"Bets On" Predicting NBA Point Differentials

Christian Lee
Chris Ratsimbazafy-Da Silva
Cameron Wright
Dave Zack

2023 April 17

Introduction

The increase in volume of sports data has led to a proliferation of research-driven experiments designed to develop and improve machine learning models and ultimately predict sports outcomes.

Past experiments include:

- Purucker (1996) - predicting National Football League (NFL) results
- Davoodi (2010) - predicting Aqueduct Racetrack horse races
- Tax & Joustra (2015) - predicting Dutch football matches

Problem Statement

Can a model using previous game data accurately predict the point differential and winners of individual NBA matches?

Why is this important?

- The global sports betting market stood at $83B in 2022 and is expected to reach a compound annual growth rate of 10.3% between 2023 and 2030
- Sports betting venues have expansive data science teams to develop algorithms that predict sports outcomes and attribute odds for sports fans to bet against
- Wider public can develop its own models to identify instances of probabilistic mismatching and opportunities for betting arbitrage

Data

Primary Data Source: NBA API

Input Data

Team-Level Game Statistics

Total Points Scored
# of Field Goals
% of Field Goals Made
# of Assists
# of Blocks
# of Rebounds

Advanced Team Statistics

Offensive Ratings
Defensive Ratings
Player Impact Estimates

Target Variable

Point Differentials
(e.g. Home Team Points - Away Team Points)

Phases of Analysis

Data Scraping

README.md

PYPI V1.2 | DOWNLOADS 5.6K/MONTH | BUILD PASSING | LICENSE MIT | SLACK NBA API

nba_api

An API Client Package to Access the APIs of NBA.com

`nba_api` is an API Client for `www.nba.com`. This package intends to make the APIs of NBA.com easily accessible and provide extensive documentation about them.

Getting Started

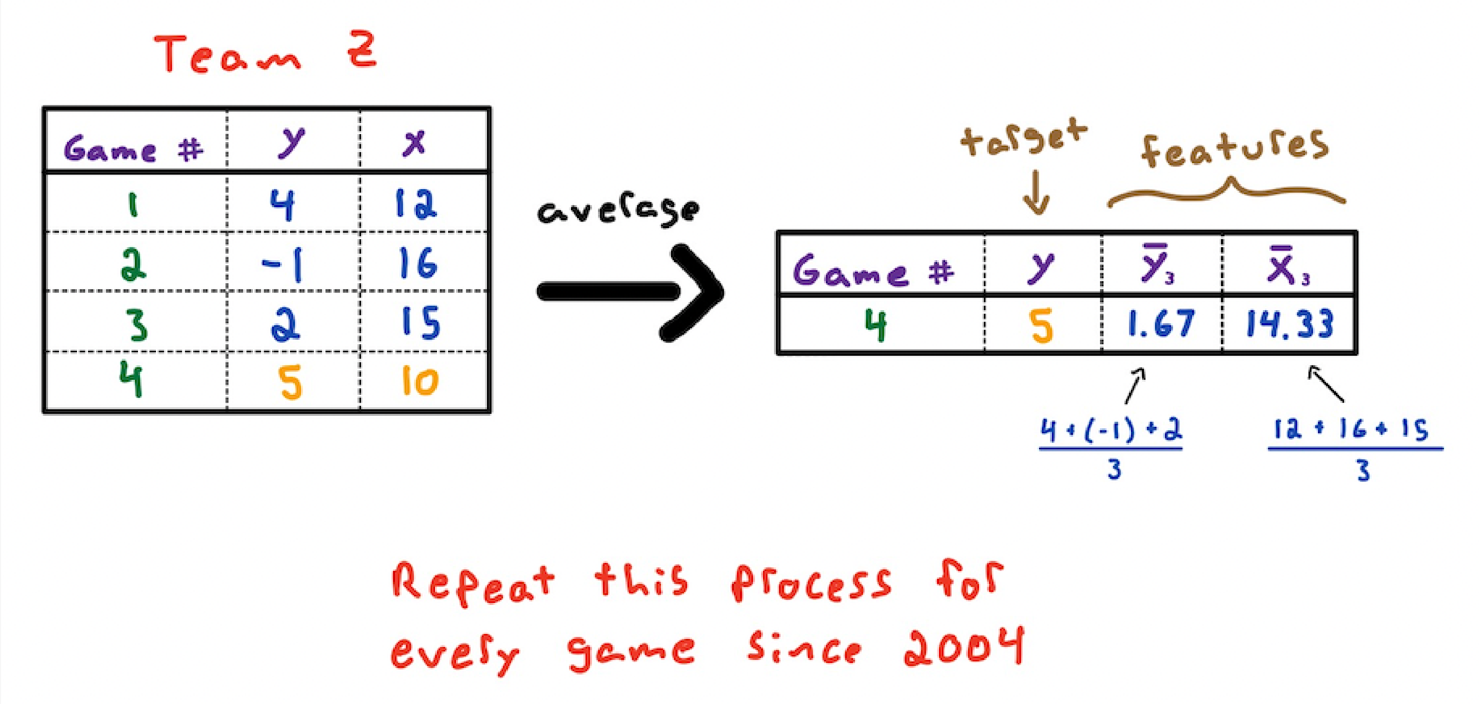
`nba_api` requires Python 3.7+ along with the `requests` and `numpy` packages. While `panadas` is not required, it is required to work with Pandas DataFrames.

```
pip install nba_api
```

All data sourced from NBA API

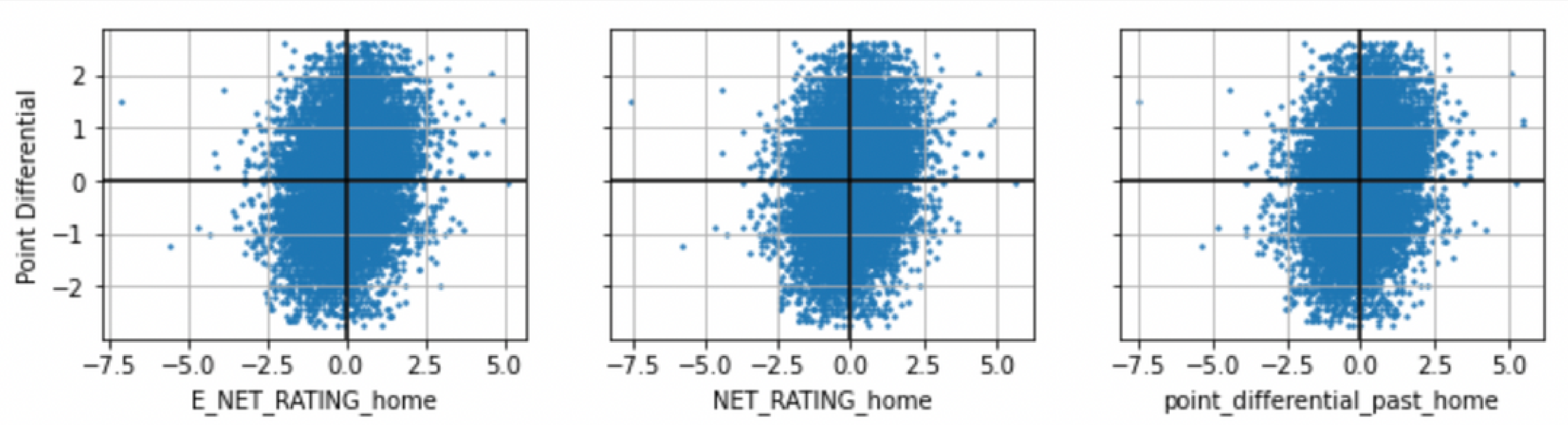
Data Cleansing & Processing

- Create the features by averaging relevant statistics over previous games

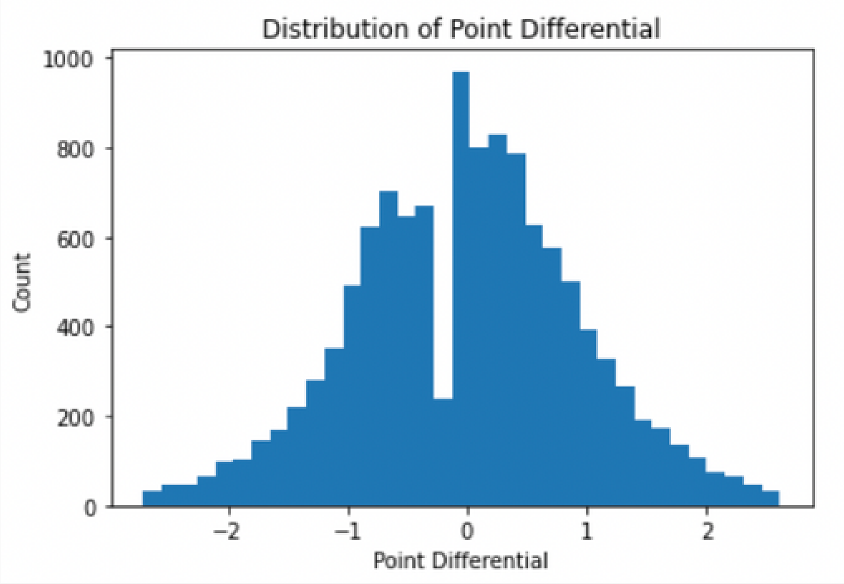


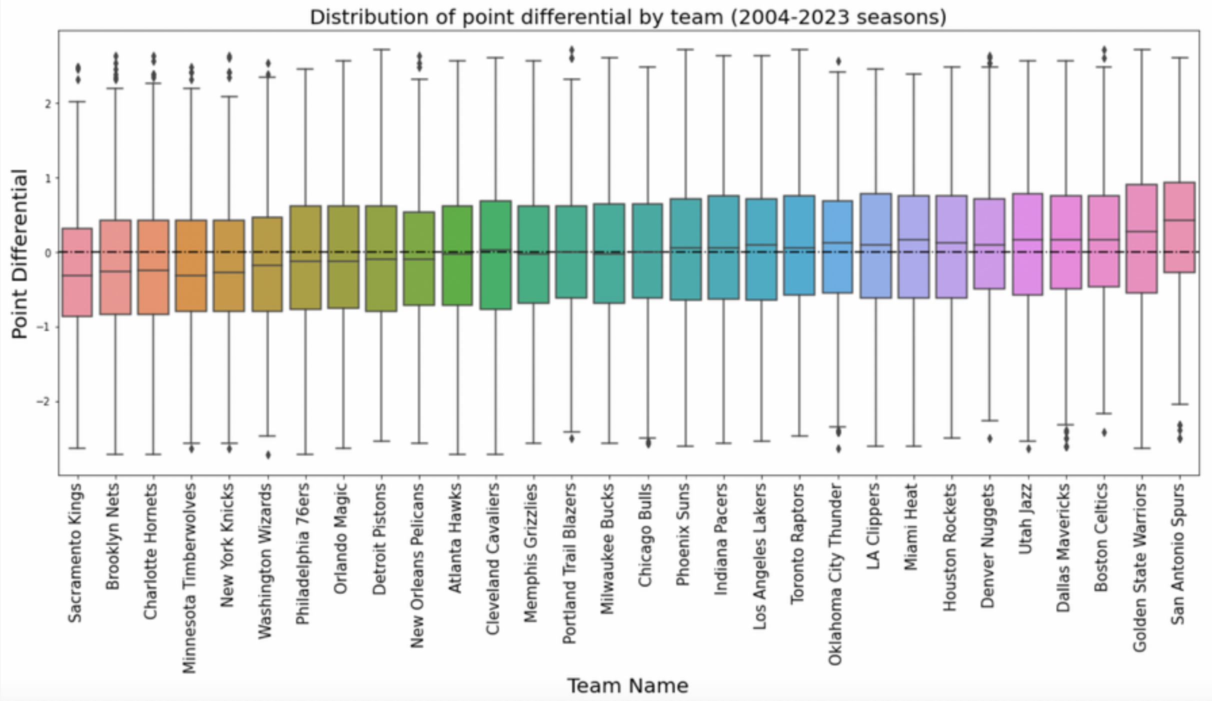
Phases of Analysis (cont'd)

Exploratory Data Analysis

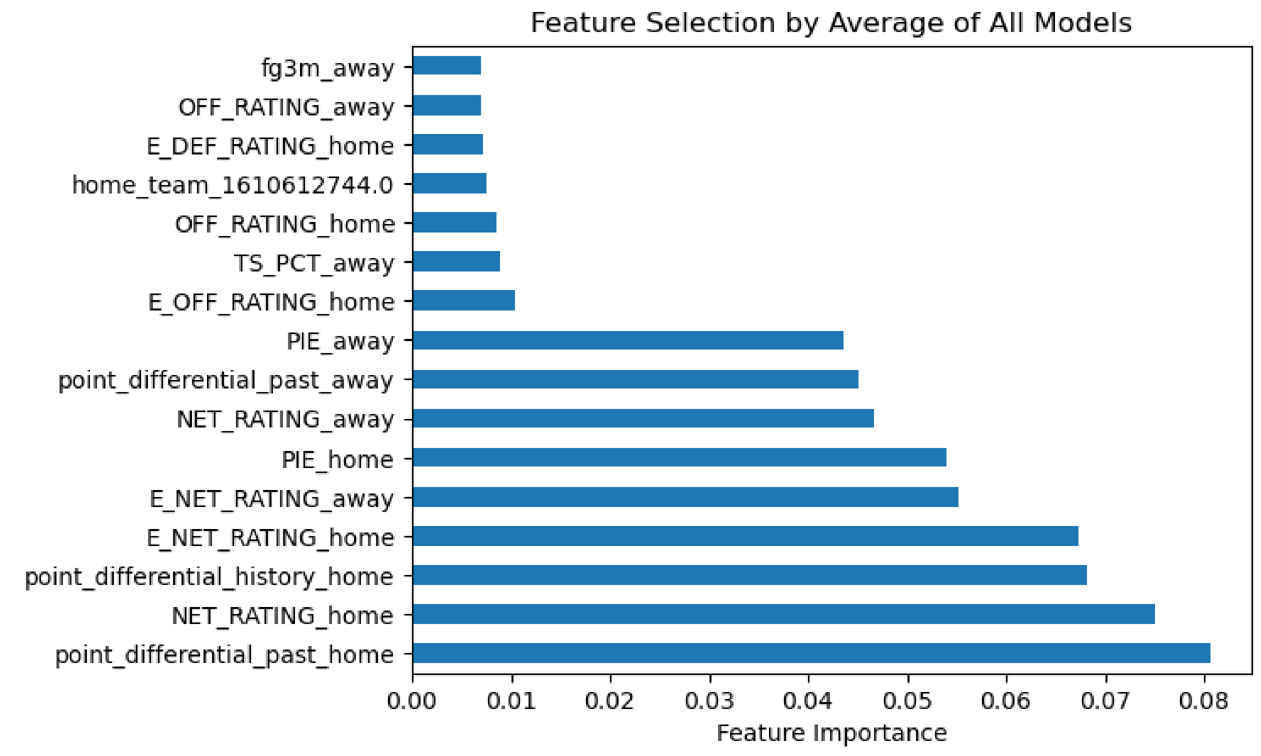


* all variables in the EDA are standardized





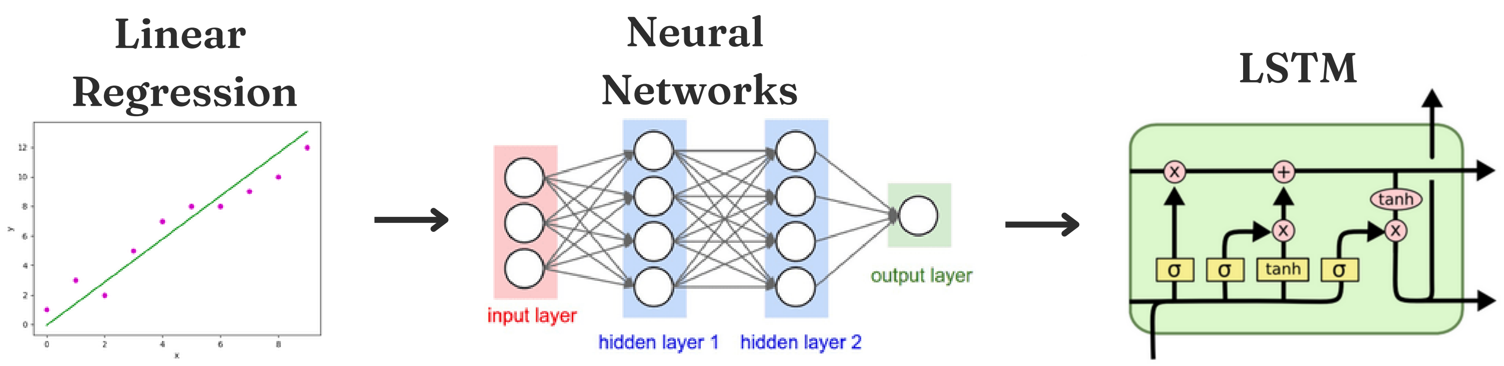
Feature Engineering



- Developed 6 random forest models to select features.
- Used GridSearchCV to tune hyper parameters at each model iterations
- MSE, MAE, R^2 were very similar for each model. (Less than 1% difference)

Phases of Analysis (cont'd)

Modeling Approach



Model Types

- Mean of target (baseline)
- Linear Regression
- Feed Forward Neural Network (FFNN)
- Neural Network with Embeddings
- Convolutional Neural Network (CNN)
- Long Short-Term Memory (LSTM)

Evaluation Approach

Quantify error using Mean Squared Error (MSE)

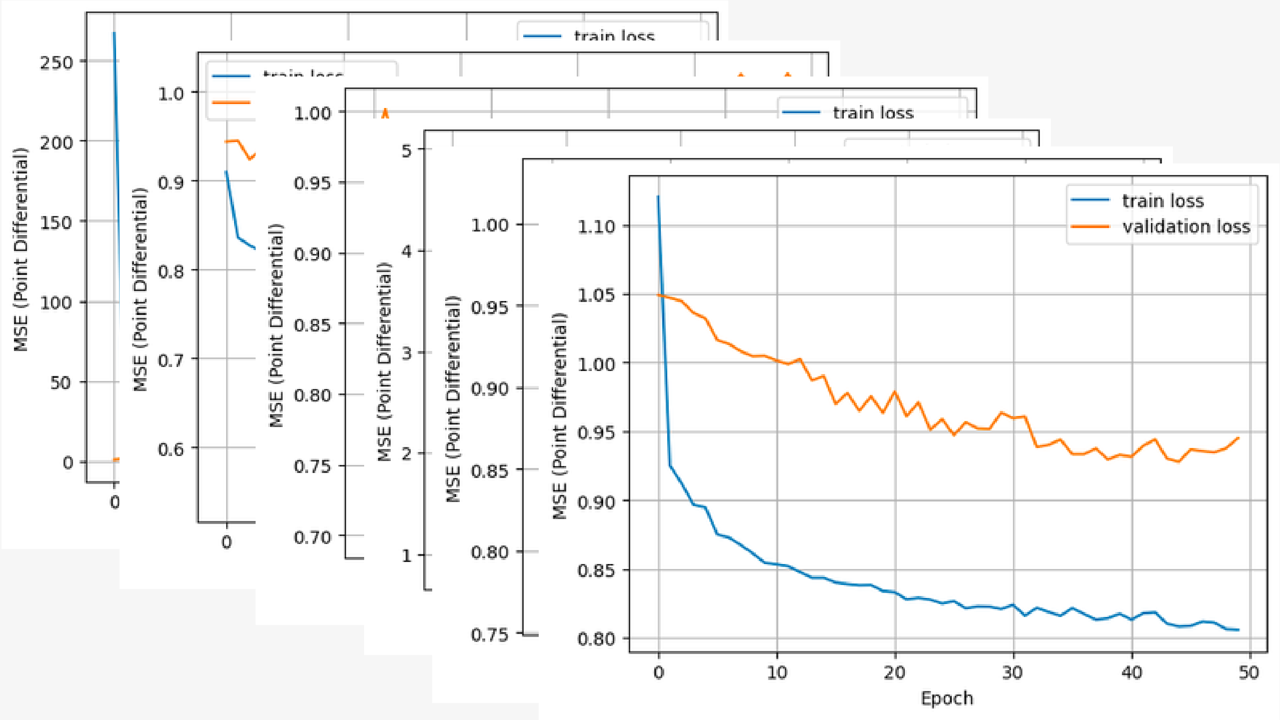
$$MSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

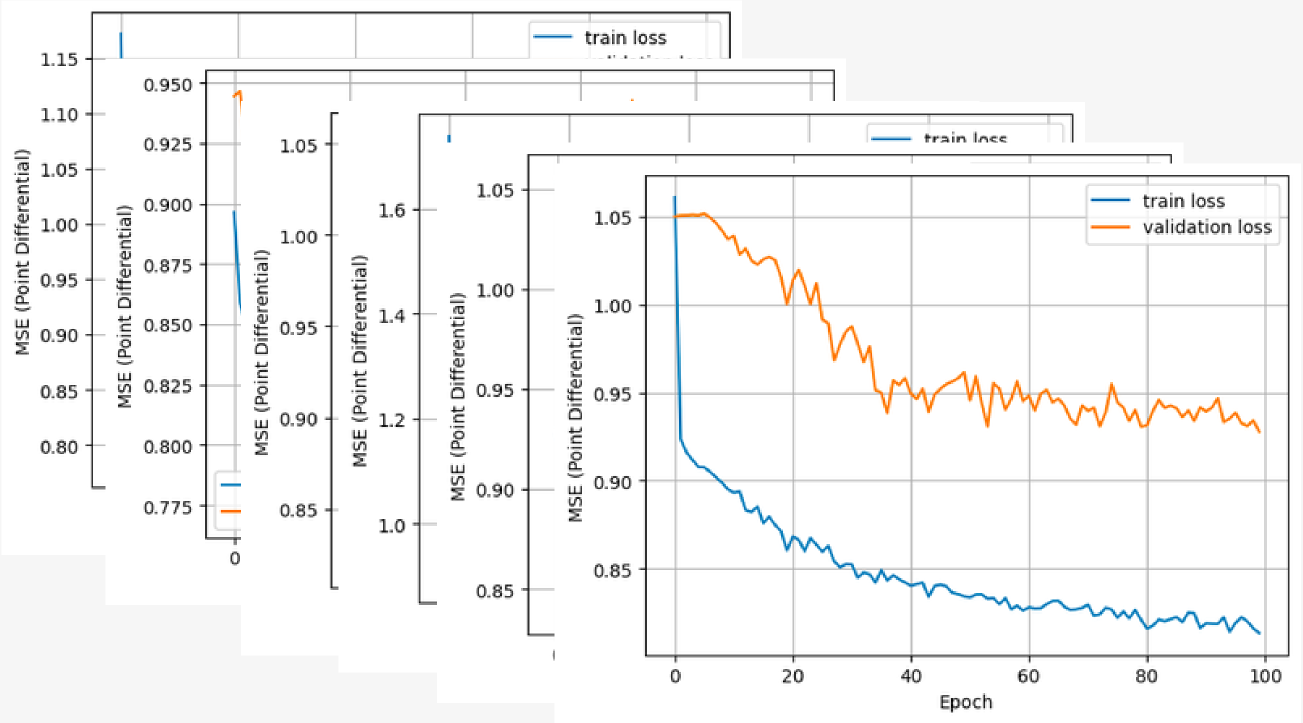
Approach #1 - Linear Regression Models (OLS)

Model	R2	Training MSE	Validation MSE	Test MSE
Baseline (predict average)	-	0.90	1.05	1.05
Previous Point Differentials (3 Features)	0.09	0.82	0.93	0.92
Top Features from Random Forest (15 Features)	0.09	0.82	0.93	0.91
Full Feature Set (132 Features)	0.11	0.80	0.92	0.91

Best linear regression test MSE: 0.91

Approach #2 - Feed-Forward Neural Networks





12 Models in Total

Hyperparameters Tuned:

- Learning Rate
- Optimizer
- Dropout Rate
- Dropout Layers
- Activation
- Epochs

Model Performance

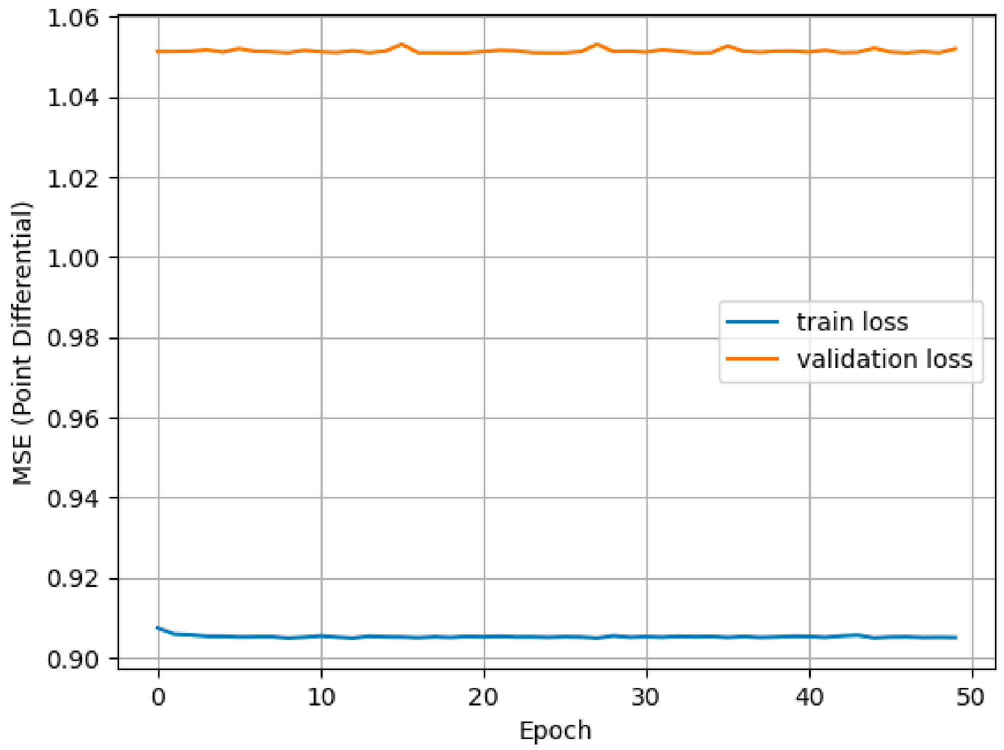
Train MSE	Validation MSE	Test MSE
0.53 - 0.90	0.93 - 1.03	0.92 - 1.03

Best linear regression test MSE: 0.91

Approach #2 - Neural Networks (cont'd)

Neural Network w/ Embeddings

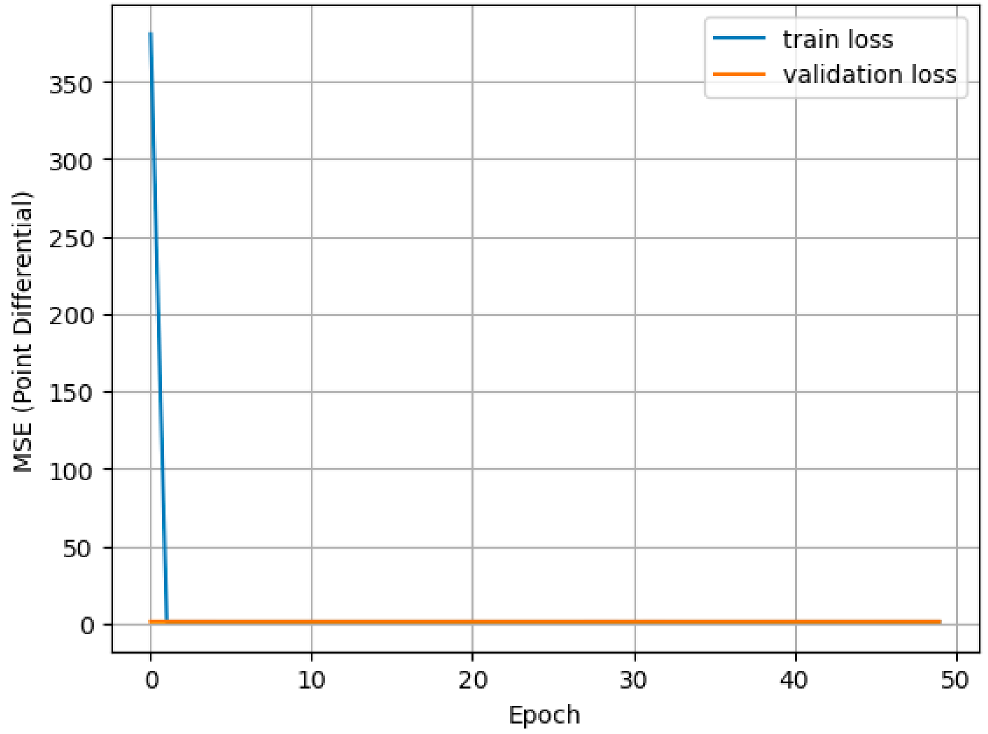
- 10 Bins
- Embedding layer
- Pooling layer
- 3 dense layers
- 2 dropout layers
- Linear output



Model Performance

Train MSE	Validation MSE	Test MSE
0.90	1.05	1.05

Convolutional Neural Network



- 3 1-D convolutional layers
- 3 max pooling layers
- 2 dropout layers
- Linear output

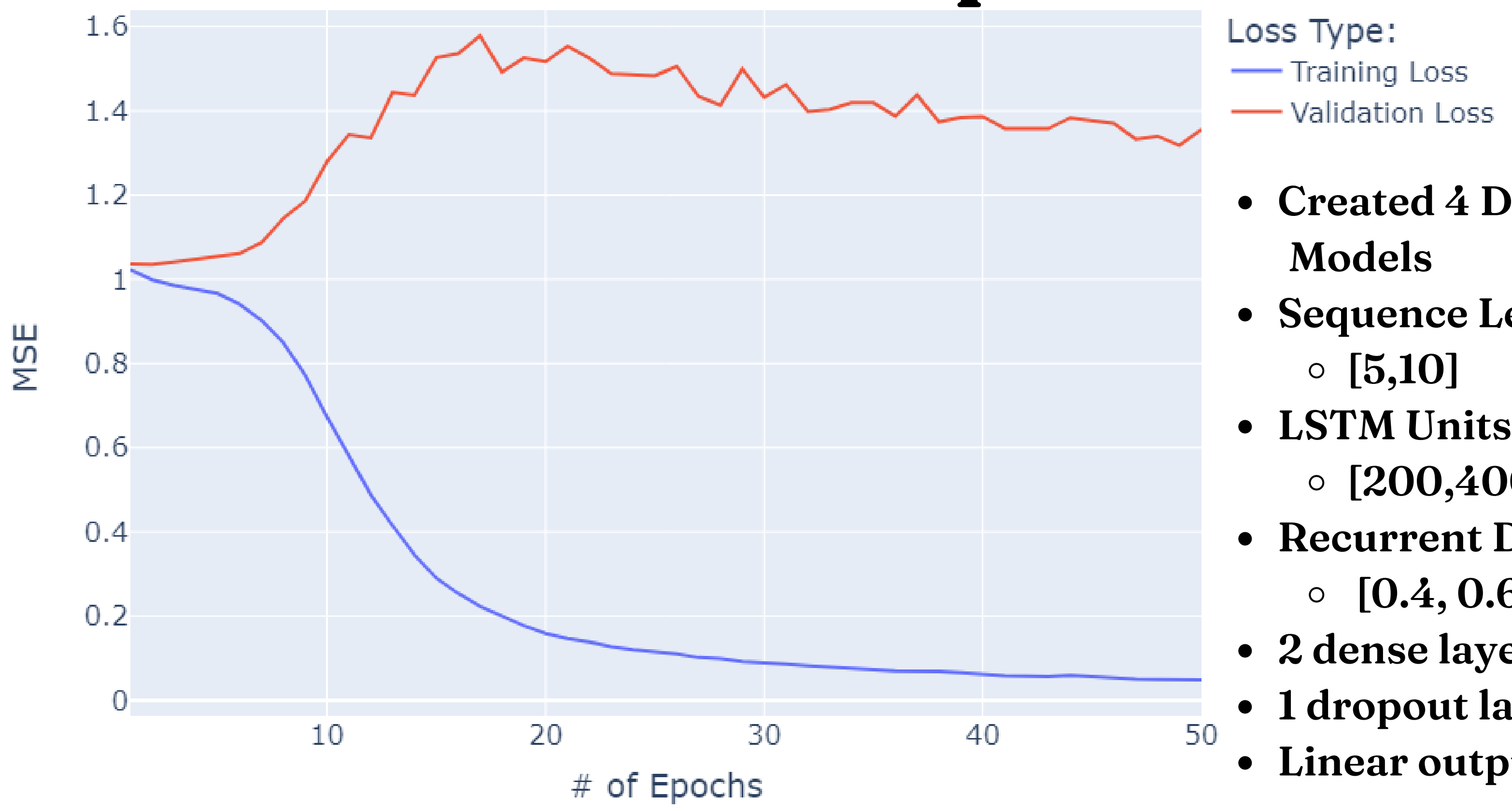
Model Performance

Train MSE	Validation MSE	Test MSE
0.91	1.05	1.05

Best linear regression test MSE: 0.91

Approach #3 - Long Short-Term Memory Models

LSTM Naive Data Manipulation



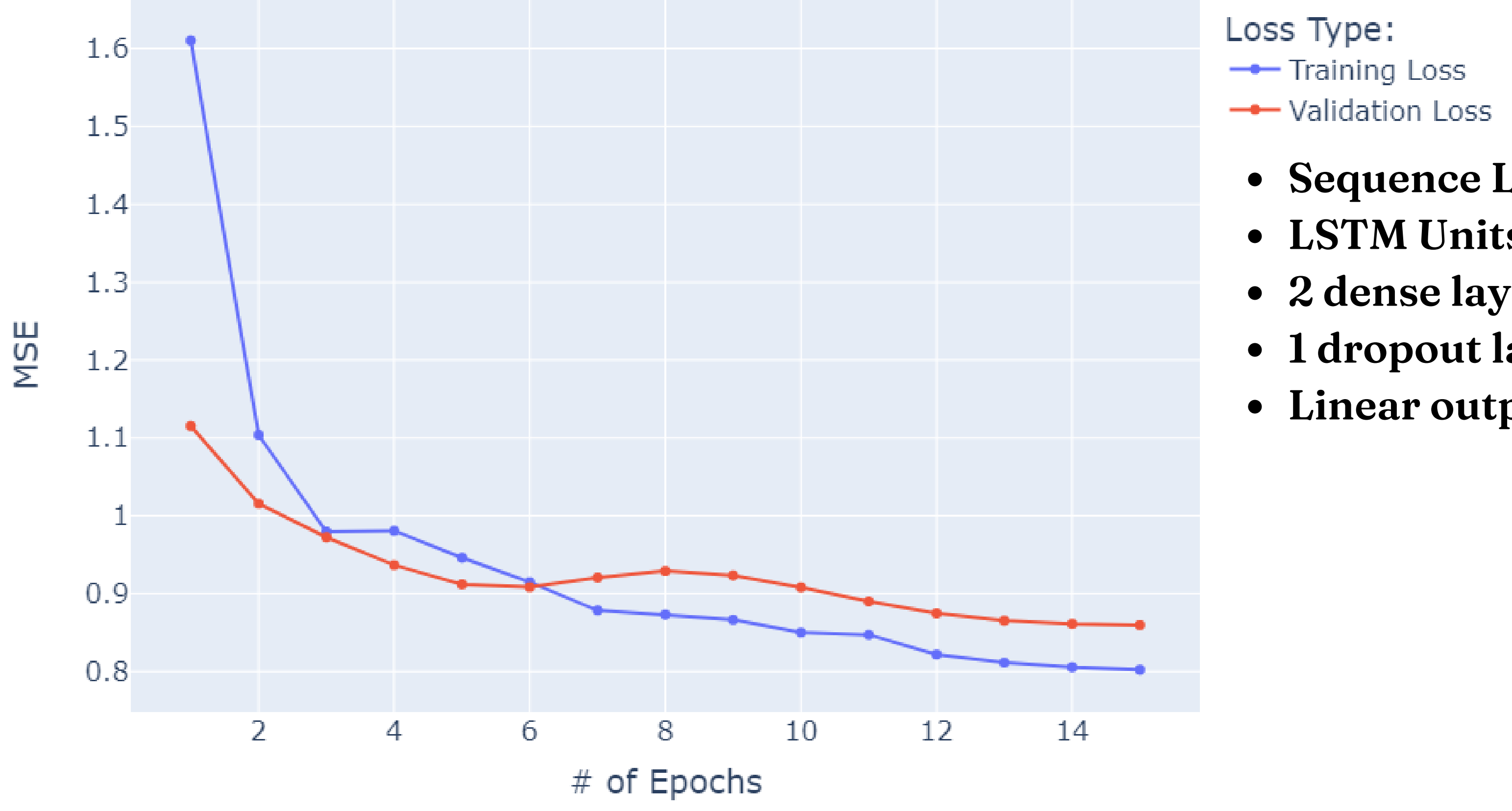
- Created 4 Different Models
- Sequence Length :
  - [5,10]
- LSTM Units:
  - [200,400]
- Recurrent Drop Out::
  - [0.4, 0.6]
- 2 dense layers
- 1 dropout layers
- Linear output

- Preprocessing Data Dimension and Setup:
  - (Point Differential, Sequence of Prior Games Point Differential, Features)

Model Performance

Train MSE	Validation MSE	Test MSE
0.05 - 0.42	1.26 - 1.38	1.26 - 1.48

LSTM Data Manipulation by Team



- Sequence Length: 663
- LSTM Units: 200
- 2 dense layers
- 1 dropout layers
- Linear output

- Preprocessing Data Dimension and Setup:
  - (Teams, Total Games (2004-2022), Features)

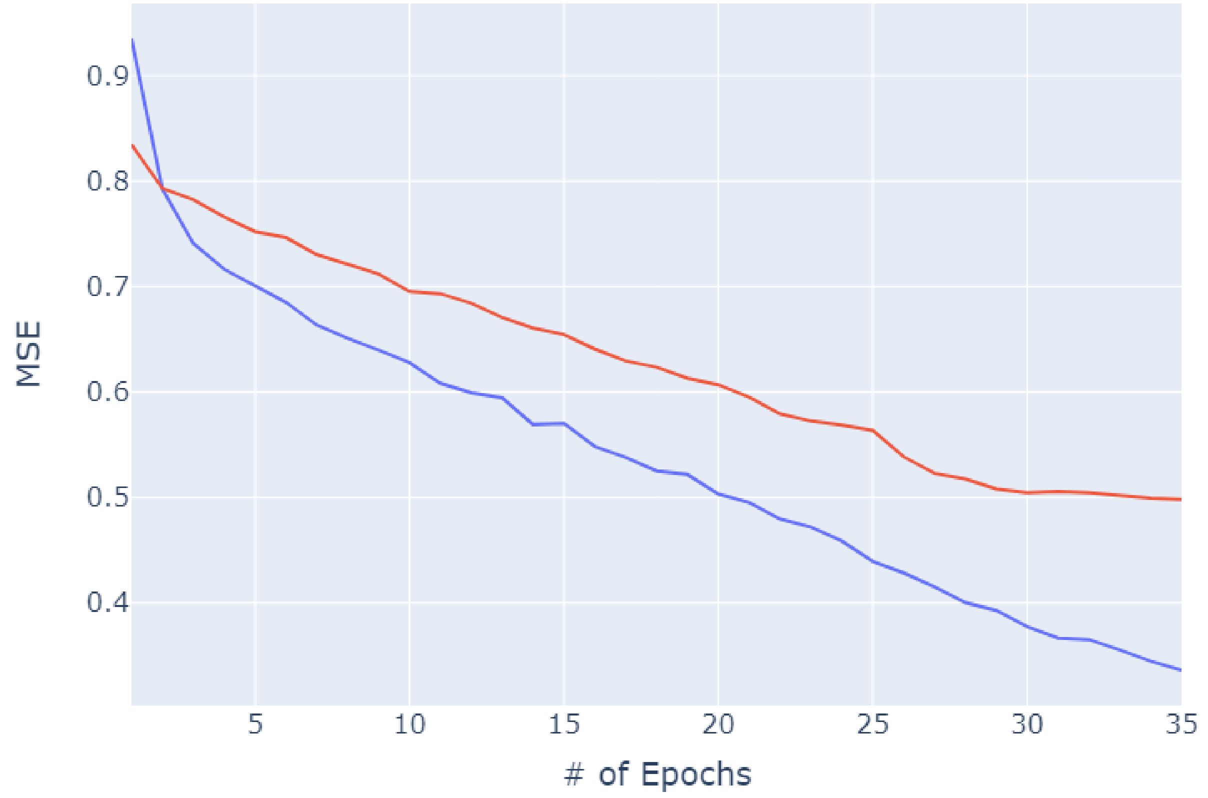
Model Performance

Train MSE	Validation MSE	Test MSE
0.78	0.83	0.85

Best linear regression test MSE: 0.91

Approach #3 - Long Short-Term Memory Models

LSTM data manipulation by home team, season



Loss Type:
Training Loss
Validation Loss

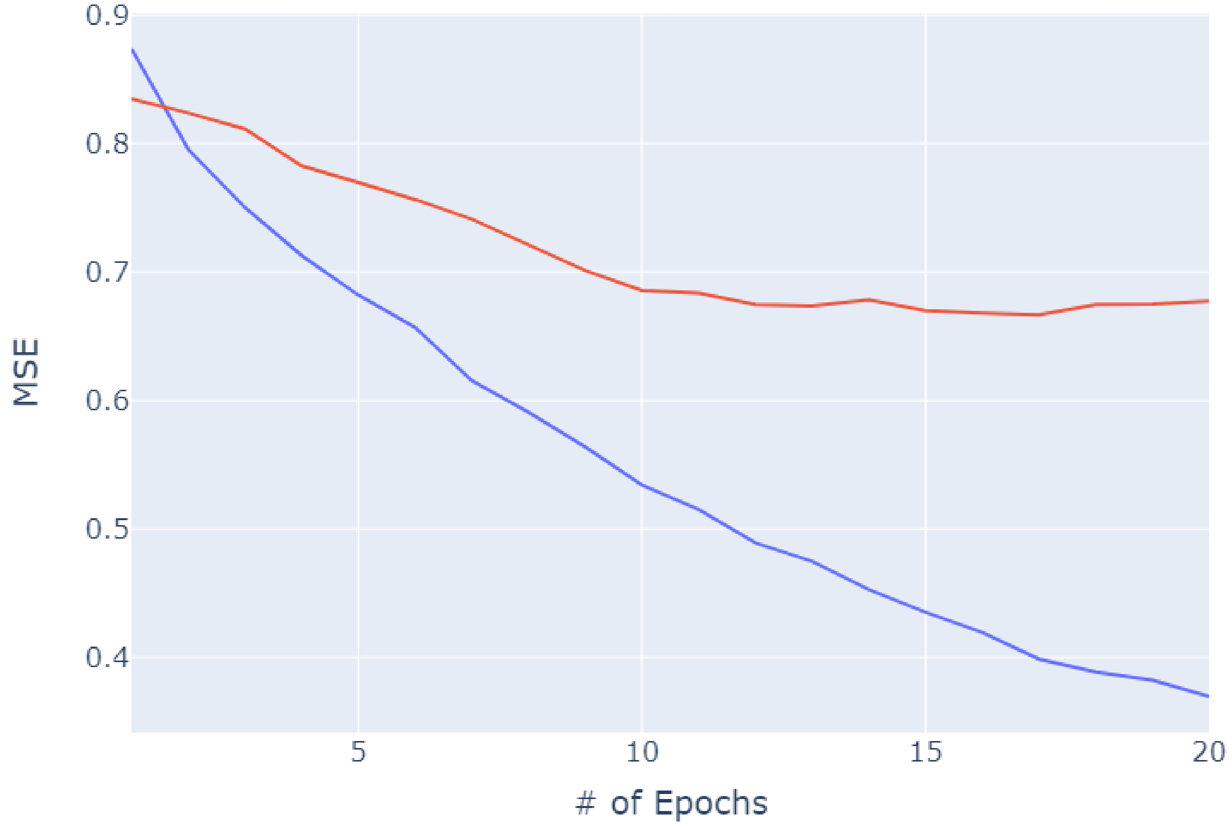
- Sequence Length: 40
- LSTM Units: 200
- 2 dense layers
- 1 dropout layers
- Linear output

- Preprocessing Data Dimension and Setup:
  - (Teams by Seasons, Home team games, Features)

Model Performance

Train MSE	Validation MSE	Test MSE
0.34	0.50	0.45

LSTM data manipulation by home team, season, bin by 5 games



Loss Type:
Training Loss
Validation Loss

- Sequence Length: 5
- LSTM Units: 200
- 2 dense layers
- 1 dropout layers
- Linear output

- Preprocessing Data Dimension and Setup:
  - (Teams|(seasons, bin), 5 games, Features)

Model Performance

Train MSE	Validation MSE	Test MSE
0.34	0.50	0.65

Model Comparisons (Evaluation)

Model	Train MSE	Validation MSE	Test MSE
Baseline (Predict Average PD)	0.90	1.05	1.05
Linear Regression	0.80	0.92	0.91
Feed-Forward Neural Network	0.79	0.93	0.92
Neural Network with Embeddings	0.90	1.05	1.05
Convolutional Neural Network	0.91	1.05	1.05
Long Short-Term Memory	0.34	0.50	0.65

Key Results

- Hyper-parameter tuning led to varying degrees of improvement across our models; however, the best performing model was an LSTM model with a MSE of 0.65.

- Future research would...
  - Incorporate player-level data in addition to team statistics
  - Explore different approaches to splitting the data into train/test folds
  - Extend the data pipelines and model building frameworks to be applied to other sports

Any questions?

Christian Lee
Chris Ratsimbazafy-Da Silva
Cameron Wright
Dave Zack

2023 April 17

Contributions

- Christian - Advanced stats data pull, EDA, data preprocessing (outliers, LSTM), model building (LSTM, Random Forest), presentation
- Chris - literature review, EDA, data preprocessing, presentation
- Cameron - regular stats data pulling, EDA, preprocessing, data housekeeping model building and formatting, presentation
- Dave - data preprocessing, (FFNN, NNE, CNN) model building, evaluation on test data, presentation