STAT 488 Project

Zacharias Culpernicus

2025-11-25

Introduction

The prediction challenge this year for the Big Data Bowl is predicting the movement of players while the ball is in the air. The data consists of the variables such as player position, speed, acceleration, orientation, angle of player motion, ball location, and more. Using these variables, a predictive model must be built to predict the (x,y) coordinates of each player specified (labelled "player_to_predict") for every frame until the play ends. There are 10 frames per second, so if the ball is in the air for 1.8 seconds, there will be 18 frames of location for the model to predict. The overall goal is to minimize the root mean squared error between the predicted and the observed target:

$$RMSE = \sqrt{\frac{1}{2N}\sum_{i=1}^{N}((x_{true,i} - x_{pred,i})^2 + (y_{true,i} - y_{pred,i})^2)}$$

To see an example of a play animation click here.

The input phase in the visualization represents the (x,y) coordinates given in the data and the output phase is an example of the data provided to use to train the model.

Methods

Initially, data cleaning is required in order to be able to build any prediction models. I manipulated certain variables to be numeric, like changing height from "6-3" to 75 (inches). After adjusting these variables, I added variables that will be used in the model. Distance to the ball, angle to the ball, and some features to get update the directional orientation of the player are applied to the data set. The overall structure of all of the first models is that I fit the model using the data from the input to predict the change in position over the entire play. These models include GAM, Elastic Net, XGBoost, Random Forest, and a physics-based model that attempts to optimize parameters to reduce the loss of speed while running towards the ball. However, these models fail to consider that the predictions of the next frame depend on the previous frame. Therefore, I fit a transformer model that updates the features that are inputted into the model at each frame. This model predicts using autoregressive techniques and I found that it had a much smaller RMSE with cross-validation. There are many hyperparameters that go into a transformer model, so I created a function that will fit 50 models that the hyperparameters are randomly selected from a set of values previously specified. This function uses a training and validation set and the minimum RMSE value found is selected as the best model. Once the final model is fit, predictions are recorded in a data frame that includes the id variables for the week, game, play, and player, as well as the predicted x and y values.

Results

After running the cross-validation using the 5 folds, the following table shows the mean RMSE over the folds of all of the models:

Model	Mean RMSE
GAM	2.14
XGBoost	1.4
Random Forest	1.42
Elastic Net	2.14
Physics (Deterministic)	2.92
Transformer	1.23

The final model hyperparameters chosen are as follows:

- Number of heads: 4
- Key Dimension: 16
- Feed-forward Dimension: 512
- Sequence length: 5
- Batch size: 32
- Dropout: 0.17
- Learning Rate: 0.000847

Looking at the same play from before, I added the predictions to compare to the actual play. The hollow stars represent the predictions from the final model output for a single play.

Link to animated gif: here

As you can see, the RMSE for this particular play was 1.88 yards. This example is one of the worst predictions for a play. However, the model is still fairly close to the actual play.

Discussion

Overall, transformer model was significantly better than any of the other models that failed to use any updating after each frame. However, given more time, I would like to have trained different types of neural nets, like LSTM, that could have potentially been a better fit to the data. Additionally, I would like to have added some sort of relationship variable between the offense and defense, as the defenders would sometimes follow the player instead of going towards the ball. Another consideration would be to use momentum instead of strictly velocity and acceleration, as that would take into account the weight of the player and how quickly the player could realistically change direction.

Appendix

Link to gitHub: https://github.com/zculp6/2026-Big-Data-Bowl

Big Data Bowl competition: https://www.kaggle.com/competitions/nfl-big-data-bowl-2026-prediction/overview