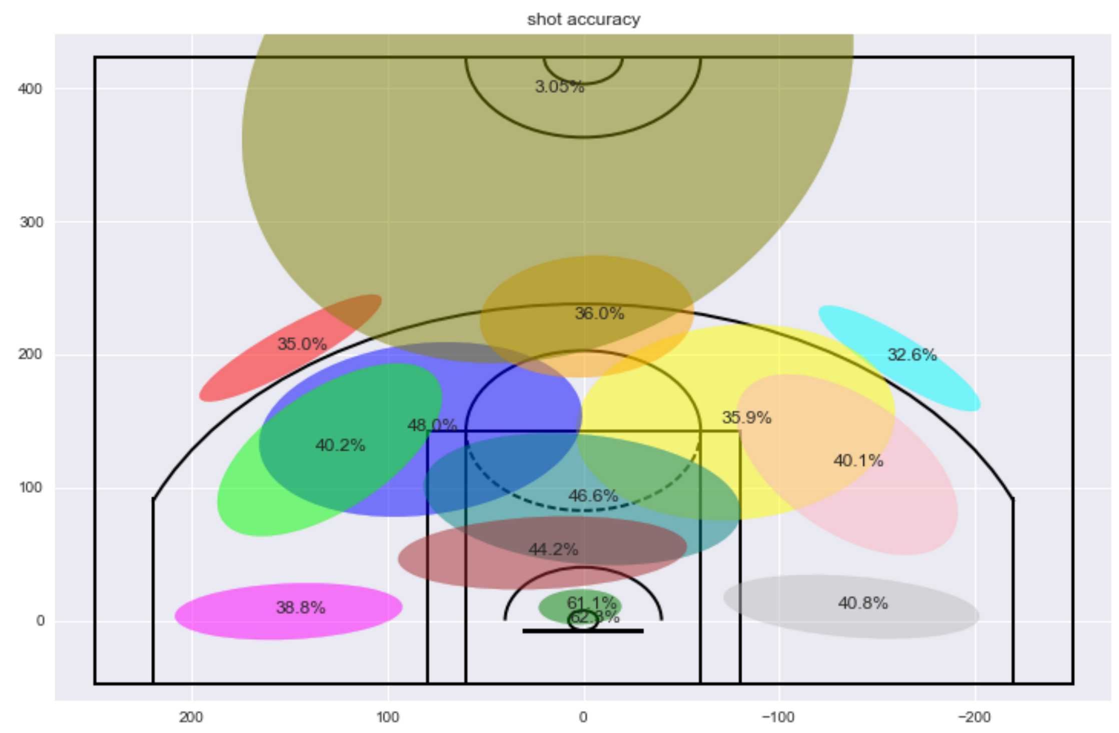

Figure 6: Accuracy of Kobe on each shot. We see results are fairly intuitive given prior knowledge.

4. LITERATURE

The methods that I'm using (logistic regression, Fisher's LDA, and random forests as well as GMMs) are fairly fundamental methods. I won't get into more detail about them.

In terms of literature about this dataset, it should be apparent that no proper analysis has been done on this exact dataset (of Kobe's jump shots). However, the general purpose of predicting which shots are good shots that you'll make is a relevant one. On Kaggle, there is the dataset [5]. This playground dataset is for basketball players in general, but it should generalize to Kobe as well. Reading the sample kernels, however, it is clear that *machine learning* methods aren't being used, i.e. no inferencing, clustering, or prediction is being done. For example, people are calculating statistics to see whether or not the "hot hand" effect exists (the idea that when a player makes a shot, he's "hot" and more likely to make the next shot). That's interesting but not exactly an inferencing type of analysis.

[6] is a very interesting paper that treats the problem of prediction as a recommender system. Specifically, they note that most player-shot location-outcome triplets are sparse. So instead, they treat this as a recommender system problem where they attempt to recommend shot attempts that a certain player should take. This is done via a latent factor matrix factorization, the sort of thing we did in rating prediction in Assignment 3. Their results outperform an SVM in predicting the probability of a shot going in. However, beyond that result, the actual recommendations are trivial. For example, it recommends that players shoot layups or go for dunks instead of jump shots, which is obvious.

Finally, we can turn to some of the publicly available kernels on the Kaggle set I am training on as a sort of literature review. [7] is an excellent comparison of models (note: I did not draw any inspiration from this kernel, as my analysis was done before looking this one up). This kernel shows that logistic regression and LDA perform about as well as more sophisticated methods such as a voting ensemble, gradient boosting, and random forests. K-NN also performs worse than logistic regression. These results agree with the results I find and discuss in Section 5: Logistic Regression and LDA perform equally well for this user, and random forests perform worse than both (though the user does not state how many trees were in his forest).

I do feel that these methods validate my choice of logistic regression. I do more creative feature engineering than this user, as I found in Assignment 1 (helpfulness) that features matter more than the model (arguably).

Finally, we arrive at the results of my model.

5. RESULTS AND CONCLUSIONS

a. Results

I tested three different models using the same feature space as outlined in section 3b. The results are shown in Table 1.

Model	Log-Loss
Logistic Regression	0.67100
Fisher's LDA	0.67103
Random Forest (10)	1.87293
Random Forest (100)	0.94860
Random Forest (1000)	0.84584
Random Forest (10000)	0.82833

In Table 1, we see that even with zero parameter tuning, Logistic Regression outperforms Random Forests up to a huge size of 10,000 trees (which takes a long time to compute). In fact, random forests perform worse than random (0.69315 was baseline, to reiterate). Furthermore, Logistic Regression and Fisher's LDA perform nearly the exact same.

The next step is to evaluate the importance of the GMM membership versus just using the absolute distance. Because GMM membership was not set up as a one-hot encoding, there is reason to believe that absolute distance provides more information than GMM membership. All comparisons are done using Logistic Regression.

Features Used	Log-Loss
Neither absolute distance nor GMM membership	0.6840
Absolute distance	0.6709
GMM membership	0.6831
Both absolute distance and GMM membership	0.6710

We can see from this table that absolute distance is far more valuable than GMM membership. But what happens if we encode GMM properly, via one hot encoding?

Features Used	Log-Loss
GMM membership	0.6702
Both absolute distance and GMM membership	0.6701

We can see in Table 3 compared to the previous Tables that the one hot encoding of GMM does outperform other representations of this feature. Furthermore, absolute distance doesn't add much additional predictive power beyond what GMM membership provides.

Finally, I look to evaluate the other three features **while using the one-hot encoding** of GMM membership but not absolute distance.

Feature Not Used	Log-Loss
All features used (baseline)	0.6702
Last 24 seconds of a period	0.6710
Season Number	0.6701
Quarter Number	0.6701

Interestingly, we find the binary feature vector of whether it is the final shot of a period (24 seconds or fewer) to reduce log-loss, but neither season number (which should have a huge effect) nor quarter number has an effect. The final test, therefore, is to make those two one hot encoded as well.

Feature Encoded as One Hot	Log-Loss
Season Number	0.6708
Quarter Number	0.6699
Both	0.6708

Encoding quarter number as one hot actually increased performance a fair amount, but season number dropped it significantly. This is a surprise to me. I had every expectation that one hot encoding season number would lead to a significant reduction in log-loss, because we would expect accuracy to be dependent on which season based on the graph in Section 1. Therefore, my final method and result is to simply encode whether the shot attempt was in his final two seasons or not, where we see a giant spike.

No Season Number	0.6699
Binary Encoding of Season 18+	0.6698

So we see here that the season number does help a little bit, but not much.

b. **Discussion**

Overall, there are a few conclusions to take away here. Firstly, while this is on the validation and not the test set, these methods are fairly competitive (though below the median) on the Kaggle leaderboard – this put me at about #880 out of 1117. Given I really only used four features with a fairly weak, non-optimized classifier, I would argue that turned out fairly well. Nevertheless, that I fell in the bottom 25% of results shows that I need to choose more useful features. Furthermore, after getting better features (and representations), *then* using a stronger classifier like random forests might give me better results.

Speaking of which, this proved a good lesson on choosing classifiers. Random Forests are certainly flexible methods, but they clearly don't categorically provide better results than Logistic Regression or LDA, two simpler methods. I suspect this is due to the feature representation used here, and a better one would allow random forests to better capture the variance in the data. If the feature representation is not precise enough, then increased representational power will simply lead the network to capture more of the *noise* in the features (i.e. overfitting). In this instance, having a weaker model is a good thing.

Furthermore, I found Logistic Regression and LDA to work more or less equally well. This also came as a surprise to me. If the classes were truly Gaussian with the same covariance matrix, shouldn't we expect better results? And if they were not, shouldn't we expect worse? Instead, they perform nearly identically. Incidentally, all these results are in line with the analysis found in [7], as discussed in Section 3. As I researched the issue, I found that LDA and Logistic Regression tend to work in practice similarly well despite LDA's assumptions about the classes' distributions – though this isn't categorically and always true [8].

In terms of feature engineering, I found that one hot encoding of categorical features is absolutely crucial. To an extent this is obvious, but I was surprised to see it true for something like season number as well (there are legitimate temporal relationships in season-by-season data that I thought a model should be able to catch).

Furthermore, I found the limits of expert knowledge here. A feature that I was convinced was useful turned out not to be, no matter the representation I choose. I'm still convinced it's correct and perhaps I just haven't stumbled on the right way to use the feature. Nonetheless, I found out the hard way how hand-crafted features aren't necessarily useful.

Finally, one question we are asked to answer is analyzing the parameters of our model. One way to interpret these parameters is to note that values near 0 don't contribute much to the decision function. Indeed, in that vein I can see that the binary "Was this season after the 17th, i.e. after Kobe tore his Achilles?" variable has a weight of 0.004, so it's no surprise that it didn't increase performance much when added. Yet if I compare it to another variable with a weight of, say, 0.1, what is the relationship between the two variables? It's hard to get a good intuition for it.

I find this as much a lesson on the beauty of *linear regression*. I consider logistic regression to be a simple model, yet it still has hard-to-interpret parameters. I didn't use linear regression in this project, but I have a newfound appreciation for the sheer interpretability of its parameters.

6. REFERENCES

[1] https://www.kaggle.com/selfishgene/kobe-bryant-shot-selection/psychology-of-a-professional-athlete

[2] https://www.kaggle.com/khozzy/kobe-bryant-shot-selection/kobe-shots-show-me-your-best-model/comments

[3] https://fivethirtyeight.com/features/an-ode-to-kobe-bryant-in-two-charts/

[4] http://grantland.com/the-triangle/courtvision-kobe-bryant-and-the-elbow-test/

[5] https://www.kaggle.com/dansbecker/nba-shot-logs

[6] Wright, Raymond E., Jorge Silva, and Ilknur Kaynar-Kabul. "Shot Recommender System for NBA Coaches."

[7] https://www.kaggle.com/khozzy/kobe-bryant-shot-selection/kobe-shots-show-me-your-best-model

[8] Pohar, Maja, Mateja Blas, and Sandra Turk. "Comparison of logistic regression and linear discriminant analysis: a simulation study." *Metodoloski zvezki* 1.1 (2004): 143.