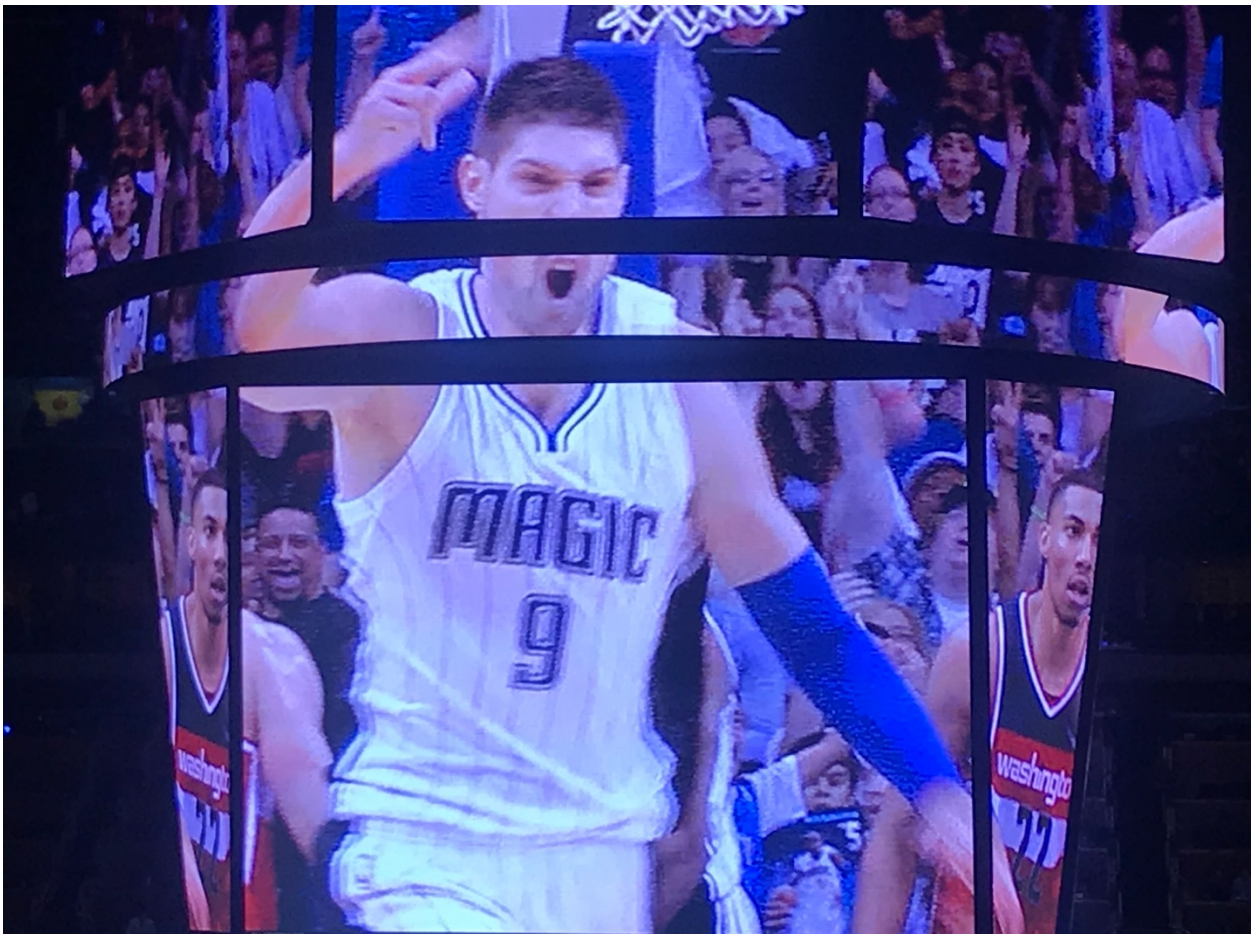
How-to Predict NBA Double-Doubles

Learn to build a logistic regression model in R to predict if NBA All-Star Nikola Vučević will score a Double-Double.



Logistic regression models allow us to estimate the probability of a categorical response variable based on one or more inputs known as predictor variables. Traditionally the responses are binary True/False values but could also be other combinations such as Pass/Fail, or even a categorical Small/Medium/Large.

An example of our goal once the model is built, is to say, with 95% confidence we can predict the outcome of the response variable 80% of the time based on the predictor variables of X, Y, and Z. The percentages would vary based on the testing specifications and quality of the model.

The article will focus on creating a model that predicts the probability a single NBA player, Nikola Vucevic, will score a double-double in an NBA basketball game. This will be demonstrated by providing a walkthrough of the steps necessary to build a logistic regression model.

Nikola Vucevic is an All-Star center for the Orlando Magic. Besides playing for my hometown team, he is also a consistent player with a long tenure on the same team, which makes his basketball statistics ideal for data science projects.

Throughout his 10-year career, Vucevic has achieved over 344 double-doubles. In the NBA a double-double is defined as obtaining 10 or more in two categories of either points, rebounds, assists, steals, or blocks. This is most often accomplished by scoring 10 or more points and 10 or more assists in a single game or 10 or more points and 10 or more rebounds in a single game.

Importing Data

The first step in building any model is to obtain an accurate dataset. Basketball-Reference tracks data points for NBA players and is often a starting point for building predictive models. From a **player's page,** there are two methods to obtain game data.

1. Use an R package like rvest to scrape player data from each season.
2. Download CSV files for each season and then upload them in R.

In Vucevic's case, you should have 10 datasets representing the 2012 through 2021 seasons.

Once the game log data is in R, add a new column "Season" to each dataset and then use rbind() to combine the individual datasets into a single "Vucevic" dataset.

```
#Add Column to Indicate Season
Vucevic2021$Season <- 2021#Use rbind() to Combine Data Frames
Vucevic <- rbind(Vucevic2012, Vucevic2013, Vucevic2014, Vucevic2015,
Vucevic2016,Vucevic2017, Vucevic2018, Vucevic2019, Vucevic2020, Vucevic2021)
```

Cleaning Data

While highly accurate, data sourced from Basketball-Reference needs a bit of cleaning before we can utilize it in our model. For this dataset in particular we need to remove rows that do not represent games played, update missing column names and update data values in the Location and WinLoss columns.

```
#Remove rows that do not correspond to a basketball game.
Vucevic <- Vucevic [!(Vucevic$GS == "Did Not Play" |
                Vucevic$GS == "Did Not Dress" |
                Vucevic$GS == "GS" |
                Vucevic$GS == "Inactive" |
                Vucevic$GS == "Not With Team"),]#Use the index method to add missing
column names
colnames(Vucevic)[6] <-c("Location")
colnames(Vucevic)[8] <-c("WinLoss")
```

In the Location column, an “@” symbolizes away games, and null represents home games. By transforming these values to “Away” and “Home” later on, we can convert it to a factor data type to test for our model.

```
#Use an ifelse() to specify “Away” and “Home” games
Vucevic$Location <- ifelse(Vucevic$Location == "@", "Away", "Home")
```

Similarly, the WinLoss column has character values following the “W (+6)” format. While a human reading a stats line can interpret “W (+6)” to mean that the game was won by six points, for model building it is more useful for the WinLoss column to contain either “W” or “L”.

```
#Split the column using str_split_fixed()
Index <- str_split_fixed(Vucevic$WinLoss, " ", 2)#Add the new column to the Vucevic dataframe
Vucevic <- cbind(Vucevic, Index) #Add Matrix to DataFrame#Remove the previous WinLoss column
Vucevic <- Vucevic %>% select(-WinLoss)#Update the new WinLoss column
names(Vucevic)[names(Vucevic) == "1"] <- "WinLoss"#Remove the column containing (+6)
Vucevic <- Vucevic %>% select(-"2")
```

Some cleaning steps are up to personal preference. Here we change “Rk” and “G” variables to the more descriptive “TeamGameSeason” and “PlayerGameSeason”.

```
#Update Column Names
names(Vucevic)[names(Vucevic) == "Rk"] <- "TeamGameSeason"
names(Vucevic)[names(Vucevic) == "G"] <- "PlayerGameSeason"
```

Data Transformation

Hand-in-hand with data cleaning is data transformation or converting data from one data type to another. Integer, numeric, and factor data types are helpful when modeling data. To understand why it is important to recall that logistic regression is a math formula where:

Response Variable = Intercept + (SlopeCoefficient1*Variable1) + Error

Solving a math formula requires using numeric, integer, or factor inputs. While factor values such as ‘Home’ and ‘Away’ appear as text labels, underneath the hood in R, factors are stored as integers.

Currently, most of the variables in the Vucevic data frame are stored as character text values. To convert multiple variables’ data types simultaneously use the hablar library along with tidyverse.

```
#View the column names in your dataset
colnames(Vucevic)#View the current datatype of an individual column variable
datatype(Vucevic$TeamGameSeason) #Convert variable datatypes
Vucevic <- Vucevic %>% convert(
```

```
            int("TeamGameSeason", "PlayerGameSeason", "FG",
              "FGA", "3P", "3PA", "FT", "FTA", "ORB",
              "DRB", "TRB", "AST", "STL", "BLK", "TOV",
              "PF", "PTS", "+/-", "PlayerGameCareer"),
            num("FG%", "3P%", "FT%", "FG%", "FT%", "3P%",
              "GmSc"),
            dte("Date"),
            fct("Team", "Location", "Opponent", "WinLoss",
              "GameStarted"))
```

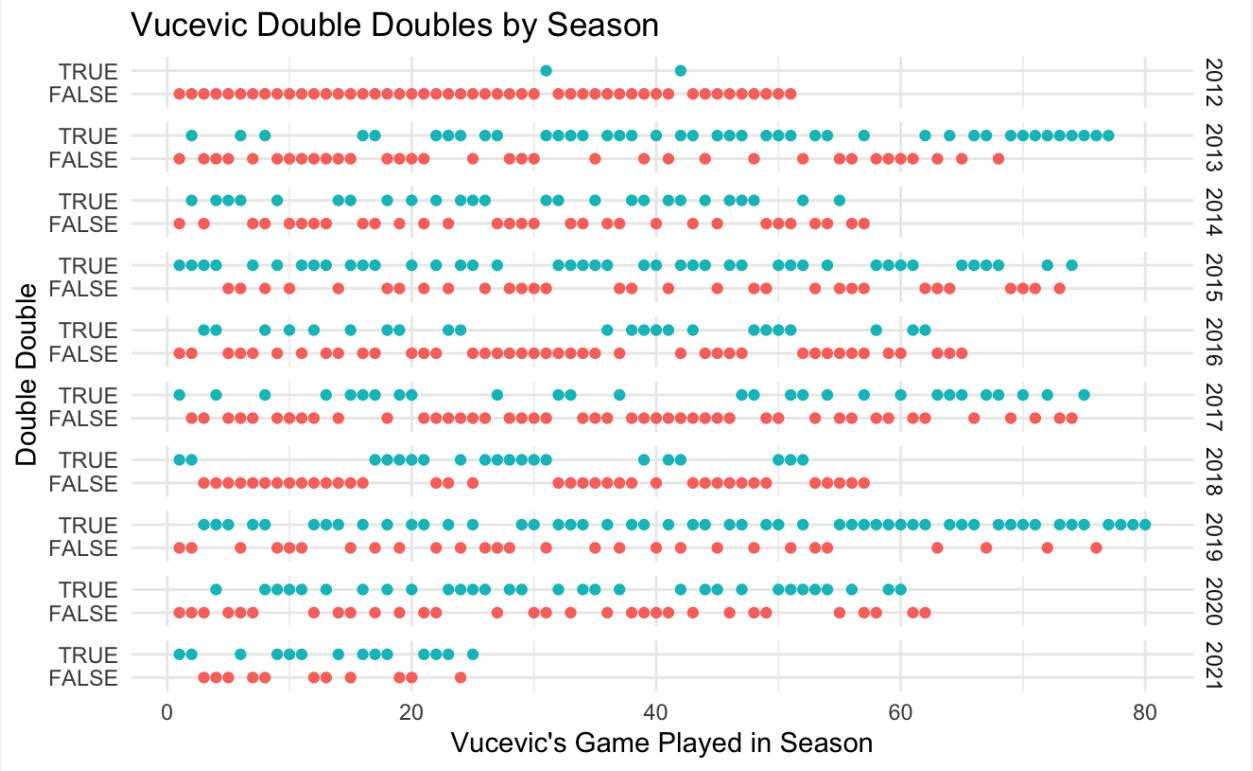
Create Double-Double Response Variable

In order to predict if Vucevic will score a double-double in the future, we need to calculate which games he has scored a double-double in the past.

As mentioned previously, a double-double is defined as obtaining 10 or more in two categories of either points, rebounds, assists, steals, or blocks.

A nested ifelse() can be used to calculate which previous games Vucevic has scored a double-double. Once the new variable is created, we can use ggplot2 to visualize the results.

```
#Create a variable that calculates DoubleDoubles
Vucevic$DoubleDouble <- ifelse(Vucevic$PTS>10 & Vucevic$AST>10,TRUE,
                ifelse(Vucevic$PTS>10 & Vucevic$TRB>10,TRUE,
                ifelse(Vucevic$PTS>10 & Vucevic$BLK>10,TRUE,
                ifelse(Vucevic$PTS>10 & Vucevic$STL>10,TRUE,
                ifelse(Vucevic$AST>10 & Vucevic$TRB>10,TRUE,
                ifelse(Vucevic$AST>10 & Vucevic$BLK>10,TRUE,
                ifelse(Vucevic$AST>10 & Vucevic$STL>10,TRUE,
                ifelse(Vucevic$TRB>10 & Vucevic$BLK>10,TRUE,
                ifelse(Vucevic$TRB>10 & Vucevic$STL>10,TRUE,
                ifelse(Vucevic$BLK>10 & Vucevic$STL>10,TRUE,
                FALSE)))))))))
```



(*Image created by the author in RStudio*)

Splitting Data into Training & Testing Sets

Before running our model, we need to split the Vucevic dataset into separate training and testing datasets. Splitting our data allows us to use one set for training our model, and one set to test how well the model works.

Use the rsample package to specify both the data split and variable to use in a stratified re-sampling approach.

Here the data uses a 70/30 split into VucevicTrain and VucevicTest datasets. Then identifies DoubleDouble to ensure there is a similar ratio of True/False values in both datasets.

```
#Identifying the split
set.seed(123)
VucevicSplit <- initial_split(Vucevic, prob = 0.7, strata = "DoubleDouble")#Creating training dataset
VucevicTrain <- training(VucevicSplit)#Creating testing dataset
VucevicTest <- testing(VucevicSplit)
```

Preparing for Multiple Logistic Regression

One method to improve model accuracy is to incorporate multiple variables when making a prediction. The following methods were utilized to determine which variables to include:

1. Determine left side leakage variables and remove them from the Vucevic dataset.
2. Remove post-game variables and others with little meaning.
3. Use a correlation matrix to remove variables from the Vucevic dataset that are statistically similar.
4. Create new variables that provide added value.

Left Side Data Leakage

In a regression model, the response variable is on the left and the predictor variables are on the right. Left Side Leakage refers to when variables that are inputs into the model are also used to calculate the value of the response variable.

When predicting if Vucevic will have a double-double, we need to ensure that the prediction is not based on any of the variables that are used to calculate a double-double.

Points, rebounds, assists, steals, and blocks are all directly related to how we calculated a double-double and will be removed. Additional variables that are a component of those 5 variables will also need to be removed. For example, offensive rebounds, free throws, and free throw attempts.

```
#Removing left side leakage variables
Vucevic <- Vucevic %>% select(-FG, -FGA, -FT, -FTA, -ORB, -DRB,
                              -TRB, -AST, -STL, -BLK, -PTS, -GmSc)#Any variable names that start with a number, or include % will need "" to remove
Vucevic <- Vucevic %>% select(-'FG%', -'3P', -'3PA', -'3P%', -'FT%')
```

Removing Post-Game Variables

The goal of our model is to ***predict*** whether Vucevic will score a double-double. While we could do that after a game when game stats list the number of turnovers and fouls, we are more likely to use the model before the game starts. To account for this, we remove all variables that are only available after the game has been played.

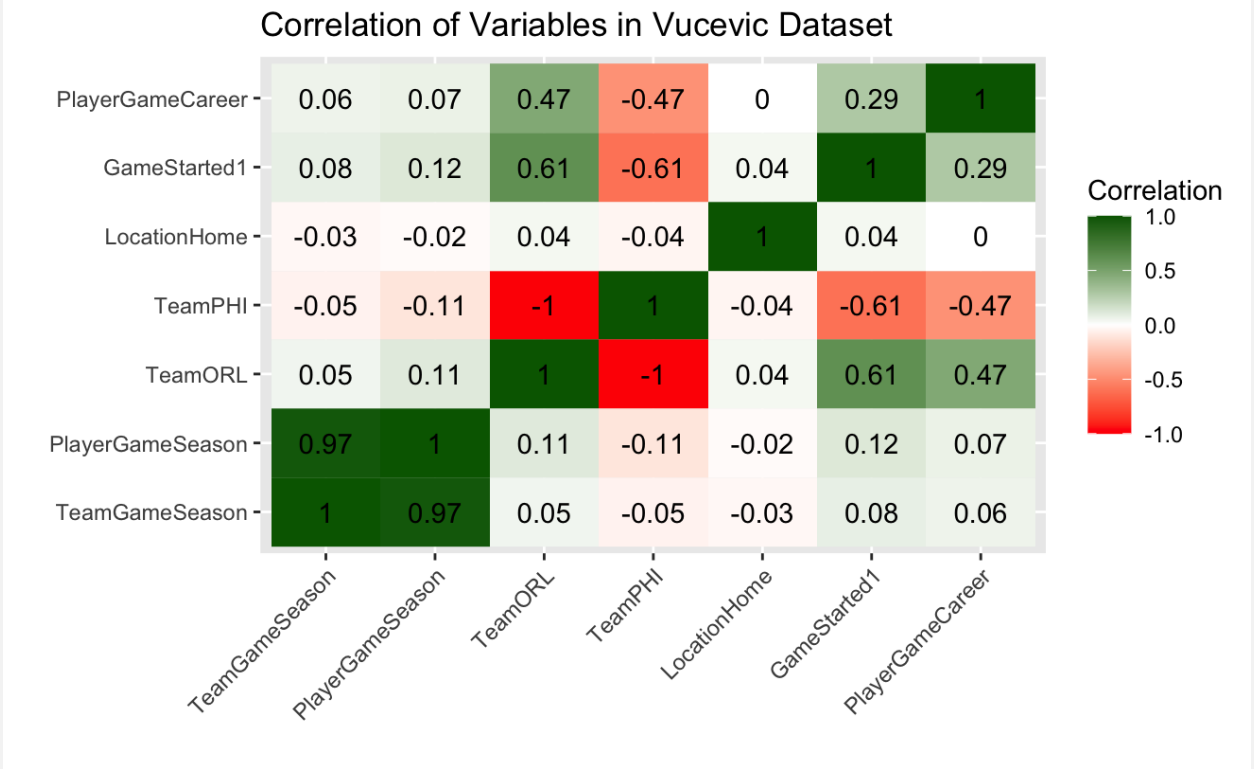
```
#Remove post-game variables
Vucevic <- Vucevic %>% select(-TOV, -PF, -"+/-", -WinLoss)
```

Remove any other variables that are unlikely to affect your model. Here MinsPlayed and Seconds are very similar to the variable Minutes and Age is like PlayerGamesCareer. The Player variable is removed from the Vucevic dataset since the only player we are analyzing is Vucevic.

```
#Remove additional variables
Vucevic <- Vucevic %>% select(-Age, -MinsPlayed, -Player, -Seconds)
```

Correlation Matrix to Identify Statistically Similar Variables

By creating a correlation matrix on the remaining variables, we can determine if any of the variables are statistically similar to each other.



(*Image created by the author in RStudio*)

The image above shows there is a strong correlation between TeamGameSeason and PlayerGameSeason as indicated by the dark green color and 0.97 value where the two variables intersect.

PlayerGameSeason is calculated by the number of games Vucevic has played in a season, while TeamGameSeason is calculated by the number of games his team has played in a season. If Vucevic had long or frequent injuries, these calculations would show more variation. Since they do not, we can get rid of either. Here the variable TeamGameSeason was removed.

Both TeamPHI and TeamORL impact multiple other variables. This is likely because Vucevic only played for Philadelphia in his rookie year. As a rookie, he was less likely to start (GameStarted1), had played fewer games (PlayerGameCareer), and was less likely to score a double-double.

We can safely get rid of the Team variable, as other variables will be better predictors.

Create Additional Variables

Now there are only four variables remaining, PlayerGameSeason, Location, GameStarted, and PlayerGameCareer to build a multiple logistic regression model.

While our model may use these four variables, additional variables may provide better insights.

- **Back to Back:** created using a mutate() and lag() to indicate if a game was played the previous night.
- **Conference:** Created using a nested ifelse() for Eastern or Western conference.
- **Time Zone:** Created using a nested ifelse() with an or operator.

```
#Create New Variable Days Since Last Game
Vucevic <- Vucevic %>%
arrange(Vucevic$Date) %>%
mutate(DaysSinceLastGame = Vucevic$Date - lag(Vucevic$Date))#Create New Variable
BackToBack
Vucevic$BackToBack <- ifelse(Vucevic$DaysSinceLastGame == 1, TRUE, FALSE)#Delete
DaysSinceLastGame
Vucevic <- Vucevic %>% select(-Date, -DaysSinceLastGame)
```

Please note the code to create Conference and Time Zone variables is truncated due to space constraints and only shows Atlanta and Boston.

```
#Create New Variable Conference
Vucevic$Conference <-  ifelse(Vucevic$Opponent == "ATL", "Eastern",
                 ifelse(Vucevic$Opponent == "BOS", "Eastern",#Create New Variable Time Zone
Vucevic$TimeZone <- ifelse(Vucevic$Location == "Home", "Eastern",
               ifelse(Vucevic$Location == "Away" &
                    Vucevic$Opponent == "ATL", "Eastern",
               ifelse(Vucevic$Location == "Away" &
                    Vucevic$Opponent == "BOS", "Eastern",
```

Running the Multiple Logistic Regression Model

After testing numerous combinations of the seven remaining independent variables in logistic regression models the following two models were chosen to examine further due to the independent variables' low p-values.

In the second model shown below, we add PlayerGameCareer even though the p-value is higher than 0.05 because it lowers the p-value of the other independent variables.

```
#Logistic Regression Model 1
LogisticRegMultiple1 <- glm(DoubleDouble ~ BackToBack +GameStarted
                    +PlayerGameSeason, family = "binomial",
                    data = VucevicTrain)#Logistic Regression Model 2
LogisticRegMultiple2 <- glm(DoubleDouble ~ +GameStarted +BackToBack
                    +PlayerGameSeason +PlayerGameCareer,
                    family = "binomial", data = VucevicTrain)
```

term <chr>	estimate <dbl>	std.error <dbl>	statistic <dbl>	p.value <dbl>
(Intercept)	-1.534510751	0.400742891	-3.829165	0.0001285787
BackToBackTRUE	-0.459923253	0.253571268	-1.813783	0.0697111423
GameStarted1	1.243550104	0.387530915	3.208906	0.0013324125
PlayerGameSeason	0.008509434	0.004644616	1.832107	0.0669354547

(*Image created by the author in RStudio*)

term <chr>	estimate <dbl>	std.error <dbl>	statistic <dbl>	p.value <dbl>
(Intercept)	-1.6631932039	0.4122606129	-4.034325	5.475954e-05
GameStarted1	1.1085864215	0.3986441440	2.780892	5.420972e-03
BackToBackTRUE	-0.4527781017	0.2541898944	-1.781259	7.487011e-02
PlayerGameSeason	0.0082451924	0.0046585752	1.769896	7.674451e-02
PlayerGameCareer	0.0007964316	0.0005510837	1.445210	1.483990e-01

(*Image created by the author in RStudio*)

We can further evaluate the accuracy of multiple models by testing these models in the evaluation stage.

Evaluating the Multiple Logistic Regression Model

Since the sample size of our total dataset is only 621 games, validation with a single testing dataset could vary significantly based on the specific games in the 30% held for the testing dataset.

K-fold cross-validation is a re-sampling method that randomly divides the training data into k groups. It fits the model on K-1 folds (aka. groups) and then the group that was left out is used to test performance. This means it tests the model K times and the average K test error is the cross-validation estimate.

```
#Convert response column to factor if necessary
Vucevic$DoubleDouble <- as.factor(Vucevic$DoubleDouble)
class(Vucevic$DoubleDouble)#Example of K-fold Cross Validation Model
set.seed(123)
cv_model3 <- train(
  DoubleDouble ~ TeamGameSeason +PlayerGameSeason +GameStarted
+PlayerGameCareer +BackToBack,
  data = Vucevic,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10),
  na.action = na.exclude)#Compare 3 Models
summary(resamples(list(
    model1 = LogisticRegMultiple1,
```

```
  model2 = LogisticRegMultiple2,
  model3 = cv_model3)))$statistics$Accuracy
```

```
           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
model1 0.5000000 0.5341142 0.5645161 0.5729135 0.6104711 0.6612903    0
model2 0.5079365 0.5341142 0.5967742 0.5811060 0.5967742 0.6666667    0
model3 0.5000000 0.5541475 0.5887097 0.5953917 0.6386329 0.7096774    0
```

(*Image created by the author in RStudio*)

While a mean model score of 57%, 58%, and 59% indicates weak models, the models can be further evaluated using a confusion matrix.

From the matrix below we see that when Vucevic does score a double-double (Reference True) the model is almost as likely to predict that Vucevic will not score a double-double (Prediction False) as he will score a double-double (Prediction True).

While the model is more accurate when predicting Vucevic will not score a double-double it is still only accurate 68% of the time.

```
# predict class
pred_class <- predict(cv_model3, Vucevic, )# create confusion matrix
confusionMatrix(
 data = relevel(pred_class, ref = "TRUE"),
 reference = relevel(Vucevic$DoubleDouble, ref = "TRUE"))
```

```
Confusion Matrix and Statistics

          Reference
Prediction TRUE FALSE
     TRUE   147   106
     FALSE  135   235

               Accuracy : 0.6132
                 95% CI : (0.5737, 0.6516)
    No Information Rate : 0.5474
    P-Value [Acc > NIR] : 0.0005254
```

(*Image created by the author in RStudio*)

Conclusion

Here the most accurate logistic regression model is only able to predict whether Vucevic will score a double-double on average 61% of the time. Which means it probably should not be deployed.

While the model is not overwhelming, the statistics still show it is a success. Using no information, the computer guessed correctly if Vucevic would score a double-double 54.74% of the time. The model, with 61.32% accuracy improved on the random guessing by 6.5%.

While the model accuracy is not spectacular by looking at the p-value we can determine that observations more extreme than this are only expected to occur randomly 5,254 times out of 10 million trials.

With 95% confidence we can predict if Nikola Vucevic will score a double-double 61% of the time based on the predictor variables of TeamGameSeason, PlayerGameSeason, GameStarted, PlayerGameCareer and BackToBack.

Future Analysis

Each data scientist will use slightly different variables and methods to create a model. In this example, we created a variable BackToBack. If your dataset did not have that variable, your model results would be different.

Similarly, Each NBA player has a different game style and is influenced differently by the same variables. Some play significantly better at home, or against a former team. Others perform poorly on the second night of a back-to-back or in certain time zones. This variation means each NBA player will have a unique logistic regression model to predict if they will score a double-double.

To increase accuracy, we could incorporate other variables from additional datasets or expand the number of observations by including other players.

An example of expanding the number of observations is to include all 4,555 NBA regular season games played by any current member of the Orlando Magic.

```
4555 samples
   5 predictor
   2 classes: 'FALSE', 'TRUE'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 4099, 4100, 4099, 4100,
Resampling results:

  Accuracy   Kappa
  0.9047171  0.53971
```

Then with 95% confidence we can predict if an Orlando Magic player will score a double-double 90% of the time based on the predictor variables of Player, Season, PlayerGameSeason, GameStarted, and PlayerGameCareer.

Please drop a comment below if you found an NBA player or variable combination that increases the accuracy of the logistic regression model.