Personal Project Report: NHL-Draft-Ranking-Model

Francis Picard

Abstract

This project aims to develop machine learning models to predict how good draft-eligible junior hockey players will be in the future. Currently, the dataset consists of 57 statistics about 914 NHL-drafted forwards who have played in the CHL (Canadian Hockey League) between 2000-01 and 2018-19. The models are meant to be used by NHL teams before the NHL Entry Draft to make projections about all the relevant CHL players, which might be used later in the ranking process.

Motivation

At the end of every regular season, the National Hockey League (NHL) hosts an Entry Draft where the best junior hockey prospects from around the world are selected by the NHL teams. During the year, teams spend countless hours studying junior hockey players to rank the best for the draft. From scout reports to hockey analytics, each player's case is thoughtfully analyzed with hope of predicting as accurately as possible how talented these young players will be. These selections are at the hearth of team rebuilds, but accurate predictions can be valuable to any team. Over the last few years, hockey analytics has been on the rise and many teams have decided to include them in the ranking process. This project is meant to be a first step in developing of machine learning models to help NHL teams make their final list for the NHL Entry Draft.

Question Definition

My initial goal was to answer the fundamental question every team is trying to answer before the draft:

- Which player still available at our drafting position will be the best for us?

This is often not an obvious question to answer and not quite measurable. First, the "best [player] for us" is ambiguous. While some people will prefer players who can play the whole 200 feet, others will love goal scorers. When two very different players have about the same strength overall, the team's need usually settles the debate on who to pick. To make the models as generally applicable as possible, I've decided to rank players based on how many points per regular season game they are projected to get in the future. I have also decided to narrow the range of players to forwards only.

- Which forward still available at our drafting position will score the most points per regular season game?

Now that we have a metric to rank the players, we need to specify a timestamp for our projection. From the team's point of view, do we want the player that will score the most points in the next season or e.g. in five years? Again, this is different from team to team. For the sake of this project, I've decided to project the number of points per regular season game five years after the draft. Arguably, young players need time to develop before we have a good idea of their peaking value, and five years is fair.

- Which forward still available at our drafting position will score the most points per regular season game five years after being drafted?

There is one final thing to add to make it a measurable question: in which league(s) do we project the statistic? Only considering the NHL might be a practical concern. Indeed, some players will stay longer in the AHL and dominate there before having a shot in the NHL simply because they need more time to develop. Thus, NHL teams arguably might want to know how good a player will be in the AHL five years after the draft if they do not make it to the NHL just yet.

- Which forward still available at our drafting position will score the most points per regular season game in either the NHL or the AHL five years after being drafted?

This is a measurable question. To compare projections in the AHL and the NHL, I have decided to turn the problem into a classification one. Briefly, the models will classify players in buckets of points per regular season game in either the AHL or the NHL. The classes/projections will consist of a league (either the AHL or the NHL) and a range of points per game (e.g. [0.6, 0.8)). This is because the skill gap between the NHL and the AHL is considerable and the same point rate per game in both leagues reflects very different player skill levels.

Data Collection

The dataset consists of several regular season, playoff, and other statistics about NHL-drafted forwards from the CHL between 2001 and 2019 (inclusive). Each record corresponds to a drafted player and each feature corresponds to a statistic about them. Among the various hockey databases freely available on the web, HockeyDB is a great place to start collecting this data. Indeed, on HockeyDB, every drafted player is attributed a unique ID that can be used to retrieve yearly statistics about such players in many different leagues, including the CHL, NHL, and AHL. The main concern with collecting data from HockeyDB alone is that there aren't a ton of statistics to work with. For example, the regular season stats only include the *GP* (number of games played), *G* (number of goals scored), *A* (number of assists), *PTS* (number of points), *PIM* (number of penalty minutes), and the +/-. Still, these statistics together with the player's name and the team they played for can almost uniquely define a player (it would be improbable that two players with the same name played for the same team, during the same year, and had all of those statistics equal).

For every drafted player from the CHL, I retrieved their regular season, playoff, and other statistics from the year they have been drafted. For example, if a player was drafted in the 2007 NHL Entry Draft, then I gathered data from his 2006-07 regular season and 2007 playoffs in the CHL. Future work could explore whether gathering data from previous junior years helps to develop better models. First, I scraped basic data from the HockeyDB player's data web page. Then, I sought more features from the CHL website, including e.g. *First* (the number of games for which the player has scored the first goal), *DS* ("Dangerous Shots": the number of shots close to the net), and *PPG* (the number of powerplay goals). After, I retrieved the player's *Weight* and *Height* at the draft from the NHL website. Finally, I retrieved the target feature, regular season *PTS/GP*, from HockeyDB five years after the corresponding NHL Entry Draft. To put all of this together, several inner joins was used between the tables. All in all, the data collection process should be thought of as collecting data from HockeyDB and then adding features to create a more complete dataset.

Since I merged different data from different sources using inner joins, it is worth addressing the potential disparities among the HockeyDB, the CHL, and the NHL website and how they have/can affect(ed) the data collection process. First, there were very few differences in the GP, G, A, and PTS statistics for the same player between the HockeyDB and the CHL website (less than a handful per year). As these errors occurred randomly, I have decided to exclude the players with at least one disparity in those features because doing this would not introduce bias. Next, I observed that most of the disparities among the three websites occurred in the players' names. From my observations, about 4% of the drafted CHL players had their names spelled differently on at least two of the three websites. I noticed that those players usually had a cognate first name, e.g. Mike and Michael, or their last name is capitalized differently. e.g. McLachlan and Mclachlan. So I wouldn't say that these errors happened randomly among the players. However, I think it is fair to assume in general that a player's name was independent of their skills. Especially, those errors did not target any type of player from the point of view of hockey statistics. Hence, I have decided to exclude the players whose names were spelled differently across at least two websites as doing so would again not introduce any bias.

The dataset obtained from the data collection consisted of 914 player records who have been drafted in the NHL Entry Draft and have played in the CHL between 2000-01 and 2018-19. There were 57 different features as well.

To retrieve the data yourself using the scripts in the repository, please consult the README.

Data Annotation: Classifying Records

There are 14 possible classes. First, the regular season PTS/GP statistic can fit in either of the following six ranges of values:

[0.0, 0.2)
[0.2, 0.4)
[0.4, 0.6)
[0.6, 0.8)
[0.8, 1.0)
[1.0, ...)

The square bracket means the value is included whereas the parenthesis indicates the value is excluded. For example, a value of 0.2 falls within [0.2, 0.4) but not within [0.0, 0.2). Also, the statistic can be retrieved in either the AHL or the NHL. Combining all possible leagues and ranges gives a total of 12 classes. The other two classes are explained later and concern the players who were not regular players in the NHL or the AHL five years after the draft.

The classification process is already implemented during the data collection; the user has nothing to do to classify the records. First, the PTS/GP statistic is retrieved from the HockeyDB scraping. Then, the class is obtained by mapping the value to the league it was measured in and the range of values that include it.

For example, if a player was drafted in the 2009 NHL Entry Draft and only played in the NHL during the 2013-14 regular season (2009 + 5 = 2014, the regular season before the 2009 NHL Entry Draft is the 2008-09 season) with a PTS/GP statistic of 0.47, then this record is classified as NHL [0.4, 0.6).

For players who have played in both the NHL and the AHL five years after being drafted, the PTS/GP statistic is taken from the league they have played the most games. For example, if a player was drafted in the 2004 NHL Entry Draft and played 47 games in the AHL with a PTS/GP of 0.87 and 23 games in the NHL with a PTS/GP of 0.20 during the 2008-09 regular season, then the record is classified as AHL [0.8, 1.0).

For players who have played the same amount of games in the NHL and the AHL five years after being drafted, the statistic is taken from the NHL. For example, if a player was drafted in the 2015 NHL Entry Draft and played 34 games in the AHL with a PTS/GP of 0.57 and 34 games in the NHL

with a PTS/GP of 0.34 during the 2019-20 regular season, then the record is classified as NHL [0.2, 0.4)

It is worth noting at this point that the PTS/GP statistic is only meaningful and accurate if the player has played enough games. To this end, I have chosen that players must have played a minimum of 20 games in either the AHL or 20 games in the NHL five years after their corresponding NHL Entry Draft to be classified into one of the 12 classes described above. What is left to do now is to handle the records that do not meet this requirement.

The players who have played less than 20 games in the AHL and less than 20 games in the NHL, but have played at least 20 games in another league, five years after their corresponding NHL Entry Draft, are classified as: "Not_In_NHL_AHL". For example, if a player was drafted in the 2003 NHL Entry Draft and didn't play in the NHL, played five games in the AHL, and played 43 games in the ECHL during the 2007-08 regular season, then they fit in this category. Players in this category are playing professional hockey five years after their corresponding NHL Entry Draft, but not as regular players in the NHL or the AHL.

Finally, players who haven't played at least 20 games in any professional league five years after being drafted are classified as: "Other". It is hard to conclude anything about those players since their absence could be due to injury, retirement, etc. Some of them will have a career as a hockey player in the future and others won't.

The 14 classes are:

NHL [0.0, 0.2), NHL [0.2, 0.4), NHL [0.4, 0.6), NHL [0.6, 0.8), NHL [0.8, 1.0), NHL [1.0, ...)

AHL [0.0, 0.2), AHL [0.2, 0.4), AHL [0.4, 0.6), AHL [0.6, 0.8), AHL [0.8, 1.0), AHL [1.0, ...)

Not_In_NHL_AHL

Other

Data Preprocessing

In this section, I will go over how I have preprocessed the raw data obtained from the data collection before feeding it to the machine learning models.

A Note on Data Leakage

In general, data leakage can occur in two flavors. First, information from the validation and testing sets can be used when training the models, scaling the features, etc. To prevent this from happening, I will be using sklearn's Pipeline and ColumnTransformer objects to structure the data transformations starting with the raw data obtained from the data collection. This will ensure that the data transformers are only fitted on the training data. Second, it can be the case that some features are consequences of the target, so they cannot be recorded before making predictions. In this problem, the junior stats can be collected before the player starts the regular season five years after the draft. However, since the models are meant to be used before the NHL Entry Draft, what needs to be ensured here is that all the junior statistics can be retrieved before the NHL Entry Draft of the same year. Since 2000, the drafts have always been occurring in the second half of June whereas the CHL playoffs end in May. Hence, the NHL teams will have about a month between the time they have all the data to use the models and the time to make their selections.

One thing worth noting here is that in the data collection process, the players' height and weight were retrieved from the NHL draft's website. This was done purely for convenience and those features are also available on the CHL website.

Missing Values

Missing values in this dataset take two different forms. First, some statistics with usually an extensive range of values have a recorded value of 0 for every player in a junior league for a given year. Although for some of them there is a slim possibility that this might be true, it is never happening in practice. For example, every player in the league can theoretically end the season with a +/- score of 0, but this is extremely unlikely. Second, some features are often filled with values that are either infeasible by themselves or impossible by considering other features. For example, every player cannot have twice as many goals as shots on goal. On another note, I noticed that all the missing values in this dataset are the fruit of statistics (features) not recorded appropriately in a given league and a given year. Furthermore, most of the missing statistics during the regular season were also missing during the playoffs. For example, during the 2000-01 regular season and the 2001 playoffs in the QMJHL, all the values were missing for the +/- and the P+/- features.

Moreover, the dataset obtained from the data collection has a feature that corresponds to the regular season year each player was drafted right after. For example, if a player was drafted during the 2015 NHL Entry Draft, then the 2014-15 year was recorded. As the models are meant to be used on future player records, the models might be learning patterns in the data with respect to the years while they will always be used on new year values. This can lead to overly optimistic estimations of the generalization error if not handled properly as we could be testing the models on some records taken from years the models have already been trained on. To prevent this from happening, I have decided to remove the years from the analysis. One consequence of this is that the missing data now becomes MNAR (Missing Not At Random) rather than MAR (Missing At Random). Future work can investigate whether using Leave-One-Year-Out Cross-Validation would be preferable.

Below, I will go into detail about how I handled each feature that was not recorded properly over a certain period in a certain league.

To begin with, the DS (Dangerous Shot) and PDS (Playoff Dangerous Shots) statistics were only recorded in the QMJHL from 2010-11 onwards and in the OHL from 2015-16 onwards. For the other seasons, the statistics were either set equal to SHOG (Shots on Goal) and PSHOG (Playoff Shots on Goal) respectively, or set to 0. Yet, in practice, not every shot is a dangerous shot and some shots are clearly dangerous shots. From the dataset, only 164 records have an accurate value for those features, which corresponds to 82% of missing values. I have decided to remove those fea-

tures from the analysis as the percentage of missing values is high. It would be useful in the future to check whether missing value imputation could still be useful.

Next, the faceoff features (FOA, FOW, FO%) and their corresponding playoff ones (PFOA, PFOW, PFO%) were missing in the OHL from 2000-01 until 2014-15 inclusively and in the WHL from 2000-01 until 2016-17 inclusively. From the raw data, these values were all set to 0. However, since there are faceoffs in every hockey game, having no faceoff in an entire season is impossible. These features have 65% missing values in the dataset. As this percentage is high, I have again decided to remove those features and it would be useful in the future to see if missing value imputation can still provide useful information.

Third, the playoff shootout statistics (PSOGP, PSO/G, PATT, PSOWG, PSO%) were set to 0 for every player in the dataset. The reason is quite simple: there is no shootout in the playoffs. In my opinion, the statistics are present to keep a homogeneous webpage structure between the regular season and playoff statistics on the CHL website. Still, for the analysis, those features do not bring anything useful and must be removed.

The last features that were missing in more than one league are the SOG and PSOG statistics. Although they were recorded in the QMJHL since 2000-01, the features were missing in the OHL from 2000-01 until 2014-15 inclusively and in the WHL from 2000-01 until 2016-17 inclusively. The missing values were either 0s for everyone in a given season or set to some very small values. When small values were used, most players had scored way more goals than they had shots on the net, which is impossible. Note that the goal-scored distributions made sense, and since there are shots on goal in every hockey game, I concluded that the SOG and PSOG features were not recorded correctly. They have 65% missing values in the dataset so the analysis will not consider those features. As mentioned above, missing value imputation should be analyzed in the future.

Now more specifically in the QMJHL, the +/- and the P+/- stats were missing from 2000-01 until 2002-03 inclusively. Indeed, every player had a +/- and P+/- stat of 0, which is theoretically possible yet infeasible in practice. The missing values represent 5% of the values for this feature. From my domain knowledge, I don't think the +/- and P+/- statistics had a very different distribution in those years compared to the more recent data when taking into account the other features. Note that the +/- statistic is a difference of two count variables, namely how many goals were scored by their team when the player was on the ice minus how many goals were scored by the opposing team when the player was on the ice. None of them are features in the dataset, and the +/- can be negative with no lower bound theoretically. Hence, I have decided to train two K-Nearest Neighbor models on the features to do model-based imputation for the missing values of the +/- and the P+/- features respectively. Future work could check if shifting the feature values into the non-negative range with some empirical lower bound and then using count regression techniques would yield better results.

In the QMJHL and the WHL, the PIM and PIM/G statistics were missing from 2000-01 until 2002-03 inclusively. Similar to the +/- stat, while it is technically possible that every player had a PIM stat of 0 (meaning that no player ever had a penalty), it would be extremely unlikely. Surprisingly, their playoff correspondents (PPIM and PPIM/G) were recorded appropriately. The missing values represent 8% of the values for those features. Again, I don't think the PIM's distribution has changed over the years when considering the other features. Hence, I have decided to use a regression model to impute the missing data. Since the PIM is a count variable, with a mean of 63.126113, a variance of 2,335.426826, and very few zeros, I have decided to use a negative binomial regression model.

Finally, the regular season shootout statistics (SOGP, SO/G, ATT, SOWG, SO%) were only recorded starting in 2005-06 for the QMJHL and the OHL and in 2006-07 for the WHL. Those statistics couldn't be recorded before because shootouts were not introduced in the regular season until then. Hence, a value of 0 for those features in this time period is technically right. Therefore, I have decided to keep those values as they are. Put differently, I am not considering them as missing values. Future work could check if feature deletion or imputation would yield better results, but one should remain mindful that other features might be affected by the overtime format (e.g. OTG: Overtime Goal).

Model-based Imputation As I mentioned, I have used model-based imputation whenever I decided to impute values for some missing data. My goal here was to keep things simple as this is not the main task of this project. Future work could investigate if using more complex models would yield better results.

For +/- and P+/- statistics, I have used KNNs with PCA beforehand to reduce the dimensionality. I have performed cross-validation using GridSearchCV with the following hyperparameter values

- number of components (PCA): [2, 5, 10, 15, 20]
- number of neighbors (KNN): [1, 5, 10, 15, 20, 50]

For +/-, the best model had a mean coefficient of determination of 0.37 during cross-validation and was using 10 components and 20 neighbors.

For P+/-, the best model had a mean coefficient of determination of 0.31 during cross-validation and was using 10 components and 20 neighbors.

For the PIM statistic, I have used a Negative Binomial regression model, and the parameter alpha was estimated using auxiliary OLS regression without a constant described here. The model had a mean coefficient of determination of 0.16 during cross-validation.

Feature Encoding

There were only two features that required to be encoded in this dataset: League and Position. Since each had only three different values (League: "QMJHL", "OHL", "WHL"; Position: "LW", "C", "RW") they were one-hot encoded. This increased the number of dimensions by four, which is few compared to the overall number of dimensions. Hence, doing so will not affect the curse of dimensionality that much. Also, since the features were nominal, no ordinality was lost in the encoding.

Outliers

Some drafted players have much higher/lower values in some statistics than the average either because they have outstanding skills compared to the others or because NHL teams see potential in a player who doesn't have super high stats. In addition, the non-missing numerical values looked all feasible. For the analysis, I have decided to use a RobustScaler to handle the univariate outliers. It would be interesting in the future to look at multivariate outliers using e.g. DB-Scan for a more complete analysis.

Feature Scaling

Some models and dimensionality reduction techniques require scaled data before being trained or applied on it. To pick the appropriate scaling strategy, I've decided to look at whether the feature was approximately normally distributed or not. I defined a feature to be normally distributed if its distribution on the training set had a skewness between -1 and 1 and a kurtosis between 2 and 4. Every approximately normally distributed feature has been scaled using StandardScaler and the other features have been scaled using RobustScaler. Note that this was implemented automatically, i.e. without having to look at the distribution of each feature.

Removing Highly Correlated Features

Features that are highly correlated do not bring much new information and increase the dimensionality. Since the numerical variables were all ratio variables, I have used Pearson correlation coefficient. Figure 1 shows the correlation heatmap of the numerical features

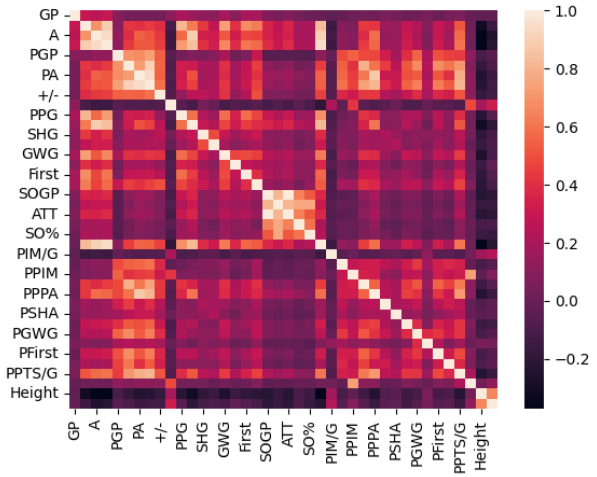


Figure 1: Heatmap of the Pearson Correlation Coefficient Matrix

The only two features that were highly correlated (absolute threshold of 0.95) were SOGP ("Shootout Games Played") and ATT ("Shootout Attempts"). The SOGP feature describes how often a shootout was necessary in a regular season game because the game was tied after three periods and nobody scored during overtime. This statistic is the same for all players of a given team in a given year. On the other hand, the ATT feature describes how often a player has been chosen to do the shootout. This is more informative since teams usually send their players with the best hands or shot to do the shootout. Therefore, I have decided to remove the SOGP feature and keep the ATT feature.

For completeness, these two features were also the only highly correlated ones when using Kendall's rank correlation. However, the PPTS and PPA features as well as the SO% and SO/G statistics were highly correlated when using Spearman's rank correlation. For this analysis, I have decided to not consider the latter since Kendall has been found to be more reliable in general and the results with Kendall are the same as with Pearson.

I didn't check if the two nominal features (League and Position) provided almost the same information as some of the numerical ones, but I doubt from personal knowledge that an absolute correlation of 0.95+ will be reached. On another note, I know from domain knowledge that the junior league and the player's position do not convey duplicate information as players from all positions are drafted from the three junior leagues.

Sampling

This dataset suffers from class imbalance. Table 1 shows the distribution of every class on the training dataset

Category	Count
Not_In_NHL_AHL	174
Other	97
AHL [0.2, 0.4)	75
AHL [0.4, 0.6)	64
NHL [0.2, 0.4)	57
AHL [0.0, 0.2)	48
NHL [0.4, 0.6)	46
AHL [0.6, 0.8)	44
NHL [0.6, 0.8)	35
NHL [0.0, 0.2)	31
AHL [0.8, 1.0)	26
NHL [0.8, 1.0)	15
NHL [1.0, ...]	10
AHL [1.0, ...]	9

Table 1: Class Distribution on the Training Set

To counter the class imbalance effects, I have either used class weights inversely proportional to the number of samples from the class in the training data or SMOTE.

Performance Metric

This is a classification problem where the classes are ordinal. Indeed, if we remove the "Other" class, the classes show how good a player is: the NHL is better than the AHL, and points per game is a ratio variable. Thus, the classic performance metrics for classification like accuracy, precision, recall, f1-score, etc. are not well-suited to measure the models' performance. To this end, I have decided to use the

Weighted Cohen's Kappa as a performance metric. The traditional Cohen's Kappa is a statistic that measures the agreement between two classifications. The weighted version accounts for how far two predictions are on the ordinal scale and so is a good performance metric for ordinal classification problems. I have chosen quadratic weighting because, in my opinion, predicting a player's success much further than the truth would have more than linear consequences.

Machine Learning Models

In this section, I will describe the models that were trained and will compare their mean performance during cross-validation using Weighted Cohen's Kappa.

The models consist of the appropriate pre-processing techniques, a dimensionality reduction technique, and a model type. The pre-processing techniques used depend on what the DR technique and the model require (e.g. Random Forest does not require feature scaling, but PCA does).

To reduce the dimensionality, I have used the following techniques: PCA, LDA, and ReBATE (ReliefF, SURF, SURF*, MultiSURF, MultiSURF*).

To do the hyperparameter tuning, I have used GridSearchCV. For PCA and LDA, I have tried the following number of components:

- number of components (PCA): [2, 5, 10]

For ReBATE, I have tried the following hyperparameters:

- number of features to select: [5, 10, 15, 20]
- number of neighbors (ReliefF): [5, 10, 15]

Softmax Regression

The baseline for the other models will be the softmax regression models. For every model, I have tried the following regularization techniques:

- penalty: [None, L1, L2]

Tables 2 and 3 show the mean performances during cross-validation of the best softmax regression models with class weights and SMOTE respectively found using GridSearchCV:

DR Technique	Mean Performance
None	-0.003
PCA	0.027
LDA	0.072
ReliefF	0.042
SURF	0.082
SURF*	0.041
MultiSURF	0.047
MultiSURF*	0.042

Table 2: Mean Performances of the Best Softmax Regression Models with Class Weights

The best softmax regression model was using SURF with 15 selected features, class weights, and L2-regularization. It had a mean performance of 0.082 during cross-validation.

DR Technique	Mean Performance
None	0.047
PCA	0.021
LDA	0.076
ReliefF	0.038
SURF	0.050
SURF*	0.044
MultiSURF	0.045
MultiSURF*	0.039

Table 3: Mean Performances of the Best Softmax Regression Models with SMOTE

Decision Tree

Another set of baseline models are decision trees. For every tree, I have tried the following hyperparameters:

- maximum tree depth: [3, 5, 7]
- minimum number of samples to split a node: [5, 10, 15]

and for the models that weren't using ReBATE algorithms, I have also tried the following hyperparameters:

- split criterion: [gini, entropy, log-loss]
- number of features selected to find the split: [sqrt, log2]
- maximum depth (in addition): [1, 9]
- minimum number of samples to split a node (in addition): [20]

For the models that were using a ReBATE algorithm, I used a gini split criterion and a number of features to find the best split equal to the square root (sqrt) of the total number of features.

Tables 4 and 5 show the mean performances during cross-validation of the best decision tree models with class weights and SMOTE respectively found using GridSearchCV:

DR Technique	Mean Performance
None	0.075
PCA	0.063
LDA	0.070
ReliefF	0.074
SURF	0.047
SURF*	0.034
MultiSURF	0.032
MultiSURF*	0.042

Table 4: Mean Performances of the Best Decision Tree Models with Class Weights

The best decision tree model didn't use a dimensionality reduction technique but was using SMOTE, a gini split criterion, a max depth of 9, 10 minimum samples to split a node, and a number of features to find the best split equal to log2 of the total number of features. It had a mean performance of 0.117 during cross-validation. Compared to the best softmax regression model, there's a slight improvement.

DR Technique	Mean Performance
None	0.117
PCA	0.110
LDA	0.102
ReliefF	0.107
SURF	0.058
SURF*	0.060
MultiSURF	0.068
MultiSURF*	0.071

Table 5: Mean Performances of the Best Decision Tree Models with SMOTE

Random Forest

Next I wanted to see if bagging could give us better results through random forests. For every model, the random forest consisted of 100 trees and I tried the following hyperparameters:

- split criterion: [gini, entropy, log-loss]
- number of features selected to find the split: [sqrt, log2]
- maximum depth: [1, 3, 5, 7, 9]
- minimum number of samples to split a node: [5, 10, 15, 20]

Tables 6 and 7 show the mean performances during cross-validation of the best random forest models with class weights and SMOTE respectively found using GridSearchCV:

DR Technique	Mean Performance
None	0.076
PCA	0.087
LDA	0.084

Table 6: Mean Performances of the Best Random Forest Models with Class Weights

DR Technique	Mean Performance
None	0.092
PCA	0.063
LDA	0.084

Table 7: Mean Performances of the Best Random Forest Models with SMOTE

The best random forest model was not using any dimensionality reduction technique but was using SMOTE, an entropy split criterion, a max depth of 9, 10 minimum samples to split a node, and a number of features to find the best split equal to the square root (sqrt) of the total number of features. It had a mean performance of 0.092 during cross-validation. Although it has a better performance than the best softmax regression model, it doesn't beat the simplier decision tree model.

XGBoost

Then I wanted to compare the performance of bagging models with the performance boosting models using XGBoost. For every model, I have tried the following hyperparameters:

- maximum depth: [1, 3]
- learning rate: [0.1, 0.3, 0.5, 0.7, 0.9]

I have also tried L1 and L2-regularization independently with the following regularization parameter values

- alpha (L1): [0.1, 1, 2]
- lambda (L2): [0.1, 1, 2]

Table 8 shows the mean performances during cross-validation of the best XGBoost models with class weights found using GridSearchCV:

DR and Regularization Techniques	Mean Performance
None, None	0.046
None, L1-Reg	0.043
None, L2-Reg	0.038
PCA, None	0.074
PCA, L1-Reg	0.096
PCA, L2-Reg	0.092
LDA, None	0.071
LDA, L1-Reg	0.056
LDA, L2-Reg	0.076

Table 8: Mean Performances of the XGBoost models

The best XGBoost model was using PCA with 2 components, a max depth of 3, a learning rate of 0.7, and L1-regularization with a regularization parameter of value 0.1. It had a mean performance of 0.096 during cross-validation. As for random forests, this model is not better than the best decision tree model but still achieves a better performance than the baseline softmax regression models.

Neural Networks

Finally, I have decided to take a look at Neural Networks.

Early Stopping All models have been trained using early stopping regularization. To determine the number of epochs to train the models, I have trained a simple neural network with each other regularization technique (mentioned below) on a portion of the training data and used the other portion as validation. Then, I plotted the loss over the number of epochs and recorded when the simple model started to overfit.

I looked at three other regularization techniques independently: L1-regularization, L2-regularization, and dropout. Figure 2 shows the plot of the loss over epochs of the simple neural network using L2-regularization.

As we can see, the validation loss oscillates after 25 epochs while the training loss diminishes, suggesting overfitting. Therefore, neural networks with L2-regularization will be trained for 25 epochs. Similarly, I have found that models without regularization (none of L1, L2, dropout) should be trained for 10 epochs, 75 epochs for models with L1-regularization, and 15 epochs for models with dropout.

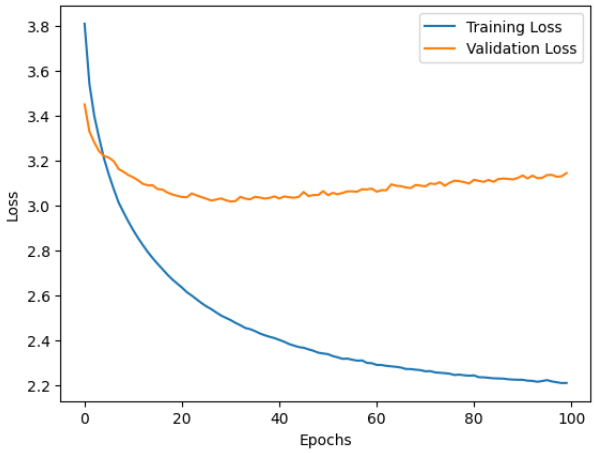


Figure 2: Training and Validation Loss over Epochs with L2-Regularization

Hyperparameter Tuning All models have been trained using an Adam optimizer. I tried the following hyperparameters for every neural network:

- nodes per hidden layer: [[]. [64], [64, 32]]
- learning rate: [0.0001, 0.001, 0.01]
- weight initialization technique: [glorot uniform, glorot normal, ones]
- β_1: [0.8, 0.9]
- β_2: [0.99, 0.999]
- batch sizes: [16, 32]

Moreover, I have set the regularization parameter of L1 and L2-regularization at 0.01 and I have used a 50% dropout.

Results Table 9 shows the mean performances during cross-validation of the best models with class weights found using GridSearchCV:

Regularization Techniques	Mean Performance
None	0.097
L1-Reg	0.087
L2-Reg	0.115
Dropout	0.104

Table 9: Mean Performances of the Neural Network models

The best neural network model was using L2-regularization, had a first hidden layer of 64 nodes and a second hidden layer of 32 nodes, the weights were initialized to all ones, and the optimizer was using a batch size of 32, had a β_1 parameter of 0.8, a β_2 parameter of 0.999, and a learning rate of 0.01. It had a mean performance of 0.115 on the validation set.

Performance of the Best Model on the Testing Set

The final step is to measure the performance of the best model on the testing set. From above, the best model is the decision tree using SMOTE, a gini split criterion, a max depth of 9, 10 minimum samples to split a node, and a number of features to find the best split equal to log2 of the total number of features. Appendix 1 shows the whole pipeline, especially how the raw dataset was processed before being fed to the decision tree.

The model had a performance of 0.129 on the testing set. This is slightly better than the mean performance during cross-validation, yet still not great enough to be reliably used in practice.

Discussion

The baseline models set the bar very low in terms of cross-validation mean performance, with a best mean performance of 0.117. However, more complex models were not able to detect better patterns in the data. In the end, we must conclude that baseline models were good enough on this dataset.

Using SMOTE over class weights have had mitigated effects overall on the cross-validation mean performance. Nonetheless, the best models were often using SMOTE rather than class weights. Future related work should not disregard the use of SMOTE in their models.

Models with ReBATE algorithms have been performing worse than models using other dimensionality reduction techniques or none most of the time. Future related work might not consider using ReBATE algorithms in their model.

The performance of the best model on the testing set shows that there is still a way to go to have reliable models to help NHL teams to rank the draft-eligible junior hockey players. This project can serve as a baseline in terms of performance for future work.

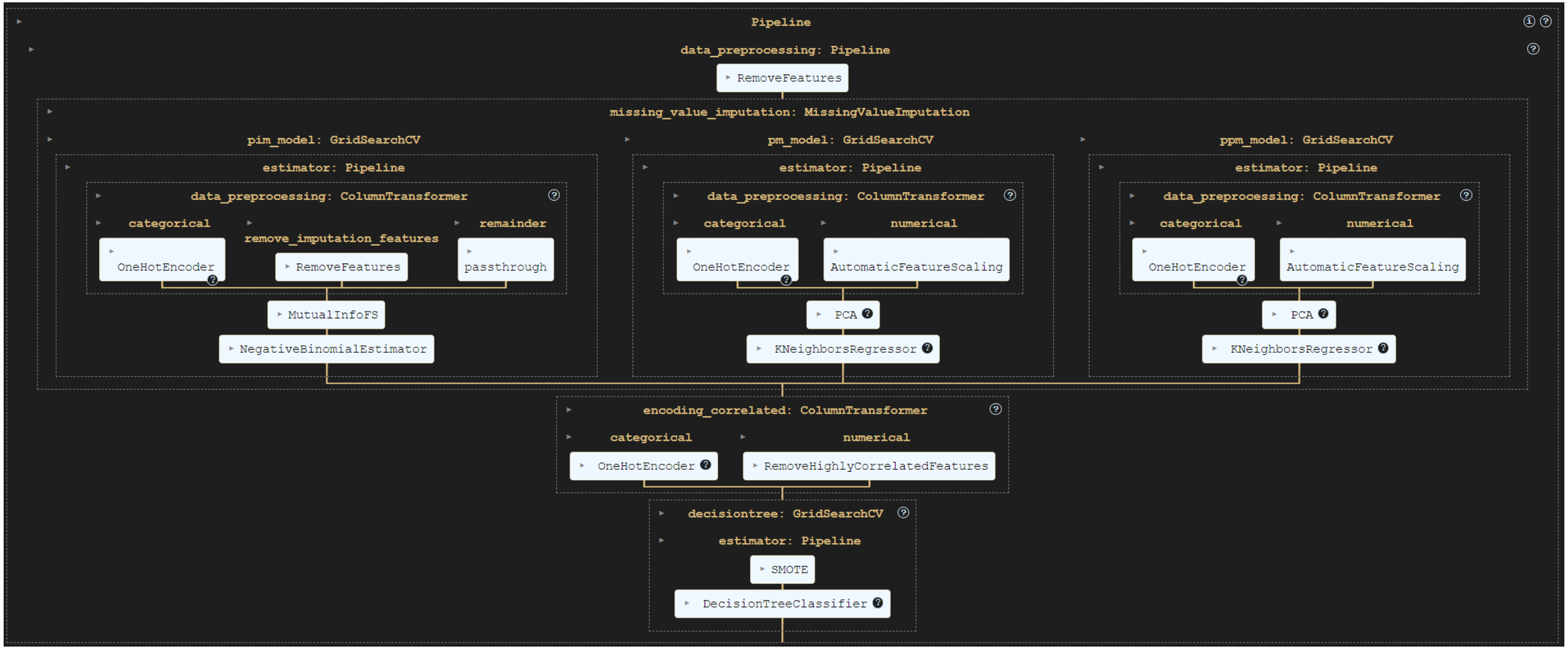
Throughout the report, I mentioned areas where future work could be done. First, the models were trained using only the junior data of the regular season and playoffs the same year as the draft. One could check whether considering e.g. the last two seasons instead would yield better results. Hockey statistics of when the players were younger can perhaps bring insights about how good they will be in the future. Second, features with a high missing value ratio maybe retained valuable information still and techniques like e.g. model-based imputation can be used to fill the missing values. Finally, different models or models with different hyperparameters be trained to see if they reach a better performance.

Personally, I think the results reflect the need to evaluate hockey players outside of hockey statistics as well. For example, players with certain skating abilities, hockey IQs, etc. might have better chances of succeeding at the professional level. Although we can argue that players' statistics are not independent of those skills, the data available online doesn't include a metric for those. Nonetheless, I believe that work can be done with e.g. scout reports to have either numeric values or ordinal categories for those skills and that those new features will play a valuable role in the development of machine learning models.

Conclusion

This project was a first step in introducing machine learning models to the world of NHL Entry Drafts. The best model had a Weighted Cohen's Kappa score of 0.129 on the testing set. Although this isn't a level of performance worthy of being used by NHL teams, future work can build on top of this project to perhaps provide valuable insights to NHL teams.

Appendix 1


Pipeline
data_preprocessing: Pipeline
RemoveFeatures
missing_value_imputation: MissingValueImputation
pim_model: GridSearchCV
estimator: Pipeline
data_preprocessing: ColumnTransformer
categorical
remove_imputation_features
remainder
OneHotEncoder
RemoveFeatures
passthrough
MutualInfoFS
NegativeBinomialEstimator
pm_model: GridSearchCV
estimator: Pipeline
data_preprocessing: ColumnTransformer
categorical
numerical
OneHotEncoder
AutomaticFeatureScaling
PCA
KNeighborsRegressor
ppm_model: GridSearchCV
estimator: Pipeline
data_preprocessing: ColumnTransformer
categorical
numerical
OneHotEncoder
AutomaticFeatureScaling
PCA
KNeighborsRegressor
encoding_correlated: ColumnTransformer
categorical
numerical
OneHotEncoder
RemoveHighlyCorrelatedFeatures
decisiontree: GridSearchCV
estimator: Pipeline
SMOTE
DecisionTreeClassifier