American Football Game Outcome Prediction with Neural Networks

David García

BU MET College

Department of
Computer Science

Professor Alexander
Belyaev

Fall, 2024

Abstract

The objective of this research is to predict the outcome of American football games using deep learning techniques. The dataset used included detailed information about football teams and game betting data, which was analyzed thoroughly and later preprocessed in order to be input to the neural networks. Seven experiments were completed during the training process, which include one logistic regression model and six neural networks, which were conducted by adjusting the parameters to improve the model's accuracy. The best-performing model achieved a 61.63% accuracy on the validation set and 59% on the test set. Despite the relatively small size of the dataset, the model demonstrated the ability to effectively learn and achieve competitive accuracy, highlighting the potential of deep learning for this type of predictive task.

INTRODUCTION 3

DATA CURATION, PREPROCESSING & FEATURE ENGINEERIN 3

a) Dropping unnecessary columns 4

b) Renaming columns 4

c) Treatment of missing values 4

d) Change datatypes 4

e) Creation of columns 4

f) Remove result columns 5

g) Target Encoder 5

h) One Hot Encoding 5

i) Perform a Train-Test Split 5

j) Standardization 5

TRAINING PROCESS 5

a) Feature Selection 7

Variable Dictionary 7

b) L1 and L2 Regularization 8

c) Dropout Layers 8

d) Batch Normalization 8

e) SMOTE 9

RESULTS 9

BIBLIOGRAPHY 12

Introduction

Advanced analytics and artificial intelligence have taken the world by storm, and the sport industry has not been the exception. American football is, alongside baseball, the sport that uses analytics the most. Teams have incorporated them not only into their business strategies, but into their sports strategy as well. NFL teams use advanced analytics for decision-making during games, tracking players' health, and signing the best players available in the yearly draft or in free agency. Sports betting has also become extremely popular in recent years, as sports leagues have changed their stance on the subject. The American Gaming Association estimated that $35 billion would be bet throughout the 2024 NFL season, a notable increase from 2023, with that number being close to $27 billion. More and more data about players and games has become available with the rise of technologies in sport, which allows for more exploration in the subject.

I intend to use neural networks for the prediction of outcomes of football games. The objective is to predict whether the home team will cover the point spread or not, which is a handicap placed by sportsbooks to balance the odds between two teams. The dataset that was used to train the neural network is comprised of betting data and information about the teams playing, like the home team spread, final score, week and season played, venue, weather, and starting quarterbacks and the coaches for each team, among others. It is a merely small dataset, which contains a little more than 6,000 observations, which is not ideal when training a complex model like a neural net. In this paper we will dive in to the process of the preprocessing of the dataset and training of the model, as well as the reasoning behind each decision that was made during the process, and the results and conclusions obtained from it.

Data Curation, Preprocessing & Feature Engineerin

An investigation was done in order to obtain a proper dataset. A Python library was discovered, named *nfl-data*-py, which provides game and seasonal data, as well as depth charts, injuries, players statistics, college prospects, among others. For the purpose of this project only weekly game data will be used. The dataset contains 6,706 observations and 46 columns. A sample of a portion of the

dataset is shown below in Figure 1.

```
# Import data from 1999 (first available) to end of last season
years = range(1999, 2024)
data = nfl.import_schedules(years)
data.tail()
```

	game_id	season	game_type	week	gameday	weekday	gametime	away_team	away_score	home_team	home_score	location	result	total	o
6701	2023_20_TB_DET	2023	DIV	20	2024-01-21	Sunday	15:00	TB	23.0	DET	31.0	Home	8.0	54.0	
6702	2023_20_KC_BUF	2023	DIV	20	2024-01-21	Sunday	18:30	KC	27.0	BUF	24.0	Home	-3.0	51.0	
6703	2023_21_KC_BAL	2023	CON	21	2024-01-28	Sunday	15:00	KC	17.0	BAL	10.0	Home	-7.0	27.0	
6704	2023_21_DET_SF	2023	CON	21	2024-01-28	Sunday	18:30	DET	31.0	SF	34.0	Home	3.0	65.0	
6705	2023_22_SF_KC	2023	SB	22	2024-02-11	Sunday	18:30	SF	22.0	KC	25.0	Neutral	3.0	47.0	

Figure 1: sample of dataset

Data preprocessing was extensive, as it usually is. Below you can find a detailed analysis of all the steps that were taken during this process to ensure data quality throughout the whole training process.

a) Dropping unnecessary columns

The dropped columns are of no utility in the training process. Some of these columns include game odds, identification variables, the referee, the stadium, and the type of surface.

b) Renaming columns

Some columns were renamed for the purpose of simplicity and/or clarification. For example, “home_qb_name” was renamed as “home_qb”, for simplicity, and “result” was renamed as “home_spread_result”, for clarification.

c) Treatment of missing values

Eight variables had missing values, which were arbitrarily treated depending on the nature of each variable. The used techniques include imputing the average for numerical features, imputing the mode for categorical features, and using the most recent value for temporal variables.

d) Changing datatypes

By changing float values to integers and objects to categories we decreased memory by 400 KB, which allows us for higher speed and less memory space.

e) Creation of columns

The “home_cover” column was created by comparing the final score to the home spread, obtaining a binary result. This feature will serve as the target variable that will be used during training.

f) Removal of score columns

The inclusion of the “home_score” and “away_score” columns would result in data leakage, as the model would already know the result of the game. It is important for both of these columns to be removed.

g) Applying Target Encoder

There are almost 200 values for quarterback and head coach. It is a tricky situation, as it is difficult to encode a variable with that many unique values. The implemented solution is Target Encoder, which encodes the value with the mean of the target variable. The result is a unique number for every distinct value, without adding dimensionality to the dataset. The number of head coaches and quarterbacks was reduced significantly by including only those with at least one season, or 16 recorded games.

h) One Hot Encoding

One hot encoding was applied to four categorical columns: “game_type”, “weekday”, “gametime” and “neutral_venue”. This introduces additional dimension to the dataset, and it will be important to assess whether this increased dimensionality is warranted based on the impact of each variable.

i) Performing a Train-Test Split

A train-test split was performed in order to prepare the data for training, allocating 70% of the data for training, 15% for validation, and 15% for testing. The model will use the training set to learn the patterns in the data, while the validation set will be used for experimentation and evaluation. Once training is complete, the final model will be assessed on the unseen test set.

j) Standardization

Finally, data was standardized using *StandardScaler* in order to make all features belong to the same scale, ensuring efficient computation and consistency.

Training Process

A total of 8 models were trained, one logistic regression and 7 neural network models were trained. Each model had a minor modification compared to the previous model, with the purpose of optimizing accuracy and loss. To assess the performance of the models three things were done: a plot comparing the training and validation loss, a confusion matrix to look at the predictions the model

is making, and a classification report used to look at the different metrics, namely accuracy, precision, and recall.

A linear regression analysis was performed to establish a baseline model. Even though it did not achieve good results, it gave an outline of the most important variables. Figure 2 shows a feature importances bar chart, outlining the impact each value has on the regression analysis.

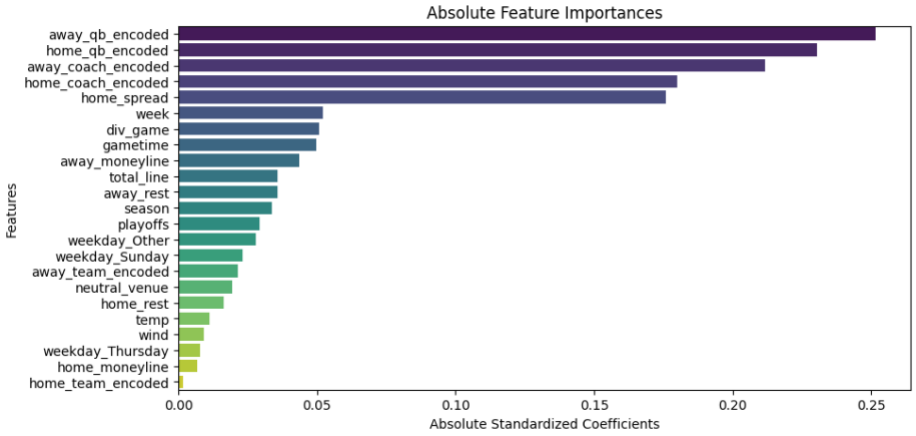

Figure 2: Feature Importances Analysis

The first neural net model did not yield great results, a product of underfitting, likely due to the great amount of features there is. In figures 3 and 4 it is possible to look at the performance of the baseline neural network, which was good but not ideal. In subsequent steps other tests will be done to improve model performance.

Validation Loss:	0.6843
Validation Accuracy:	0.5895
Correct Predictions:	593/1006

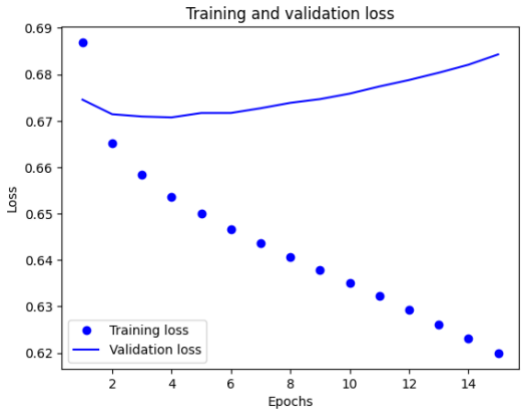

Figure 3 Baseline Neural Network Loss

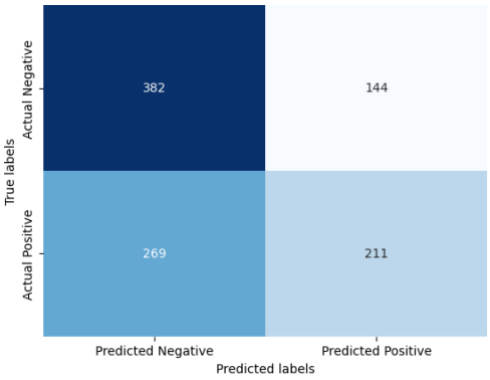

Figure 4: Baseline Neural Network Confusion Matrix

Other steps were completed to improve the model, which are detailed below:

a) Feature Selection

Only nine out of the 24 initial columns were selected after the training of the baseline model. It is possible to find below the variable dictionary. These variables were selected taking into account the feature selection analysis done on the logistic regression model.

Variable Dictionary

Variable	Description
week	The week of the seasonin which the game is being played.
gametime	The time in which the game starts.
home_spread	The spread assigned to the home team.
div_game	Whether it is a divisional game or not.
home_qb_encoded	The encoding of the quarterback of the home team.
away_qb_encoded	The encoding of the quarterback of the away team.
home_coach_encoded	The encoding of the coach of the home team.
away_qb_encoded	The encoding of the coach of the away team.
home_cover_spread	Whether the home team covers the spread or not.

Slightly better results were achieved with the selection of these features, as both validation loss and accuracy improved.

Validation Loss:	0.6660
Validation Accuracy:	0.5964
Correct Predictions:	600/1006

b) L1 and L2 Regularization

In another attempt to decrease overfitting, models with L1 and L2 regularization were tested. These regularizers prevent overfitting by adding a penalty term to the loss function. Lasso, also known as L1 regularization, provided better results, and will eventually be included in future tests.

L1 - Lasso

Validation Loss:	0.6794
Validation Accuracy:	0.6083
Correct Predictions:	612/1006

L2 - Ridge

Validation Loss:	0.6777
Validation Accuracy:	0.5954
Correct Predictions:	599/1006

c) Dropout Layers

Dropout layers remove a specific percentage of the neurons from training in a random manner, which decreases complexity, enhances robustness and improves the model's capacity for generalization. This experiment has obtained the best performance so far, and it is highly probably that it represents the optimal solution.

Validation Loss:	0.6744
Validation Accuracy:	0.6163
Correct Predictions:	620/1006

d) Batch Normalization

Batch normalization layers standardize the inputs before inputting them to the next

layer, increasing the stability and consistency of the network. These were added to the model, but negative results were obtained, leading to their removal in future experiments.

Validation Loss:	0.7604
Validation Accuracy:	0.5984
Correct Predictions:	602/1006

e) SMOTE

The last experiment involves SMOTE, which stands for Synthetic Minority Over-sampling Technique. It is used to handle class imbalance and create artificial observations based on existing data. The data has a slight imbalance towards the positive class, which is slightly higher than 53%. Close to 300 new observations were created. The experiment did not work as expected, and will likewise not be included in the final solution.

Validation Loss:	0.7101
Validation Accuracy:	0.6004
Correct Predictions:	604/1006

Results

The results were promising, as an accuracy of 61.63% in the validation set was achieved, a great result considering the limited size of the dataset. The model with the best performance is experiment III, which includes two Dense layers with 64 and 32 neurons, two Dropout layers with 25% dropout rate between the Dense layers, Lasso regularization with a value of .001 in Dense layers, and the Dense output layer with sigmoid activation. *reLu* activation was used for Dense layers, *Adam* was used as an optimizer, *binary* crossentropy was used as the model loss and *accuracy* as metric. The architecture of the model can be seen in Figure 5 below.

Model: "sequential_8"

Layer (type)	Output Shape	Param #
dense_24 (Dense)	(None, 64)	576
dropout_8 (Dropout)	(None, 64)	0
dense_25 (Dense)	(None, 32)	2,080
dropout_9 (Dropout)	(None, 32)	0
dense_26 (Dense)	(None, 1)	33

Total params: 2,689 (10.50 KB)
Trainable params: 2,689 (10.50 KB)
Non-trainable params: 0 (0.00 B)

Figure 5: Architecture of best model

The model was then evaluated using test data, which is completely unseen by the model, as only the train and validation sets were used during the training process. The analysis of the results of predictions on test for the final model are shown in the figures below.

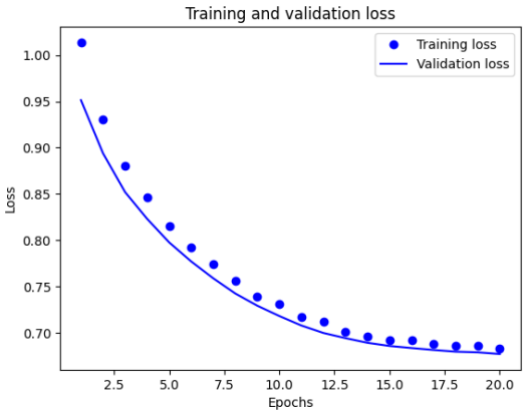

Figure 7: Final Model Validation Loss Graph

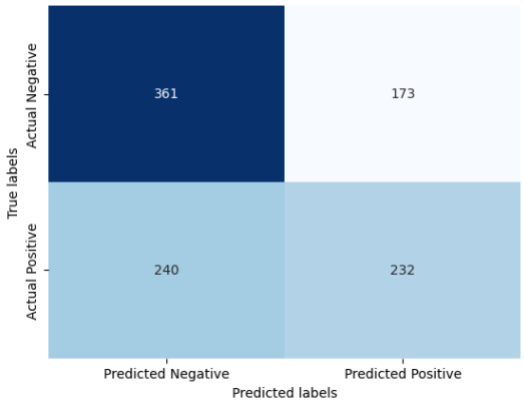

Figure 7: Final Model Confusion Matrix

Classification Report:

	precision	recall	f1-score	support
0	0.60	0.68	0.64	534
1	0.57	0.49	0.53	472
accuracy			0.59	1006
macro avg	0.59	0.58	0.58	1006
weighted avg	0.59	0.59	0.59	1006

Figure 8: Classification Report of final model

The model shows a better capacity for generalization than previous models,

as evidenced by the validation loss curve. The model did not lose much capacity for prediction on the test set, as it can predict results with 59% accuracy on completely unseen data, compared to a 61.63% accuracy on the validation set.

An intriguing observation is the drastic difference in the identification of true values between classes. While 68% of the observations of the negative class were identified correctly, only 49% of the positive class observations were accurately predicted. This indicates that the model consistently commits Type II Errors, leading to a higher number of false negatives.

Additionally, the model predicts the negative class 60% of the time. It is worth noting that both classes are predicted with similar accuracy: the negative class is predicted correctly 60% of the time, while the positive class is predicted correctly 57% of the time.

The model was retrained on all data, in the hopes that the extra 30% of the data will improve the accuracy of the model. Results are promising considering that most experts achieve a record of 55-60% ATS (against the spread) during a season, which means that the predictions of the trained model are on par, or even surpass, those of top experts.

Bibliography

2024 NFL Wagering Estimates - American Gaming Association. (2024, September 3). American Gaming Association. https://www.americangaming.org/resources/2024-nfl-wagering-estimates/

nfl-data-py. (2024, September 20). PyPI. https://pypi.org/project/nfl-data-py/

Pickwatch. (2024). *NFL Pickwatch - Week 14 2024 Against the Spread NFL picks from every media expert*. Nflpickwatch.com. https://nflpickwatch.com/nfl/picks/ats/experts