Predicting the NHL Draft

Ethan O'Brien
Western University
eobrien.hba2024@ivey.ca

James Hutchins
Western University
jhutch63@uwo.ca

Shaz Khan
Western University
skan746@uwo.ca

***Abstract*—This paper aims to develop a predictive model that accurately forecasts if a player will be selected in the first round of the National Hockey League (NHL) draft based on amateur performance statistics. To achieve this goal, the datasets were pre-processed by removing null values, outliers, and highly correlated variables. All categorical variables were converted to one-hot encoded variables, and columns with vast variabilities were removed to improve the quality of the dataset.**

Class imbalance was checked, and the class weights in each classification model were balanced to ensure accurate predictions. The dataset was then standardized and split into training and testing sets, and Principal Component Analysis (PCA) was used to reduce the number of features in the dataset.

Several classification models were tested, including Logistic Regression, K-Nearest Neighbours, Decision Trees, Support Vector Machine, and Multi-layer perceptron feedforward neural network. These models were evaluated based on their accuracy, precision, recall, and F1 score.

To improve the base models, ensemble methods such as Random Forests, Bagging, and Boosting were used, and the hyperparameters were fine-tuned using a GridSearch algorithm. After evaluating the performance of different models, a bagging SVM model was selected as the final model with C equal to 100 and gamma equal to 1, achieving an accuracy of 84%.

Overall, this model provides NHL teams with valuable insights to inform their draft strategies, allowing them to make data-driven decisions and choose high-performing players while saving time and resources. By exploring various classification models, this study has demonstrated the effectiveness of machine learning algorithms in predicting NHL draft outcomes.

I. Introduction

NHL teams use substantial resources and time scouting players to determine what round to draft them. Consequently, teams must find ways to make more informed and data-driven decisions, as draft pick mistakes are costly and can directly influence the team's future success.

A. Motivation

The NHL draft is an important event for all teams, and selecting the proper player can significantly impact the team's performance. However, predicting the draft round in which a player will be selected is a difficult task that requires evaluation of various factors such as a player's skill set, physical attributes, and performance history. According to a study conducted by the NHL in 2018, NHL teams spent an average of $320,000 on amateur scouting during the 2016-2017 season [1]. However, poor draft picks don't only cause monetary costs. For example, in a 2018 interview with Sports Illustrated, former NHL general manager Craig Button stated:

> the cost of a first-round mistake can be catastrophic. A bad draft pick can set a team back for several years in terms of both on-ice performance and financial investment. [2]

Therefore, a predictive model for the NHL draft would provide teams with valuable insights to inform their draft strategies, improving their chances of selecting high-performing players while saving resources.

B. Related Works

Previous research has explored the NHL draft and attempted to predict the success of drafted players. For instance, one study utilized machine learning algorithms to predict a player's career performance based on their pre-draft statistics [3]. Another study focused on identifying the critical factors that contribute to a player's success in the NHL, such as their point production and draft position [4]. However, no research focuses specifically on predicting the round in which a player will be drafted. Therefore, this paper aims to address this gap in the literature by developing a predictive model that can accurately forecast draft rounds.

C. Problem Definition

The objective of this paper is to create a predictive model that can accurately forecast the draft round in which a player will be selected in the NHL draft. Specifically, the scope will focus on the first round as these players produce or cost organizations the most capital, as they can become superstars or a bust. To accomplish this objective, several player attributes will be analyzed, such as player position, physical measurements, and performance history, to evaluate how these fields influence draft-round selection. Ultimately, the goal is to provide teams with a valuable tool to inform their draft strategies, enabling them to choose high-performing players while saving time, and resources, and allowing the team to make data-driven decisions.

II. Methodology

A. Data Processing

The purpose of this paper was to develop a classification model to predict the likelihood of a professional hockey player being drafted in the first round based on their amateur performance statistics. To achieve this, two datasets were utilized. The first contains NHL draft round picks [5] and the other performance data from the Elite Hockey Prospects website [6], which was obtained using an off-the-shelf scraper.

The datasets were merged using the player's name and only the most recent amateur year's stats for each player was kept.

First, the dataset was examined for missing values and duplicates using the Pandas library in Python. Fortunately, there were only a few null values that could be removed as they did not compromise the dataset.

Next, the dataset was tested for outliers using the z-score technique provided by the Scipy stats library. The test revealed extreme values for columns P (points) and GP (games played). These rows consisted of some first-round picks with low stats and some non-first-round picks with high stats, predominantly from international leagues. External research validated that this data was incorrect and that the error resulted from misinformation on the Elite Hockey Prospect website. Hence, the outliers were removed to improve model performance.

All categorical variables, such as player position, were converted to one-hot encoded variables and columns with vast variabilities, such as league and nationality, were removed. The collinearity of the dataset was evaluated using a correlation matrix. The analysis identified that columns A (assists), G (goals), and P were highly correlated, indicating collinearity. This issue was addressed by removing the P column from the dataset.

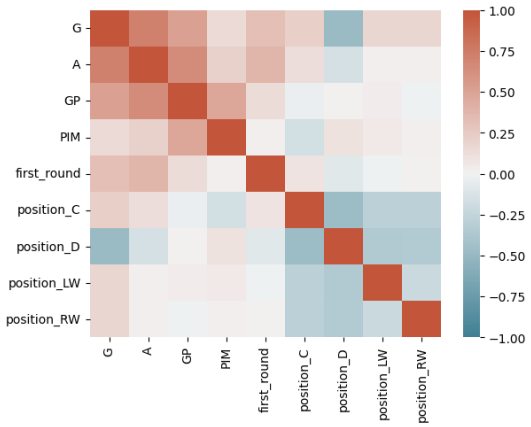

Fig. 1. Correlation Matrix.

The dataset was then checked for class imbalance. 9% of the data was first-round picks, which is consistent with the number of rounds in the NHL draft. Therefore, the dataset is unbalanced and so the class weights in each classification model must be balanced to ensure that the first-round and non-first-round picks are equally represented.

After cleaning the data, it was processed for analysis. It was divided into features (x) and target (y) variables. The features dataset was standardized using the Scikit-Learn library to ensure that the scale of the data did not affect the classification models. This involved transforming the data to a standard score by subtracting the mean and dividing it by the standard deviation. The standard scaler was saved as a joblib file so that it could be used to transform new data during the prediction stage.

Finally, the dataset was split into training and testing sets using Scikit-Learn, with a ratio of 80:20. This will allow the machine learning models to be trained on a subset of data and tested on another.

B. Feature Engineering

Principal Component Analysis (PCA) was used to reduce the number of features in the dataset while preserving as much information as possible. PCA transforms the data into a lower-dimensional space by identifying the most important features, known as principal components [7].

A PCA algorithm was employed by calculating a covariance matrix and the resulting eigenvectors and eigenvalues. These are used to rotate the n-dimensional space into a smaller space. The information proportion was calculated by dividing each eigenvalue by the sum of all eigenvalues, producing the below Scree plot which demonstrates the proportion of information explained by each principal axis. The algorithm was verified by matching the results to Scikit-Learn's PCA module. The analysis revealed that six axis', G, A, GP, PIM, position_C, and position_D, account for 85% of the data variation.

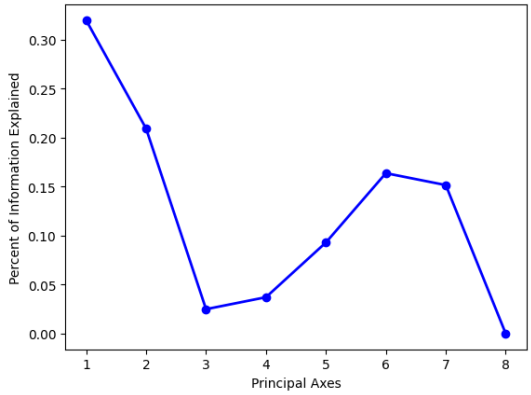

Fig. 2. Scree Plot.

In addition, wrapper methods were used to verify the best subset of features for the model. Sequential forward and backward feature selection was used with and without replacement, as well as recursive feature selection with logistic regression as the test model. Sequential forward selection (SFS) iteratively adds features to the model based on their performance until a predetermined number of features is reached, while Sequential backward selection (SBS) iteratively removes features from the model [8]. Both SFS and SBS can be used with or without replacement, depending on whether a feature that has been selected or removed is allowed to be reconsidered in subsequent iterations. Recursive feature elimination (RFE) iteratively removes the feature that has the lowest importance score. The analysis revealed that the model performed the best for the same 6 features as the PCA analysis.

Overall, feature engineering reduced the dataset's dimensionality, improving computational efficiency and allowing for more sophisticated models to be deployed given CPU restrictions on the test computer.

C. Model Selection

To determine the best classification model for the proposed problem, various machine and deep learning models were employed, such as logistic regression, k-nearest neighbors, decision trees, support vector machines, and neural networks.

The first model applied was a simple Logistic Regression model (LR). LR models use the log-odds or sigmoid function to model the connection between a group of independent variables and a binary dependent variable. The model determines the probability that an observation belongs to a specific class and then classifies the observation based on a predetermined threshold [9]. The Scikit-Learn library was used to train this model.

Next, a k-nearest neighbors (KNN) model was employed, which assumes that data points belonging to the same class will be close to each other in space. To classify a new data point, KNN calculates the distance between the new data point and all the known data points of a particular class. Based on the value of K (i.e., the number of closest neighbors to evaluate), it applies the majority rule to assign the new data point to a particular class [10]. The Scikit-Learn library was used to train this model.

In addition, a decision tree (DT) model was used. DT models function by recursively splitting the data based on the values of the features that best differentiate the data into distinct groups. Each split creates two new branches, and the algorithm continues splitting until it reaches a stopping criterion, such as a minimum number of observations in each group or a maximum tree depth [10]. The resulting tree is a hierarchical structure that can be used to classify new data points by following the branches from the root node down to the leaf node corresponding to the predicted class. The splitting criterion used was the Gini index, which measures the probability of misclassifying a randomly selected observation in a given subgroup. The max depth was set to 3 to avoid overfitting; however, this will be tuned. The Scikit-Learn library was used to train this model with the architecture in figure 3.

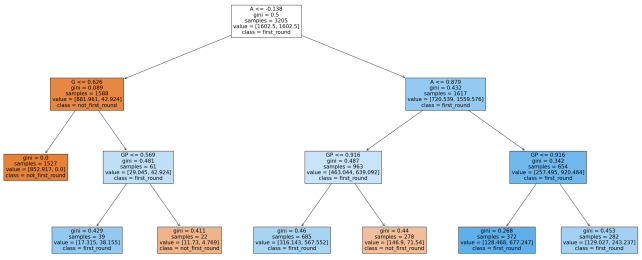

Fig. 3. Decision Tree Plot.

A support vector machine (SVM) model was used. It functions by defining a decision boundary that maximally separates the data points of different classes. The decision boundary is typically a hyperplane in a high-dimensional space that is defined by the support vectors, which are the data points closest to the decision boundary. SVM seeks to maximize the margin between the support vectors on either side of the decision boundary. In cases where the data cannot be linearly separated, SVM uses kernel functions to transform the data into a higher-dimensional space where linear separation is possible [10].

The radial basis function (RBF) was used. It operates by mapping the data into a high-dimensional space where a linear decision boundary can be defined. The RBF kernel calculates the similarity between two data points using a Gaussian distribution centered at the data point [10]. Other hyper-parameters, C, and gamma were set to defaults until tuning. The Scikit-Learn library was used.

Finally, a Multi-layer Perceptron (MLP) feedforward neural network classifier was built using the Keras and TensorFlow libraries. Each layer of the MLP model consists of a set of neurons, where each neuron takes inputs, performs a linear transformation, and then applies a non-linear activation function to the output [11]. The weights and biases of the neurons are adjusted during training to improve the accuracy of the predictions. The output of one layer serves as input to the next layer, and the process repeats until the final output is produced.

The model deployed has four hidden layers, each with 128 or 64 neurons, and the ReLU activation function is applied to the output of each hidden layer. The ReLU activation function sets all negative values to zero and preserves positive values, which allows the model to learn complex non-linear relationships between the input and output variables [12].

Batch normalization is a technique used to improve the performance and stability of deep neural networks. It normalizes the inputs of each layer by subtracting the mean and dividing by the standard deviation of the batch, which helps to reduce the effects of input covariance shift and can improve training convergence [13].

Dropout regularization is a technique used to prevent overfitting in neural networks. It randomly drops out (sets to zero) a proportion of the neurons during each training iteration, which reduces the interdependence of the neurons and can improve the generalization performance of the model [13].

The final output layer of the model has one neuron and a sigmoid activation function, which outputs a probability value between 0 and 1 that represents the predicted probability of the input belonging to the positive class. The binary cross-entropy loss function is used to measure the difference between the predicted output and the actual output, and the Adam optimizer is used to minimize the loss during training by adjusting the weights and biases of the neurons [13].

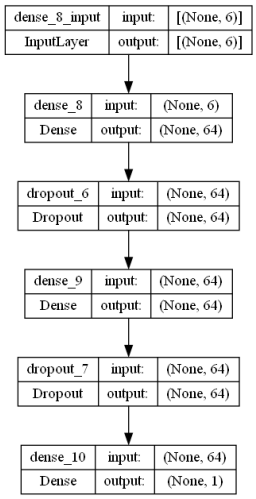

Fig. 4. Neural Network Architecture.

After training and testing each model, the performance was evaluated against accuracy, precision, recall, F1 score, and a confusion matrix.

D. Model Improvement

1) Ensemble Methods: After the base models were explored, ensemble methods were used to improve the performance. Ensemble methods were applied to the LR, DT, and SVM models.

One ensemble method that was utilized was the Random Forests method, which was applied to the DT base model. This method involves building multiple decision trees that collaborate to classify new data points. The final classification is based on the most popular classification from all the trees. Bootstrapping is used to create new datasets for each tree, which reduces the variance of the model and leads to better generalization [14].

The scikit-learn Bagging Classifier was also used to improve the DT, LR, and SVM models. This technique involves training multiple models on different subsets of the dataset and aggregating their predictions to form the final output. This helps to reduce over-fitting and increase the stability of the model [14]. Figure 5 provides a visualization of the bagging process [15].

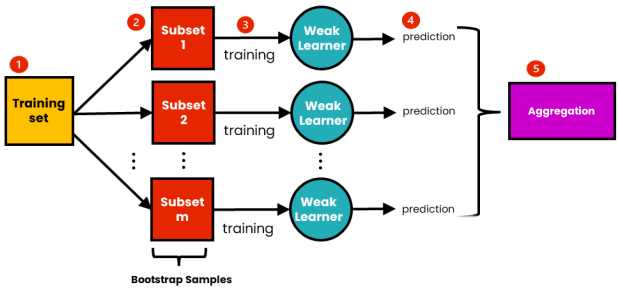

Fig. 5. The Bagging Process.

Boosting was also explored to improve performance. This is a sequential learning technique where each base model is a weak learner and builds off the previous model. In Adaptive Boosting or AdaBoost, each base model is reweighted to reduce error, and subsequent models attempt to fix the errors in the previous stage. This approach helps to improve the model's accuracy and reduces bias [14].

2) Hyper-parameter Tuning: A Grid Search algorithm was used to tune the hyper-parameters of each base model. This technique searches through a specified parameter space to find the optimal combination of hyper-parameters. The Grid Search algorithm tested each combination of specified hyper-parameters by training the model on the training data and evaluating its performance on the validation data. The combination of hyper-parameters that produced the highest accuracy was selected as the optimal combination.

III. Results

The discussed base models were trained and tested on subsets of the dataset. The model performance is as follows:

TABLE I
BASE MODEL PERFORMANCE RESULTS.

Model	Accuracy	Precision	Recall	F1
LR	77%	64%	78%	65%
KNN	88%	68%	58%	60%
DT	66%	62%	78%	58%
SVM	78%	66%	84%	68%
MLP	70%	63%	81%	61%

The KNN model performed the best with an accuracy of 88%; however, this was because the Scikit-Lean KNN Classifier does not support model weighting and so the classes were imbalanced, shown by the lowest Recal Score of 58%. Therefore, this result must be disregarded and instead the best-performing base model was the SVM with 78% accuracy. The MLP Neural Network also performed well with a Recal Score of 81% indicating high performance at classifying the under-weighted first-round pick class.

The LR, DT, and SVM models were then improved using the discussed ensemble methods.

TABLE II
TUNED MODEL PERFORMANCE RESULTS.

Model	Accuracy	Precision	Recall	F1
Random Forests	77%	64%	79%	65%
Bagging DT	69%	62%	78%	60%
Bagging SVM	82%	65%	73%	67%
Bagging LR	77%	63%	77%	65%
AdaBoost DT	67%	62%	77%	58%
AdaBoost SVM	88%	44%	50%	47%

The DT base model was improved by 11% to an accuracy of 77% using the Random Forests Method. Similarly, the Bagging SVM model was improved by 4% to an accuracy of 82%. However, this was at the expense of a decrease in the Recall score which is an important metric considering the class imbalance. The Bagging DT resulted in a slight

accuracy improvement and the Bagging LR saw no change in performance. The AdaBoost SVM model had the highest accuracy at 88%; however, it had the lowest recall at 50%. The model did not correctly classify any of the first-round picks and so must be disregarded.

Overall, the Bagging SVM model performed the best. It predicted 38 false negatives and 105 false positives. These results are expected as the stats for players drafted at the bottom of the first round compared to players drafted at the top of the second round are very similar. However, this analysis could indicate that a portion of the players drafted in the first round did not deserve to be. It would be interesting to explore if these players were draft busts or lived up to first-round expectations.

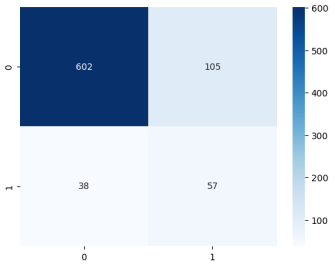

Fig. 6. Bagging SVM Confusion Matrix.

The GridSearch algorithm was used to fine-tune the hyper-parameters of the Bagging SVM and Random Forest DT classifier and further improve model performance. For the Bagging SVM, the optimal hyper-parameters were C=100 and gamma=1. For the Random Forest DT, the optimal hyper-parameters were max_depth=7 and criterion='gini'. The Bagging SVM accuracy was improved by 2% to 84% and the Recall Score by 5% to 78%.

Overall, this paper concluded that an SVM model using the Bagging technique, and hyper-parameters of 100 for C and 1 for gamma resulted in the greatest model performance. NHL teams can now predict if a player deserves to be drafted in the first round with 84% accuracy, potentially saving teams millions of dollars by avoiding draft busts and reducing scouting costs.

IV. Conclusion

The NHL draft is an essential event for all teams, and selecting the right player can significantly impact the team's performance. Poor draft picks can cause substantial costs, and therefore, a predictive model for the NHL draft would provide teams with valuable insights to inform their draft strategies, improving their chances of selecting high-performing players while saving resources.

Based on the analysis conducted, a Bagging SVM model with tuned hyper-parameters was found to be the best predictor of whether a player will be drafted in the first round of the NHL draft. The model achieved an accuracy score of 84% and a Recall score of 73%.

Moving forward, future research could explore the prediction of other rounds in the NHL draft and analyze the impact of additional variables, such as player character and work ethic, on draft-round selection. Expanding the model would greatly increase the value to NHL teams. In addition, it would be interesting to explore in more detail which players were classified as false negatives. The model could be validated by matching false negatives to draft busts and false positives to eventual superstars. It can also be tested on the upcoming NHL draft.

In conclusion, this paper addressed the gap in the literature by developing a predictive model that could accurately forecast the first round of the NHL draft. It employed a thorough Machine Learning process including data cleaning, data exploration, feature engineering, model testing, ensemble methods, and hyper-parameter tuning. It demonstrated the importance of a clean and viable dataset and how base models can be improved with additional machine-learning techniques.

References

[1] NHL. (2018). NHL teams invest more in amateur scouting and development. Retrieved from https://www.nhl.com/news/nhl-teams-invest-more-in-amateur-scouting-and-development/c-299201546

[2] Button, Craig. "The Cost of a First-Round Mistake Can Be Catastrophic." Interview with Dan Falkenheim. Sports Illustrated, 25 Apr. 2018, https://www.si.com/nhl/2018/04/25/nhl-draft-scouting-players-craig-button-interview

[3] O'Reilly, N., & Williams, P. (2019). Predicting career performance of NHL players using pre-draft statistics and physical measurements. Journal of Sports Analytics, 5(2), 95-107

[4] Larivière, J., Gauthier, A., & Lapierre, M. A. (2019). The importance of point production and draft position in predicting the success of NHL players. Journal of Quantitative Analysis in Sports, 15(3), 157-170

[5] NHL Draft Hockey Player Data 1963-2022 by Matt O'Pray, Kaggle (2021). Retrieved from https://www.kaggle.com/datasets/mattop/nhl-draft-hockey-player-data-1963-2022

[6] Elite Prospects Hockey Stats & Player Data by Mjavon, Kaggle (2021). Retrieved from https://www.kaggle.com/datasets/mjavon/elite-prospects-hockey-stats-player-data

[7] Abdi, H., & Williams, L. J. (2010). Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics, 2(4), 433-459. https://doi.org/10.1002/wics.101

[8] Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. Journal of machine learning research, 3(Mar), 1157-1182.

[9] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825-2830.

[10] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.). Springer-Verlag.

[11] TensorFlow. (n.d.). Guide to Keras Sequential Model. Retrieved from https://www.tensorflow.org/guide/keras/sequential_model

[12] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press

[13] Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. International Conference on Machine Learning (ICML), 448-456.

[14] Kumar, D., & Sharma, A. (2021). Ensemble Learning for Heart Disease Classification using Machine Learning Algorithms. In 2021 6th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-6). IEEE.

[15] Analytics Vidhya. (2023, January 30). Ensemble Learning Methods: Bagging, Boosting, and Stacking. Retrieved February 27, 2023, from https://www.analyticsvidhya.com/blog/2023/01/ensemble-learning-methods-bagging-boosting-and-stacking/