Will Kobe Bryant Make His Next Shot: Quadratic Discriminant Analysis and Logistic Regression using R

Paul Adams
Reannan McDaniel
Jeff Nguyen

Master of Science in Data Science, Southern Methodist University, USA

Contents

1 Abstract:

This project investigates the correlation between multiple potential explanatory variables and Kobe Bryant's ability to make a shot while playing for the NBA team Los Angeles Lakers using data gathered from 1996-2015.

2 Introduction

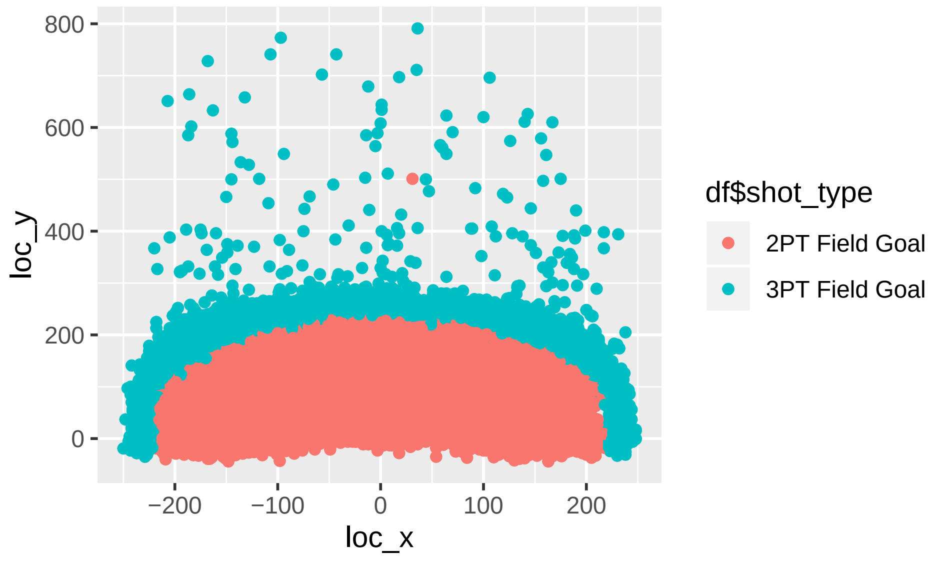
Kobe Bryant, a professional basketball player, spent his entire 20-year career with one team, the LA Lakers as a guard. Bryant started with the Lakers just after high school and went on to win five NBA Championships and 18 All-Star titles. Kobe Bryant's career average points scored per game is 25-points during the regular season and 26-points during playoffs. In addition to these statistics, using the `KobeData` provided, we were able to determine the odds of Kobe making a shot decreases with respect to his distance from the hoop and the probability of him making a shot decreases after taking into consideration his distance from the hoop and analyze the relationship distance and making a shot during the playoffs. During model selection we considered Area Under the Curve (AUC), Mis-Classification Rate, Sensitivity, Specificity and objective / loss function when making model comparisons.

3 Exploratory Data Analysis

3.1 Outlier Check & Transformations

First, we performed a brief outlier check, which included a graphical analysis of all shots taken, by loc_x and loc_y. This graphical analysis indicated a 2PT (2-point) Field Goal was recorded from the 3PT (3-point) range. Upon inspection of other attributes - such as action type and shot_zone_range - we verified this shot to be a member member of the 3-point level of shot_type. Under the assumption shots from beyond the 300 inch mark are more likely to have been incorrectly recorded as 2 points rather than an incorrectly recorded location y, we modified our programming to transform all shots where $loc_y > 300$ to be recoded as `3PT Field Gloal`.

Additionally, `action_type` was recoded where shots types were categorized into `short` and `not_short`. However, modeling performance decreased because the parameter became less descriptive. Further transformations included using indicator variables for `shot_made_flag`, depending on the phase of analysis and modeling.

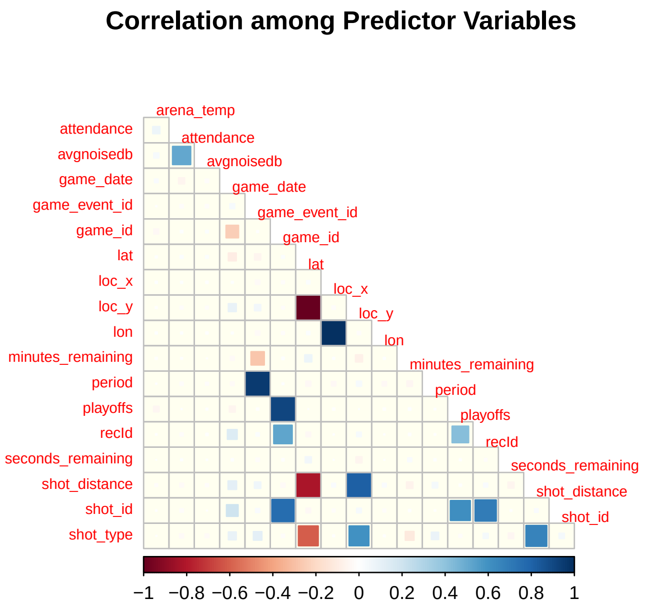


3.2 Variable Elimination

Next, we removed one-level factors. These will never change so are not useful to the model; including can cause issues with model sensitivity since linear trajectories will be down-weighted. Therefore, their significance will be lessened by the constant state of the additional parameters. While this is may not be significant, it is not condusive to model quality.

3.3 Addressing Multicollinearity: Correlation Heat Map for Visual Data Exploration

To address multicollinearity among quantitative predictor variables, a correlation heat map (below) was created for visual inspection of correlation. Red corresponds to negative correlation (where an increase in a predictor causes a decrease in the value of its collinear counterpart) whereas blue corresponds to positive correlation.



3.4 Post-Correlation Heat Map Variable Elimination

Following our correlation heat map, we decided to eliminate some collinear terms. However, some of the collinearity is useful to capture the instances where the terms are unique. For example, `combined_shot_type` (factor variable) is collinear with `shot_distance` (quantitative variable), but it also accounts for the method Kobe may use to make a shot. For example, distance may be relatively the same between 10 and 11 feet, but the factor levels used to derrive their `short` or `far` indications may differ. This difference could be whether Kobe makes a potentially more accurate heel-planted shot or if he is forced to lean forward and take a riskier shot at basket; the difference in distance may only be one foot, but the difference in technique could measure significant relative to the odds of success.

3.5 Addressing Multicollinearity: Correlation Matrix for Numerical Analysis

After deselecting the most obvious collinear terms through visually inspection of the correlation plot, a correlation matrix for analyzing the remaining results. Collinear quantitative data was preliminarily removed following correlation plot analysis to desaturate the model to an extent that allows more distinction among significance measures for terms in the correlation matrix. Following the removal of predictor variables after visually inspecting the correlation heat map, we analyzed a correlation matrix. However, the matrix itself did not identify any remaining collinearity at a threshold of correlation necessitating removal of like-terms. Consequently, no further predictor variables are removed. Please refer to the table below for a list of the remaining top 10 collinear terms following first-pass variable removal. While this risks over-fitting, we believed the variables were different enough to capture useful information for the model.

	Correlation Predictor Variable	Correlation Response Variable	Correlation	p-Value
25	loc__y	shot__distance	0.81812	p < 0.0001
56	recId	shot__id	0.69172	p < 0.0001
44	shot__distance	shot__type	0.66861	p < 0.0001
61	playoffs	shot__id	0.61299	p < 0.0001
40	loc__y	shot__type	0.60662	p < 0.0001
104	attendance	avgnoisedb	0.51092	p < 0.0001
11	recId	playoffs	0.42527	p < 0.0001
66	game__date	shot__id	0.20932	p < 0.0001
46	recId	game__date	0.14773	p < 0.0001
38	game__event__id	shot__type	0.11238	p < 0.0001

4 Quadratic Discriminant Analysis

As requested within the requirements of this study, a Linear Discriminant Analysis must be assessed and provided. Discriminant analysis is an operation that compares a categorical response variable against measures of quantitative predictor variables. As a result, analysis for this section is performed on the numerical predictors, which include `recId`, `game_event_id`, `game_id`, `loc_x`, `loc_y`, `minutes_remaining`, `seconds_remaining`, `shot_distance`, `shot_made_flag`, `shot_type`, `game_date`, `shot_id`, `attendance`, `arena_temp`, `avgnoisedb`, controlling collinearity by eliminating a member of each collinear pair prior to model development.

`Linear Discriminant Analysis` requires a linear boundary between the predictor variables, respective of the response. If the boundary between predictors and response is not linear, `Quadratic Discriminant Analysis` (QDA) must be used. `Wilks' Lambda` distribution is used to assess the nature of boundary linearity, which is a required understanding to develop a well-fit discriminant classification model. However, because of the large dimensions of the data set analyzed in this study, an approximation of Wilks' Lambda must be used, rather than Wilks' Lambda itself. `Bartlett's Test` is an approximation of Wilks' Lambda that can be used for models with large dimensions by applying a measure against the `Chi-Square distribution`. This method is applied herein to assess linearity.

4.1 Bartlett's Test

The result of the Bartlett's test returned statistically significant results, indicating the null hypothesis of linearity must be rejected in favor of the alternate, which is that the discriminant boundary is non-linear. Consequently, we proceed with a model based on `Quadratic Discriminant Analysis` to provide predictive responses from a discriminant model. However, we proceed with caution, as the quadratic version of the discriminant analysis is at greater risk for over-fitting to the data than Linear Discriminant Analysis as the boundary is required to conform more closely to the data rather than to the mean of the data. This was also taken into consideration when assessing the results of the Logistic Regression model development that occurs afterward. Bartlett's Test of this data set yielded a significant p-value, where $p < 0.0001$, indicating that the proportion of distribution beyond the derrived test statistic is beyond that which could be explained by chance. Therefore, we must reject the null hypothesis that the boundary for analysis is linear; the boundary is non-linear. Thus, an analysis using Quadratic Discriminant Analysis is applied. Please refer to the table `Bartlett Test's Wilks' Lambda Approximation` below.

4.1.1 Bartlett Test's Wilks' Lambda Approximation

	Statistics
Chi Square Statistic	1037.24251
Degrees of Freedom	14
Wilks' Lambda	0.9511
p-Value	p < 0.0001

5 Quadratic Discriminant Analysis: Internal Cross-Valdiation and Model Development

Following removal of significant levels of multicollinearity from the dataset and partitioning into a 75% training / 25% testing split, internal cross-validation is performed. The specifics of this test involves 25 folds of the data - meaning the 75% training data is divided into 25 partitions. The model is then trainied on 1/25th of the original 75%, then tested against the remaining 24/25ths, 1/25ths at-a-time. This test is repeated 5 times, with each repeat involving a different random partitioning of the 25 specified `folds` of the data. Finally, the model developed using the 75% training split is then applied to the 25% testing split and predictions are measured against the actuals of that split to develop model statistics such as `Accuracy`, `Misclassification`, `Precision`, `Sensitivity` and `Specificity`.

This is internal cross-validation and its objective is to assist in identifying a model that can perform well across different data sets of the same source by using partitions of data from the same set to simulate an environment where those partitions are actually data from different sets. This is typically performed prior to external cross-validation.

6 Quadratic Discriminant Analysis: External Cross-Valdiation and Model Development

After building a model using internal cross-validation, which applied 5 repeated internal cross-validations across the 25 folds of training data, a confusion matrix was constructed and analyzed. Next, we applied the model developed using the 75% training split to make predictions against the entire portion of data that includes values for `shot_made_flag` in order to assess how closely the model can predict against the entire data set compared to the actuals.

Applying the model to the entire dataset as `external cross-validation` provides the model an opportunity to test against different data and more closely simulate a real-life scenario than internal cross-validation. Internal and external cross-validation is performed for later Logistic Regression models as well.Following external cross-validation of both models, the metrics are compared to determine the best model (Quadratic Discriminant Analysis versus Logistic Regression).

A confusion matrix is a table of results from cross-validation. Some key metrics provided by a confusion matrix include `Accuracy`, `Precision`, `Sensitivity` and `Specificity`. `Accuracy` is the number of all correct predictions divided by the number of all predictions. `Precision` is the ratio of the number of correctly classified positive predictions divided by the number of all positive predictions. `Sensitivity` (also called `Recall`) is the number of correctly classified positive predictions divided by all positive actuals - this is similiar to precision, except that sensitivity measures against actual values. `Specificity` is the number of correctly classified negative predictions divided by all negative actuals. Simplistically, sensitivity is the true positive rate wherease specificity is the true negative rate. Higher Accuracy, Precision, Sensitivity, and Specificity is desireable.

Another important component for cross-validation is the `Misclassification Rate`. The Misclassification Rate is a descriptor of how often a model is wrong. This value is equal to the total number of False Positives plus the False Negatives divided by all predictions. A lower misclassification rate is desireable.

In addition to the misclassification rate, Accuracy, Precision, Sensitivity, Specificity and Misclassification Rate, the Logarithmic Loss function is applied to measure . A lower logarithmic loss value is desireable as logarithmic loss increases as predicted probability diverges from the actual response values and conversely decreases as predicted probability moves converges toward the actual response values.

Two final metrics used in this analysis are the `Area Under the Curve (AUC)` and `Receiver Operating Characteristic (ROC) curve`. The ROC bounds an area the area which the AUC describes. As a discrimination threshold changes, the ROC visually represents the correct diagnostic ability of a binary classification model and is a plot of the true positive against the false positive rate at those varied thresholds. As the AUC describes the area under this curve, a higher AUC is more desireable than a lower AUC. As mentioned, these metrics will be analyzed when comparing internal to external cross-validation to ensure consistency as well as between the QDA and Logistic Regression models.

7 Quadratic Discriminant Analysis: Internal vs. External Cross-Validation

Using the two confusion matrix output tables immediately below, the performance across internal and external cross-validations of the QDA model can be compared. As indicated in those figures, the model performed highly similarly across both cross-validation techniques, indicating the model is consistent and reasonably fit, after controlling for the variables selected for modeling.

7.0.1 Confusion Matrix Results for Quadratic Discriminant Analysis

	Internal CV Statistics	External CV Statistics
Sensitivity	0.51431	0.51187
Specificity	0.66761	0.66993
Precision	0.56104	0.55695
Accuracy	0.59826	0.59917
Misclassification Rate	0.40174	0.40083
Logarithmic Loss	0.70127	0.70127
Area Under the Curve	0.40904	0.59090

8 Logistic Model Development Using Ordinary Least Squares

Logistic Regression is a classification technique that is best suited for dichotomous response variables - in the case of the Kobe data the response is '0' for shot missed, or '1' for shot made. Compared to discriminant analysis techniques, multiple explanatory variables, interactions, and categorical variables can be used, allowing for a potentially more descriptive model. For this type of regression, coefficients are in log-odds where each coefficient needs to be exponentiated to yield odds ratios - this is done for ease of interpretation. Logistic Regression can also be used to generate predictions that yield the probability of an observation having the desired traits of the response variable as occurring or not.

8.1 Logistic Regression: Model Selection

A preliminary, manual variable elimination process was performed during the analysis of multicollinear terms in preparation for model development. Below we perform logistic regression using Ordinary Least Squares (OLS). In preparation for the model development, a starting model and a finishing model must be developed to provide the scope of variable selection. These initial models are used by forward, backward, and stepwise model selection methods are used to help select a combination of variables that result in the lowest Residual Deviance and/or AIC. The selection method that generates the model with the lowest AIC/Residual Deviance is then used for internal cross validation to further tune the model which allows for better prediction. Below are models that each selection method generated:

Forward Selection Model: $shot\_made\_flag = action\_type + attendance + arena\_temp + game\_event\_id + season + seconds\_remaining + minutes\_remaining + loc\_y + game\_date + loc\_x$

Backward Elimination Model: $shot\_made\_flag = recId + action\_type + game\_event\_id + loc\_x + minutes\_remaining + season + seconds\_remaining + shot\_distance + game\_date + shot\_id + attendance + arena\_temp$

Stepwise Regression Model: $shot\_made\_flag = recId + action\_type + game\_event\_id + loc\_x + minutes\_remaining + season + seconds\_remaining + shot\_distance + game\_date + shot\_id + attendance + arena\_temp$

Based on the fit-statistics generated from each model selection method, the backwards and stepwise models are identical in fit-statistics with an AIC at 25167.77, and the residual deviance at 25001.77. The forward model selection out-performs both backwards and stepwise models with an AIC of 25166.48, but has a higher a residual deviance at 25001.48. Compared to the forward selected model, the backwards and stepwise models have lower residual deviances, although their residual deviances are very close in value to the forward model, their AIC values are larger compared to the forward selected model. Based on the evidence, the forward selected model will be used for the internal and external cross validation process.

Selection Type	AIC	Residual Deviance
Forwards	25168.23	25004.23
Backwards	25167.77	25001.77
Stepwise	25167.77	25001.77

8.2 Logistic Regression: Internal Cross Validation and Model Development

After identifying that the forward selected model is the best candidate based on it's AIC and residual deviance, its features are tuned using an internal cross validation. A training set is generated by randomly selecting 75% of the observations, with the remaining 25% serving as the validation (test) set. The model is tuned using 25 folds and is repeated 5 times, this process is the same as described in section 5.0.1. Prior to the internal cross validation process, action types that rarely occur in the data are recoded to similar, but differ action types, i.e. "Running Tip Shot" (1 observation) to "Tip Shot" (many observations). If levels within the evaluation data exist but are not present in the training data, a trained model will have difficulty making predictions. Infrequently occurring action types are recoded to similar action types to avoid this situation.

8.3 Logistic Regression: External Cross Validation and Model Development

External cross-validation is used to evaluate a model after it has been tuned in the internal cross-validation process. External cross validation confusion matrix statistics, ROC/AUC, and misclassification rates can be compared to the internal cross-validation statistics to help assess performance. As with the "Quadratic Discriminant Analysis: External Cross-Validation and Model Development" section, confusion matrices are used to assess model performance where we will be focusing on `Accuracy`, `Precision`, `Sensitivity`, `Specificity`, `Misclassification Rate`, `AUC`, and `Log Loss`. When looking at model performance high values for `Accuracy`, `Precision`, `Sensitivity`, `Specificity`, `AUC` are desirable; and low values for `Misclassification Rate`, and `Log Loss` are desirable. For descriptions of the mentioned terms please refer to the "Quadratic Discriminant Analysis: External Cross-Validation and Model Development" section for more information.

8.4 Logistic Regression: Internal vs. External Cross-Validation

The misclassification rate, and Log loss for both internal and external cross validations are close in values. However, the external cross-validated model has higher `Sensitivity`, `Precision`, and `Accuracy`; but lower `Specificity` and `Area Under the Curve` compared to the internal cross-validated model. This suggests that model `Sensitivity`, `Precision`, and `Accuracy` improves, but its predictive performance decreases when evaluated using the external cross validation.

	Internal CV Statistics	External CV Statistics
Sensitivity	0.51431	0.86230
Specificity	0.66761	0.46417
Precision	0.56104	0.66502
Accuracy	0.59826	0.68406
Misclassification Rate	0.31136	0.31136
Logarithmic Loss	0.74323	0.74323
Area Under the Curve	0.70361	0.59090

8.5 Logistic Regression: Fitted Model

The model was selected for prediction based on having the lowest AIC and residual deviance compared to backwards and stepwise selected models. The forward selection model was tuned using k-fold internal cross-validation. The forward selection model generated is listed below where Logit coefficents for each feature can also be found in the "Fitted Logistic Regression Model" portion of the appendix:

$shot\_made\_flag = action\_type + attendance + arena\_temp + game\_event\_id + season + seconds\_remaining + minutes\_remaining + loc\_y + game\_date + loc\_x$

Several notable features that this model generated are several of the `action_types`. The odds ratios of Kobe missing a shot are revealed once the coefficients are exponentiated. In the table below we can see that the odds ratios are all very small. This suggests that when Kobe performs one of the listed levels of `action_type`, his chances of missing the shot are very low. The frequency of the `action_type` can also be ascertained by looking at the standard errors from the model output. Small standard errors for a coefficient suggest that there are a large number of observations with that attribute with a similar outcomes, and a large standard error would suggest that there are fewer observations with a similar outcome. When moving from the base `action_type` of "Alley Oop Dunk Shot" to "Jump Shot" for every one unit increase in "Jump Shot" the logit value for `shot_made_flag` is approximate -4; in terms of odds ratios, compared to using the "Alley Oop Dunk Shot," Kobe's chances are missing the shot when using the "Jump Shot" is at a factor of approximately 0.016. The "Jump Shot" also has a standard error of approximately 0.723, this value is much lower compared to the other top shots suggesting that Kobe uses the "Jump Shot" frequently and rarely misses this shot type when he chooses to use it. When looking at the "Running Finger Roll Shot" the odds ratio of missing this shot when starting from a base level of "Alley Oop Dunk Shot" is approximately 4.5e-8 with a logit standard error of approximately 308; this suggests that when Kobe chooses to use this shot he will rarely use it, but due to the high standard error the number of times that Kobe uses this shot is very low. When a frequency table of the `action_type` for the whole dataset is created, the total number of times the "Running Finger Roll Shot" occurs is 4 - compared to 12712 observations for "Jump Shot."

	Coefficient (logit)	Odds Ratio
‘action_typeRunning Finger Roll Shot‘	-16.915327	0.0000000
‘action_typeRunning Pull-Up Jump Shot‘	-16.791811	0.0000001
‘action_typeRunning Reverse Layup Shot‘	-4.523732	0.0108485
‘action_typeTip Shot‘	-4.194048	0.0150851
‘action_typeJump Shot‘	-4.111135	0.0163892
‘action_typeDriving Jump shot‘	-4.109410	0.0164175

9 Shot Made Odds Analysis Using Logistic Regression

When analyzing odds for basketball many are interested in the spread, or by how much will the favored team win or lose. There are many other odds played in professional sports, but we are focused on the odds Bryant making a shot or not relative to his distance from the hoop during regular season games and the playoffs.

There is sufficient evidence to suggest that the shot made odds for Bryant being further from the hoop are less than him being closer to the basket (p-value < 0.041422) with respect to all other covariates.

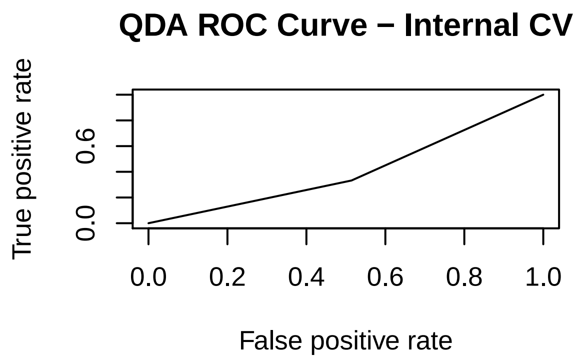
For every additional foot in distance from the hoop, the estimated odds of making the shot change by eˆ 0.00050349=1.0005036. This is for every additional foot in distance, the estimated odds of making the shot decrease by 100.05% (1.0005036)100. A 95% confidence interval for the multiplicative change is (0.00001922857, 0.00098719309) holding all other variables constant.

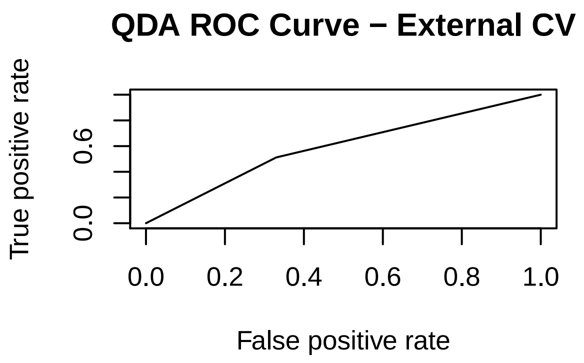
We also were curious if the odds of Bryant making a shot further away from the hoop had a difference during the playoffs. The shot distance is still significant but there is virtually no difference in the odds if the shot was made in a regular season game.

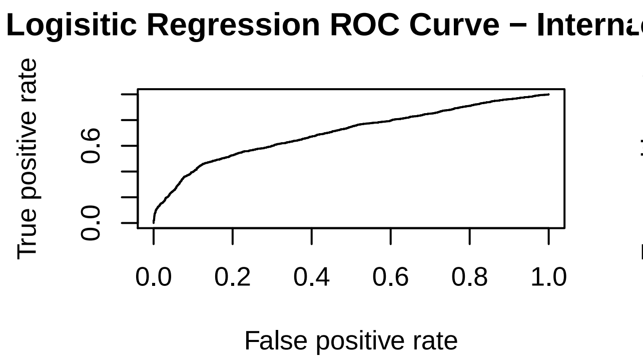
10 Appendix A: Figures and Tables

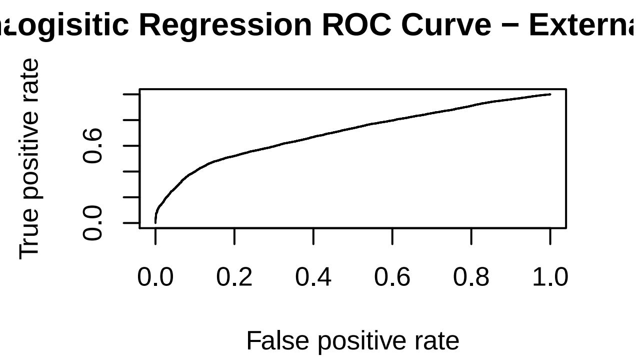
10.0.1 *Appendix A contains content related to the paper*

10.1 Receiver Operator Characteristic (ROC) Curves









10.2 Fitted Logistic Regression Model

	Coefficient (logit)	Odds Ratio
(Intercept)	-0.5406091	0.5823934
‘action_typeAlley Oop Layup shot‘	-2.3125586	0.0990076
‘action_typeCutting Layup Shot‘	-1.9200208	0.1466039
‘action_typeDriving Bank shot‘	10.6747918	43251.7007167
‘action_typeDriving Dunk Shot‘	0.4927331	1.6367836
‘action_typeDriving Finger Roll Layup Shot‘	-1.1244474	0.3248319
‘action_typeDriving Finger Roll Shot‘	-1.8877604	0.1514105
‘action_typeDriving Floating Bank Jump Shot‘	9.9403844	20751.7192693
‘action_typeDriving Floating Jump Shot‘	-3.6656405	0.0255878
‘action_typeDriving Hook Shot‘	-2.8993874	0.0550569
‘action_typeDriving Jump shot‘	-4.1094104	0.0164175
‘action_typeDriving Layup Shot‘	-2.2506621	0.1053295
‘action_typeDriving Reverse Layup Shot‘	-1.9553431	0.1415159
‘action_typeDriving Slam Dunk Shot‘	0.1657104	1.1802312
‘action_typeDunk Shot‘	-2.2960202	0.1006587

(continued)

	Coefficient (logit)	Odds Ratio
‘action_typeFadeaway Bank shot‘	-0.6729202	0.5102165
‘action_typeFadeaway Jump Shot‘	-3.0523906	0.0472458
‘action_typeFinger Roll Layup Shot‘	-2.3467101	0.0956834
‘action_typeFinger Roll Shot‘	-3.0929933	0.0453660
‘action_typeFloating Jump shot‘	-2.2745444	0.1028438
‘action_typeFollow Up Dunk Shot‘	-1.1106209	0.3293544
‘action_typeHook Bank Shot‘	10.2687917	28819.0438353
‘action_typeHook Shot‘	-3.9216063	0.0198092
‘action_typeJump Bank Shot‘	-2.0558189	0.1279880
‘action_typeJump Hook Shot‘	-2.1372195	0.1179824
‘action_typeJump Shot‘	-4.1111346	0.0163892
‘action_typeLayup Shot‘	-3.8709890	0.0208378
‘action_typePullup Bank shot‘	-2.9677237	0.0514202
‘action_typePullup Jump shot‘	-2.3056159	0.0996974
‘action_typePutback Dunk Shot‘	-2.7340623	0.0649549
‘action_typePutback Layup Shot‘	-2.6452777	0.0709856
‘action_typeReverse Dunk Shot‘	0.0718912	1.0745384
‘action_typeReverse Layup Shot‘	-2.8641128	0.0570337
‘action_typeReverse Slam Dunk Shot‘	10.3104940	30046.2750438
‘action_typeRunning Bank shot‘	-1.5071851	0.2215327
‘action_typeRunning Dunk Shot‘	-1.6818534	0.1860289
‘action_typeRunning Finger Roll Layup Shot‘	-2.8650540	0.0569801
‘action_typeRunning Finger Roll Shot‘	-16.9153274	0.0000000
‘action_typeRunning Hook Shot‘	-1.9042649	0.1489321
‘action_typeRunning Jump Shot‘	-2.2720781	0.1030977
‘action_typeRunning Layup Shot‘	-2.8672407	0.0568556
‘action_typeRunning Pull-Up Jump Shot‘	-16.7918113	0.0000001
‘action_typeRunning Reverse Layup Shot‘	-4.5237315	0.0108485
‘action_typeSlam Dunk Shot‘	0.7787543	2.1787564
‘action_typeStep Back Jump shot‘	-2.5994483	0.0743146
‘action_typeTip Shot‘	-4.1940480	0.0150851
‘action_typeTurnaround Bank shot‘	-1.9673093	0.1398326
‘action_typeTurnaround Fadeaway shot‘	-3.0814350	0.0458934
‘action_typeTurnaround Finger Roll Shot‘	10.0616178	23426.3747389
‘action_typeTurnaround Hook Shot‘	-3.6153476	0.0269076
‘action_typeTurnaround Jump Shot‘	-2.9350977	0.0531255
attendance	0.0001723	1.0001723
arena_temp	0.0383777	1.0391236
game_event_id	-0.0003392	0.9996608
seconds_remaining	0.0027254	1.0027291
minutes_remaining	0.0120657	1.0121388
loc_y	0.0005358	1.0005359
game_date	-0.0000377	0.9999623
loc_x	0.0001772	1.0001772
playoffs	0.0045673	1.0045778

10.3 Action Type Frequency Table

Var1	Freq
Alley Oop Dunk Shot	76
Alley Oop Layup shot	59
Cutting Layup Shot	6
Driving Bank shot	1
Driving Dunk Shot	196
Driving Finger Roll Layup Shot	47
Driving Finger Roll Shot	52
Driving Floating Bank Jump Shot	1
Driving Floating Jump Shot	3
Driving Hook Shot	13
Driving Jump shot	19
Driving Layup Shot	1335
Driving Reverse Layup Shot	67
Driving Slam Dunk Shot	38
Dunk Shot	176
Fadeaway Bank shot	22
Fadeaway Jump Shot	693
Finger Roll Layup Shot	21
Finger Roll Shot	23
Floating Jump shot	75
Follow Up Dunk Shot	10
Hook Bank Shot	5
Hook Shot	61
Jump Bank Shot	223
Jump Hook Shot	16
Jump Shot	12712
Layup Shot	1734
Pullup Bank shot	10
Pullup Jump shot	318
Putback Dunk Shot	3
Putback Layup Shot	9
Reverse Dunk Shot	52
Reverse Layup Shot	276
Reverse Slam Dunk Shot	15
Running Bank shot	35
Running Dunk Shot	14
Running Finger Roll Layup Shot	5
Running Finger Roll Shot	4
Running Hook Shot	28
Running Jump Shot	620
Running Layup Shot	42
Running Pull-Up Jump Shot	1
Running Reverse Layup Shot	6
Slam Dunk Shot	264
Step Back Jump shot	93
Tip Shot	121
Turnaround Bank shot	50
Turnaround Fadeaway shot	299
Turnaround Finger Roll Shot	1
Turnaround Hook Shot	8

(continued)

Var1	Freq
Turnaround Jump Shot	739

11 Appendix B: Source Code

11.0.1 *Appendix A contains source code created for this project*

```
library(pacman)
p_load(rrcov, MASS, dplyr, purrr, ggplot2, Hmisc, pcaPP, knitr, kableExtra, caret, cluster, robustbase, RO

df <- read.csv("./modelingKobeData.csv", header=T, sep=",", strip.white=T, stringsAsFactors = F, na.string
df.preds <- read.csv("./predictionKobeData.csv", header=T, sep=",", strip.white=T, stringsAsFactors = F, na
```

```
# Check for Missing Values
sapply(df, function(cnt) sum(length(which(is.na(cnt)))))
```

```
# Outlier Plot
ggplot(data.frame(df$loc_x, df$loc_y, df$shot_distance), aes(x=loc_x, y=loc_y)) +
  geom_point(aes(colour = df$shot_type))
```

```
# Outlier Imputation
df[which(df$loc_y > 300),"shot_type"] <- "3PT Field Goal"
df.preds[which(df.preds$loc_y > 300),"shot_type"] <- "3PT Field Goal"
```

```
# Convert 2PT Field Goal and 3PT Field Goal to 2 and 3
df$shot_type <- ifelse(df$shot_type=="2PT Field Goal", 2, 3)
df.preds$shot_type <- ifelse(df.preds$shot_type=="2PT Field Goal", 2, 3)
```

```
# Convert all characters to factors with levels the models can use
df <- df %>% mutate_if(is.integer, as.numeric) %>% mutate_if(is.character, as.factor) %>% data.frame()
```

```
# Drop categorical variables
df <- df %>%
subset(select=-c(team_id, # dropping since this is a uniform distribution of data
                       team_name, # dropping since this is a uniform distribution of data. Also collinear
                       combined_shot_type, # dropping this in favor of action_type
                       shot_zone_area, # this is ambiguous and less descriptive than geospatial data
                       matchup # removing in favor of opponent; Kobe only played for LAL so that will neve
                      )
              )
```

```
    # Numeric Data Frame for Modeling and Predicting
    df.numeric <- df %>% keep(is.numeric)

    df.preds <- df.preds %>% mutate_if(is.integer, as.numeric) %>% mutate_if(is.character, as.factor) %>% 
    df.preds <- df.preds %>%
      subset(select=-c(team_id, # dropping since this is a uniform distribution of data
                       team_name, # dropping since this is a uniform distribution of data. Also collinear
                       combined_shot_type, # dropping this in favor of action_type
                       shot_zone_area, # this is ambiguous and less descriptive than geospatial data
                       matchup # removing in favor of opponent; Kobe only played for LAL so that will neve
                      )
              )
    # create numeric dataframe for correlation plot
    df.numeric.preds <- df.preds %>% keep(is.numeric)
```

```
# Correlation Heat Map
corr.plot <- corrplot::corrplot(cor(df.numeric %>% subset(select=-c(shot_made_flag)))
                   , title = "Correlation among Predictor Variables"
                   , type = "lower"
                   , tl.pos = "ld"
                   , method = "square"
                   , tl.cex = 0.65
                   , tl.col = 'red'
                   , order = "alphabet"
                   , diag = F
                   , mar=c(0,0,5,0)
                   , bg="ivory1"
                   ,tl.srt=.05
)
```

```
# Dropping variables

df <- df %>% subset(select=-c(lat, # dropping lat because it is collinear with loc_y and shot_distance
                              lon, # dropping lon because it is collinear with loc_x and shot_distance
                              period, # dropping period in favor of game event id because game event id is
                              game_id # dropping playoffs for game_id; game ID can capture playoffs season
                             )
                    )

df.preds <- df.preds %>% subset(select=-c(lat, # dropping lat because it is collinear with loc_y and shot_
                              lon, # dropping lon because it is collinear with loc_x and shot_distance
                              period, # dropping period in favor of game event id because game event id is
                              game_id # dropping playoffs for game_id; game ID can capture playoffs season
                             )
                    )

df.numeric <- df %>% keep(is.numeric) %>% mutate_if(is.integer, as.numeric)
df.numeric.preds <- df.preds %>% keep(is.numeric) %>% mutate_if(is.integer, as.numeric)
```

```
# Correlation Matrix
flattenCorrMatrix <- function(cormatrix, pmatrix) {
  ut <- upper.tri(cormatrix)
  data.frame(
    row = rownames(cormatrix)[row(cormatrix)[ut]],
    column = rownames(cormatrix)[col(cormatrix)[ut]],
    cor  =(cormatrix)[ut],
    p = pmatrix[ut]
  )
}

options(scipen=999)
options(max.print=100000)

#See what variables are correlated with eachother, p-values
correlation.matrix <- Hmisc::rcorr(as.matrix(df.numeric), type="pearson")
corDF <- data.frame(flattenCorrMatrix(correlation.matrix$r, correlation.matrix$P))

corDF.ordered <- data.frame(corDF[order(-corDF$cor),])
cordDF.ordered.Top10 <- data.frame(head(corDF.ordered, 10))

colnames(cordDF.ordered.Top10) = c("Correlation Predictor Variable", "Correlation Response Variable", "Cor
cordDF.ordered.Top10$Correlation <- round(as.numeric(as.character(cordDF.ordered.Top10$Correlation)), digi

cordDF.ordered.Top10$`p-Value` <- ifelse(as.numeric(as.character(cordDF.ordered.Top10$`p-Value`)) < 0.0001
```

```
# Correlation Matrix Table
kable(cordDF.ordered.Top10, format="latex", booktabs = T)  %>%
  kable_styling(latex_options="striped", position = "center")

dfTrain.numeric <- df.numeric[which(!is.na(df.numeric$shot_made_flag)),]
prediction.Data.numeric <- df.numeric[which(is.na(df.numeric$shot_made_flag)),]
```

```
# Convert shot_made_flag to Qualitative
dfTrain.numeric$shot_made_flag <- as.factor(dfTrain.numeric$shot_made_flag)
dfTrain.numeric$shot_made_flag <- ifelse(dfTrain.numeric$shot_made_flag=="1", "made", "not_made")
dfTrain.numeric <- dfTrain.numeric %>% mutate_if(is.integer, as.numeric) %>% mutate_if(is.character, as.fa
```

```
# Bartlett's Test for Wilks' Lambda
Bartlett_ChiSq <- rrcov::Wilks.test(shot_made_flag ~ ., data=dfTrain.numeric, method = "c", approximation =

Wilk's Lambda produces significant p-value in Bartlett's test so we need to use a Quadratic Discriminant A
format(round(Bartlett_ChiSq$p.value, 2), nsmall=4)

Wilks' Lambda plus degrees of freedom used in Bartlett's chi-squared test
WilksDegreesofFreedom <- rbind(as.numeric(paste0(Bartlett_ChiSq$parameter, sep = " ")))

p-value from Bartlett's test
Bartlett_ChiSq$p.value
Bartletts_p <- format(round(as.numeric(Bartlett_ChiSq$p.value), 2), nsmall=4)

Because Bartlett's p-value is less than 0.0001 (indicated above), updating to shorter form:
Bartletts_p = ifelse(Bartlett_ChiSq$p.value < 0.0001, "p < 0.0001", Bartlett_ChiSq$p.value)
#Bartletts_p <- "p < 0.0001"

dfBartlett <- data.frame(WilksDegreesofFreedom, Bartlett_ChiSq$wilks, Bartletts_p)
colnames(dfBartlett) <- c("Chi-Square Statistic", "Degrees of Freedom", "Wilks' Lambda", "p-Value")

dfBartlett$`Chi-Square Statistic` <- round(as.numeric(as.character(dfBartlett$`Chi-Square Statistic`)), di
dfBartlett$`Wilks' Lambda` <- round(as.numeric(as.character(dfBartlett$`Wilks' Lambda`)), digits=5)

bartlettsTest <- data.frame(rbind(dfBartlett$`Chi-Square Statistic`,dfBartlett$`Degrees of Freedom`,dfBart
rownames(bartlettsTest) <- c("Chi Square Statistic","Degrees of Freedom","Wilks' Lambda","p-Value")
colnames(bartlettsTest) <- "Statistics"

kable(bartlettsTest,
      format="latex", booktabs = T)  %>%
  kable_styling(latex_options="striped", position = "center")
```

```
dfTrain <- df[which(!is.na(df$shot_made_flag)),]
prediction.Data <- df[which(is.na(df$shot_made_flag)),]

# Full Data train/test split for Logistic
test_sample_size <- floor(0.75 * nrow(dfTrain))
set.seed(123)
train_ind <- sample(seq_len(nrow(dfTrain)), size = test_sample_size)
subDF.Train <- dfTrain[train_ind, ] #75% training
subDF.Test <- dfTrain[-train_ind, ] # 25% testing

# Numeric Data train/test split for QDA
test_sample_size <- floor(0.75 * nrow(dfTrain.numeric))
set.seed(123)
train_ind <- sample(seq_len(nrow(dfTrain.numeric)), size = test_sample_size)
subDF.Train.numeric <- dfTrain.numeric[train_ind, ] #75% training
subDF.Test.numeric <- dfTrain.numeric[-train_ind, ] # 25% testing


kobe.qda <- qda(shot_made_flag ~ ., CV=T, data=dfTrain.numeric)

data.frame(mean(kobe.qda$posterior[,1]), mean(kobe.qda$posterior[,2]))

shot_made_flag_Posterior <- rbind("0", "1")
proportion_Posterior <- rbind(mean(kobe.qda$posterior[,1]), mean(kobe.qda$posterior[,2]))
priori <- data.frame(shot_made_flag_Posterior, proportion_Posterior)

subDF.Train.numeric$shot_made_flag <- as.factor(subDF.Train.numeric$shot_made_flag)
subDF.Train.numeric$shot_made_flag <- ifelse(subDF.Train.numeric$shot_made_flag=="1", "made", "not_made")
subDF.Train.numeric <- subDF.Train.numeric %>% mutate_if(is.integer, as.numeric) %>% mutate_if(is.characte

# k=25 folds, repeat random folding for internal cross-validation 5 times:
train.Control <- caret::trainControl(method = "repeatedcv",
                              number = 25,
                              repeats = 5,
                              summaryFunction = mnLogLoss,
                              classProbs = T) # classProbs=T to get mnLogLoss (also for twoClassSummary)

Build Model using the 75% partitioned from the internal dataset (the set with all shot_made_flag response 
qda.filtered <- train(shot_made_flag ~ .
                , data = subDF.Train.numeric
                , method = "qda"
                , trControl=train.Control
                , preProcess = c("center", "scale", "spatialSign")
                , metric = "logLoss"
                 )

Test Model on Test Set (25%):
internal_cv.predicted.qda <- suppressWarnings(predict(qda.filtered, newdata = subDF.Test.numeric))

# Confusion matrix for Internal Cross-Validation - Quadratic Discriminant Analysis
confusion_matrix_results.internal<- confusionMatrix(table(internal_cv.predicted.qda, subDF.Test.numeric$sh

apply the model developed above to the entire (75 + 25 = 100%) internal dataset
external_cv.predicted.qda <- suppressWarnings(predict(qda.filtered, newdata = dfTrain.numeric))

compare the predicted results to the actual results to make sure model still performs as intended:
confusion_matrix_results.external <- confusionMatrix(table(external_cv.predicted.qda, dfTrain.numeric$shot

# Internal Cross-Validation Metrics - Quadratic Discriminant Analysis
SpecSense.confusion.internal <- data.frame(confusion_matrix_results.internal$byClass)
AccuracyP.confusion.internal <- data.frame(confusion_matrix_results.internal$overall)

Accuracy.confusion.internal <- AccuracyP.confusion.internal[1,]
Sensitivity.confusion.internal <- SpecSense.confusion.internal[1,]
```

```r
Specificity.confusion.internal <- SpecSense.confusion.internal[2,]
Precision.confusion.internal <- SpecSense.confusion.internal[5,]

# External Cross-Validation Metrics - Quadratic Discriminant Analysis
SpecSense.confusion.external <- data.frame(confusion_matrix_results.external$byClass)
AccuracyP.confusion.external <- data.frame(confusion_matrix_results.external$overall)

Accuracy.confusion.external <- AccuracyP.confusion.external[1,]
Sensitivity.confusion.external <- SpecSense.confusion.external[1,]
Specificity.confusion.external <- SpecSense.confusion.external[2,]
Precision.confusion.external <- SpecSense.confusion.external[5,]

misclassification.QDA.external <- (confusion_matrix_results.external$table[2,1] + confusion_matrix_results

misclassification.QDA.internal <- (confusion_matrix_results.internal$table[2,1] + confusion_matrix_results

print(logLoss)
logLoss.quadratic <- qda.filtered$results[1,2]
logLoss_StDev <- qda.filtered$results[1,3]

dfTrain.numeric$shot_made_flag <- ifelse(dfTrain.numeric$shot_made_flag=="made",1,0)
# Internal Cross-Validation - Quadratic Discriminant Analysis
internal_cv.predicted.qda <- ifelse(internal_cv.predicted.qda=="made",1,0)

AUCpredStep.internal <- prediction(internal_cv.predicted.qda, as.numeric(subDF.Test.numeric$shot_made_flag
perf_step.internal <- performance(AUCpredStep.internal, measure = "tpr", x.measure = "fpr")

AUC.internal <- performance(AUCpredStep.internal, measure = "auc")
AUC.internal <- AUC.internal@y.values[[1]]

# External Cross-Validation - Quadratic Discriminant Analysis
external_cv.predicted.qda <- ifelse(external_cv.predicted.qda=="made",1,0)

AUCpredStep.external <- prediction(external_cv.predicted.qda, as.numeric(dfTrain.numeric$shot_made_flag))
perf_step.external <- performance(AUCpredStep.external, measure = "tpr", x.measure = "fpr")

AUC.external <- performance(AUCpredStep.external, measure = "auc")
AUC.external <- AUC.external@y.values[[1]]

# Internal Cross-Validation Confusion Matrix - Quadratic Discriminant Analysis
confusionFrame.internal <- data.frame(rbind(round(Sensitivity.confusion.internal, digits=5),round(Specific

rownames(confusionFrame.internal) <- c("Sensitivity","Specificity","Precision","Accuracy","Misclassificati

# External Cross-Validation Confusion Matrix - Quadratic Discriminant Analysis
confusionFrame.external <- data.frame(rbind(round(Sensitivity.confusion.external, digits=5),round(Specific

rownames(confusionFrame.external) <- c("Sensitivity","Specificity","Precision","Accuracy","Misclassificati

confusionFrame <- data.frame(confusionFrame.internal, confusionFrame.external)
colnames(confusionFrame) <- c("Internal CV Statistics", "External CV Statistics")

# Cross-Validation Confusion Matrices for Quadratic Discriminant Analysis
kable(confusionFrame, format="latex", booktabs = T)  %>%
  kable_styling(latex_options="striped", position = "center")
```

```
# Prediciton Code for Quadratic Discriminant Analysis
pred.qda.filtered <- suppressWarnings(predict(qda.filtered, newdata = df.numeric.preds))
df.preds$shot_made_flag <- pred.qda.filtered
write.csv(df.preds, "Predicted_Results.csv", row.names = F)


# Logistic Regression Models
**Forward Selection Model:**
$shot\_made\_flag = action\_type + attendance + arena\_temp + game\_event\_id + season + seconds\_remainin

**Backward Elimination Model:**
$shot\_made\_flag = recId + action\_type + game\_event\_id + loc\_x + minutes\_remaining + season + second

**Stepwise Regression Model:**
$shot\_made\_flag = recId + action\_type + game\_event\_id + loc\_x + minutes\_remaining + season + second


# Initial Models
model.forward.Start <- glm(shot_made_flag~1, family=binomial(link='logit'), data = dfTrain)
model.Allvar <- glm(shot_made_flag ~ ., family=binomial(link='logit'), data = dfTrain)


# Forward Selection
model.Forward <- stepAIC(model.forward.Start, direction = "forward", trace = F, scope = formula(model.Allv

summary(model.Forward)
model.Forward$anova

forward.glm <- glm(shot_made_flag ~ action_type + attendance + arena_temp +
    game_event_id + season + seconds_remaining + minutes_remaining +
    loc_y + game_date + loc_x + playoffs
                    , family=binomial(link='logit')
                    , data=dfTrain)

summary(forward.glm)


# Backward Elimination
model.Backward <- stepAIC(model.Allvar, direction = "backward", trace = F, scope = formula(model.forward.S
summary(model.Backward)
model.Backward$anova

back.glm <- glm(shot_made_flag ~ recId + action_type + game_event_id +
    loc_x + minutes_remaining + season + seconds_remaining +
    shot_distance + game_date + shot_id + attendance + arena_temp
    , family=binomial(link='logit')
    , data=dfTrain)

summary(back.glm)
back.glm$aic



# Stepwise Regression
model.Stepwise <- stepAIC(model.Allvar, direction = "both", trace = F)
```

```r
summary(model.Stepwise)
model.Stepwise$anova
Stepwise Regression Model Suggestion
step.glm <- glm(shot_made_flag ~ recId + action_type + game_event_id +
    loc_x + minutes_remaining + season + seconds_remaining +
    shot_distance + game_date + shot_id + attendance + arena_temp
    , family=binomial(link='logit')
    , data=dfTrain)

summary(step.glm)
step.glm$aic

# Model Selection Statistics AIC and RMSE
model_stats <- data.frame(cbind(rbind("Forwards","Backwards","Stepwise"),rbind(round(forward.glm$aic, digi
colnames(model_stats) <- c("Selection Type", "AIC", "Residual Deviance")

kable(model_stats,format="latex", booktabs = T) %>%
  kable_styling(latex_options="striped", position = "center")

# Recategorize rare shot types into more common, but similar shot types
dfTrain$action_type = as.character(dfTrain$action_type)
dfTrain = dfTrain %>%
  mutate(action_type = if_else(action_type == "Running Tip Shot", 'Tip Shot', action_type))%>%
  mutate(action_type = if_else(action_type == "Tip Layup Shot", 'Tip Shot', action_type)) %>%
  mutate(action_type = if_else(action_type == "Putback Slam Dunk Shot","Slam Dunk Shot",action_type)) %>%
  mutate(action_type = if_else(action_type == "Running Slam Dunk Shot","Slam Dunk Shot",action_type))
dfTrain$action_type = as.factor(dfTrain$action_type)

df.preds$action_type = as.character(df.preds$action_type)
df.preds = df.preds %>%
  mutate(action_type = if_else(action_type == "Turnaround Finger Roll Shot", 'Finger Roll Shot', action_ty
  mutate(action_type = if_else(action_type == "Putback Slam Dunk Shot",'Slam Dunk Shot',action_type)) %>%
  mutate(action_type = if_else(action_type == "Running Slam Dunk Shot",'Slam Dunk Shot',action_type))
df.preds$action_type = as.factor(df.preds$action_type)

# K-fold Cross-Validation
set.seed(100)
Train <- createDataPartition(dfTrain$shot_made_flag, p=0.75, list=FALSE)
training <- dfTrain[ Train, ]
testing <- dfTrain[ -Train, ]

ctrl <- trainControl(method = "repeatedcv",
                     number = 25,
                     repeats = 5,
                     classProbs = T)

# Generalized Linear Model Training
mod_fit <- train(shot_made_flag ~ action_type + attendance + arena_temp +
    game_event_id + seconds_remaining + minutes_remaining +
    loc_y + game_date + loc_x + playoffs,
    data=training, method="glm",
    family="binomial",
    trControl = ctrl,
    tuneLength = 5,
    metric = "logLoss")
```

```r
# Interal Model Performance Metrics
pred = predict(mod_fit, newdata=testing)
cf = confusionMatrix(table(data=as.numeric(pred>0.5), testing$shot_made_flag))
misclassificationRateInternal = (cf$table[2,1]+cf$table[1,2]) / sum(cf$table)

# ROC/AUC
Compute AUC for predicting Class with the model
pred <- prediction(pred, testing$shot_made_flag)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")

aucInternal <- performance(pred, measure = "auc")
aucInternal <- aucInternal@y.values[[1]]

#LOG LOSS AND PREDICTION FOR TRAINING DATA
testing$prob = predict(mod_fit, newdata=testing)
loglossTraining = testing %>%
  mutate(logloss = testing$shot_made_flag * log(1-testing$prob) + (1-testing$shot_made_flag)*log(1-testing

  #Will generate log loss value
  loglossValue = -1/5174 * sum(loglossTraining$logloss)

# External Model performance Metrics
ex.pred = predict(mod_fit, newdata=dfTrain)
ex.cf = confusionMatrix(table(data=as.numeric(ex.pred>0.5), dfTrain$shot_made_flag))
misclassificationRateEx = (cf$table[2,1]+cf$table[1,2]) / sum(cf$table)

# ROC/AUC
Compute AUC for predicting Class with the model
expredROC <- prediction(ex.pred, dfTrain$shot_made_flag)
experf <- performance(expredROC, measure = "tpr", x.measure = "fpr")

aucEx <- performance(expredROC, measure = "auc")
aucEx <- aucEx@y.values[[1]]

# LOG LOSS AND PREDICTION FOR External CV
dfTrain$prob = predict(mod_fit, newdata=dfTrain)
loglossTraining = dfTrain %>%
  mutate(logloss = dfTrain$shot_made_flag * log(1-dfTrain$prob) + (1-dfTrain$shot_made_flag)*log(1-dfTrain

  #Will generate log loss value
  loglossValue = -1/20697 * sum(loglossTraining$logloss)

# Logistic Internal Cross-Validation Metrics
lr.SpecSense.confusion.internal <- data.frame(cf$byClass)
lr.AccuracyP.confusion.internal <- data.frame(cf$overall)

lr.Accuracy.confusion.internal <- AccuracyP.confusion.internal[1,]
lr.Sensitivity.confusion.internal <- SpecSense.confusion.internal[1,]
lr.Specificity.confusion.internal <- SpecSense.confusion.internal[2,]
lr.Precision.confusion.internal <- SpecSense.confusion.internal[5,]

#  External Cross-Validation Metrics
lr.SpecSense.confusion.external <- data.frame(ex.cf$byClass)
lr.AccuracyP.confusion.external <- data.frame(ex.cf$overall)
lr.Accuracy.confusion.external <- lr.AccuracyP.confusion.external[1,]
lr.Sensitivity.confusion.external <- lr.SpecSense.confusion.external[1,]
lr.Specificity.confusion.external <- lr.SpecSense.confusion.external[2,]
lr.Precision.confusion.external <- lr.SpecSense.confusion.external[5,]


# Internal Cross-Validation Confusion Matrix
lr.confusionFrame.internal <- data.frame(rbind(round(lr.Sensitivity.confusion.internal, digits=5),round(lr

rownames(lr.confusionFrame.internal) <- c("Sensitivity","Specificity","Precision","Accuracy","Misclassifica

# External Cross-Validation Confusion Matrix
lr.confusionFrame.external <- data.frame(rbind(round(lr.Sensitivity.confusion.external, digits=5),round(lr

rownames(lr.confusionFrame.external) <- c("Sensitivity","Specificity","Precision","Accuracy","Misclassifica

lr.confusionFrame <- data.frame(lr.confusionFrame.internal, lr.confusionFrame.external)
colnames(lr.confusionFrame) <- c("Internal CV Statistics", "External CV Statistics")

suppressWarnings(kable(lr.confusionFrame, format="latex", booktabs = T)) %>%
  kable_styling(latex_options="striped", position = "center")


topshots = as.data.frame(coef(mod_fit$finalMode))
names(topshots)[1] = "Coefficient (logit)"
topshots$`Odds Ratio` = exp(topshots$`Coefficient (logit)`)
topshots =  topshots[order(topshots$`Odds Ratio`),]
```

```r
# Prediciton Code for Logistic Regression
df.preds$prob = predict(mod_fit, newdata=df.preds)
logistic_prediction =  df.preds %>%select(recId, prob)
write.csv(logistic_prediction,"logistic_prediction.csv")
```

"'r suppressWarnings(kable(head(topshots), format="latex", booktabs = T)) %>% kable_styling(latex_options="striped", position = "center")