Project Report: Team A$^+$
Live Football Bet Pricing

Anjul Kumar Tyagi (111482304), Yu-Jung Ko (111563423), Eugenia Soroka (111494044)

{aktyagi,yujko,ysoroka}@cs.stonybrook.edu
Department of Computer Science, Stony Brook University,
Stony Brook, NY 11794-4400

1 Introduction

Our challenge is predicting the probability of bet-clearance with a given point spread for the favorite team during a live football match. The source code of the project is available at `https://github.com/tyagi-iiitv/NFLLive`. The demo is on `https://nfllivedemo.herokuapp.com`

2 Background

The National Football League (NFL) is a professional American football league consisting of 32 teams, divided equally between the National Football Conference (NFC) and the American Football Conference (AFC). Every NFL team is based in the contiguous United States.
The NFL season format consists of a four-week preseason, a seventeen-week regular season (each team plays 16 games), and a twelve-team single-elimination playoff culminating in the Super Bowl, the league's championship game.
Game play in American football consists of a series of downs, individual plays of short duration, outside of which the ball is dead or not in play. These can be plays from scrimmage – passes, runs, punts, or field goal attempts (from either a place kick or a drop kick) – or free kicks such as kickoffs and fair catch kicks.
In this project, we are interested in sports betting, i.e. predicting sports results and placing a wager on the outcome. In particular, for NFL we consider spread betting (where wagers that are made against the spread).

3 Literature Review

Apart from sources mentioned in our Progress Report, we have studied additional academic literature on predicting NFL betting, with the objective of broadening our knowledge of the problem and approaches utilised earlier for solving it.
In particular, in [1], K. Gimpel explores the problem of using machine learning to automatically choose which team to bet on in an NFL football game. Neither Pearson correlation coefficient approach, nor measuring the performance of each feature individually and using the resulting accuracies to guide the grouping together of features was alone sufficient for feature selection and no simple feature combination yielded significantly-higher results, which is why he performed randomized feature search. The logistic regression classifier K. Gimpel used to evaluate his feature set on new data, performed similarly on the test data as it did in cross-validation on the training data, suggesting that the feature set they obtained as a result of the FEATURESEARCH algorithm is an effective feature set for this problem.

In [2], authors present three tests of efficiency of the NFL betting market for the years 1994-2000:
(1) Is the betting market's point spread an unbiased predictor of the actual point difference?
(2) Are the fundamentals that determine the outcome of games fully incorporated into the betting line?
(3) Are there betting strategies that can beat the market, i.e. are profitable?

The betting spread on NFL games is intended to balance the size of the bets placed on the home and visiting teams. Even though it is not intended to be an unbiased or efficient estimate of the actual outcomes, the results show that the hypothesis of weak form informational efficiency is not rejected. It was also found that, betting on games in which the predicted margin of victory differed substantially from the betting spread was marginally profitable but the proportion of bets won was not statistically significantly different from the break-even level. In short, no information was found beyond the point spread that would explain the outcome of games as measured by the difference in points, although further examination of filter rules seems warranted.

1. Gimpel, K.: Beating the NFL football point spread (2006), `https://pdfs.semanticscholar.org/7ee4/e8c78415b4beb524ae64fa24ded7d22ab441.pdf`.
2. Bryan L. Boulier, H. O. Stekler, and Sarah Amundson. Testing the efficiency of the national football league betting market. Applied Economics, 38(3):279–284, February 2006.

4 Data Sets

We have extended our data set, so now for our model we use the NFL games data from **seasons of 2009-2017**. For all the matches, we have two types of data available: **Point Spread Data and Play by Play data for each game.** The contents of each of these data sets are shows in the tables below.

Match-id	Season	Away Team	Home Team	Week	Home Score
Away Score	Time	Day of the week	Favorite	Underdog	Spread

Table 1: Point Spread data columns. Source [www.footballlocks.com, www.fantasydata.com]

Match-id	Drive	Quarter	Down	Play
Time	Play Time Difference	Side Of Field	Yardline	Yards To Go
Goal To Go	First Down	Possessing Team	Defensive Team	Description
Yards Gained	Punt Result	Play Type	Passer	Pass Outcome
PassLength	Air Yards	Yards After Catch	Pass Location	Rusher
Run Location	Receiver	BlockingPlayer	Returner	Tackler 1, 2
Field Goal Result	Field Goal Distance	Fumble	Sack	Challenge Replay
Accepted Penalty	PenaltyType	Penalized Team	Penalized Player	Penalty Yards
Possession Team Score	Defense Team Score	Score Difference	Home Team	Away Team
EPA	Home WP (pre, post)	Away WP (pre, post)	WPA	Season

Table 2: Play by Play data columns. Source [https://github.com/ryurko/nflscrapR-data]

5 Baseline Model

We use buckets to tabulate possible range of features and assign the spread covering probability. Our current baseline model would provide more informative and reasonable probability prediction than the previous baseline model, where *logistic regression* was used to predict probability of a given play. Instead of predicting the chance of clearing the bets by logit function, we could introduce some intuition, such as probability would rise if the score difference passes 20, by looking into the relation of probability among a sequence of buckets.

In the baseline model, we use the features *Time Left (Secs) and Score Difference.* We consider 9 classes for score difference from -35 to 35 with step 10 and 61 classes for time left, where a single class covers a minute. Details of creating the buckets are explained in the advanced model section. The formation of the baseline model follows the steps: 1) Initially add **1** covered(the bet was covered for this play) play and **1** uncovered play into each buckets (add 1 discount approach), 2) put the plays from history data into defined buckets, and 3) design smoothing function to deal with the buckets without real data in them. More details on the implementation of bucketing and probability calculation for each bucket are given in the advanced model section.

In this model, we first adapt **linear regression fitting** as the smoothing function with different score difference across time left. While fitting, the buckets with 0.5 probabilities (which are considered as the ones without data) are filtered out. To ensure that the probability does not leave the range $[0, 1]$ and remains within reasonable scope, we apply the algorithm shows in Algorithm 1.

Algorithm 1 Adjustment for baseline smoothing function

```
1: if score diff $< 0$ and prob $> 0.5$ then prob $= 0.5$
2: else if score diff $< 0$ and prob $< 0$ then prob $= 0$
3: else if score diff $> 0$ and prob $< 0.5$ then prob $= 0.5$
4: else if score diff $> 0$ and prob $> 1$ then prob $= 1$
5: end if
```

The process of creating the smoothing function is shown in Fig. 1.

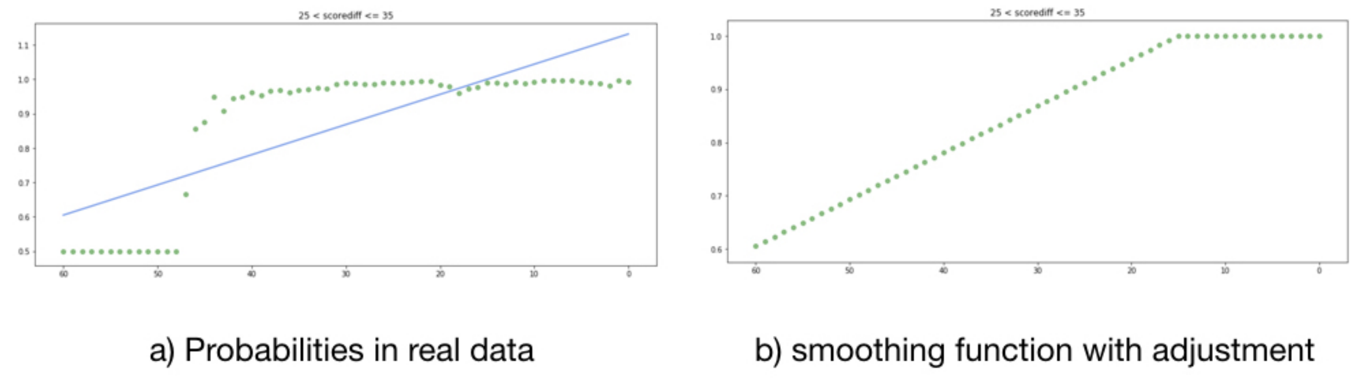


Fig. 1: Process of creating smoothing function with 25 < score difference <= 35

The result comparison of previous Logistic regression baseline model and current bucketing baseline model is shown in Fig 2. The result of bucket model makes more accurate predictions towards the score changes in a game. Moreover, as the score difference varies through the game, it would be more reasonable to predict that their probability of covering the spread would fluctuate in the middle of the game. The evaluation results is shown as Fig. 3. From the evaluation, the current buckets model covers the range of probability within [0, 0.2] and [0.8, 1]. However, due to the fitting process, probabilities around 0.5 are not covered by baseline bucketing model. Based on this result, we state our observation and improve our baseline model.

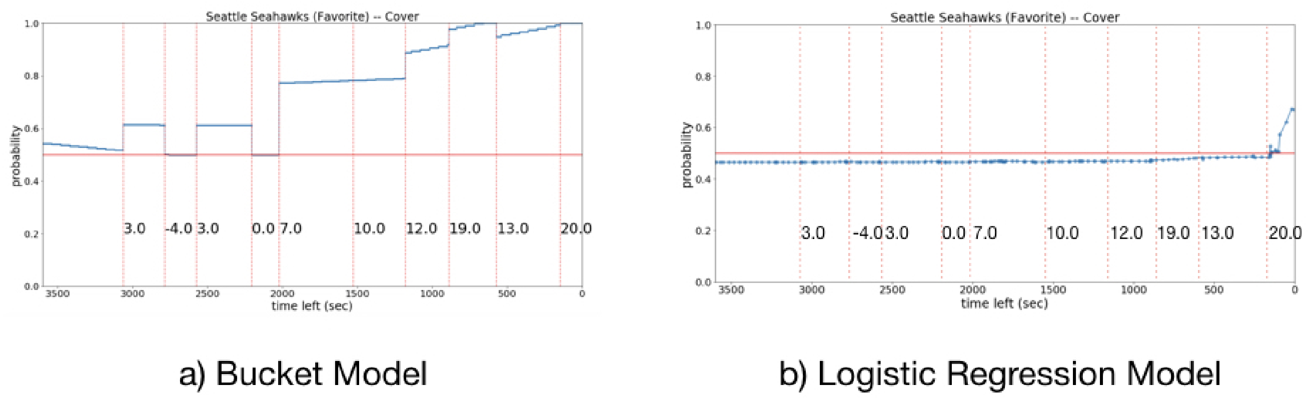


Fig. 2: Predicting probability of Green Bay Packers @ Seattle Seahawks in 2014 week 1

6 Observations

According to Fig. 3, we observe that the probabilities around 0.5 are not covered by our baseline bucket model. This might be caused by the features we select (score difference and time left only) and our smoothing function design. From Fig. 4 (b), we notice that there is a probability gap when the Favorite starts to take a lead (from 0.625 not 0.5) which can be explained by by the fact that we filtered out the buckets without real data i.e. buckets with probability 0.5, while fitting the model. Also, the baseline bucket model predicts monotonically decreasing probabilities towards the end of the game even when the score difference fluctuates within the range [-5, 5]. This phenomenon is against our intuition that the probabilities should vary around 0.5.

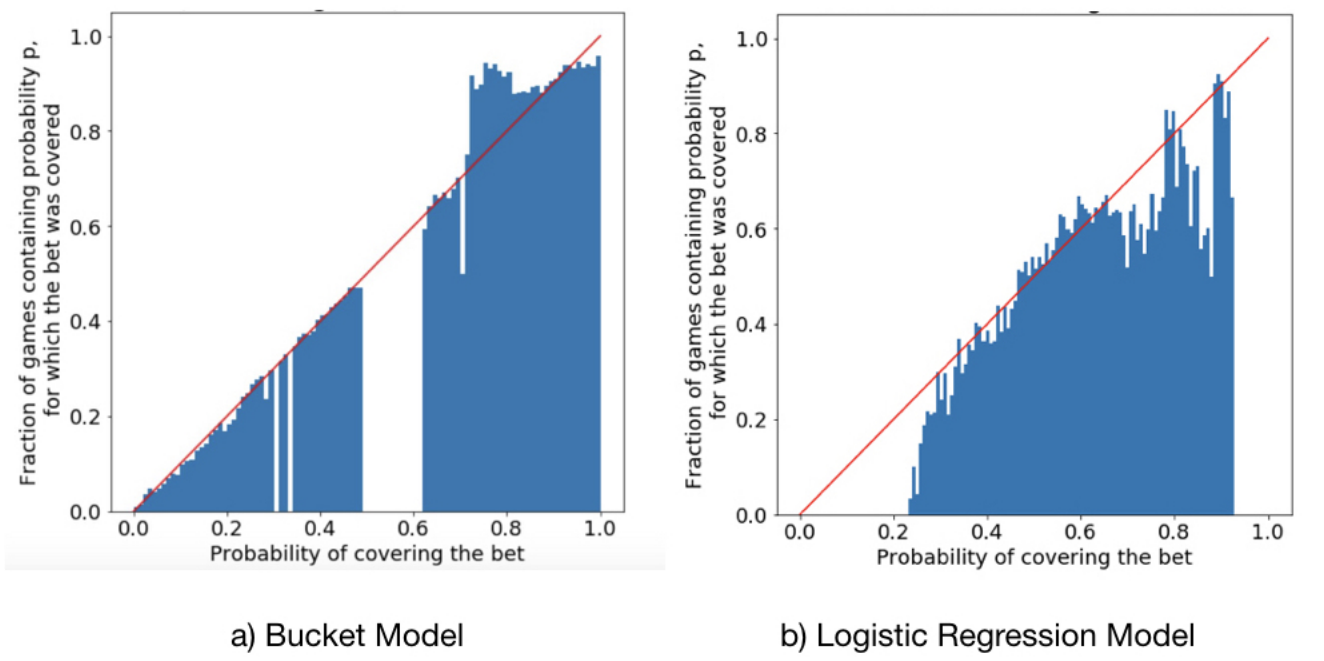


Fig. 3: Evaluation of matches from 2009 to 2017

To solve this problem, we have the following consideration for the advanced model.

– Include more features

- Vary the range of the score difference buckets, and separate the bins [-5, 5] to [-5, 0] and [0, 5].
- Create a smoothing function that can work with more than 2 features.

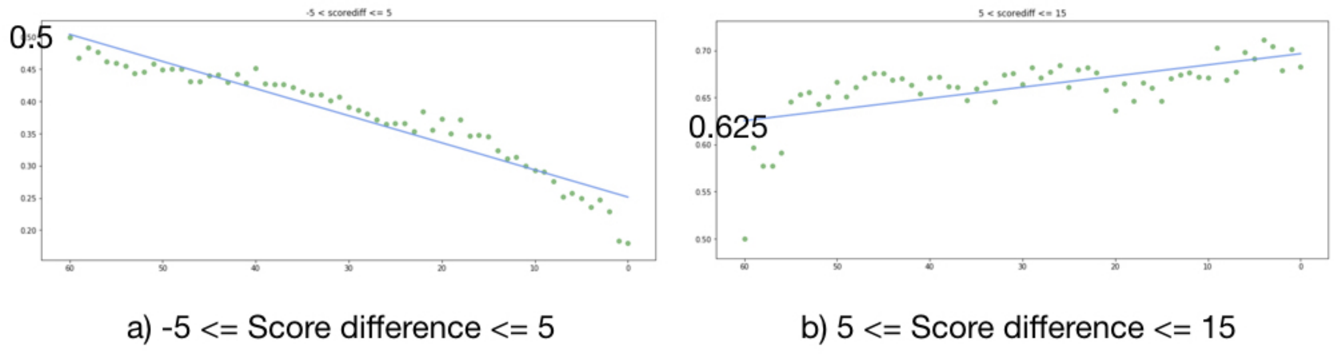


Fig. 4: Evaluation of matches from 2009 to 2017

7 Advanced Model

7.1 Brief description

One of the main ideas of the advanced model is to extend the relationship of each feature with the final probability to n dimensions. For example, features that are covered in the advanced model are Score Difference, Time Left, Downs and Yard Line, and we know the relationship of Time Left with the probability of clearing the bet irrespective of other features. Given these individual relationships of features, the advanced model tries to extend this to all selected features altogether. These individual relationships when extended to four dimensions, give us a smoothing function and prevent abrupt changes from one bucket to another. Several steps involved in this algorithm are discussed in detail below.

7.2 Creating buckets of features

The first step of this model involves assigning each play a bucket based on its values of four features: [Score Difference, Down, Time Left and Yard Line]. To do this, first the range of all of the four variables is calculated for the real data:

Feature Name	**Max Value**	**Min Value**
Time Left	3600	0
Yard Line	100	0
Score Difference	59	-45
Down	4	1

Table 3: Min and Max values of each variable used in the model.

Class Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Range	(-∞,-35]	(-35,-25]	(-25,-20]	(-20,-15]	(-15,-10]	(-10,-5]	(-5,-0]	(0,5]	(5,10]	(10,15]	(15,20]	(20,25]	(25,35]	(35, ∞)

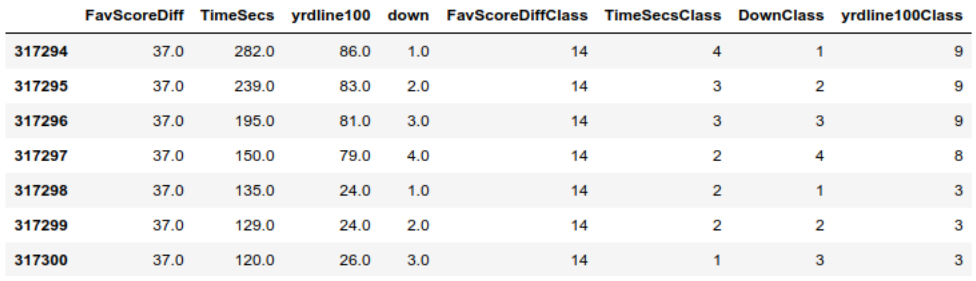
Table 4: Dividing Score Difference into classes and giving labels to classes.

The range of each variable was divided into smaller ranges and each subrange was given a number (the class number for that variable). As an example, the division of classes for the Score Difference is shown in the Table 4.

Notice that the range of score difference isn't constant to 5 points towards the left and right ends of the table because of scarcity of data. Very few data points were available for score difference of values greater than 25 or less than -25. Hence, the interval is increased to accumulate more data points into this class for ease of analysis.

Analogously, each variable was divided into several classes. Except for the Score Difference, classes for all other variables were of similar step size. **Downs were divided into 4 classes** based on the down value of [1,4]. **Time Left was divided into 60 classes** with each class containing a time interval of 60 seconds. **Yard Line was divided into 10 classes**, each class containing 10 yards covering from 0 to 100. This makes the total number of buckets equal to **14*60*10*4 = 33600.**

After giving labels (assigning numbers) to classes of features, each play in the real data was assigned its class number for all the features, defining the bucket in which this play belongs to. (Fig. 5) shows how each play is assigned a class based on its values of the features.



	FavScoreDiff	TimeSecs	yrdline100	down	FavScoreDiffClass	TimeSecsClass	DownClass	yrdline100Class
317294	37.0	282.0	86.0	1.0	14	4	1	9
317295	37.0	239.0	83.0	2.0	14	3	2	9
317296	37.0	195.0	81.0	3.0	14	3	3	9
317297	37.0	150.0	79.0	4.0	14	2	4	8
317298	37.0	135.0	24.0	1.0	14	2	1	3
317299	37.0	129.0	24.0	2.0	14	2	2	3
317300	37.0	120.0	26.0	3.0	14	1	3	3

Fig. 5: Obtaining classes numbers for each play

The main task of this step is to calculate the probability of clearing the bet for each bucket from the real data. Since a bucket is now represented by the class numbers of each feature, we can find the plays with the given class labels for each bucket in the real data, and see how many of these plays were a part of a match in which the bet was cleared. Using this technique, the probabilities of clearing the bet for each of the bucket was calculated.

As it is pretty clear, many of the buckets were empty i.e. **there were no plays that have the labels of that bucket.** To tackle this situation, we used the discounting approach: each bucket was given 2 plays (one clearing the bet and one not clearing the bet), s.t. each bucket had the initial probability of 0.5 before the analysis.

After each bucket had it's probability, each play in the original data was matched to its bucket and assigned the probability of the bucket, which we will call $\mathbf{P}_{org}$.

7.3 Analysis of individual features with probability

Now that each play is assigned its bucket and an initial probability P_{org}, we use this to analyze how each feature affects P_{org}. For instance, probability distribution for each Yard Line Class was analyzed fixing the other features to a fix value and the corresponding box plot (Fig. 6) was obtained. Similarly, the Time Left was analyzed for its probability distribution as shows in (Fig. 7) and (Fig. 8).

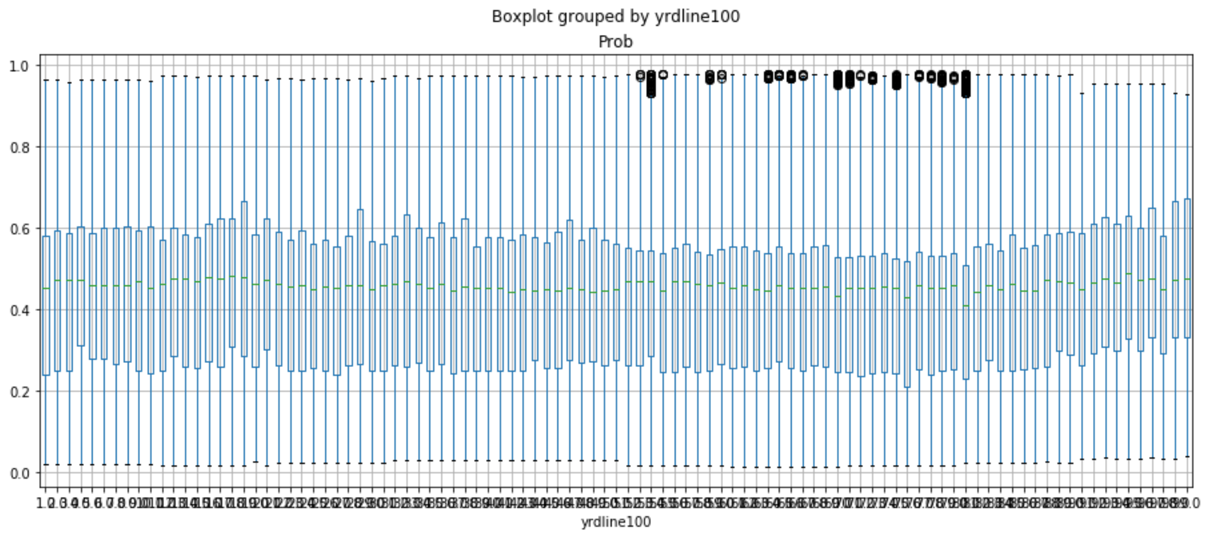


Fig. 6: Distribution of probability for each yard line class

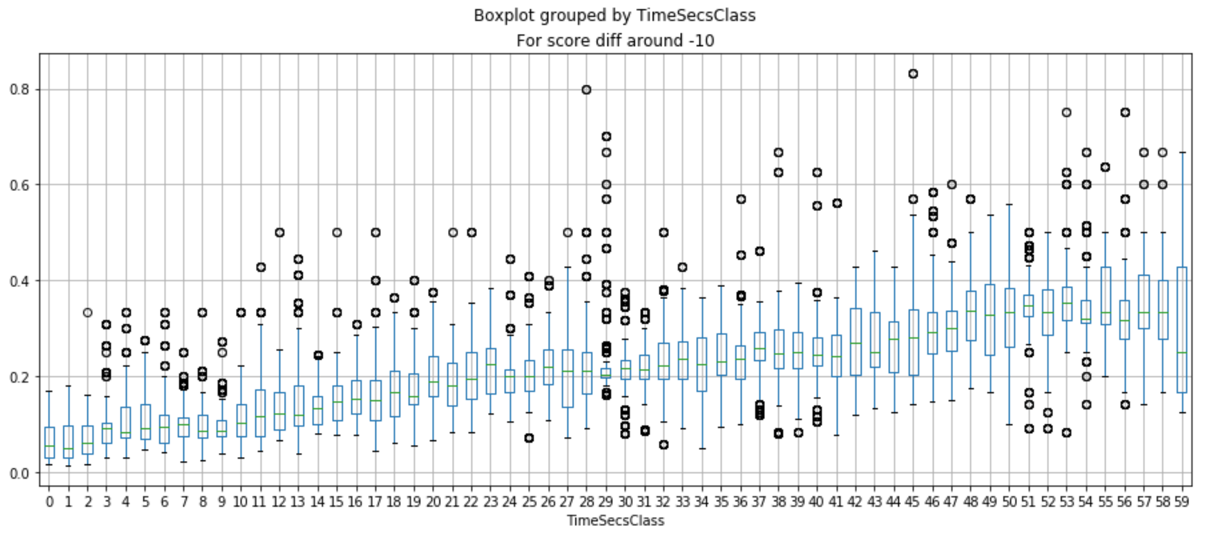


Fig. 7: Distribution of probability for each Time Left class

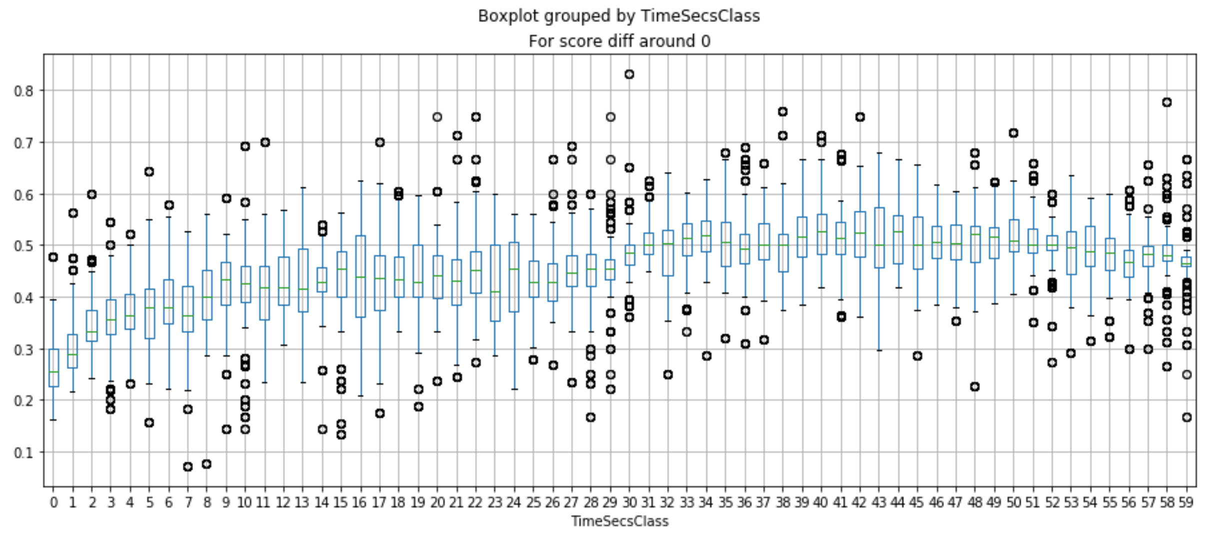


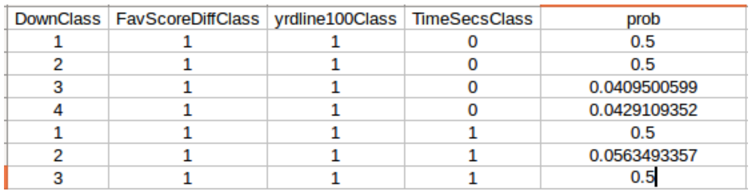
Fig. 8: Distribution of probability for each Time Left class

A bunch of these plots gave us some idea about how each feature was affecting the actual probability of that bucket. Using this idea, our advanced model tried to find this relationship using linear regression and extend this to all 4 features.

7.4 Building up the smoothing function

This is the most complicated part of the advanced model. Presently, each bucket is assigned a probability P_{org}, which was calculated from the real data. Since many buckets were empty (as no real play data fell in that bucket), they were assigned a $P_{org} = 0.5$, which resulted in the variation in the buckets being not smooth. To solve this problem, we need a smoothing function so that the transition of one bucket to other closely related bucket is smooth.

Data for smoothing function. The data used for smoothing function is shown in Fig. 9. These are the P_{org} for each bucket with many values being 0.5. With this data, Algorithm 2 was used to generate four new probabilities:



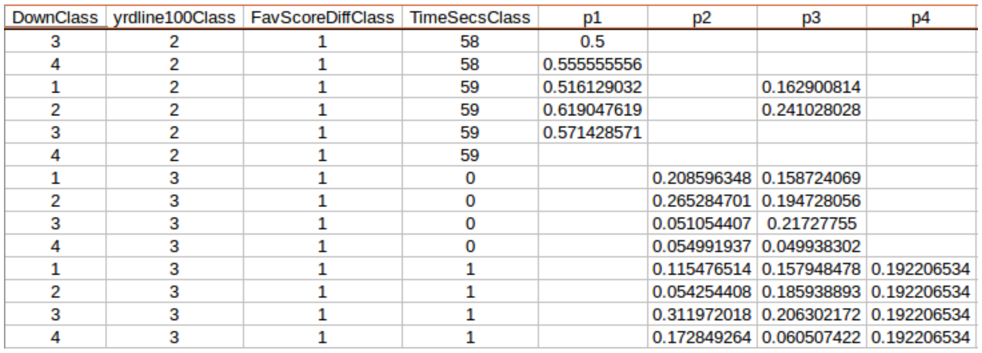
DownClass	FavScoreDiffClass	yrdline100Class	TimeSecsClass	prob
1	1	1	0	0.5
2	1	1	0	0.5
3	1	1	0	0.0409500599
4	1	1	0	0.0429109352
1	1	1	1	0.5
2	1	1	1	0.0563493357
3	1	1	1	0.5

Fig. 9: Data fed into the smoothing function

Algorithm 2 Finding Individual probabilities from P_{org}

```
1: for feature in {DownClass, FavScoreDiffClass, yrdline100Class, TimeSecsClass} do
2:     for bucket in {bucketData} do
3:         $reg$ = linear-regression.train($real - data[feature, prob]$) by fixing other features to value $bucket[other - features]$
4:         $Prob[Feature] = reg.predict(row[feature])$
5:         $Prob - Feature[bucket] = Prob[Feature]$
```

The Algorithm calculates the probabilities for each feature in each bucket by fixing the values of other features according the values in that bucket. These values are smooth, because all of these values are obtained from the Linear regression models. Now the data looks like in Fig. 10. Here p1, p2, p3, p4 are probabilities from DownClass, Yard Line Class, Score Difference Class and Time Left Class respectively for each bucket. Some buckets are empty because no data existed for those values.



DownClass	yrdline100Class	FavScoreDiffClass	TimeSecsClass	p1	p2	p3	p4
3	2	1	58	0.5			
4	2	1	58	0.555555556			
1	2	1	59	0.516129032		0.162900814	
2	2	1	59	0.619047619		0.241028028	
3	2	1	59	0.571428571			
4	2	1	59				
1	3	1	0		0.208596348	0.158724069	
2	3	1	0		0.265284701	0.194728056	
3	3	1	0		0.051054407	0.21727755	
4	3	1	0		0.054991937	0.049938302	
1	3	1	1		0.115476514	0.157948478	0.192206534
2	3	1	1		0.054254408	0.185938893	0.192206534
3	3	1	1		0.311972018	0.206302172	0.192206534
4	3	1	1		0.172849264	0.060507422	0.192206534

Fig. 10: Probabilities from individual features.

Training the feature probabilities to predict $\mathbf{P}_{org}$. Now that the probabilities of each feature are available, we have the individual contributions from each feature towards P_{org}. The next task is to combine these probabilities to give a new smooth probability for each bucket. To get the new final probability which we call P_{model}, we train the four feature probabilities to predict P_{org} (which is the only representation of real data we have). So we build up a regression model with X = feature probabilities and Y = P_{org} and use this to predict P_{model} for all the buckets. Now P_{model} become our final probabilities for each bucket. P_{model} is smooth as we will see in the Evaluation section and gives a decent intermediate probability for buckets with no data.

7.5 Evaluation

All of the above implementation were done separately for the defense and attack data, i.e. data was separated based on plays where the Favorite team was attacking or vice verse. The evaluation results are given in Fig. 11.

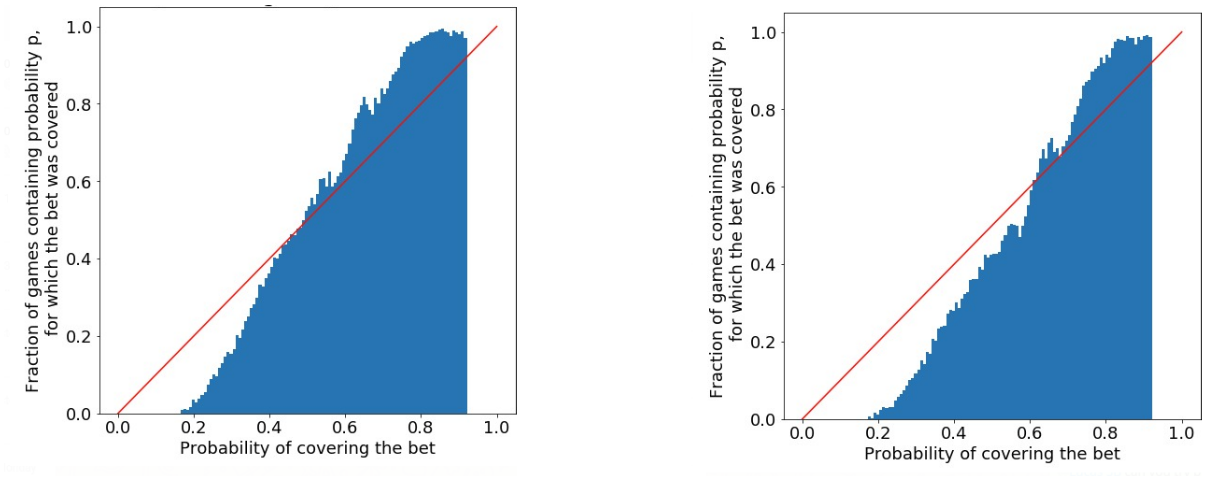


Fig. 11: Evaluation of Attack (left) and Defense (right).

The evaluation is almost linear and is close to $x = y$ line. However, there is very little or **no prediction with probabilities in range of 0 to 0.2 and from 0.9 to 1.** This could be explained by the fact that we blindly gave every play a probability of 0.5 without analyzing the type of bucket it was in. For example, giving 0.5 probability to a bucket with Score Difference of -35 clearly isn't a good idea. They should be given a smaller probability and, similarly, the buckets with higher score difference should be given a higher probability. So, to improve the model, **the buckets with score difference less than -25 were given random probabilities in the range of 0 to 0.2 and similarly, the buckets with score difference greater than 25 were given random probabilities in the range 0.9 to 1.** After this change, the whole model was rerun and we obtained the evaluation results as shown in Fig. 12. Clearly, the new evaluation model is covering more probabilities than the previous model.

In addition, the example match (2014 week 1 Packers @ Seahawks) comparison between our final model and baseline one is shown in Fig. 13. Notice that the result of our final model depict the variance within the games. Besides this, the situation that before the 2nd quarter of the game, Seahawks are trailing on the score difference and then later, they comeback in the game. Our final model does demonstrate this situation.

8 Conclusions

In the course of the project, we have tackled the problem of predicting bet pricing for NFL live data.

The baseline model has been developed and implemented, providing preliminary probabilities of covering the bet based on the play by play data. The bucketing approach and the Logistic regression approach for the baseline model, both have their own advantages and disadvantages which are taken into consideration while building the advanced model.

The advanced model for producing probabilities has been developed, which uses real data to assign initial probabilities by counting the plays in buckets, and incorporates linear regression to build smoothing functions for probability for separate features while other selected features are fixed and find out updated probabilities. The obtained probabilities are then fit to the real data, giving the final probabilities of covering the bet. As evaluation has shown, additional work on the data was required to increase the efficiency, e.g. assigning reasonable probabilities to empty buckets at the margins of score differences.

Now that we have built and trained the advanced model, it can be used for predicting probabilities of covering the bet for live matches, by plugging data from the live stream into the model.

It is worth mentioning several challenges that occured throughout the project:

- We tried training our data with K-Means and tried to assign each cluster the mean probability of that cluster. But in this case, to get smooth probability transitions from one cluster to another, a lot of clusters were required. So it wasn't feasible.
- We also tried simply training the given data to predict P$_{org}$ but the results were bad. The probabilities were predicted with values beyond the range of 0 and 1.
- Dealing with missing buckets data was a problem and we had to come out with a good solution to this problem so that the evaluation model isn't affected by this random data.

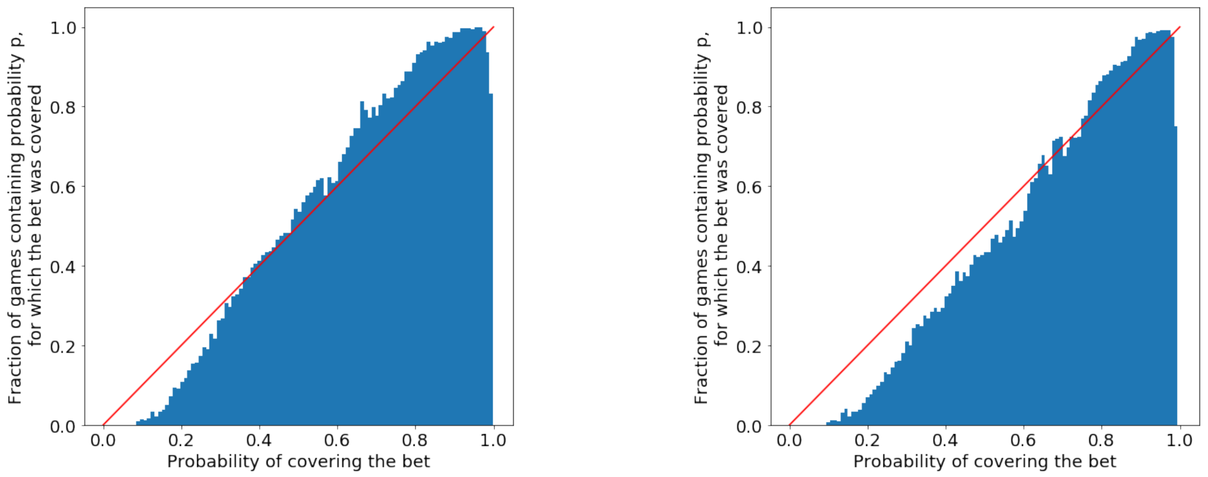


Fig. 12: New evaluation of Attack (left) and Defense (right).

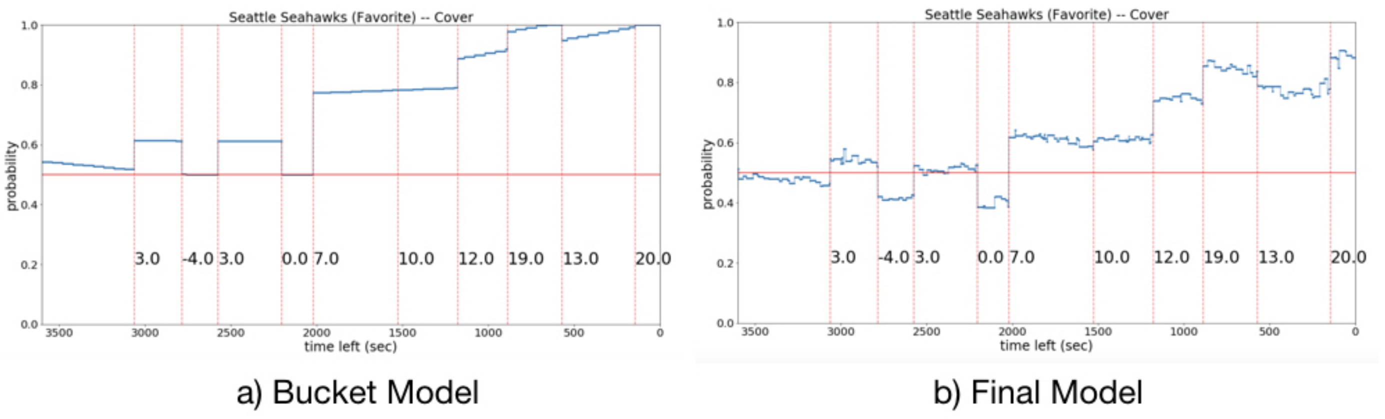


Fig. 13: Predicting probability of Green Bay Packers @ Seattle Seahawks in 2014 week 1.

Addition:

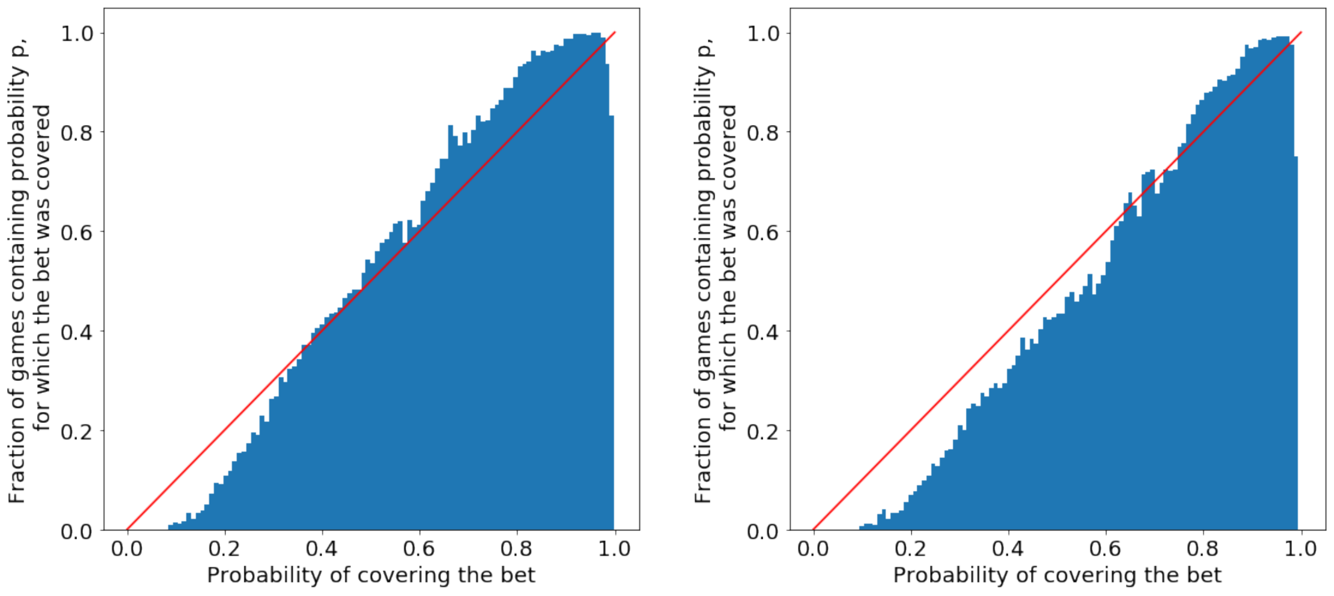


Fig. 12: New evaluation of Attack (left) and Defense (right).

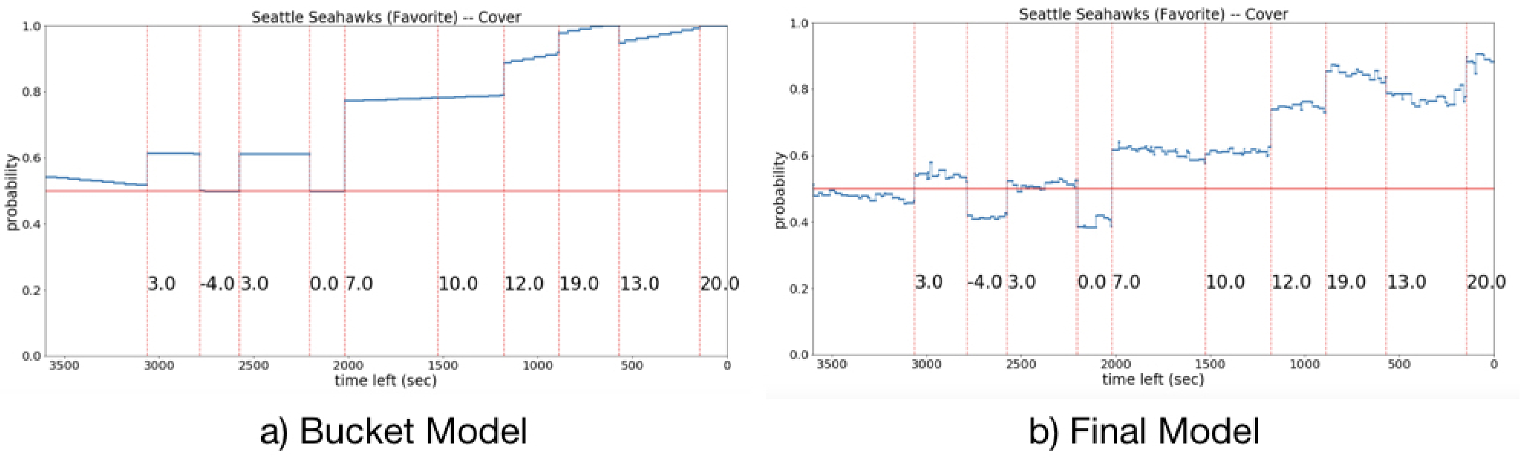


Fig. 13: Predicting probability of Green Bay Packers @ Seattle Seahawks in 2014 week 1.