UNIVERSITY OF CALIFORNIA

Los Angeles

Using Recurrent and Mixture Density Network Architectures to Model National Basketball Association In-Game Win Probabilities

A thesis submitted in partial satisfaction of the requirements for the degree Master of Applied Statistics

by

Henry Cleveland Poole

2020

ABSTRACT OF THE THESIS

Using Neural Networks to Model National Basketball Association In-Game Win Probabilities

by

Henry Cleveland Poole

Master of Applied Statistics

University of California, Los Angeles, 2020

Professor Ying Nian Wu, Chair

There are a number of possible machine learning approaches to modeling the win probability of an NBA game. Previous publically available research suggests that a mixture density network performs best in modeling win probability. In this paper, I explain and reproduce a number of previously shared approaches to modeling win probability. Unlike previous research, I fit each model with an identical set of inputs to fairly evaluate and compare the performance of each model. Furthermore, I create a recurrent mixture density network approach based off the recommendation of previous research. I find that the recurrent mixture density network has the highest measured accuracy in comparison to all other tested models.

The thesis of Henry Cleveland Poole is approved.

Rick Paik Schoenberg

Vivian Lew

Ying Nian Wu, Committee Chair

University of California, Los Angeles

2020

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF TABLES CONT.

1. Introduction

The National Basketball Association (NBA) has a long history of professional teams using statistical analyses in search of identifying a competitive advantage in the pursuit of a championship. In comparison to many other sports, professional basketball lends itself relatively well to statistical analysis. Official box score data that tracks the number of shots taken by individual players on each team dates back to the NBA's inaugural season in 1947. Over time more counting stats measuring different elements of an individual players performance, such as rebounds and steals, were added to the box score. As a result, the NBA maintains a remarkably rich dataset of summary statistics for each game in the league's history.

During the 1996-1997 regular season, the NBA began recording play-by-play data for each game. This was a significant step forward in the development of statistical analysis in basketball. Before the introduction of play-by-play data, the only publically available statistics for the NBA were derived from box score data. The significance of the introduction of play-by-play data was the ability break down the summary counting stats from the box score into a sequence of events describing a basketball game as it progresses. For example, instead of simply knowing that Joel Embiid scored 30 points and secured 15 rebounds in a given game by looking at the box score at the end of the game, play-by-play data could describe the exact sequence of events when he scored 2 of his 30 total points on a 10 foot jump shot with 3:00 left in the 3rd quarter to extend his team's lead to 95-75 after grabbing a defensive rebound off of a missed free throw by Jared Dudley who plays on the opposing team. The introduction of this extra level of detail found in the NBA's play-by-play data facilitated the growth of basketball analytics and

allowed for the study of a number of new areas of research, one of which is the modeling of win probability.

In the context of the NBA, a win probability model attempts to assign a probability to the likelihood of each team winning the game given a current game state. Typically, the model specifically calculates the likelihood that the home team will ultimately win the game. For example, if the Boston Celtics are playing the Los Angeles Lakers in Los Angeles, a win probability model would calculate the likelihood of the Lakers winning the game given that the score is tied 100-100 with ten seconds left in the fourth quarter and that the Celtics have possession of the ball. A win probability model has practical value to both NBA front-office executives and coaches. Executives may use a win probability model to measure how much an individual player or a five-man lineup contributed to a team's win probability over the course of a game as a measure of their performance. Coaches may be interested in using a win probability model to inform their in-game decision making, such as determining the optimal time to remove their best players from a game they are winning or losing by a large margin in order to minimize those players' risk of injury. The objective of this paper is to first briefly describe and recreate a choice selection of previously studied methods for modeling win probability and then to propose a novel approach based off of a recurrent mixture density neural network architecture.

2. Description of Dataset

The data used for this project comes from the dataset shared in conjunction with the publication of "The Problem With Win Probability" at the 2018 MIT Sloan Sports Analytics Conference by Sujoy Ganguly and Nathan Frank of the STATS LLC AI Group [1]. The dataset contains over 8.7 million play-by-play events from each game spanning the 2002-2003 through 2016-2017 NBA regular seasons. For each play-by-play event in the dataset, there are 352 associated features that can be classified into 5 different groups.

The first group is the set of base features describing each play-by-play event P_t including the total game time in seconds, a binary feature describing whether the home team or away team has possession of the ball, and the current score differential defined as the home team's score minus the away team's score.

The second group of features for each event, I_t, contains integer labeled identities for both the home and away team as well an integer labeled identity for the event itself. For example, these event labels could correspond to a field goal attempt, an offensive rebound, a steal, and so on.

The third group of features for each event, X_t, is a set of box score statistics for both the home and away teams aggregated in-game up until the time of the given play-by-play event. The team-level box score statistics tracked in this set of features are assists, blocks, fouls, offensive rebounds, defensive rebounds, team rebounds, steals, and turnovers.

The fourth group of features for each event, L_t, is a player lineup vector describing the players that are on the home and away teams' rosters. For each player on a team's roster for the given game, a vector is constructed containing an integer label for the player's identity, a binary

flag for whether or not the player started the game, a binary flag for whether or not the player was unavailable to play the game due to injury, the number of games to date in the season that the player has played, the number of games to date in the season that the player has started, the number of minutes to date in the season that the player has been on the court, the player's total plus-minus rating to date in the season, the player's average per game plus-minus rating to date in the season, and the player's average number of fouls committed per game to date in the season. The player lineup vector, L_t, is the concatenation of 30 such individual player vectors corresponding to the maximum 15 players allowed on each NBA roster. For rosters with less than 15 players, the corresponding individual player vector for a missing player is padded with zeroes.

The fifth and final set of features for each event, O_t, is a set of binary flags describing which of the 30 combined players on the home and away team were active on the court for a given play-by-play event. Each of these 30 binary flags is aligned with the ordering of players in the player lineup vector, L_t.

In addition to each of these 352 features that could potentially be used as inputs to a win probability model, two different features are associated with each play-by-play event for potential use as the target variable in the model. First, there is a binary flag corresponding to whether or not the home team eventually won the game. Second, there is a continuous variable corresponding to the eventual final score differential at the end of the game calculated as the home team's final score minus the away team's final score.

Similar to the methodology used by Ganguly and Frank in their paper, for all models constructed in this project I consider data from the 2002-2003 through 2013-2014 NBA seasons as an internal season set used for training and in-season testing. I then consider the 2014-2015

through 2016-2017 NBA seasons as an external set used for out-of-season testing. From the internal seasons set, I create an internal training set consisting of all play-by-play events from a randomly selected set of 4,440 games and an internal testing set of all play-by-play events from a randomly selected set of 500 games. From the external seasons set, I create an external testing set of all play-by-play events from a randomly selected set of 500 games. No games overlap between the internal training set, internal testing set, and external testing set. Using the training and testing sets facilitates a fair comparison of the performance of different types of models.

3. Logistic Regression

A logistic regression is perhaps the most straightforward model to construct in an attempt to model NBA win probability. The dataset lends itself easily to approaching the task as a simple binary classification problem. Others have already developed a logistic regression solution for modeling win probability. Perhaps the most widely such circulated solution is maintained by Michael Beuoy and can be found on his website inpredicatble.com [2]. Beuoy's approach utilizes a locally weighted logistic regression except for the final seconds of game time where he then uses a decision tree. For this project, I create a logistic regression to function as the simplest baseline for modeling win probability. Similar to Beuoy's approach, for a given game state in the play-by-play data I consider just three features from the play-by-play dataset as inputs for the model.

Formally, the logistic regression win probability model can be defined as follow: for a given event and corresponding game state in the play-by-play data, let x_1 represent the amount of time measured in seconds that has passed during the game at the exact time of the event, let x_2 represent the score differential, and let x_3 be a binary variable representing whether or not the home team currently has possession of the ball. Lastly, let Y be a Bernoulli response variable where $P(Y = 1)$ represents the probability of the home team ultimately winning the game given x_1, x_2, and x_3. Thus, with a logistic regression, $P(Y = 1) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$, where β_0 is a fitted intercept and each of β_1, β_2, and β_3 are the corresponding regression coefficients for each input feature.

This logistic regression is constructed in Python using the scikit-learn machine learning library. I fit the model on the aforementioned internal training seasons training set and then make win probability predictions on both the internal and external seasons' testing sets.

Table 3.1 displays the odds ratio derived from the regression coefficients in the fitted logistic regression model.

Table 3.1

	β_0	β_1	β_2	β_3
Odds Ratio	0.617	0.999	1.183	0.980

To an extent, the reported odds ratios from the fitted logistic regression make intuitive sense. The intercept term suggests that the estimated ratio of play-by-play events ultimately leading to the home team winning the associated game in the model training set is approximately 62%. Given that the team playing at home generally has an advantage over their opponent, this intercept value makes sense. The translated odds ratio for β_2 also makes intuitive sense; all else being equal a one point increase in the current score differential in favor of the home team should increase the odds of the home team ultimately winning the game.

Interpreting the fitted regression coefficients for the effect of game time and ball possession is more challenging. The translated odds ratios suggest that neither the passage of time nor possession of the ball has a large impact on win probability. While this may make sense in the early stages of game, these results do not seem reasonable for end-of-game scenarios. Generally, the logistic regression model appears to ineffectively capture the effect of game time on win probability.

Tables 3.2, 3.3, and 3.4 show confusion matrix evaluation metrics for predictions made from the logistic regression model on the internal training, internal testing, and external testing sets respectively.

Table 3.2

	Precision	Recall	F1 Score
Away Team Win	0.7133	0.5905	0.6461
Home Team Win	0.7625	0.8471	0.8026

Table 3.3

	Precision	Recall	F1 Score
Away Team Win	0.7434	0.5984	0.6631
Home Team Win	0.7354	0.8439	0.7859

Table 3.4

	Precision	Recall	F1 Score
Away Team Win	0.7389	0.6089	0.6676
Home Team Win	0.7182	0.8225	0.7668

Three results illustrated in the preceding tables are of particular interest. First, there are generally incremental performance decreases when evaluating predictions made on the internal training set versus both the internal and external seasons testing sets. This is expected as the model is not trained on any play-by-play event data from games that were randomly selected to be part of the testing. While the performance is expectedly worse in the testing sets, the

differences in evaluation metrics found in the preceding confusion matrices are not so large to indicate issues with over-fitting.

Another interesting result from the preceding confusion matrices is the performance differences in recall. Given the model formulation, recall in the event of an away team win corresponds to the model's specificity while recall in the event of a home team win corresponds to the model's sensitivity. The preceding logistic regression model is notably more sensitive than it is specific, meaning that home team wins are correctly predicted by the logistic regression model at a higher rate than away team wins are correctly predicted. In most binary classification tasks there is a trade off between a model's sensitivity and specificity. In the context of modeling NBA win probability, I suspect that the disparity is a reflection of the concept of home court advantage. The idea of the home team having an advantage is a long acknowledged trope of basketball and sports analysis even though its exact causes and effects are both contested and highly variable from game-to-game. The preceding results suggest that this baseline logistic regression win probability model may be slightly biased too much in favor of the home team.

Table 3.5 shows the overall accuracy of the classifier calculated with predictions made on the internal training, internal testing, and external testing sets respectively.

Table 3.5

Data Set	Accuracy
Internal Seasons' Training	0.7465
Internal Seasons' Testing	0.7382
External Seasons' Testing	0.7259

Similar to the confusion matrix metrics, there is an expected slight decline in predictive performance comparing internal training accuracy versus both internal and external testing accuracy but the decline is not large enough to suggest over-fitting.

Figures 3.1, 3.2, and 3.3 each plot a receiver operating characteristic curve, or ROC curve, alongside the associated area under the curve metric, AUC. These plots were made with predictions on the internal training, internal testing, and external testing sets, respectively.

Figure 3.1

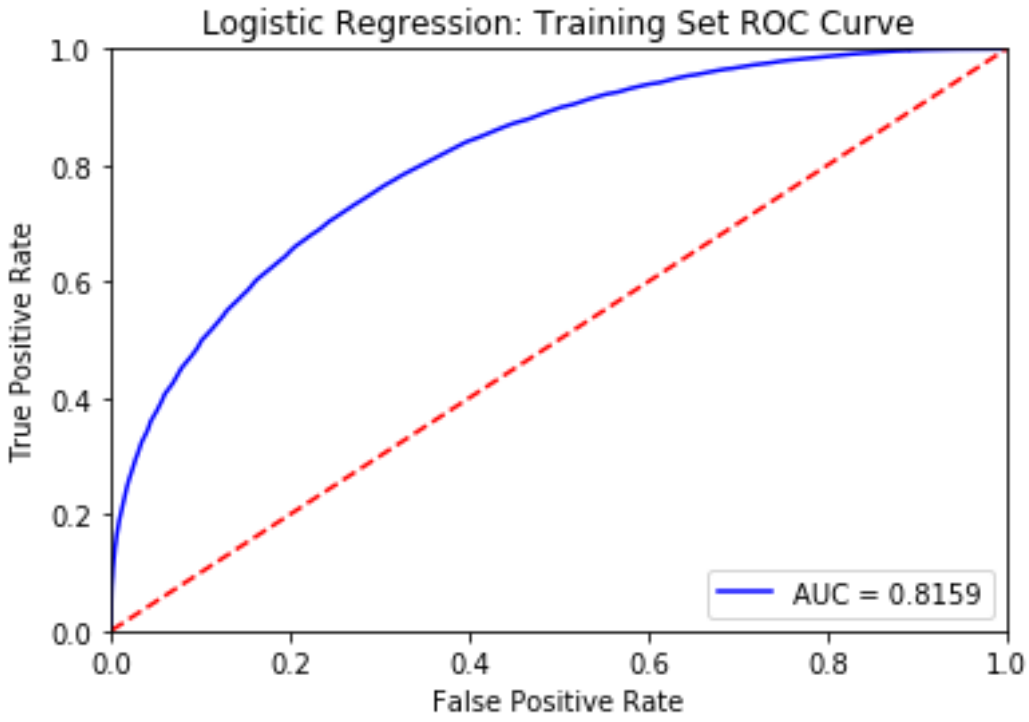


Figure 3.2

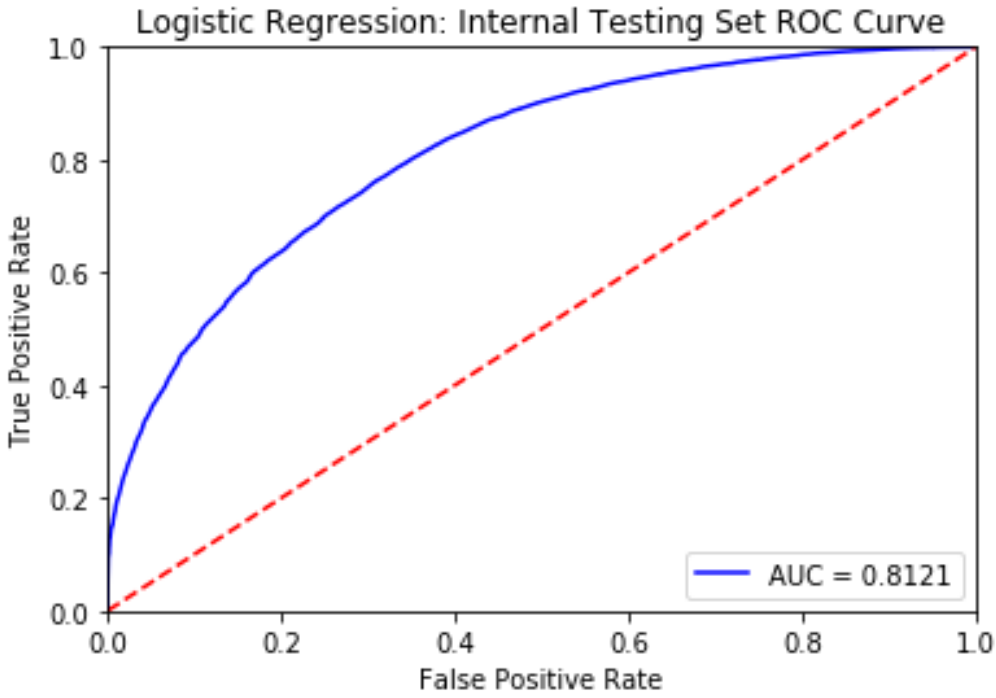
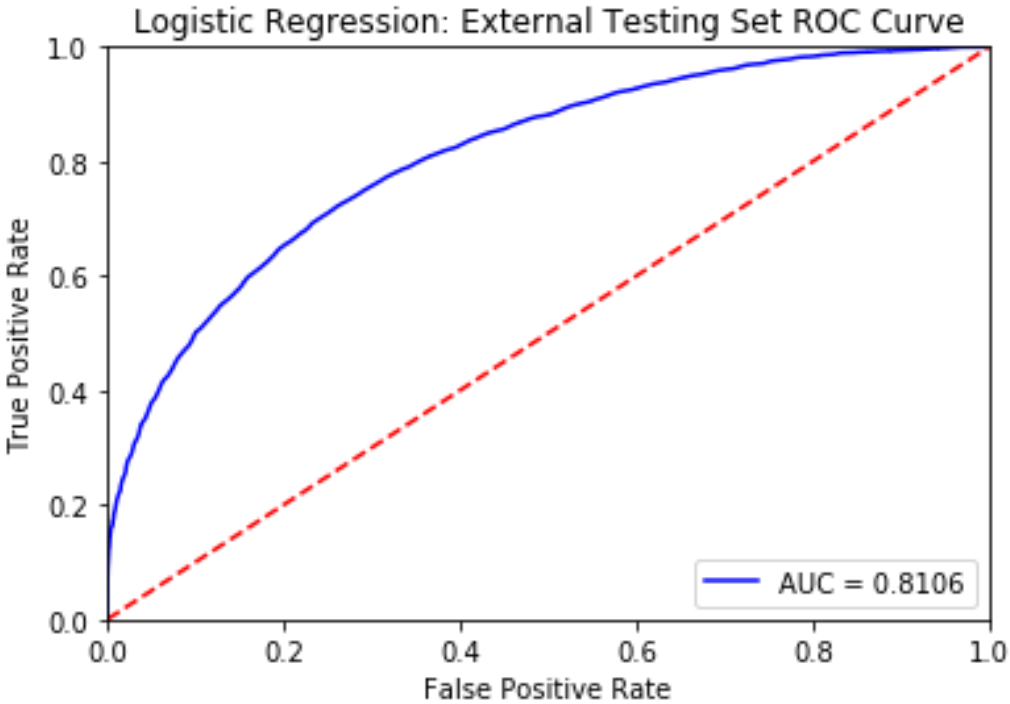


Figure 3.3



Similar to the preceding results, there is an expected slight decline in predictive performance measured by AUC for internal training predictions and both internal and external testing predictions. Thus, there are no clear signs of over-fitting or other aberrations in the model. The ROC curves are shaped similarly across each of the three data sets. On the external testing set in particular, with a false positive rate of about 40%, there is a true positive rate of about 80%.

The preceding results suggest that a logistic regression win probability model performs adequately in predicting the ultimate outcomes of games, even when making predictions outside of the training data sample. While the model can effectively serve as a baseline for comparison for predictive performance with other types of models, I expect other model formulations to perform better, specifically in their ability to account for the effect of game time and ball possession.

4. Random Forest

A random forest model is also an appropriate choice for modeling NBA win probability. Similar to a logistic regression model, a random forest model is a valid approach to modeling win probability if the task is structured as a binary classification problem. A potential advantage of using a random forest instead of a logistic regression for this task is that the random forest may more effectively be able to capture the effect of any non-linearly separable input variables to the model that significantly affect win probability. While the results of the logistic regression model suggest that game time and ball possession do not have a significant effect on win probability, this does not make intuitive sense and may be a result of logistic regression's relative inability to capture the effect of non-linearly separable input variables on a binary output variable.

For this project, I train a random forest with the same play-by-play input features as the preceding logistic regression in order to evaluate the predictive accuracy of the two different models. The random forest model is built using the scikit-learn machine learning library. Similar to the baseline random forest model presented by Ganguly and Frank in their aforementioned research on win probability, the random forest I construct consists of 20 trees each of which sees at most the square root of available features at each level in the tree and splits on a minimum of 200 samples with a minimum of 100 samples per leaf node.

Table 4.1 lists the feature importance values from the fitted model.

Table 4.1

Input Feature	Game Time	Score Differential	Possession
Feature Importance	0.1463	0.8536	0.0001

Similar to the coefficient values of the fitted logistic regression model, the feature importance values of the fitted random forest model suggest that the current score differential is the most important factor in modeling win probability. Game time is also significant, albeit to a significantly lesser degree than score differential, while the importance of possession appears to be negligible. These feature importance values suggest that the random forest model may be modeling the non-linearly separable effect of game time on win probability more effectively than the logistic regression.

Tables 4.2, 4.3, and 4.4 list confusion matrix metrics calculated using predictions made with the fitted random forest model on the internal training, internal testing, and external testing sets respectively.

Table 4.2

	Precision	Recall	F1 Score
Away Team Win	0.7365	0.5793	0.6485
Home Team Win	0.7617	0.8665	0.8107

Table 4.3

	Precision	Recall	F1 Score
Away Team Win	0.7511	0.5739	0.6507
Home Team Win	0.7267	0.8562	0.7861

Table 4.4

	Precision	Recall	F1 Score
Away Team Win	0.7479	0.5866	0.6575
Home Team Win	0.7105	0.8368	0.7685

The preceding confusion matrix metrics show similar results to the logistic regression model. Again, there are expected slight declines in performance from the training set to the external testing set; however, the difference is not large enough to suggest that there are serious issues with over-fitting. Also similar to the logistic regression, the random forest model is much more sensitive than it is specific meaning that it correctly predicts home team wins at a higher rate than away team wins.

Table 4.5 shows the overall accuracy of the random forest classifier evaluated with predictions made on the internal training, internal testing, and external testing sets respectively.

Table 4.5

Data Set	Accuracy
Internal Seasons' Training	0.7540
Internal Seasons' Testing	0.7347
External Seasons' Testing	0.7237

The accuracy results for the random forest model and the logistic regression model are nearly identical, with the logistic regression model performing slightly better. Figures 4.1, 4.2,

and 4.3 show the ROC curves and associated AUC metrics made with random forest predictions on the internal training, internal testing, and external testing sets respectively.

Figure 4.1

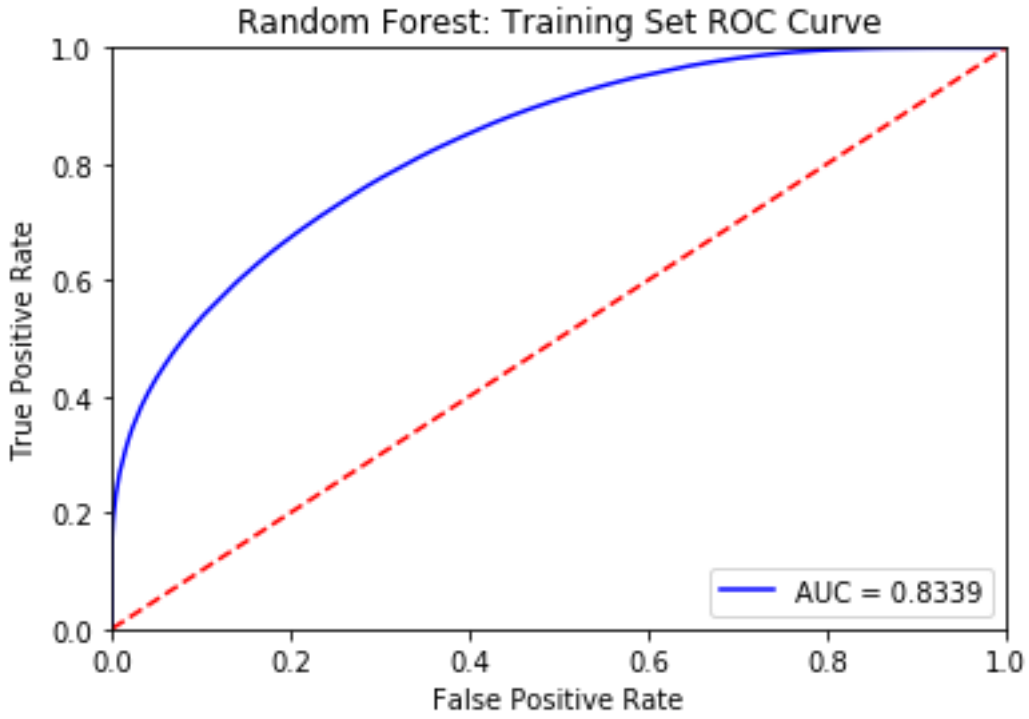


Figure 4.2

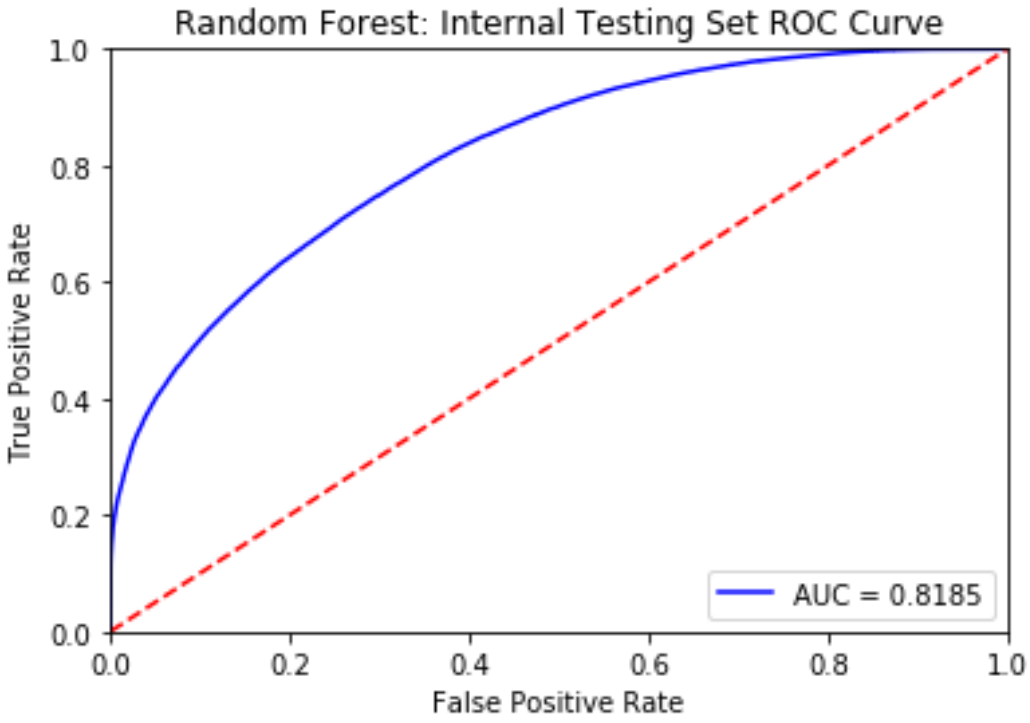
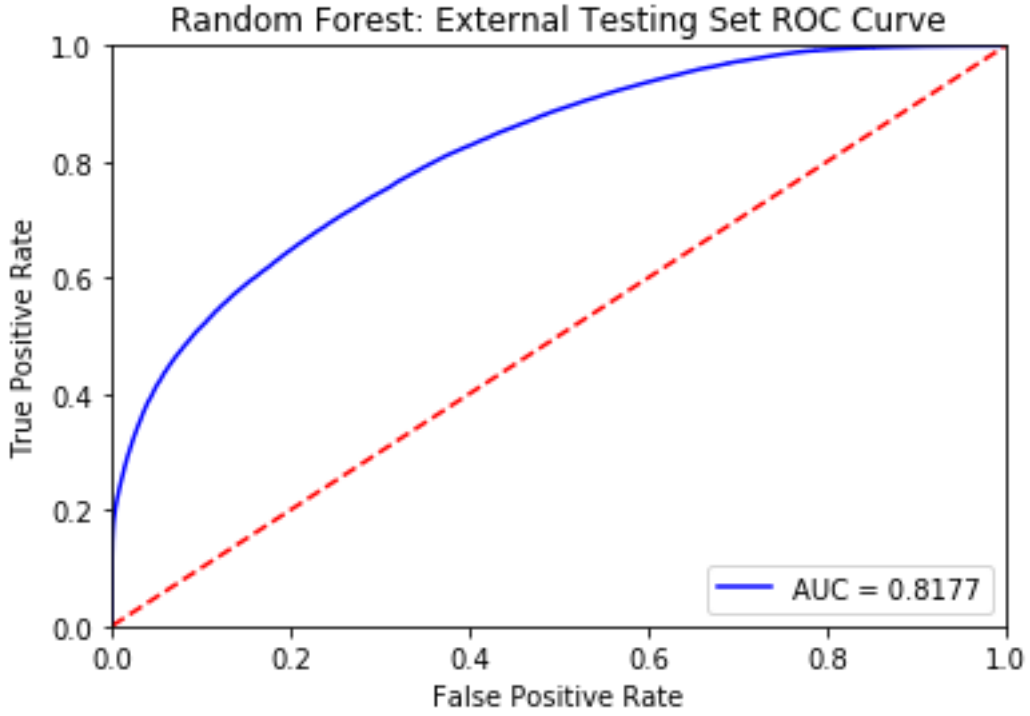


Figure 4.3



The preceding random forest ROC curves are shaped similarly to the logistic regression ROC curves. The AUC for the random forest model is slightly higher than the AUC for the logistic regression model, which is one indicator that the random forest model may perform better than the logistic regression model.

Overall, the predictive performance of the random forest model is similar to that of the logistic regression model. In theory, the random forest model can more effectively model any non-linearly separable input features from the play-by-play data; however, that did not translate to a significantly more accurate predictive model in this case.

5. Neural Networks For Binary Classification

Neural networks are also a valid model choice to predict NBA win probability. Similar to random forests, neural networks can appropriately model the effect of any non-linearly separable play-by-play input features on a binary output variable. In addition, the number of samples in the play-by-play dataset is large enough to reasonably expect that a neural network trained on such a dataset should generalize well enough to make predictions on out-of-sample data points.

In Ganguly and Frank's research on win probability, they fit a mixture density network model using every feature in the play-by-play dataset as an input to the model. Their mixture density model performs much stronger than that of a number of baseline random forest models they also construct using different limited subsets of input features in the play-by-play dataset. In addition, they suggest that adapting their mixture density network model into a recurrent architecture may further boost predictive performance. They theorize that a recurrent architecture could more effectively represent the effect of game time on win probability.

In this paper, I construct 4 different neural networks to model NBA win probability. Each model is built in Python using the Keras library as a higher level API to a Tensorflow backend. Unlike Ganguly and Frank's methodology, I restrict the input features to these different neural network models to the same three play-by-play data point used to fit the preceding baseline logistic regression and random forest models. I restrict the inputs to examine whether any increased predictive performance can be attributed to different model architectures as opposed to simply increasing the number of features used as inputs to the model.

5.1 Fully Connected Network For Binary Classification

The first neural network I consider for modeling win probability is a fully connected architecture that accepts game time, score differential, and ball possession as inputs and predicts a binary win classification output variable. The network contains a single hidden layer with four nodes and a hyperbolic tangent activation function. The output layer contains a single node with a sigmoid activation function. The model is trained on the internal training seasons set for 5 epochs with a batch size of 32 samples. The binary crossentropy loss function is used with the Adam optimizer during training.

Tables 5.1, 5.2, and 5.2 show the confusion matrix metrics calculated with predictions made by the fully connected neural network win probability model on the internal training, internal testing, and external testing sets respectively.

Table 5.1

	Precision	Recall	F1 Score
Away Team Win	0.7333	0.5625	0.6366
Home Team Win	0.7549	0.8682	0.8076

Table 5.2

	Precision	Recall	F1 Score
Away Team Win	0.7612	0.5700	0.6519
Home Team Win	0.7268	0.8648	0.7898

Table 5.3

	Precision	Recall	F1 Score
Away Team Win	0.7535	0.5834	0.6576
Home Team Win	0.7103	0.8426	0.7708

The confusion matrix metrics for the fully connected binary classification win probability model show similar overall performance to the baseline logistic and random forest models. On the external testing set, the neural network's F1 score is slightly higher than that of the random forest model for both away team wins and home team wins, which is an indicator that the neural network model may have greater predictive performance than the random forest model.

Table 5.4 shows the overall accuracy of predictions made by the fully connected neural network on the internal training, internal testing, and external testing sets, respectively.

Table 5.4

Data Set	Accuracy
Internal Seasons' Training	0.7484
Internal Seasons' Testing	0.7379
External Seasons' Testing	0.7254

The overall accuracy on the external testing set is slightly higher for the fully connected neural network than that of the random forest model, which suggests that the fully connected neural network may have stronger predictive performance.

Figures 5.1, 5.2, 5.3 show the ROC curves and corresponding area under the curve metrics calculated using predictions made on the internal training set, internal testing set, and external testing set respectively.

Figure 5.1

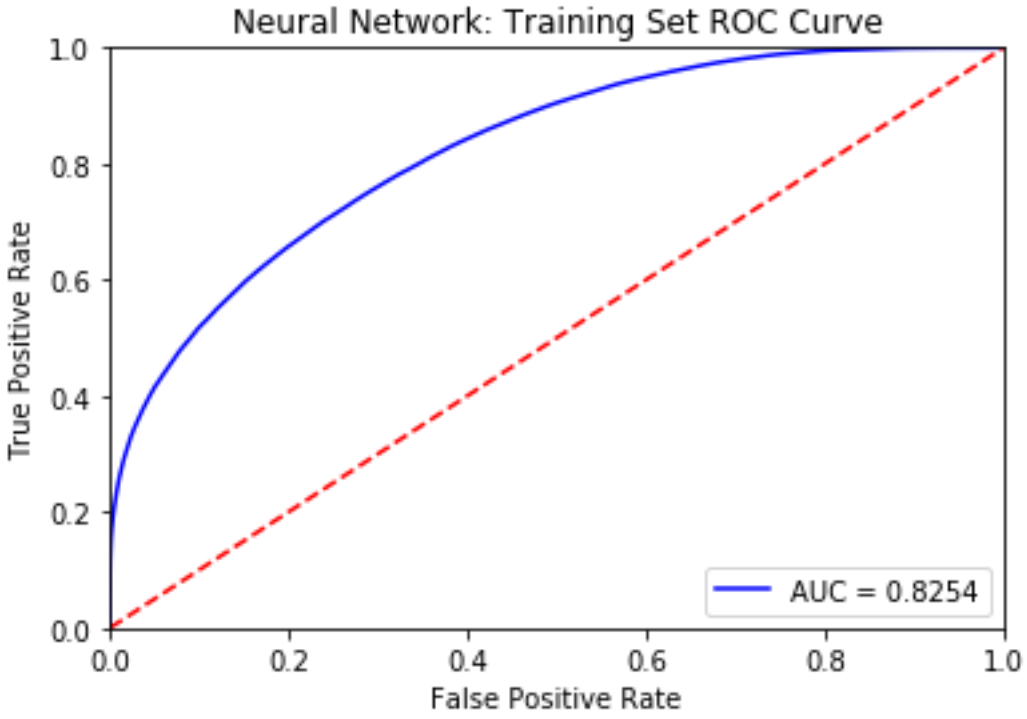


Figure 5.2

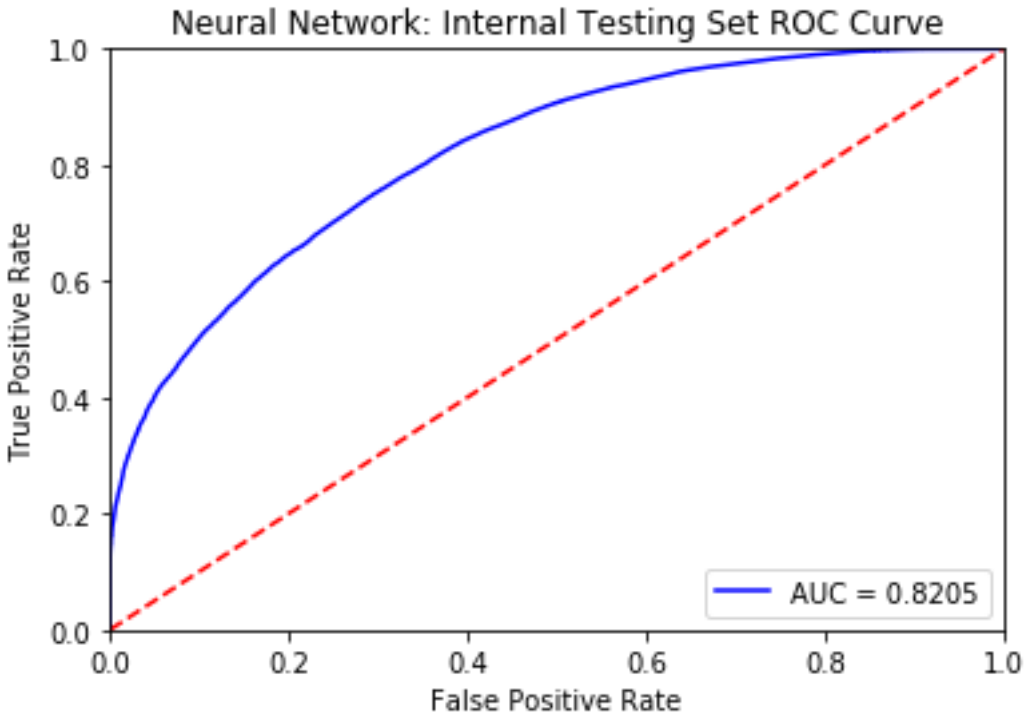
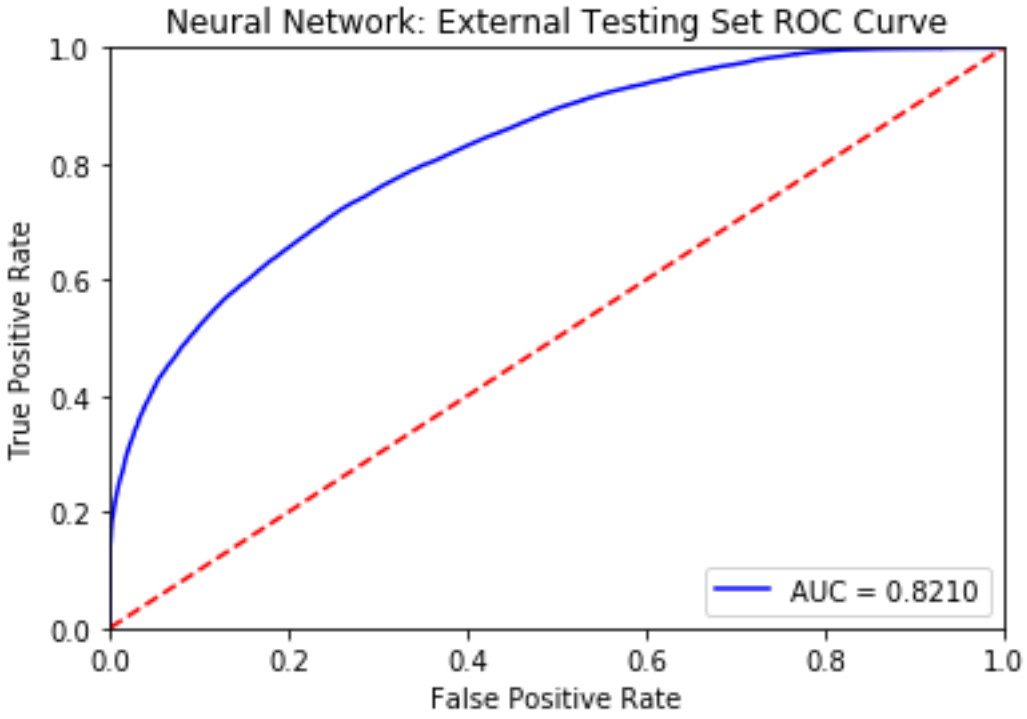


Figure 5.3



The neural network's AUC for predictions made on the external testing set is slightly higher than that of the random forest AUC. Similar to the confusion matrix metrics and overall accuracy, this is another indicator that a fully connected neural network may perform better than a random forest for modeling win probability.

5.2 Recurrent Fully Connected Network For Binary Classification

Ganguly and Frank suggest in their win probability research that a recurrent neural network architecture may better account for the effect of game time than simply using game time as an input to a fully connected network. There are additional intuitive reasons to believe that a recurrent neural network model architecture may work well for modeling win probability. First, the sequential nature of the play-by-play data lends itself easily to fitting a recurrent model architecture. Furthermore, it is reasonable to expect that a recurrent model may be able to identify patterns during training in sequences of play-by-play events that lead to greater predictive performance as opposed to only considering a single event in isolation during training, if such patterns indeed exist. Basketball games are often determined by sequences of dominant play by one team for a stretch of game time followed by a dominant sequence by the opposing team. Some basketball analysts may refer to this sequential phenomenon of strong performance over a series of play-by-play events as momentum. It is difficult to formally establish whether or not momentum is real in the context of a basketball; however, a recurrent network should learn any significant sequential patterns in the data that materially effect win probability, if they do in fact exist.

After fitting the fully connected binary classification neural network for modeling win probability, I construct three different recurrent binary classification neural networks. Each of these networks has a similar architecture. Each contains a single hidden fully connected recurrent layer with 4 nodes and a hyperbolic tangent activation function. The output layer contains one unit and a sigmoid activation function. The model is trained for 5 epochs in batches of 32 sequences with a binary crossentropy loss function and the Adam optimizer. The difference among the three models is the length of the input sequence data. For this project, I arbitrarily experiment with sequence lengths of 10, 20, and 50 play-by-play events. For sequences early in the game, I pad zero values to the beginning of the sequence to ensure that each contains the exact same length. Each sequence contains the same three input features used in the preceding win probability models: game time, current score differential, and ball possession.

Table 5.5 shows the area under the ROC curve metric values calculated on the external seasons testing set for each of the three recurrent network models with different input sequence lengths.

Table 5.5

Recurrent Model Input Sequence Length	External Testing Set AUC
10	0.8207
20	0.8210
50	0.8206

The preceding AUC results suggest that an input sequence length of 20 produces the most predictive recurrent model. The difference is small, and there is no monotonic pattern suggesting that a larger or smaller input sequence length produces a more accurate model. The AUC

calculated from the recurrent model with an input sequence length of 20 is identical to the AUC calculated from the non-recurrent fully connected binary classification model.

Tables 5.6, 5.7, and 5.8 contain confusion matrix metrics calculated with predictions made by the recurrent fully connected neural network model with an input sequence length of 20 on the internal training, internal testing, and external testing sets respectively.

Table 5.6

	Precision	Recall	F1 Score
Away Team Win	0.7509	0.5327	0.6232
Home Team Win	0.7464	0.8862	0.8103

Table 5.7

	Precision	Recall	F1 Score
Away Team Win	0.7763	0.5392	0.6364
Home Team Win	0.7170	0.8826	0.7912

Table 5.8

	Precision	Recall	F1 Score
Away Team Win	0.7674	0.5524	0.6424
Home Team Win	0.7001	0.8619	0.7726

Table 5.9 contains the overall accuracy scores calculated from predictions made on the internal training, internal testing, and external testing sets, respectively.

Table 5.9

Data Set	Accuracy
Internal Seasons' Training	0.7476
Internal Seasons' Testing	0.7348
External Seasons' Testing	0.7254

Figures 5.4, 5.5, and 5.6 show the ROC curve and corresponding area under the curve metric calculated with predictions made on the internal training, internal testing, and external testing sets respectively.

Figure 5.4

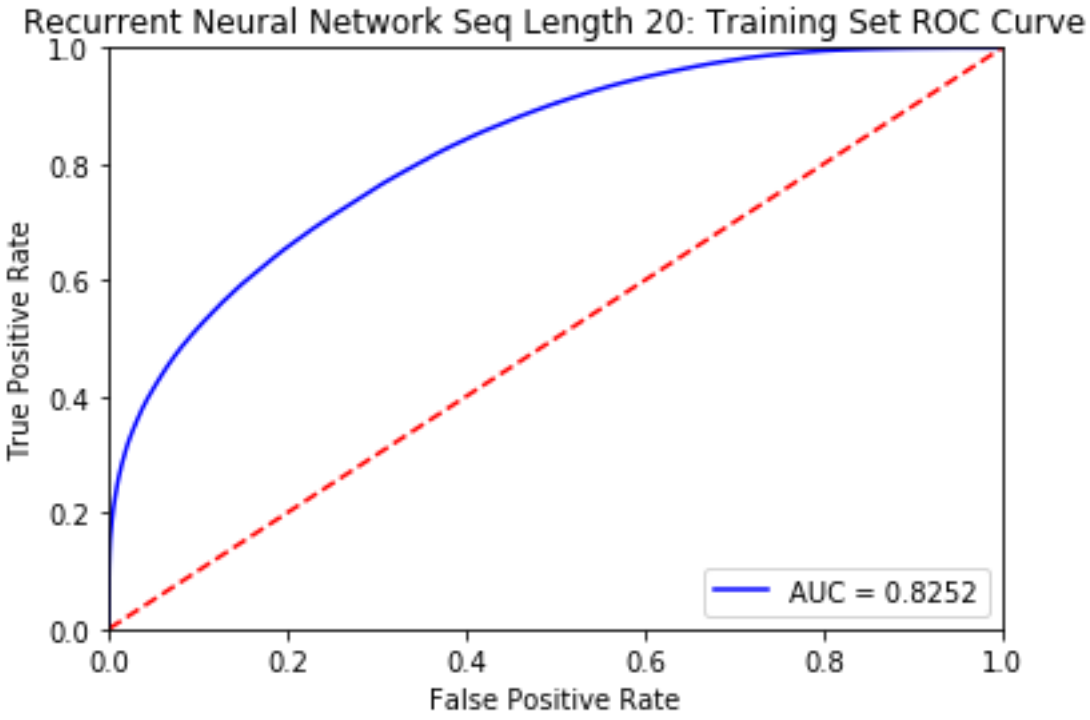


Figure 5.5

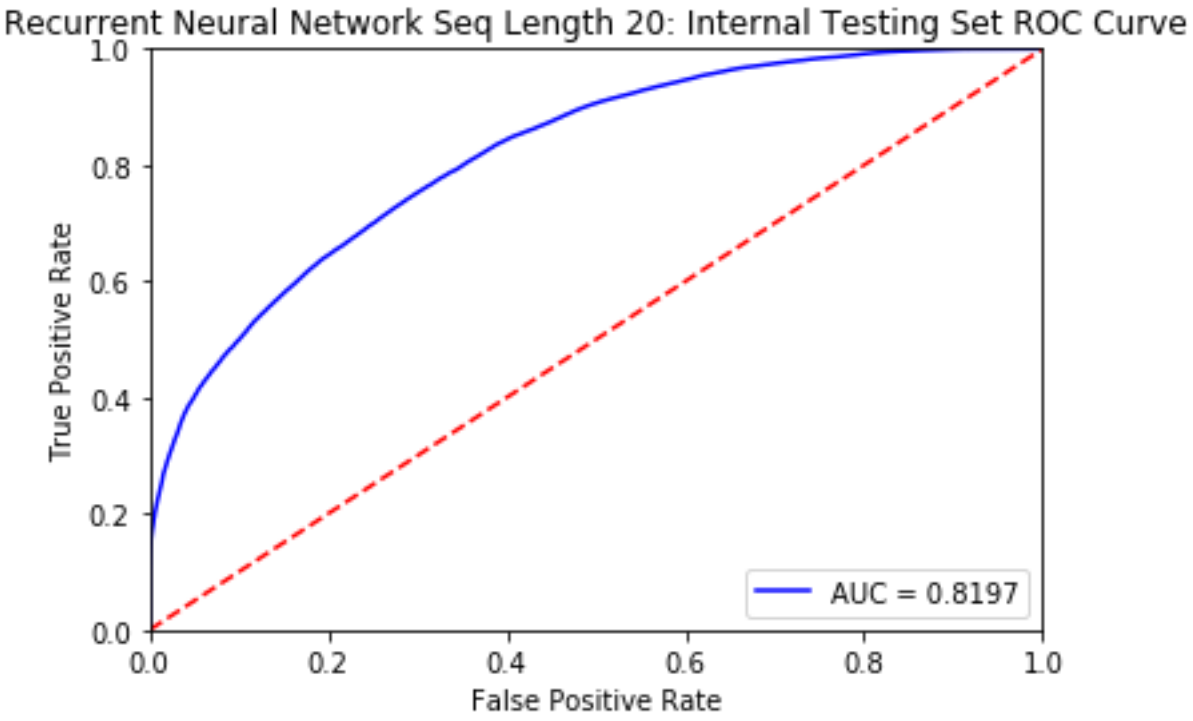
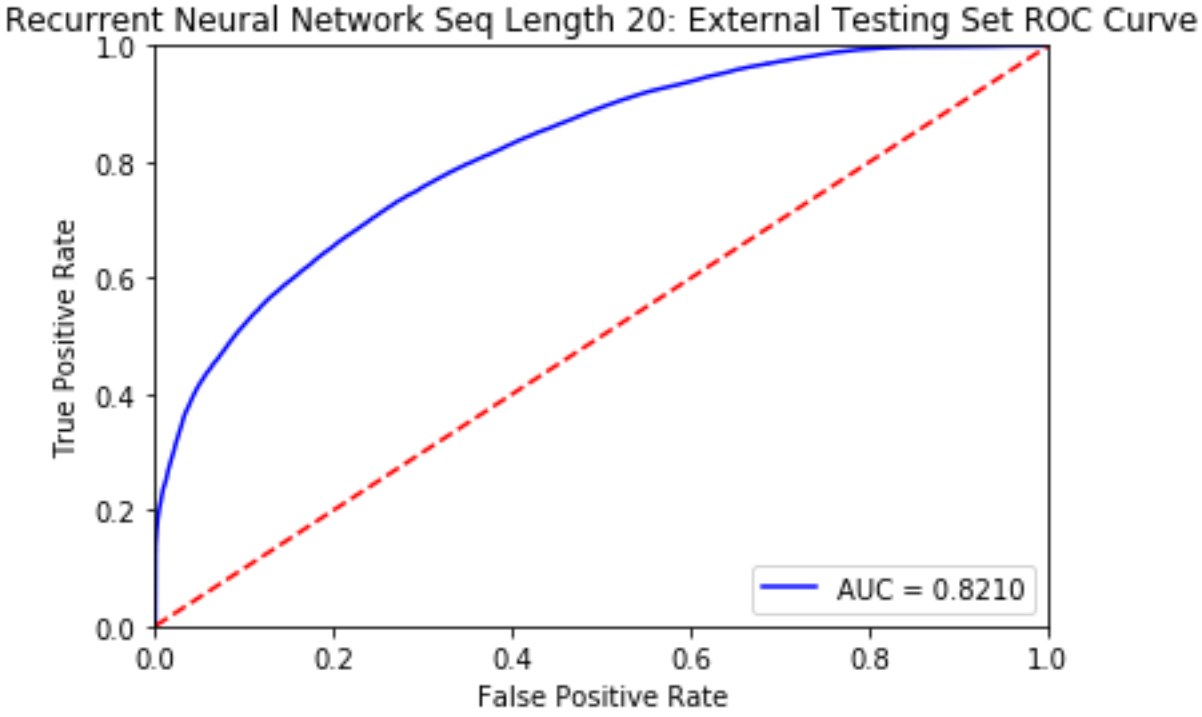


Figure 5.6



Overall, the predictive performance of the fully connected recurrent neural network with an input sequence length of 20 performs similarly to the fully connected non-recurrent neural network. From these results, there are no apparent significant predictive performance gains from using a recurrent architecture instead of a non-recurrent architecture when approaching the win probability model task as a binary classification problem.

6. Mixture Density Networks

Christopher Bishop first proposed mixture density network architectures in 1994 in his research titled ‘Mixture Density Networks’. A mixture density network accepts an input vector and produces a conditional probability distribution of a continuous output variable given the input vector. Bishop states that mixture density networks can “represent arbitrary conditional probability distributions in the same way that a conventional neural network can represent arbitrary functions” [4].

Mixture density networks fit conditional probability distributions by first mapping an input vector to a parameter vector with a neural network. The outputted parameter vector contains the parameters to a mixture model that then represents a conditional probability distribution. Commonly, and for this particular win probability project, mixture density networks are used to fit Gaussian mixture models. A key parameter to the architecture of a mixture density network is the selection of the number of mixture component distributions that make up the larger mixture model. For example, if 3 mixtures are selected before fitting the Gaussian mixture density network; the resulting parameter vector mapped during model training will contain a fitted conditional mean and standard deviation for each of the three Gaussian distributions that comprise the mixture model as well as three mixing components that represent the weighting of each individual Gaussian mixture in the larger mixture model. Even though the output variable being modeled may not necessarily follow a Gaussian mixture model, Bishop cites research stating that Gaussian mixture models “can approximate any given density function to arbitrary accuracy, provided the mixing coefficients and the Gaussian parameters are correctly chosen.” The correct choice for mixing coefficients and Gaussian parameters can be ensured by selecting

the correct loss function to minimize using gradient descent during the training of the neural network that ultimately outputs the conditional probability density's parameter vector. Bishop proves in his research that a negative log-likelihood function is the appropriate choice for fitting a mixture density network.

Ganguly and Frank construct a mixture density network in their aforementioned NBA win probability research. In the context of modeling NBA win probability, a mixture density network can accept any number of inputs to create the conditional probability distribution of the final score differential. From the fitted conditional final score distribution, home team win probability can be calculated by computing the final score cumulative distribution function to find the probability that the final score differential will be less than or equal to zero. Previous research proves that the cumulative distribution function of a Gaussian mixture model can be calculated with a weighted average of the cumulative distribution functions of the individual Gaussian mixture component distributions where the weights are identical to the mixture weights of the each component distribution [4]. In Ganguly and Frank's work, their mixture density network outperforms their constructed baseline random forest models, albeit using a larger input vector with the mixture density network than the baseline random forest models. In this paper, I fit two separate mixture density models using the same three input features used in all preceding models. First, I fit a mixture density network with a standard feed-forward fully connected network mapping the mixture model parameter vector. Then, I fit a mixture density network with a recurrent fully connected network mapping the model parameter vector to see if a recurrent architecture produces any improvements in predictive performance over a non-recurrent architecture.

6.1 Mixture Density Network For Final Score Distribution Estimation

To evaluate the performance of a mixture density network for modeling NBA win probability, I first fit three separate models with a different number of Gaussian mixture components. Each network contains a single hidden layer with 4 nodes and a hyperbolic tangent activation function before the mixture model parameter layer. Similar to all preceding models in this paper, the model accepts game time, score differential, and ball possession as inputs to the model. Table 6.1 describes the area under the ROC curve metrics for mixture density networks with 1, 3, and 5 mixture components respectively.

Table 6.1

MDN Mixture Components	External Testing Set AUC
1	.8186
3	.8155
5	.8145

From the three tested models, it appears that AUC decreases slightly as the number of components in the mixture density network increases.

Tables 6.2, 6.3, and 6.4 show confusion matrix metrics calculated from predictions made with a mixture density network with a single mixture component on the internal training, internal testing, and external testing sets respectively.

Table 6.2

	Precision	Recall	F1 Score
Away Team Win	0.7375	0.5527	0.6318
Home Team Win	0.7519	0.8732	0.8080

Table 6.3

	Precision	Recall	F1 Score
Away Team Win	0.7662	0.5627	0.6489
Home Team Win	0.7247	0.8702	0.7908

Table 6.4

	Precision	Recall	F1 Score
Away Team Win	0.7564	0.5715	0.6511
Home Team Win	0.7058	0.8482	0.7705

The preceding confusion matrix results suggest that the mixture density network performs similarly to, if not slightly worse than, the fully connected binary classification neural network. Table 6.5 shows the overall accuracy metrics calculated with predictions made by the mixture density network with one mixture component on the internal training, internal testing, and external testing sets, respectively.

Table 6.5

Data Set	Accuracy
Internal Seasons' Training	0.7485
Internal Seasons' Testing	0.7390
External Seasons' Testing	0.7263

Interestingly, the predictive accuracy of the mixture density network on the external test set is slightly higher than that of all preceding models fitted in this project. This reinforces Ganguly and Frank's findings in their aforementioned research that mixture density networks perform better than random forests for accurately modeling win probability.

Figures 6.1, 6.2, and 6.3 show the ROC curve and associate area under the curve metrics for predictions made from the mixture density network with a single Gaussian mixture component on the internal training, internal testing, and external testing sets respectively.

Figure 6.1

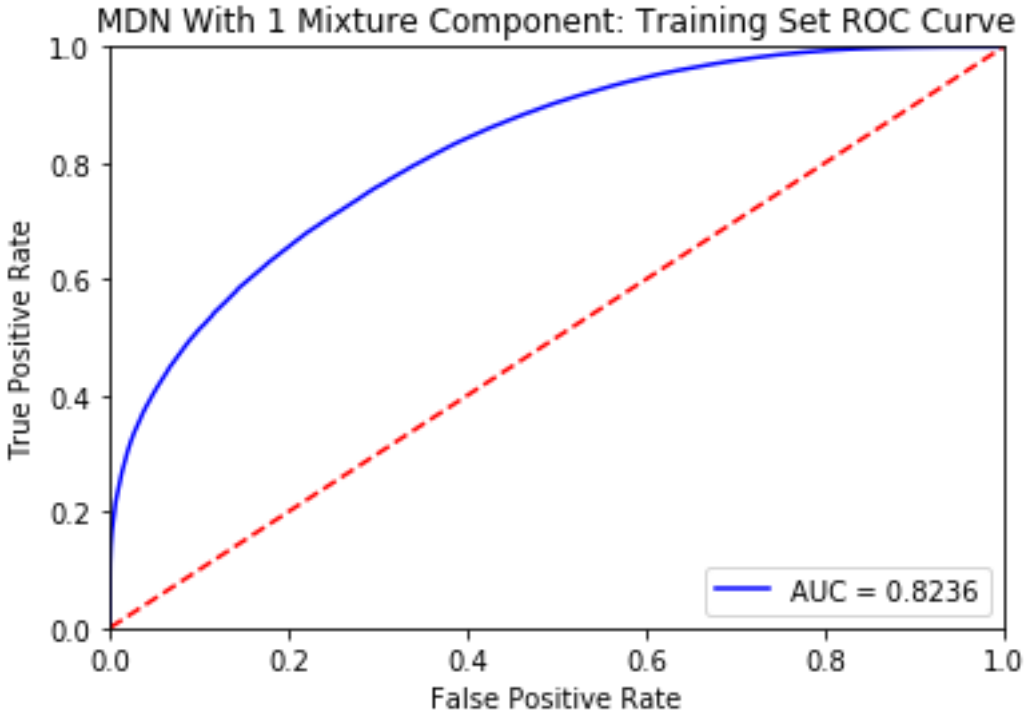


Figure 6.2

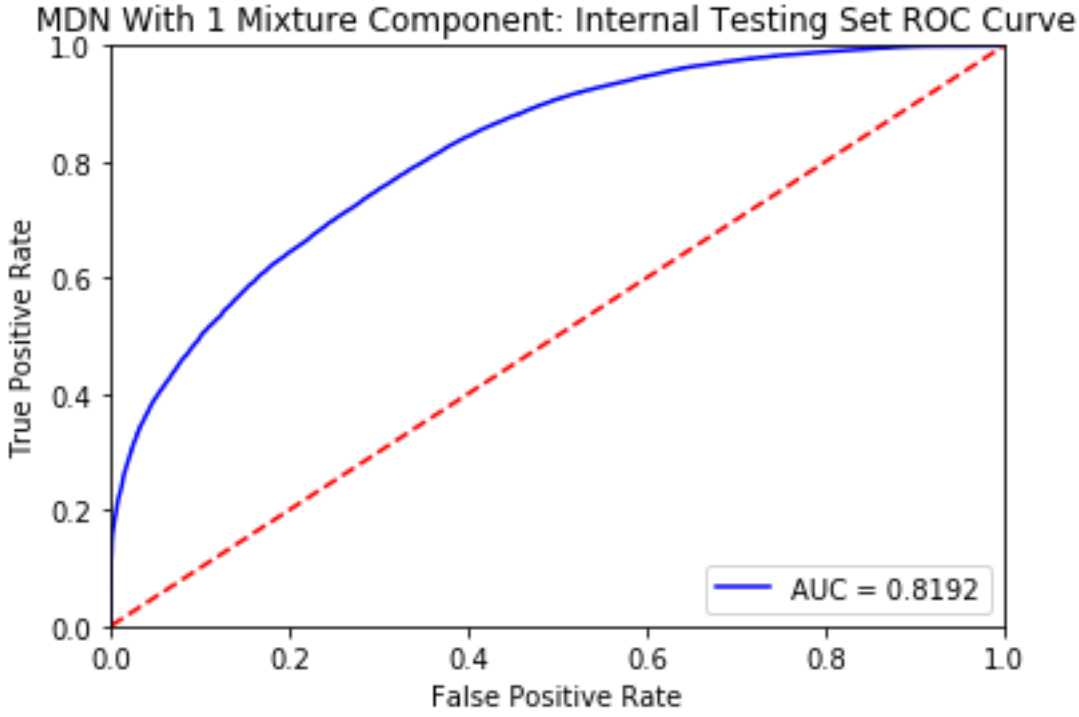
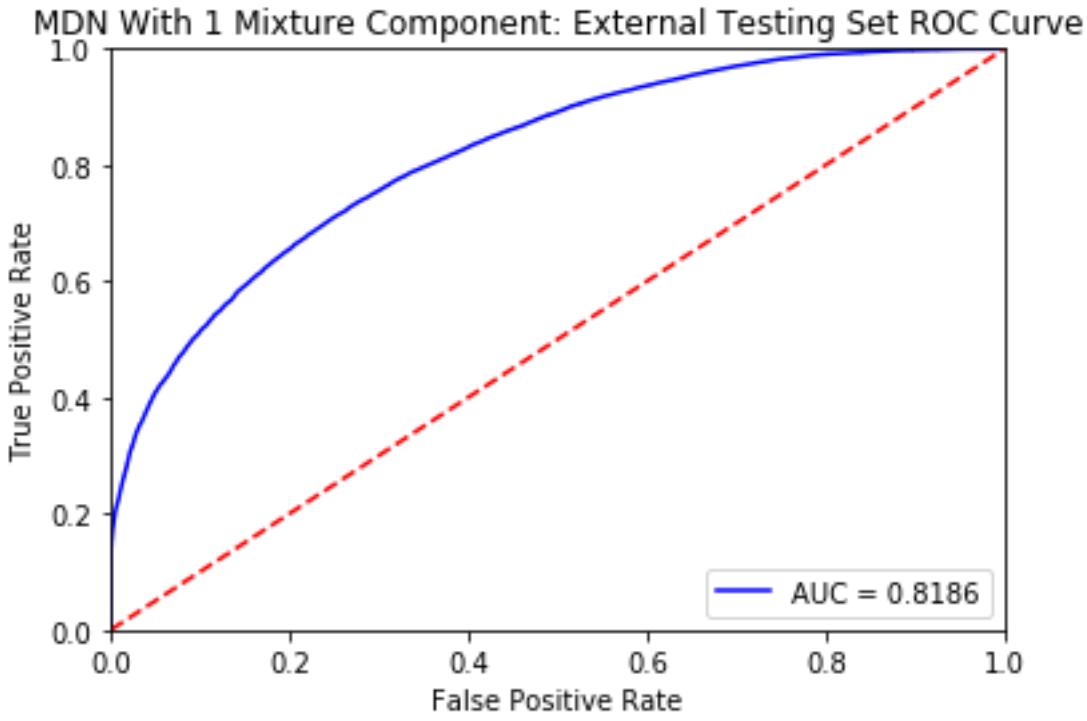


Figure 6.3



Unlike the preceding accuracy metrics, the ROC curves and associated AUC metrics for the mixture density network suggest that the model performs slightly worse than the fully connected binary classification network. Overall, it appears that the mixture density network performs similarly to both the fully connected binary classification network as well as the recurrent fully connected binary classification network for predicting NBA win probability with an equivalent set of inputs.

6.2 Recurrent Mixture Density Network For Final Score Distribution Estimation

Similar to the preceding binary classification networks, I fit a recurrent mixture density network model to compare the predictive performance of a recurrent mixture density network architecture versus a non-recurrent mixture density network architecture. The fitted recurrent mixture density model accepts input sequences of length 20 and contains a single Gaussian mixture component. The network contains a single recurrent fully connected layer with 4 nodes and a hyperbolic tangent activation function before the mixture model parameter layer.

Tables 6.6, 6.7, and 6.8 show confusion matrix metrics calculated from predictions made with the fitted recurrent mixture density network on the internal training, internal testing, and external testing respectively.

Table 6.6

	Precision	Recall	F1 Score
Away Team Win	0.6997	0.6131	0.6536
Home Team Win	0.7692	0.8304	0.7986

Table 6.7

	Precision	Recall	F1 Score
Away Team Win	0.7276	0.6195	0.6692
Home Team Win	0.7414	0.8247	0.7808

Table 6.8

	Precision	Recall	F1 Score
Away Team Win	0.7304	0.6318	0.6775
Home Team Win	0.7267	0.8076	0.7650

One of the most interesting results from the preceding external test set confusion matrix metrics is that the away team recall, equivalent to the overall model sensitivity, is the highest for the recurrent mixture density network in comparison to all other preceding fitted models. This means that away teams are correctly predicted to ultimately win the game at the highest rate in comparison to all preceding models, which suggests that the recurrent architecture may be capturing information in the sequential input data that more accurately reflects the aforementioned concept of momentum for teams playing away from their home arena. Otherwise, the confusion matrix metrics are similar to both the non-recurrent mixture density network as well as the binary classification networks.

Table 6.9 shows the overall accuracy of the recurrent mixture density network calculated with predictions made on the internal training, internal testing, and external testing sets, respectively.

Table 6.9

Data Set	Accuracy
Internal Seasons' Training	0.7453
Internal Seasons' Testing	0.7364
External Seasons' Testing	0.7281

The overall accuracy on the external testing set is the highest with the recurrent mixture density network in comparison to all previously fitted models, including the non-recurrent mixture density network. This supports the hypothesis put forward by Ganguly and Frank in their aforementioned research that a recurrent model architecture may more effectively model win probability than a non-recurrent architecture.

Figures 6.4, 6.5, and 6.6 show the ROC curves and associated area under the curve metrics for the recurrent mixture density network calculated with predictions made on the internal training, internal testing, and external testing sets, respectively.

Figure 6.4

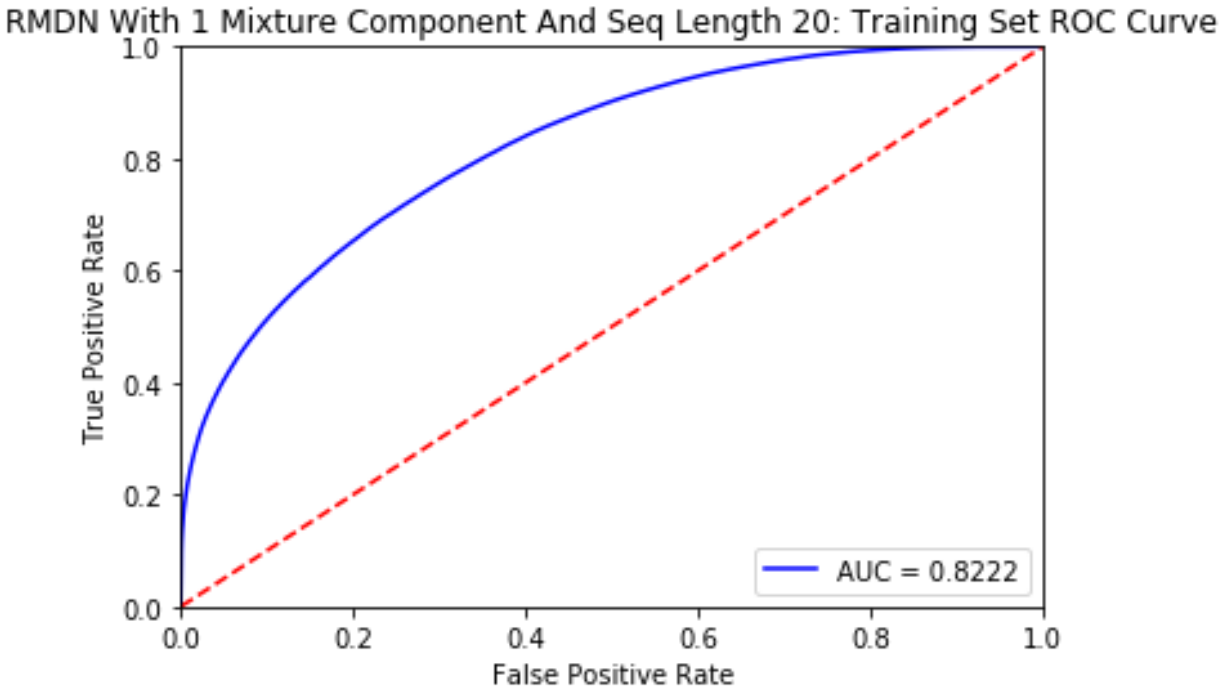


Figure 6.5

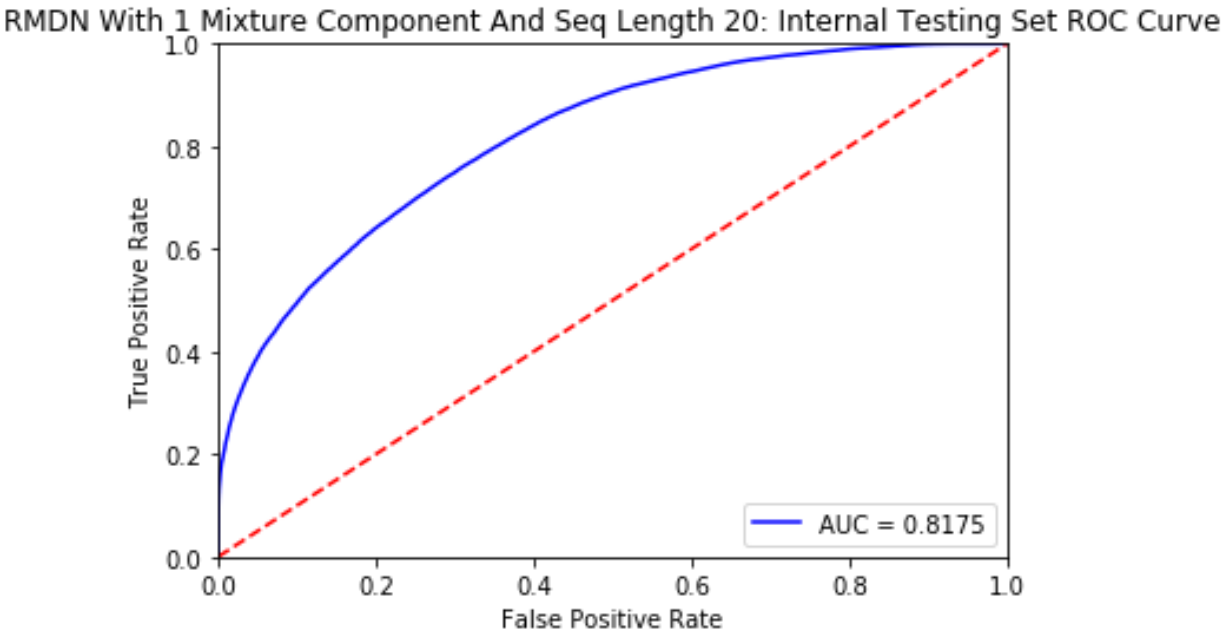
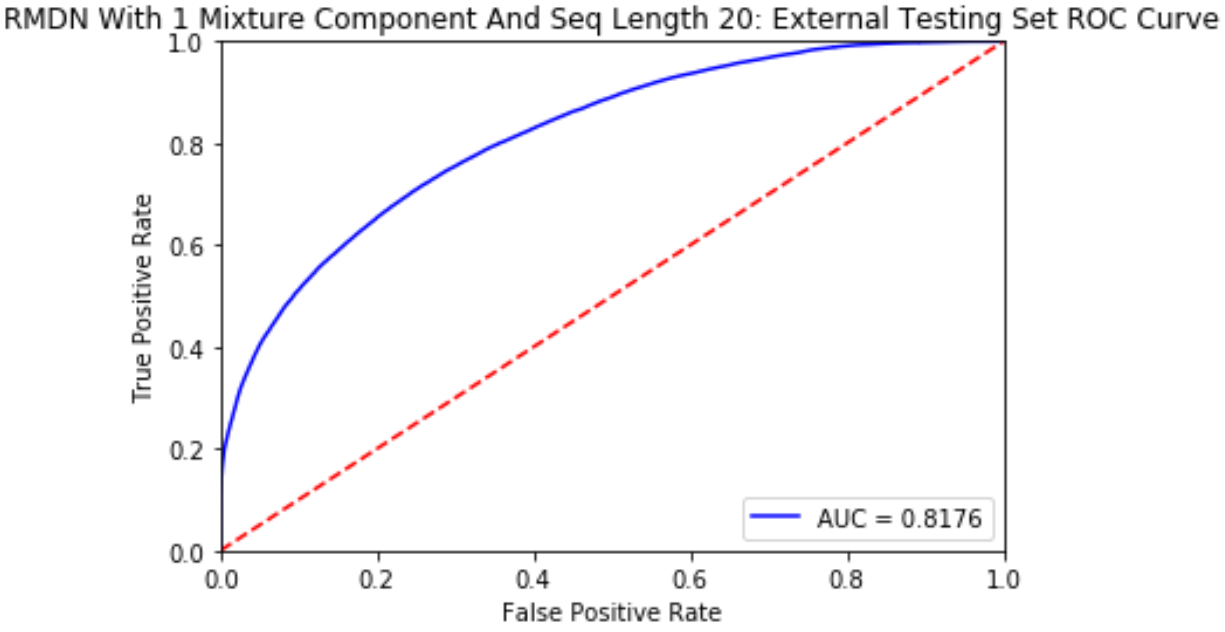


Figure 6.6



In contrast to the preceding accuracy results, the area under the curve metric calculated on the external testing set is not the highest of all previously fitted models. This may be a result

of an unusual proportion of home teams winning in the randomly sampled set of games comprising the external testing set.

Overall, the recurrent mixture density network has the highest measured accuracy and performs similarly in other evaluation metrics to the previously fitted non-recurrent mixture density network and binary classification networks.

7. Conclusion

In sum, the preceding fitted models perform relatively similarly in predicting NBA win probability when given a common set of play-by-play data inputs. That being said, neural networks achieve slightly better predictive results in comparison to the baseline logistic regression and random forest models. Furthermore, as Ganguly and Frank suggest in their research that motivates this paper, recurrent model architectures produce, at the least, equivalent predictive performance to non-recurrent model architectures for modeling win probability. Perhaps most notably, the recurrent mixture density network produces the highest measured predictive accuracy amongst all other models fitted in this project.

There is plenty of opportunity for further work in modeling NBA win probability. One such opportunity is to fit a recurrent mixture density network model with the full play-by-play dataset and compare its predictive accuracy with Ganguly and Frank's mixture density network. In addition to experimenting with a greater number of inputs, there is also an opportunity to fit mixture density networks that utilize neural networks with a greater number of hidden layers to output the mixture model parameter layer. Another interesting possible direction for further work is to utilize the NBA's spatial-temporal player tracking data, or other advanced basketball metrics, as inputs to a mixture density network. Based this project as well as other previous research, win probability models tend to perform better when used in conjunction with the most sophisticated input data, so I would expect the level of detail found in the player tracking data to produce significant performance gains if modeled appropriately.

All model fitting and evaluation code for models fitted in this paper can be found on Github [8]. Data preprocessing code is excluded due to the proprietary nature of the data.

REFERENCES

[1] Sujoy Ganguly and Nathan Frank, “The Problem With Win Probability.”, *2018 MIT Sloan Sports Analytics Conference*, 2018.

[2] Michael Beuoy, “Updated NBA Win Probability Calculator.”, https://www.inpredictable.com/2015/02/updated-nba-win-probability-calculator.html

[3] Christopher M. Bishop, “Mixture Density Networks.”, *Neural Computing Research Group*, 1994.

[4] Hari Bandi, Dimitris Bertsimas, and Rahul Mazumder, “Learning a Mixture of Gaussians via Mixed Integer Optimization”, *Operations Research Center, Massachusetts Institute of Technology,* 2018.

[5] Charles Martin, “Keras Mixture Density Network Layer.”, https://github.com/cpmpercussion/keras-mdn-layer

[6] Francois Chollet, “Deep Learning with R.”, *Manning Publications Co.,* 2018

[7] Data Source: STATS, copyright 2020

[8] Code: https://github.com/henrypoole/nba-win-prob