BASIC MACHINE LEARNING MODELLING TO PREDICT FINAL TOTAL SCORE USING TEAM-BASED FEATURES IN THE NATIONAL HOCKEY LEAGUE

Oscar Jaimes

CMPUT 466

University of Alberta

December 2021

Contents

1 Introduction and Motivation

Using mathematics and statistics to gain an insight on player and team performance in sports has been of growing interest since it was first seen successfully applied in baseball in the early 2000s. In most sports, it is evident that both individual athlete and aggregate team performance contribute to the final outcome of a game. A lot of research in sports statistics has been primarily focused towards combined individual athlete performance and aggregate team performance or solely individual athlete performance. To their credit, these strategies have produced very accurate results in many sports. The trade-off with this research is that the learning methods are often complex and replication is not necessarily an easy task, or is focused on individual athlete outcomes instead of final game-state aggregate outcomes.

Although individual performance of athletes is key in a semi-static sport such as baseball (i.e the pitcher and batter are the only two people playing for a considerable period of time), sports such as basketball and hockey are significantly more team-based and dynamic in play, and overall team performance seems to outweigh individual player performance (although a really good player can significantly impact overall team performance positively).

This project aims to explore the validity of exclusively using team-based aggregate statistics in the National Hockey League (NHL) as features to build basic machine learning systems that can predict the final total score of a given game. Concretely, I aim to answer the following question through this problem: Are team-based aggregate statistics significant enough to use as features in basic machine learning systems to make predictions on the final total score (FTS) of a given game in the NHL?

2 Problem Formulation

As stated in the introduction, the goal of this project is to be able to accurately predict the final total score in a given NHL game. In hockey, a goal is scored when a player of a team successfully shoots the puck into the net of the opposing team.

This project looks at this learning task as both a regression and classification problem. It is important to note that the set of final scores usually only contains integer values from 1..10, which is why we can label this as a classification problem. Conversely, even though the set of final scores is usually constrained to the set of integers 1..10, it does not necessarily have to be, and thus this assumption could impact predictions, especially for outlying data points.

2.1 Data Gathering

Almost all sports organizations, including the NHL, collect data on each game played in their organization. Usually this data is also collected by related sports companies such as ESPN. Luckily, the NHL hosts a public but undocumented REST API to view real-time and historic game data . The root endpoint for the API is: https://statsapi.web.nhl.com/api/v1/.

The NHL's API provides a wide range of data for every game played since the early 2000s. Although this API is very convenient for fetching data, a naively-written sequential script to pull a significant amount of data from the server would take hours to run. In order to reduce the time of the data gathering process, the go programming language was used for this portion of the project. Using golang's native concurrency features, we can quickly and easily pull tens of thousands of detailed game records in less than 1 minute.

The main data gathering script is located in the **gen/** folder of the project repository and is called gen_nhl_game_stats.go. This script pulls data for every game played between the 2000-2001 and 2020-2021 seasons, and a subset of the 2021-2022 season.

2.2 Feature and Target Selection

A total of 8 team-based statistics were chosen as features for this problem: blocked shots, face-off win percentage, giveaways, hits, penalty infraction minutes, power play percentage, shots, and takeaways.

Although only 8 statistics were chosen as features for this problem, there will be 16 baseline features for models as we are training with these statistics for both the away and home teams in a given game.

2.3 Proposed Models

This project aims to predict the final total score in a given NHL game utilizing simple and fundamental machine learning models. The following models are implemented and evaluated in this project:

1. Closed Form Linear Regression
2. Linear Regression with Gradient Descent
3. K-Nearest-Neighbours Regression
4. Multinomial Logistic Regression (Classification)

3 Feature and Historical Data Exploration

Before diving into model development, observing the historical distribution of final total scores and the relationship between our features and target is needed to establish a foundation of expectation for results as well as a reference frame for this problem.

3.1 Historical Final Total Score

Final Total Score (FTS) is our target variable in each model implemented in this project. **Figure 1** displays the historical distribution of FTS in the National Hockey League for over 10,000 games.

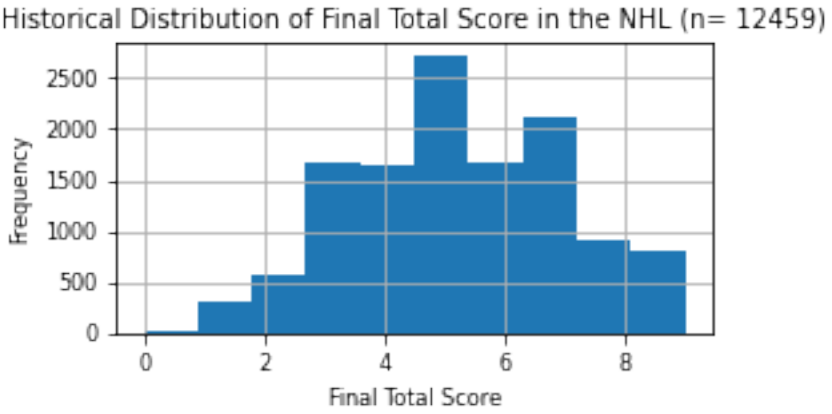

Figure 1: Historical Distribution of FTS

This distribution can be classified as approximately normal (although slightly left-skewed) with a mean and standard deviation of 5.30 and 2.00 goals respectively.

3.2 Feature Correlations to Target

It is important to also note how features impact the FTS target relative to each other. The following table shows the correlations of each baseline feature against the FTS target:

Feature	Correlation to FTS Target
home_pim	0.087542
home_shots	0.114609
home_powerPlayPercentage	0.264350
home_faceOffWinPercentage	-0.000054
home_blocked	-0.035579
home_takeaways	0.027947
home_giveaways	0.017017
home_hits	-0.043786
away_pim	0.088469
away_shots	0.122245
away_powerPlayPercentage	0.243005
away_faceOffWinPercentage	-0.002987
away_blocked	-0.055242
away_takeaways	0.028610
away_giveaways	0.027930
away_hits	-0.033793

We can see that the aggregate features that have the biggest relative postive impact on the FTS target are power-play scoring percentage, shots taken, and penlty infraction minutes. Conversely, we observe that blocked shots, hits, and face-off win percentage are inversely correlated with the FTS target.

4 Implementation and Evaluation

This section will outline the evaluation metrics for each model implemented in this project. Hyper-parameters and loss results are reported on a per-model basis, and associated jupyter notebooks containing training code will be linked.

4.1 Closed Form Linear Regression

4.1.1 Formulation

There exists a closed form solution to the minimization of the MSE for linear regression. This solution can be expressed in matrix-vector representation as:

$$\mathbf{w} = (X^T X)^{-1} X^T \mathbf{t}$$

Where **X** is our feature matrix, **t** is our target vector, and **w** is the optimal weights minimizing MSE.

Using a feature matrix consisting of 12,627 rows, the initial data was split into two sets: 80% Training and 20% Testing. A validation set was not needed as this method does not perform numerical optimization.

It is important to note that a bias term was added to the feature matrix for a total of 17 predictors.

4.1.2 Training and Testing Results

The results for the training data were adequate. **Figure 3** shows the FTS distribution produced by linear regression prediction vs the true target distribution on training data.

The mean square error for the training data was 3.28. The mean FTS for the prediction distribution was 5.30 with a standard deviation of 0.82 goals.

Tested on 1,246 unseen data points, the model seems to generalize adequately. **Figure 4** shows the FTS distribution produced by linear regression prediction vs the true target distribution on unseen testing data.

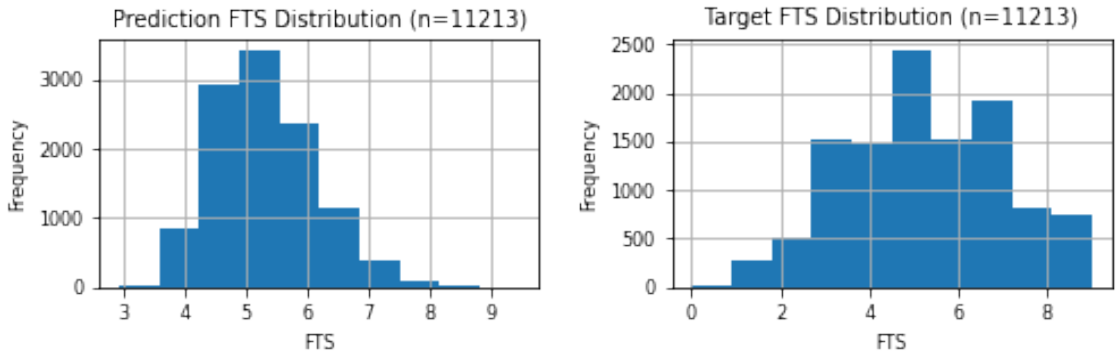

Figure 2: Linear Regression Prediction and Target FTS Distributions (Training Data)

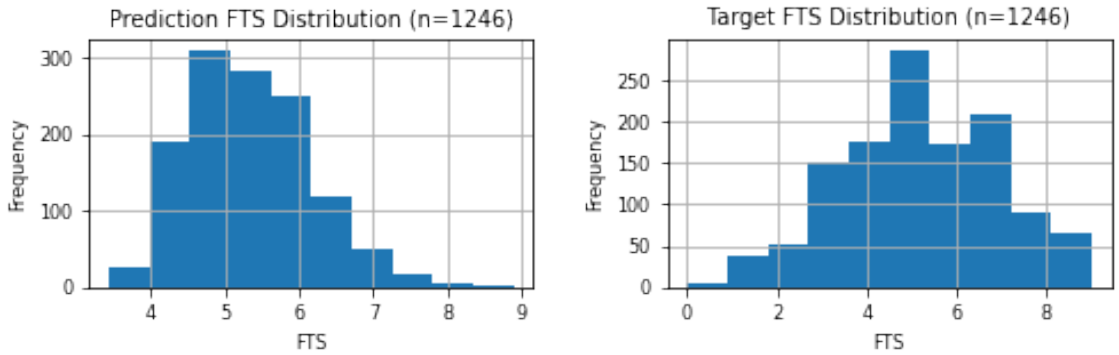

Figure 3: Linear Regression Prediction and Target FTS Distributions (Test Data)

The MSE for the test data was 3.18. The mean FTS for the prediction distribution was 5.33 compared to a mean of 5.25 for the true target distribution. Additionally, the prediction distribution and true distribution had standard deviations of 0.82 and 1.96 goals respectively.

The jupyter notebook containing the implementation of closed form linear regression can be found in the closed_form_lr.ipynb jupyter notebook within the expl/ directory of the repository.

4.2 Linear Regression with Gradient Descent

Using mini-batch gradient descent to find optimal weights by minimizing the mean square error loss yields slightly different results than the closed-form solution to the minimization problem.

Using a feature matrix consisting of 12,627 rows, the initial data was split into three sets: 80% Training, 10% Validation, and 20% Testing.

4.2.1 Hyper-parameters

The initial approach to choosing learning rates was an iterative one. Starting with a standard baseline learning rate, we run the gradient descent algorithm for a particular model and perform the following operation on alpha after each iteration:

$$\alpha_i = \alpha_{i-1} * 1.0e^{-1}$$

where the initial learning rate is

$$\alpha_0 = 1.0e^{-2}$$

We run this algorithm for 10 iterations and check which learning rate yields the lowest average loss. The goal of this algorithm is not to find an optimal alpha, but to find a reasonably close estimate within an order of magnitude. Based on the result, the chosen learning rates are then altered slightly in a guess-and-check manner for individual model purposes.

The learning rate used in the gradient descent algorithm was

$$\alpha = 1e^{-5}$$

And the maximum number of epochs was set to 100. **Figure 4** shows the minimization of the mean square error loss function over epoch iterations.

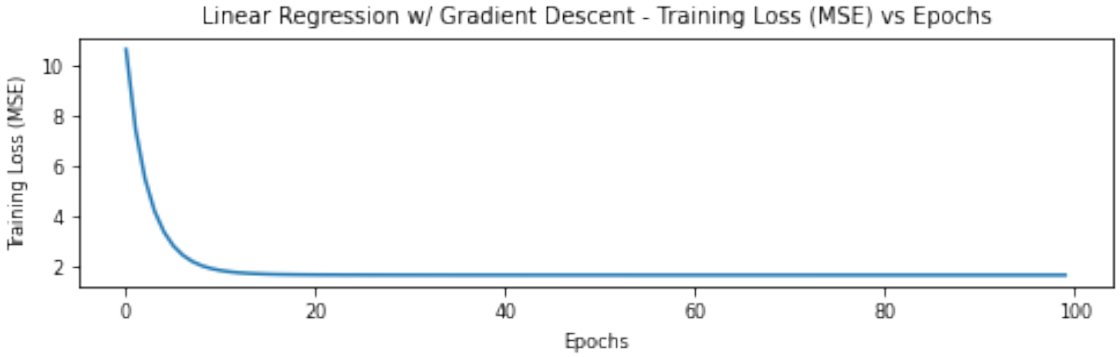

Figure 4: Minimization of MSE over Epoch Iterations

4.2.2 Training and Testing Results

Testing against the validation set, the best performing epoch was 42 with a MSE of 1.61 goals. **Figure 5** shows the distribution produced by computing predictions with the weights given by the mini-batch gradient descent algorithm and the true target predictions.

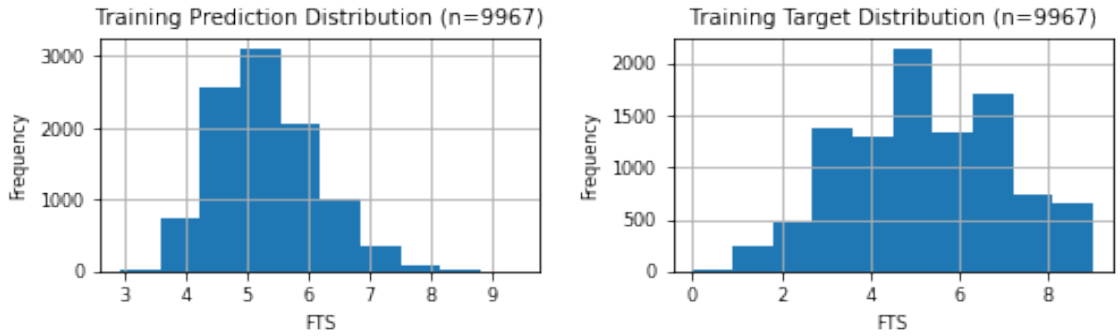

Figure 5: Linear Regression (Gradient Descent) Prediction and Target FTS Distributions (Training Data)

After acquiring optimal weights through training, we can see how the model performs against unseen data. **Figure 6** shows the distributions of predictions on unseen data, the true FTS targets.

The MSE for the test data was 3.18. The prediction distribution had a mean of 5.33 goals while the true target distribution had a mean of 5.25 goals. Additionally, the prediction distribution and true distribution had standard deviations of 0.82 and 1.96 respectively.

The jupyter notebook containing the implementation of gradient descent optimized linear regression can be found in the grad_des_lr.ipynb jupyter notebook within the expl/ directory of the repository.

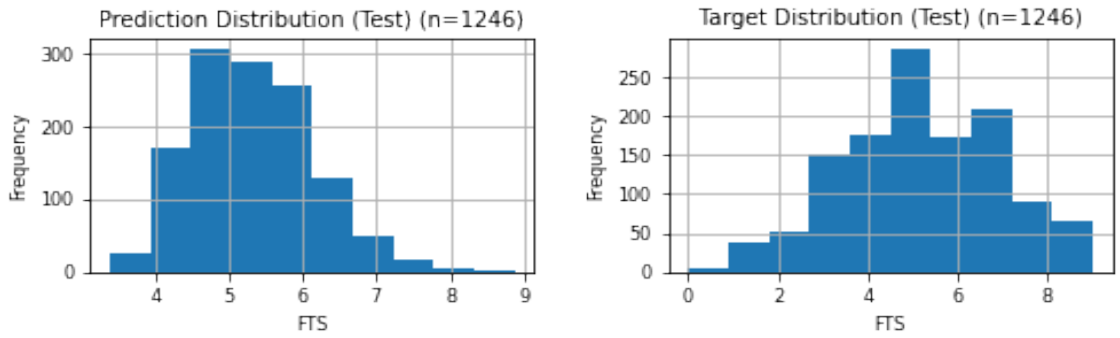

Figure 6: Linear Regression w/ Gradient Descent Prediction and Target FTS Distributions (Test Data)

4.3 K-Nearest-Neighbours Regression

Although both the closed form and gradient descent approaches of linear regression were implemented from scratch with numpy as the main library, the scikitlearn library was used to formulate the K-Nearest-Neighbours Regression Model. The code containing model implementation can be found in the knn.ipynb jupyter notebook within the expl/ directory of the repository.

4.3.1 Hyper-parameters

Through the use of cross validation, splitting up data into 80% training, 10% validation, and 10% testing, we were able to systematically choose a k-value such that it minimizes the MSE loss for the validation set. Testing values of k from 1 to 75, it was found that the best value for k is 73. This yields the lowest MSE on the validation set with 3.29 goals. **Figure 7** shows the MSE vs K value.

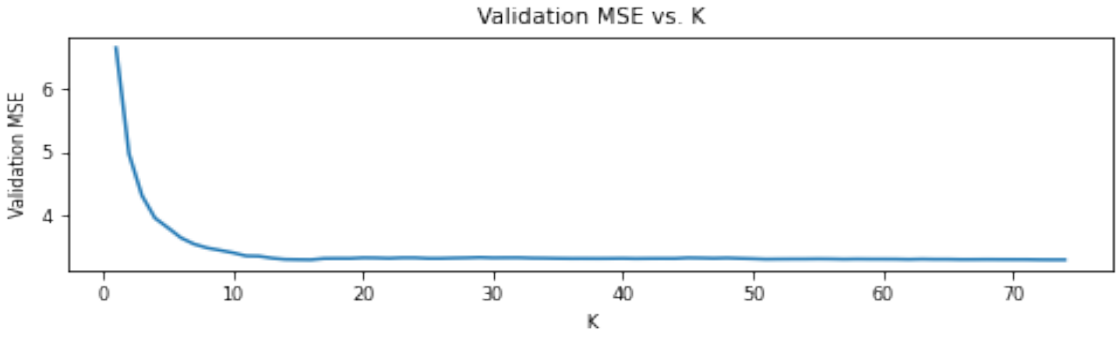

Figure 7: Validation Mean Square Error vs K-Value

4.3.2 Training and Testing Results

Figure 8 shows the distribution of the training predictions vs the distribution of the true training targets. The training predictions and true training targets had means of 5.23 and 5.31 goals and standard deviations of 0.59 and 1.99, respectively.

The model performed slightly different on unseen data. **Figure 9** shows the distribution of the test predictions bs the distribution of the true test targets. The test predictions and true testing targets had means of 5.24 and 5.25 goals and standard deviations of 0.59 and 1.96, respectively. The MSE loss for the testing data was 3.275.

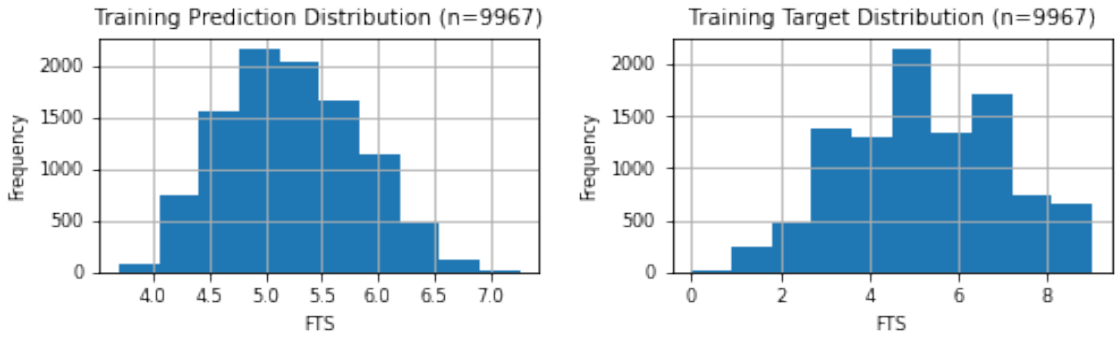

Figure 8: KNN Prediction and Target FTS Distributions (Training Data)

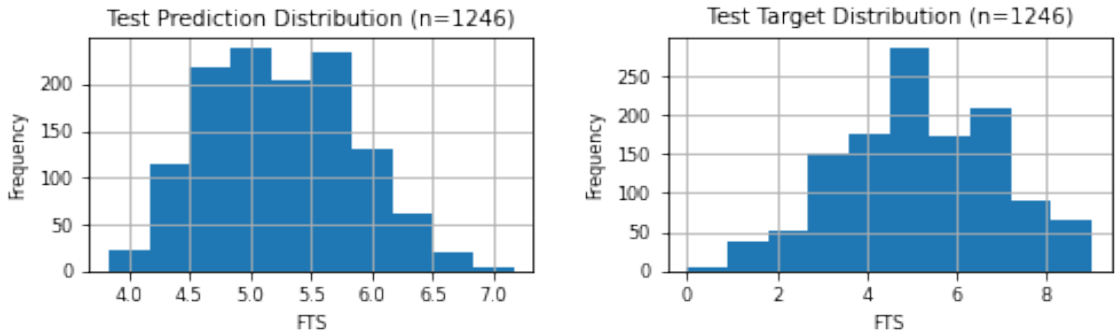

Figure 9: KNN Prediction and Target FTS Distributions (Test Data)

4.4 Multinomial Logistic Regression (Classification)

Similar to the KNN model, the implementation of multinomial logistic regression was implemented with the use of the sci-kit-learn library. The main objective of trying a classification model is to see how it would compare to regression models, given that this problem can be looked as a classification one.

If evaluated using MSE, this model yields an error of 3.86 on unseen data. The accuracy of this model in prediciting the true score was 22%. **Figure 10** and **Figure 11** display the distributions of the model for training and testing data against the true target final total score distributions respectively.

The code containing model implementation can be found in the multinomial_logistic_reg.ipynb jupyter notebook within the expl/ directory of the repository.

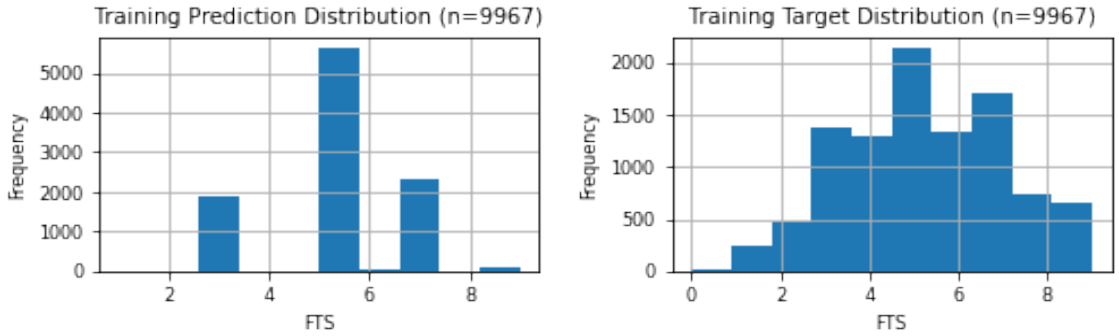

Figure 10: Multinomial Logistic Regression Prediction and Target FTS Distributions (Training Data)

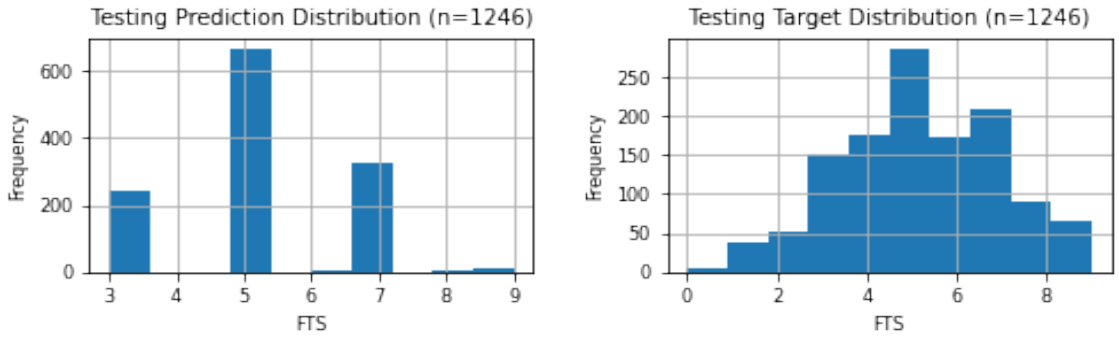

Figure 11: Multinomial Logistic Regression Prediction and Target FTS Distributions (Test Data)

5 Conclusion

5.1 Interpretation of Results

The following table outlines the main results of each model on testing data in comparison to the true distribution of final total scores in the NHL.

Distribution Source	Mean	Std. Dev	MSE	Accuracy
Linear Regression (CF)	5.33	0.82	3.18	-
Linear Regression (GD)	5.33	0.82	3.18	-
KNN Regression	5.24	0.59	3.86	-
Multinommial Logistic Regression	5.17	1.39	3.28	22%
True FTS Distribution	5.25	1.98	-	-

Both flavours of linear regression seemed to have performed almost identically, while the KNN regression performed slightly worse than both other models. The true distribution of final total scores in the NHL has a standard deviation of approximately 2 goals, but it is evident that the distributions of models produced in this project fall short of this statistic. This means that the linear regression and KNN models do not generalize well to outlying data points that are significantly far from the mean.

In regards to the multinomial logistic regression model, we can see from the distributions in **Figure 10** and **Figure 11** that there are gaps in the prediction distributions. Specifically for the training data for the model, it seems as though the prediction labels correspond to the three most significant "peaks" in the training target distribution of 3, 5, and 7. The testing accuracy of the model for the testing data was 22%, which was quiet low and surprising. This could be due to the nature of the data and would be an interesting question to explore.

To answer the questions proposed in the introduction, although semi-succesfull results were achieved, the MSE between predictions and true data for all models was approximately 3. Although low, this means the absolute error is approximately 1.73 on average. So, on average, the models are off by 1.73 goals in their predictions. Given that the standard deviation of the true distribution of final total score in the NHL is 1.98, this means that on average, our predictions are off by almost 1 standard deviation. It is evident that the distribution of final total score in the NHL is tight, and the proposed basic machine learning systems trained using team-based features can semi-adequately make predictions on the final total score of a game.