An Exploration of the Analysis of Ice Hockey Statistics in the National Hockey League (NHL) and Strategy Recommendation

UNIVERSITY OF
LINCOLN

Sam Davey
17684258

17684258@students.lincoln.ac.uk

School of Computer Science

College of Science

University of Lincoln

Word Count: 9026
Submitted in partial fulfilment of the requirements for the
Degree of BSc(Hons) Computer Science

Supervisor: Dr. Kamaran Fathulla

May 2022

Acknowledgements

Firstly, I want to thank my supervisor, Kamaran Fathulla, for his guidance and encouragement throughout the project. I would also like to thank my family for their emotional support, motivation and for introducing me to as well as creating my love for ice hockey, specifically Martin and Nigel Davey. Finally, I would like to thank James Luscombe, Michael Murrin, Daniel Butcher and Michael Apim for their advice and support throughout the project.

Abstract

Data analytics is now a common part of the professional sports industry is starting to utilize this beneficial technology for both viewers as well as, strategically, for coaches to make observations and game plans out of the conclusions drawn from analytics. This project will outline the research undertaken as well as the steps to create an artefact which produces data analysis of ice hockey statistics and trains a machine learning model to predict future statistics.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

Ice hockey is a game that is played with 12 players, 5 players are attacking and defending with 1 player in goal for each team. Common statistics that are tracked from the game are: Goals, shots, assists and time on ice (how long a player plays each game) line ups change every couple of minutes to continuously rest the players. In order for the line ups to change, coaches create lines which contain 5 players which can swap with the players on the ice when needed, usually at a timed interval.

Statistics gathering is commonplace within sports, however the analysis of these statistics is relatively new, beginning towards the end of the 20th century, with ice hockey beginning to use the technology in the 21st century (Physical Education and Sport Pedagogy, 2022). This is, in large part, due to the assistance of computers which can perform quick calculations on large datasets (recorded statistics) to find average values, perform data visualisation, to aid understanding and to make judgements on the data, as well as to find positive or negative correlations. This use of data analytics has enabled sports teams to take a scientific and mathematics-based approach to strategy as opposed to the traditional method of relying on coaches' previous experience which can be unreliable when compared to data analytics which can show clear patterns that can be used to improve coaching of players or finding areas of weakness within opposing teams such as where players are more likely to score a goal on the ice against a certain goalkeeper as well as finding weak shooting points for players.

Machine learning within data analytics, particularly in sports, has also grown in prominence due to its use in injury prediction (Sports Biomechanics, 2020). This new technology is creating new opportunities for prediction as sports prediction can aid coaches in informing strategies, selecting players for their team as well as finding methods of play that ensure the best outcomes for their team at the end of the season. This project endeavours to create an artefact which performs analysis and visualisation on player statistics from the National Hockey League which include graphical representations, machine learning and player ranking while discovering how useful these tools are to stakeholders such as: NHL coaches to fans of the sport that take part in fantasy leagues which require analysis tools to make decisions on player drafting and trades. Due to the variety of stakeholders the artefact needs to be accessible to all, easy to use with easy-to-understand outputs and visualisations. Research will be undertaken which looks for the viability of these objectives as well as the best practices for achieving these goals.

Objectives

The objectives of the project are:

- Research analytics within hockey and similar projects
- Design an application which displays statistics for a team with data visualisation
- Research machine learning and its uses in ice hockey analysis
- Train a machine learning model using player statistics

1.2 Referencing

Physical Education and Sport Pedagogy. (2022). *The validity and reliability of a performance assessment procedure in ice hockey*. [online] Available at: https://www.tandfonline.com/doi/full/10.1080/17408980701444718?casa_token=0fGVNEkTZYsAAAAA%3ATuQknhmmotRqpXbxXipFMx1ffoPQ7OHYHkaJ79yDNcr7-iSjFx4tQZKot4JOOwsnvfgHK7VUSjU [Accessed 17 May 2022].

Sports Biomechanics. (2020). *Machine learning in sports science: challenges and opportunities*. [online] Available at: https://www.tandfonline.com/doi/full/10.1080/14763141.2021.1910334 [Accessed 17 May 2022].

Chapter 2

Literature Review

2.1 Background

The use of machine learning for strategy review is uncommon within the NHL, this is due to the complicated nature of ice hockey as many different factors influence the outcome of a game which results in a lower amount of development of data analysis in ice hockey. This is different from sports such as baseball as for a run to be scored in baseball a player has to reach home plate which is a direct result of the ball being hit within the boundaries and the player not being thrown out or being caught out, because of this statistics such as batting averages and on base percentages can be used to predict outcomes of games more accurately than statistics from football and ice hockey as base-ball is a slower and more simple game, whereas ice hockey is faster paced with many more moving pieces. General managers of sports teams have discovered the benefit of data analytics when it comes to managing their team, previously management had to rely on their own experiences and experiences of scouts to attempt to make judgement decisions for their teams, however this approach to managing their teams was filled with bias and prejudice which as a result made some players over valued while other players were overlooked. This was revealed when saber metrics was introduced to base-ball, a famous example being the Oakland Athletics' win streak in 2002, the key to the success of the Oakland Athletics was by finding players that would get on base on average more than other players by using an evidence based approach rather than coach and spectator's perceptions (Cullen, Myer and Latessa, 2009)

regardless of the outcome after that, as a result players that had been overlooked previously became big players for the team and the Oakland Athletics began winning more than teams with larger budgets and players that had been perceived as being better. Data analytics allows sports analysts to find trends that might not be recognized by just watching the games, in baseball the previously used example of the on-base percentage is valuable, as is the use of batting averages, however with further investigation information such as the speed of the baseball when a player is most likely to hit can be discovered as well as how far the ball goes on average when hit at that speed by each player, from this a manager can decide when the best time to put a player on the field would be as they now know which players deliver the best outcomes under specific circumstances, some-thing that would have been missed had the data not been recorded and checked for particular occurrences. (Hamilton et al., 2014) used machine learning as a data analytics tool to predict pitch information such as speed and angle, this is beneficial to managers who can use it for batting training, planning line-ups as well as assessing their own pitchers and evaluating who should play in future games depending on their skill set. As previously mentioned, hockey is vastly more complicated due to the speed and many different moving parts, instead of one team getting a chance to attack while the other defends until the end of an inning both teams have the ability to defend or attack based on who has the puck, while this is happening defending players will try and shield at-tacking players while others attempt to retrieve the puck from the other team, attacking players will try to find routes to score. Although analytics are more complicated in hockey they are still useful, the majority of research that has been conducted has been on injuries within hockey, the game is violent in nature as physical “hits” are encouraged (slamming players) and fights are often allowed to take their course, because of this players often sustain injuries, some of these injuries cause players to miss games or entire seasons, to avoid this researchers have been using machine learning to identify when and how injuries are likely to occur, this can be beneficial to the management of hockey teams as they can coach players on how to avoid these injuries, particularly the players that are more prone to certain injuries. This is another example of where coaches and spectators may not be able to predict certain events and would have to apply their best guess which usually ends up being costly for the team, the use of data analytics in this scenario is not only safer for players but also better for the team as they will not lose key players, while experience from coaches is still valuable human perception can be erroneous and patterns may not be found due to previous games being forgotten and recollection not being as clear after games, this makes data analytics important as ac-curate data can be analysed to find patterns which would not have been found otherwise which can give financial and tactical advantages.

Attitudes towards data analytics are changing, while some are still hesitant to the benefits (Online MBA, 2019) major sports teams are including analytics when making trades, planning strategies as well as evaluating previous performance. Because of these developments more research is available than in previous years, while analytics in hockey is a recent development there have been groups which have used machine learning in regard to predicting the outcome of games played in the NHL such as Gu et al., 2019. These researchers have applied their own rankings to players based on previous performance data, by applying a ranking that is comparative to other players less information needs to be passed into the machine learning models while still delivering the same outcome as giving the models all the previous performance data. Giving all the previous performance data would be very hardware intensive and would take a long time to process as a result, by creating a player ranking system, the process is stream-lined which makes it quicker without sacrificing accuracy, the training part of machine learning will be able to find the similarities between player scores and successful games which contributes to the accuracy of the models. Player rankings will be utilized in the project to make the program more efficient while also aiding the player comparison aspect as the strengths of players will have already been considered and a ranking will have been calculated so an initial decision about which player has the higher ranking will show which player is better overall, while individual statistics will show which player is better in individual areas. The previously mentioned research paper used multiple machine learning models in an effort to get the most accurate predictions with an emphasis on neural networks. Liu and Schulte, 2018also took a neural network ma-chine learning approach however instead of predicting the outcome of games they used machine learning for the evaluation of players, this is an efficient process as large datasets of previous player performance were used to train the model resulting in accurate appraisals of players' overall performances. Bunker and Susnjak, 2019took a mixed approach to machine learning as they used a combination of neural networks, decision tree algorithms as well as rule set algorithms such as K-nearest-Neighbours, by using algorithms that take different approaches to machine learning the algorithms that produce the most

accurate results can be identified and used for any further analysis. This is a direct benefit over only using neural networks to make predictions. The paper also discovered that while in other sports neural networks delivered accurate predictions, for ice hockey the alternative algorithms were superior in terms of accuracy, this sup-ports the need for testing multiple machine learning algorithms in order to find the best fit as accuracy is the most important factor in predictions. A famous and commonly used method for ranking teams is the ELO ranking method, which has been used previously for ice hockey (Hockey Analytics, 2016). The ELO ranking method accounts for home ice advantage (teams playing at their home arena) as well as teams which are performing at a higher level than others, the ranking method is adaptable depending on which teams win and which teams lose, over time an accurate ranking of teams is developed. However, a negative of the ELO ranking method is that it is only applicable for teams and not for players, therefore it ignores the contributions of individual players which takes away its value when trying to analyse the data for reasons of a team's success or shortcomings. While machine learning models are important to the prediction of NHL outcomes, the evaluation of players is also crucial to getting accurate results for predictions as well as creating an accurate metric for comparing players and their individual abilities. Schulte., et al., 2017 uses context-based valuation of player's actions, this includes location for example which takes account of home ice advantage due to players feeling more confident at their home arenas. The researchers make use of the Markov Game formalism which finds the likely consequences of actions to find how hockey is actually played as opposed to optimal play in order to value players' actions fairly, an example being that goals scored to break a tie is more valuable than a goal scored when a team already has the lead, as a result players who bring significant value to a team will be ranked higher for bringing needed benefit in difficult situations such as a needed tie breaker. Adaptive player rankings are valuable for player evaluation as they are an accurate representation of how a player is performing, because of this they can be used as part of the data which is used to train machine learning models in order to give the models a greater understanding of how statistically better players can result in

positive outcomes for a team by finding correlations between the two. Outside of private research, Amazon has started offering data analytics for the NHL starting with the 2021 playoffs (NHL, 2021) with the goal being to give viewers a greater understanding of the game. However, the analysis being offered such as save and shot analysis will be a benefit to coaches as they will be able to identify where goal tenders' weaknesses are in terms of goal position and where attackers are at their strongest such as which side they are shooting from and distance from the goal. The data generated by Amazon's analysis can be useful for machine learning projects regarding team and player performance as they provide direct figures for how a team is either succeeding or failing which can bring about greater accuracy for machine learning models attempting predictions on NHL game data.

Chapter 3

Methodology

3.1 Project Management

Various tools were utilised throughout development of the artefact, starting with the project proposal, to ensure the project followed a structure that enabled tasks to be completed in a timely manner allowing the artefact to reach full completion by the end of the Software Development Life Cycle.

The first tool used was GitHub Projects, this is a simple application that allows tasks to be set with the project manager creating columns which signify different stages of development. Within this project there was a To-do column for stages of the project that had not been started yet, an in-progress column for parts of the project that were in the process of being completed and finally there was a completed column for stages that were finalised. Towards the end of development, a fourth column was added for

stages of development that had been created but subsequently removed due to a redundant need for them in the project. The use of GitHub Projects was largely successful and aided organisation of the project as well as motivation due to breaking down the development into smaller tasks and being able to acknowledge when tasks had been completed to a satisfactory standard. GitHub Projects also allowed for new tasks to be added to the development when tested warranted changes or when designs changed with the interface. GitHub Projects is perfectly suited to the Kanban methodology as Kanban's main focus is visualising workflow with tasks at different stages of development in order to limit the number of tasks that are in the work-in-progress phase by having one task being the main focus (Kirovska and Koceski, 2016). This was beneficial to a one-person team such as the one used for this project as tasks were not being juggled which gave each task the full attention required for it to be completed.

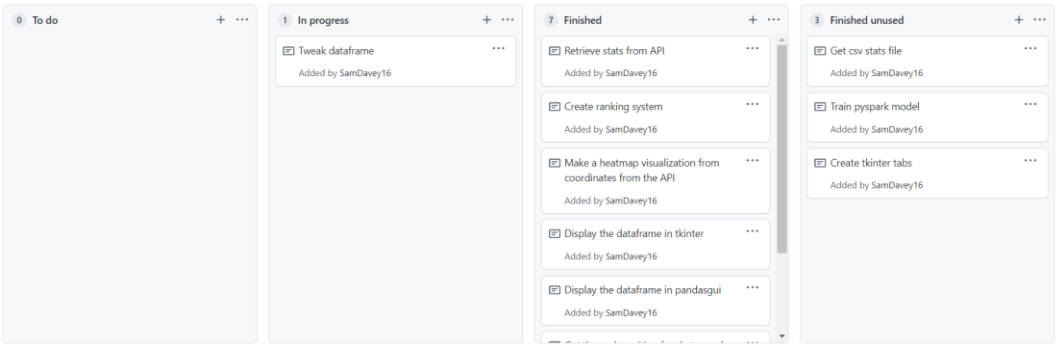

Figure 1 - GitHub Projects

A Gantt chart was developed during the project proposal to reference when in the development certain tasks would be undertaken. A Gantt chart is a visual display across a calendar that shows when events will take place between two dates, in this case a time frame for which a task was to be completed. The Gantt chart was useful in the initial planning stage of the project for the purpose of identifying tasks that would have to be undertaken for the project as well as giving timeframes and milestones. However, as the project progressed the use of the Gantt chart dwindled as challenges arose which resulted in certain deadlines being delayed or new

tasks being completed, due to this the reliance on the Gantt chart was switched to GitHub Projects as it was more suited to a project that had to be rearranged after testing without disrupting the workflow. Negatives of Gantt charts have been noted by Wilson (2003) who notes that large scale companies gave struggled to implement them due to the charts being abandoned as teams got later into production stages.

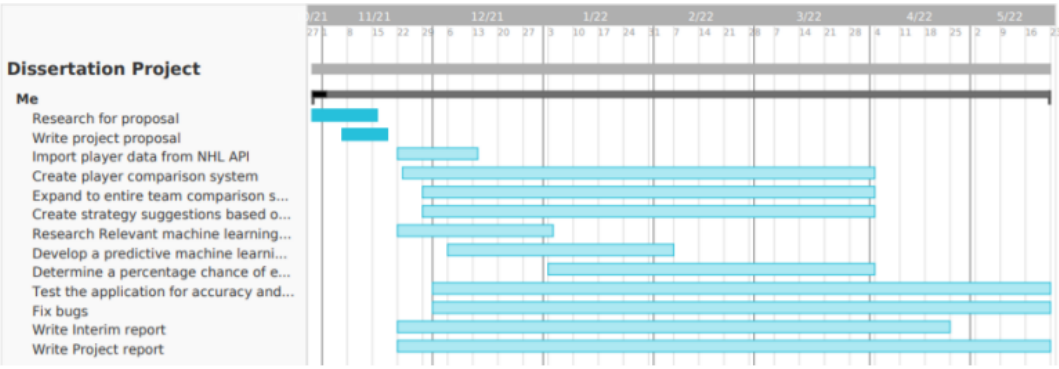

Figure 2 - Gantt chart

3.2 Software Development

The first methodology considered for the software development aspect of the project was Waterfall, due to the team working on the project only consisting of one person it would be easier to maintain an organised workflow that relied on one task being undertaken at a time. Each step of the project would also naturally flow on from the previous tasks as for example each subsequent step relies on the retrieval of the statistics and each step after storing the data relies on the data being shown meaning that in this project tasks needed to be completed in a sequential order. The other methodology that was considered and ultimately used was the Cleanroom methodology which focuses on continuous testing and development until the task reaches its requirement at which point it is maintained while other tasks are started. The methodology, in theory, should result in no defects being present at the end of development due to each individual part of the application having been maintained with errors fixed before deployment.

This is an ideal methodology for a project where the testing is self-reported as issues with the development can be fixed by the person doing the testing which allows for issues to be resolved quickly instead of a wide array of issues to be solved during testing at the end of the project, as alluded to by Mingwei (2020). Due to the potentially wide range of statistical experience the users could have, ease of use and ease of understanding is of paramount importance, therefore the constant testing and fixing offered by the Cleanroom methodology is beneficial to the project as large changes to the type of statistics and visualisations displayed, at the end of the project, would require a complete overhaul of the application as the display of statistics and visualisations is the artefact's primary function. This would jeopardise the project as towards the end there is usually only enough time for bug fixing rather than another entire development cycle to redesign the application in order to create an improved version that has visualisations and statistics that are more suited to the wide range of stakeholders. This was the deciding factor in using the Cleanroom methodology over Waterfall as time management was key and testing for this project was more efficient when it was performed routinely rather than in the final stages of development.

3.3 Toolsets and Machine Environments

As outlined in the Project Management section, only one toolset was required to outline the tasks to be completed using project management, which was GitHub Projects, due to GitHub being a web service, it is easily accessible on multiple platforms and therefore is not restricted to certain operating systems making project management a simple task that could be accessed regardless of where the project was being worked on.

For software development two IDEs (Integrated development environments) were used, initially the python IDE Thonny was used due to it being a solely python IDE which was the programming language chosen to create the artefact, as well as its library manager where python libraries

could be easily downloaded with all dependencies through a search on its built-in pypi index (an index of python libraries). Thonny was eventually abandoned as it failed to install the python library Pandasgui (which was an essential resource to the project) due to an error installing pyarrow. After installing Pandasgui using pip on the windows command line with no issues another IDE had to be found that used the python installation already available on the development PCs. Visual Studio Code was then chosen as it is a lightweight version of Visual Studio, another IDE which had more features that would not be needed which resulted in the lighter weight version being chosen. All python libraries were then installed from the command line and stored in the python installation files, where they could then be accessed by Visual Studio Code. The python libraries that were used are as follows:

- Pandas – Pandas was used for creating dataframes to store the statistics in which could later be used to be displayed to the user.
- Tkinter – Tkinter was used to create a graphical user interface, it is a built-in python library and is commonly used for making lightweight applications within python.
- Pandasgui – Pandasgui is a library built to display pandas dataframe in an interface as opposed to on the command line where data can be difficult for users to interpret.
- Matplotlib – Matplotlib is a library which is used for plotting data on graphs and plots. Matplotlib was used within the artefact to create a heatmap visualisation.
- Hockey_rink – Hockey rink is a library that works in conjunction to plot coordinates shot data on a visualisation of an NHL ice rink.
- Seaborn – Seaborn is another plotting tool which was also used in conjunction with Matplotlib and Hockey rink to create a heatmap visualisation.

- Requests – Requests is a built-in python library for parsing information from the internet. For this project the requests library was used to parse JSON data.

Python was chosen as the programming language due to the amount of data analysis and visualisation libraries that were available for the language that are also considered desirable data analysis tools (Stancin and Jovic, 2019) as well as the requests library which allows JSON to be parsed from APIs which can allow data to be retrieved, beneficial when using the official NHL API. Python is also a popular language which resulted in libraries that were well maintained which would prevent bugs in libraries from making the artefact unusable in the short term as well as future proofing the artefact.

Chapter 4

Design, Development and Evaluation

4.1 Software Development

Requirements elicitation, collection and analysis

The requirements for the project were quite simple as the goal of the application was to produce statistics that were easy for users to interpret as well as inform their choices on who they believed were the strongest and weakest players in an NHL team, due to this the number of requirements were low and the risk of not achieving these was also low. The requirements are as follows:

- Retrieve NHL game, team and player statistics from a source
- Display player statistics to the user
- Produce data visualisations
- Create a system to rank players and how they rank in the team for shots, goals and assists
- Have all outputs easy to understand to people of every level of understanding of ice hockey

Design

When considering the design, the requirements need to be taken into account as the design of the artefact is essentially the plan of how to create an application that matches the requirements. Identifying stakeholders is key in design as the artefact needs to be appropriate for the target audience. The stakeholders in the project are coaches of NHL teams as well as NHL fans that want to engage in fantasy leagues. Due to the wide range of experience with hockey within the stakeholders group the application has to be accessible to all, therefore the statistics output needs to be in a simple format that is easy to understand and the visualisations need to be free of any hockey-related jargon that may deter less experienced users and negatively impact the experience they have with the application. The application must retrieve statistics from a source, store them and be able to display these to a user for interpretation, the display of these statistics should be part of a user interface as opposed to being displayed on the command line. A machine learning model also had to be chosen and trained to discover how useful machine learning could be to representing statistics in the artefact. For this a decision tree classifier was chosen. A decision tree model works by having each node with a test with each "branch" being an outcome of that test, as with other models the training data is passed through first so the model can "learn" to detect (in this case) which team the statistics are describing through a series of decision trees and then the test data is passed in where it attempts to predict, based on its earlier training, which team the model is being fed without the full context the model had earlier (Song and Lu, 2015). The decision tree model was chosen due to the model asking a series of "questions" which train the model as all the statistic categories were going to remain the same, therefore making the "questions" the model had learned earlier useful as they could be used repeatedly, in theory, to obtain accurate predictions. To aid understanding of the statistics as well as create a visual aid to influence strategy creation, The artefact also needs to include data visualisation, this is a key part of data analysis as complex statistics can be reduced to graphs and charts that accurately reflect data and engage the users (Azzam et al., 2013) while aiding understanding, especially for comparisons. As previously mentioned the user interface was designed to be a central part of the application, all functions could be completed without the use of a graphical user interface, however, to reach the requirement of making the data easy to understand and interpret the application must also be easy to use which becomes more difficult with a command line application, especially for users who have little experience with command line applications. The graphical user interface was always intended to have a main menu page

which the user could interact with to choose which analysis they wanted to perform whether it be visualisations or which dataset they want to use. Parmar et al., (2018) states that the primary steps to data analysis are: gathering the data, checking the quality (sometimes known as normalisation if there are missing or repeated values) which is not required in this project due to the simplicity of the data and the only data being recorded being that of players who have played in NHL games, with the final step being statistical modelling for machine learning, the first and last steps were used throughout the project.

The initial design for the user interface is shown below. This design was ultimately abandoned due to issues with the user interface library Tkinter as will be noted in development section.

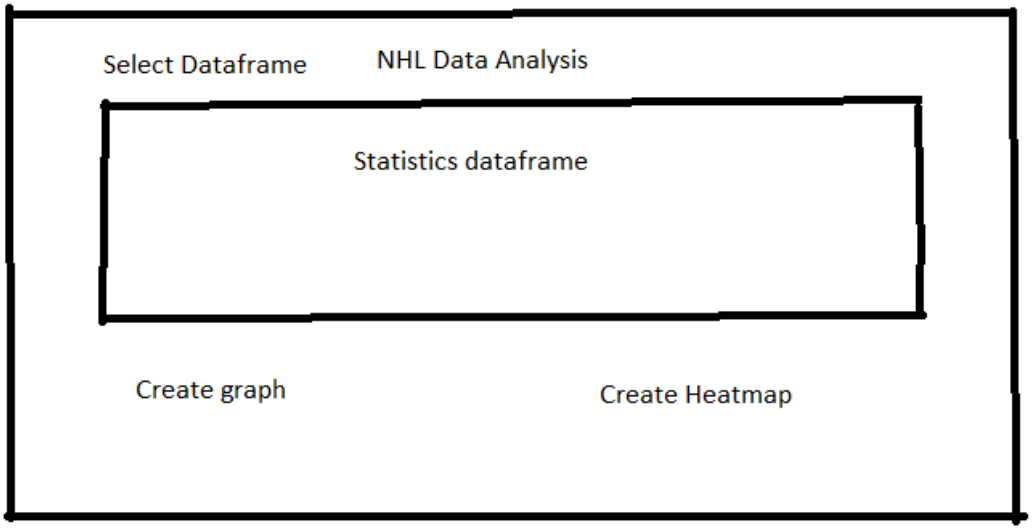

Figure 3 - Initial user interface design

Building the application

The first stage of development of the artefact was using the python library pyspark to train a machine learning model (the decision tree classifier described in the Design section) from a CSV file that contains the statistics of every active NHL player from every game of the past 10 years. Pyspark is built upon Apache Spark and it is a tool that is used for data analysis with a wide variety of tools such as dataframes which are used to store data and statistics which can be analysed using built-in pyspark functions or by user created functions using python. The dataset was from the website moneypuck.com which collects data directly from the NHL and splits them into season data, team data, skater data or goalie data. For the purpose of training the model the skaters file was used which was split into a 70% training set and 30% testing set, however this caused

problems immediately as the file contained too many classes of data for the model to be successfully trained and pyspark threw an error. Upon investigation it was revealed that this was due to trying to get the model to predict the player ID based on the statistics from the file, there was too many players as pyspark is limited to 100 individual values which the models can identify for prediction. To adequately test the model the prediction column was changed to teams as there are far fewer teams than players. To facilitate this change a string indexer was applied to the teams in the dataframe as machine learning models predict numerical values. Once the string indexer had been applied, the binary class classification evaluator threw an error this is because it can only evaluate a model when the index only has two values, to fix this the muti class classification evaluator was used instead to successfully evaluate the accuracy of the model. To test the accuracy the model is evaluated by its error rate, specificity and sensitivity. Unfortunately, the error rate was high and the model only had an accuracy of 62%. This low level of accuracy can arguably be attributed to the human element within hockey which can include varying response times due to different levels of fatigue and emotional state, varying levels of strength as well as player motivation. Another reason for the low accuracy can be that hockey is a fast-paced game with many players constantly moving with multiple events that can be taking place at any given time, this makes it difficult to reduce the game to simple numbers and statistics. This is different from games such as baseball and tennis that have events that follow each other with a ball going in the direction of a player who will either hit or miss the ball which in turn leads to another outcome, therefore how often a player hits the ball can be indicative of a player's skill and therefore can be used to predict future performance with reasonable levels of accuracy when compared to hockey where there are many reasons why a player may miss certain opportunities that may not be recorded in any form of numerical statistics. Ultimately the low accuracy led to the machine learning model not being included in the final application as its inclusion wasn't necessary and its presence impacted the accuracy of the data analysis due to all other statistics and visualisations being based on recorded statistics whereas the machine learning aspect was only a prediction and not a true or clear analysis of a player's performance which was the primary purpose of the application.

The second phase of development was to access data for the purpose of analysis. For this the NHL publish their own statistics to an official API for public access, accessing this data was quicker than using the CSV files from moneypuck.com as money puck's data was per game as opposed to offering a total of players' statistics, due to this the statistics would have to be added up by going through every game in the file for

the given season. This would be inefficient due to the large amount of processing time, making the API the better choice as a data source, due to it being official data from the NHL as well as the much quicker data retrieval time. The data was in the JSON format that required the requests python library in order to parse the data. The first task attempted was to create a heatmap, which was developed in its own file so that it could be merged with the rest of the application later. The heatmap shows where goals are most often scored from by a single player. Initial research was undertaken and the "hockey_rink" python library was found which is built on top of matplotlib and seaborn which are python libraries which are used to plot data on various graphs and plots. Hockey rink creates an image of a third of a hockey rink containing a goal with the ability to plot coordinates data that will display on the rink in the correct game location in the form of a heatmap. Firstly, a message is displayed to the console that lets the user know that the heatmap is loading as the display is not immediate. As previously mentioned the data was collected by looping through each game in the 2021-2022 NHL season, following this filtering the event data for when a goal was scored (labelled as "G" in the JSON output) by the player that the user enters during the input. The goal data is then appended to two lists, the first contains the coordinates for the x axis and the second contains the coordinates for the y axis. Once the for loop is complete, the two lists are passed into the "NHLRink" part of the hockey rink library via the hex bin and seaborn which plots the coordinate points on the rink. The hockey rink library then "draws" the rink and creates the heatmap which has been set to red with the darkest shade representing the locations where goals are scored from most frequently and the lighter shades showing where goals are scored from less frequently.

Once the heatmap had been created a general stats display had to be created to output not only the goal statistics of an individual player but all of the key stats for any player on the roster. The process of retrieving the statistics was very similar to the heatmap, however instead of looping through each game to calculate the required statistics the NHL API offers all the up-to-date information on every player from the current season, this information is stored in each player's API call rather than being stored in one place for the entire time. To access each player's information two functions had to be created which would retrieve each player's player ID or each goalie's ID from the roster information provided by the API. For this a for loop was created which loops through every value in the roster section (which is every player) and appends each ID to a list of IDs, the function ultimately returns the list of IDs. Once this function was completed another function had to be made to retrieve the statistics of each player and then produce an output. To store the

information the pandas python library was used which similar to pyspark has data analysis tools and can produce dataframes. Pandas was chosen as pyspark produces errors that are difficult to handle due to their lack of accurate descriptions of the problem whereas pandas was only required to produce a simple dataframe and has manageable errors. Once the pandas dataframe had been initialised with each column representing a certain piece of information or name of a type of statistic, a loop was made to loop through the ID list returned by the ID functions (get_player_ids or get_goalie_ids) the functions then call the API with each ID to access basic information such as names, goals, assists and their time on the ice. All of this information is added to a new row in the pandas dataframe. Once the loop completes each player has been added to the dataframe and it is ready to be displayed to the user. Initially the dataframe was displayed on the console, this resulted in the output being in order of the player ID rather than in order of desirable stats. To fix this a ranking system was added which was calculated by goals plus shots plus assists divided by 23 (the number of players allowed in the active roster), these statistics were chosen as they are the most desirable and a higher amount of them deems a player more skilled than those with a lower amount of each respective statistic.

At this stage in the development the user interface development was started to display the dataframe, as on the console output the dataframe had too many columns to display each row on one line making the dataframe hard to interpret due to statistics being shown in different rows and columns to the rest of the data on its row. Python supplies a built-in library for producing user interfaces called Tkinter, research was conducted to determine how a pandas dataframe could be displayed in a Tkinter window which resulted in using Treeview which is part of the Tkinter library. Treeview allows the dataframe to be shown in a Tkinter window with a scroll bar for navigation. Treeview was implemented without issues until additional items were added to the page. The initial item that was added was the goalie dataframe that was created at the same time as the get_goalie_data function, this function operates in the same way as get_player_data with the difference being that it takes goalkeeper IDs and gathers goalkeeper statistics such as powerplay saves, shots against and goals against and adds these statistics to a row in the goalkeeper dataframe. Similar to the get_player_data function, the output is the dataframe. When the goalkeeper dataframe was added to the Tkinter window only one dataframe would be displayed despite attempting to adjust the placement. To fix this tabs were implemented to give each dataframe a page, however Treeview was displayed over the tabs which prevented their use, the final attempt to solve this issue was to add buttons however when another dataframe was opened the

dataframe already being displayed would become blank and the other window would also contain a blank dataframe. Treeview was promptly abandoned as multiple dataframes being displayed became a requirement as well as implementing the ability to analyse an individual player's data. Further research revealed that a library "Pandasgui" could display a pandas dataframe as well as offer additional functionality such as the ability to select a column and display every row in order of the highest or lowest values from that column as well as the ability to create graphs from the statistics contained in the dataframes, giving the user the choice of what type of analysis they want to see. For example, a user can choose to create a bar chart of shots which visualises how many shots (shots on target) each player had taken in the season compared to other players on the roster. To make this interface accessible to the user, changes had to be made to the Tkinter window. As previously mentioned Treeview was removed, in Treeview's place buttons were added which gave users the option to analyse roster data, analyse individual player data or analyse goalie data. Each button was made from Tkinter's button function which executes a command upon a button being clicked. The code from the statistics functions were altered to show the pandas dataframe using the Pandasgui show function upon the event of the respective button being clicked. A common bug occurred when the Tkinter window was ran after the inclusion of the buttons with the first button automatically running its function without the button being clicked, upon researching the bug it was revealed that the function should be called using lambda to prevent the functions being automatically called.

Once the lambda functions had been incorporated to the buttons the individual player data function had to be created. This function named "individual_player_data" takes in a string user input which contains the name of a player on the roster (in this instance a player on the active Toronto Maple Leafs roster). Once a player has been named the function creates three lists to contain the goal data, shot data and assist data for each player on the roster to be used later to find where the named player ranks for each statistic in comparison to other players on the roster. The rest of the program operates in the same way as the get_player_data function except it filters out the rest of the players on the roster and only adds the chosen player's statistics to the dataframe. Despite the other players' data being ignored for the dataframe, their goal, shot and assist data is still appended to the respective lists. Three almost identical functions were created to find where the chosen player ranks in comparison to the other players on the roster, the sole difference being which statistic is being ranked. Each function uses a counter to loop through the list and counts how many players have a higher amount of the selected statistic. Once the chosen player's number of that statistic is

found the counter is stopped and the position in the list is displayed to the user as the rank in the team in which the player is positioned for the selected statistic. For example, Auston Matthews was ranked first for goals in the team, fifth for assists and first for shots (in the regular season, not including end of season playoffs) as each one of his respective statistics were in those locations in the ordered lists. Once the chosen player's statistics have been shown to the user, the chosen player's goal-scoring shot locations are shown using the heat map function, after the user has inspected the heat map the ranking of the chosen player's goals shots and assists are displayed in the console.

Testing

The application was largely self-tested in line with the original requirements set out for its development as the meeting of these requirements were clear to identify. Bug testing was common throughout the development with a test being ran with every addition to the python file api.py in order to address bugs in a time efficient manner as opposed to dealing with a large array of bugs after numerous different additions which would ultimately make the solution harder to identify, particularly when bugs are coming from code that was added during an earlier stage in development that was some time ago and therefore not in recent memory.

At the end of development when self-testing had ended three people agreed to test the application to identify if the requirements had been met. Participant one is a long-time ice hockey fan and had previous experience using websites that provide ice hockey statistics, largely for fantasy NHL leagues. Participant two had watched ice hockey before but had limited knowledge of current players. Participant three had never watched ice hockey. Participant one could provide useful feedback due to their previous experience with hockey statistics, being able to compare the artefact to already existing solutions. Participants two and three were valuable as their lack of experience with hockey allowed for an investigation into how easy the application and its data were to understand to ice hockey beginners, as per the requirement outlined earlier. Participant one noted that the visualisations were "useful and insightful" however "the app could look better, it looks a bit 1.0" referring to the basic Tkinter window as well as the artefact requiring different windows for Pandasgui, the heat map, the ranking outputs as well as the main menu. Participants two and three had similar opinions to each other with participant two stating "The stats screen is good, and the ranking helps" they also added that "a search function could be good so that you could switch between players" in reference to the analyse

roster window. Participant three noted that “It would be better if the ranking bit was in the same window as the rest”. Once the testing was complete no changes were made, largely due to the satisfaction with the statistics gathered and the visualisations which were the key requirements of the project. A drawback to implementing changes was that the primary issues were with the user interface which would of have required major changes to the artefact as the Pandasgui is used for the visualisations required when analysing the roster data and matplotlib (which is used by hockey rink) opens in another window to show the heat map. To make the recommended changes would have involved losing the visualisations which were essential to the artefact. The user interface was also a secondary priority as the focus was gathering the statistics, displaying them to the user and creating visualisations from the statistics to aid understanding and to influence decisions on strategy, an aim that was achieved by the end of development and confirmed through both methods of testing.

Operation

The current operation of the application is a stable version of what was set out in the requirements. The user is greeted with a home page that gives them the choice of three options: Analyse roster data, analyse individual player data or analyse goalie data. As discussed in development the application retrieves statistics from the NHL API which is then displayed to the user in the form a pandas dataframe, which to a regular user would seem like any other table that they may be used to such as a Microsoft Excel table. The user then has the choice to visualise any of the statistics using a variety of charts and graphs including bar and pie charts, they also have the choice of which axis to place the data columns on. If the user chooses to analyse individual player data they can give an input of a player name from the active Toronto Maple Leafs roster, the statistics of that player is then displayed to the user as well as a heatmap of where the player scores goals from and the frequency of scoring from these locations. Finally, the user is shown where the player ranks in terms of taking shots, scoring goals and providing assists in goals within the roster; this output is displayed on the command line due to issues with placing text and subsequently removing it within a Tkinter window. After the user has made a choice the primary Tkinter window is still running which allows them to go back and make another selection after they finish with their first choice. If the user wishes to end the application, they simply have to select the close tab at the top right of the screen, signified by a red X.

The operation of the application is displayed sequentially below:

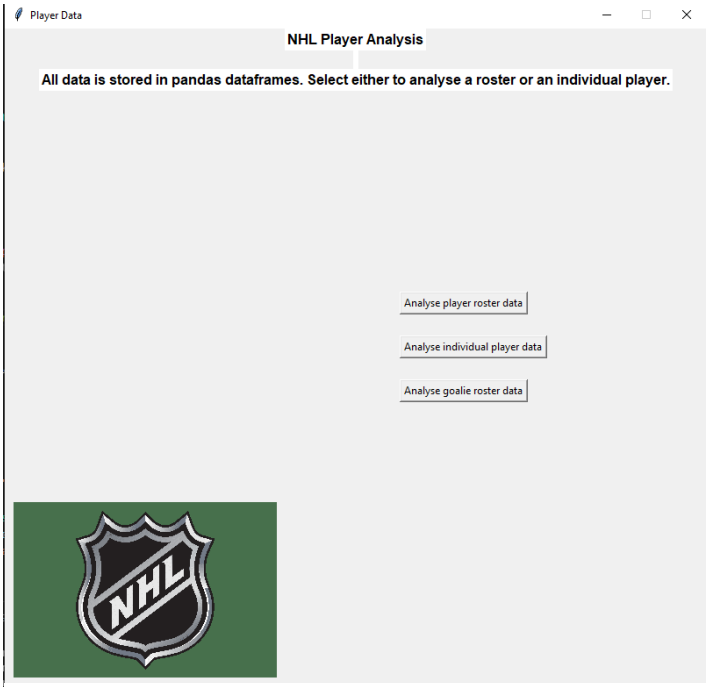

Figure 4 - Main menu

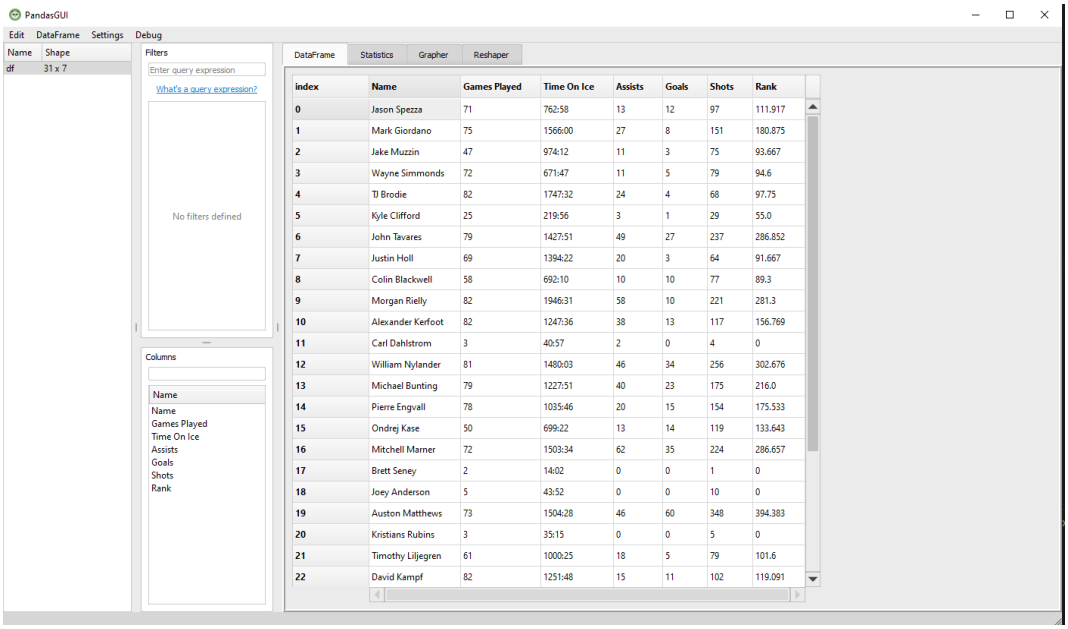

Figure 5 - Roster statistics and ranking

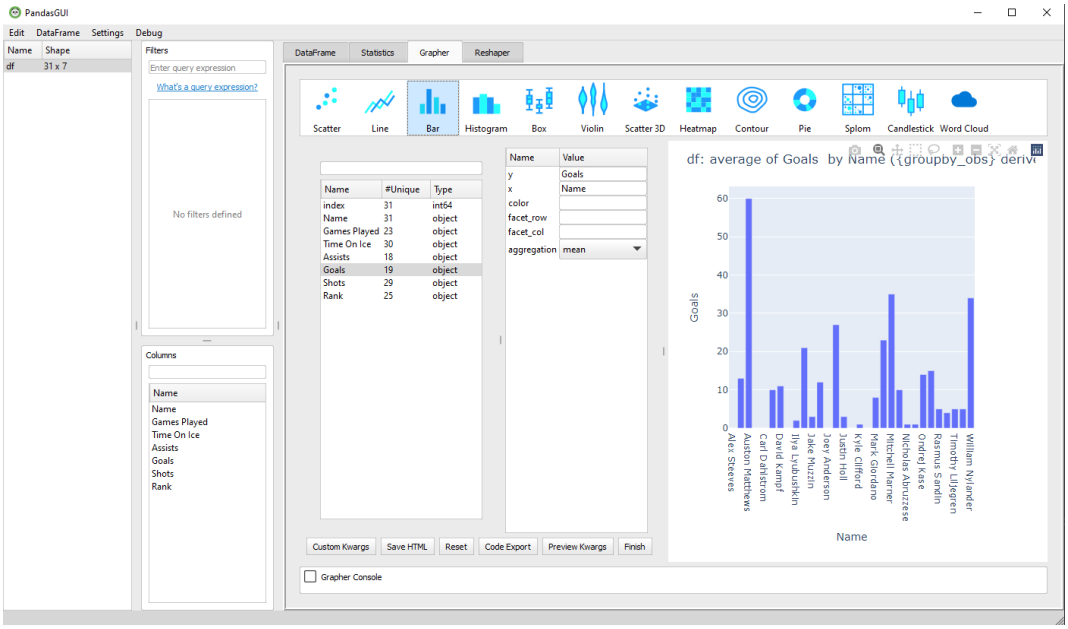

Figure 6 - Data visualisation

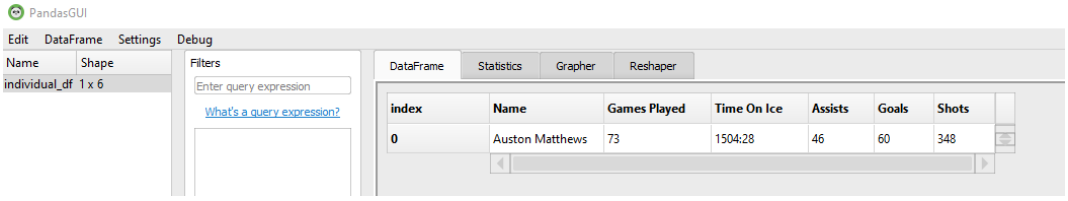

Figure 7 - Individual player statistics

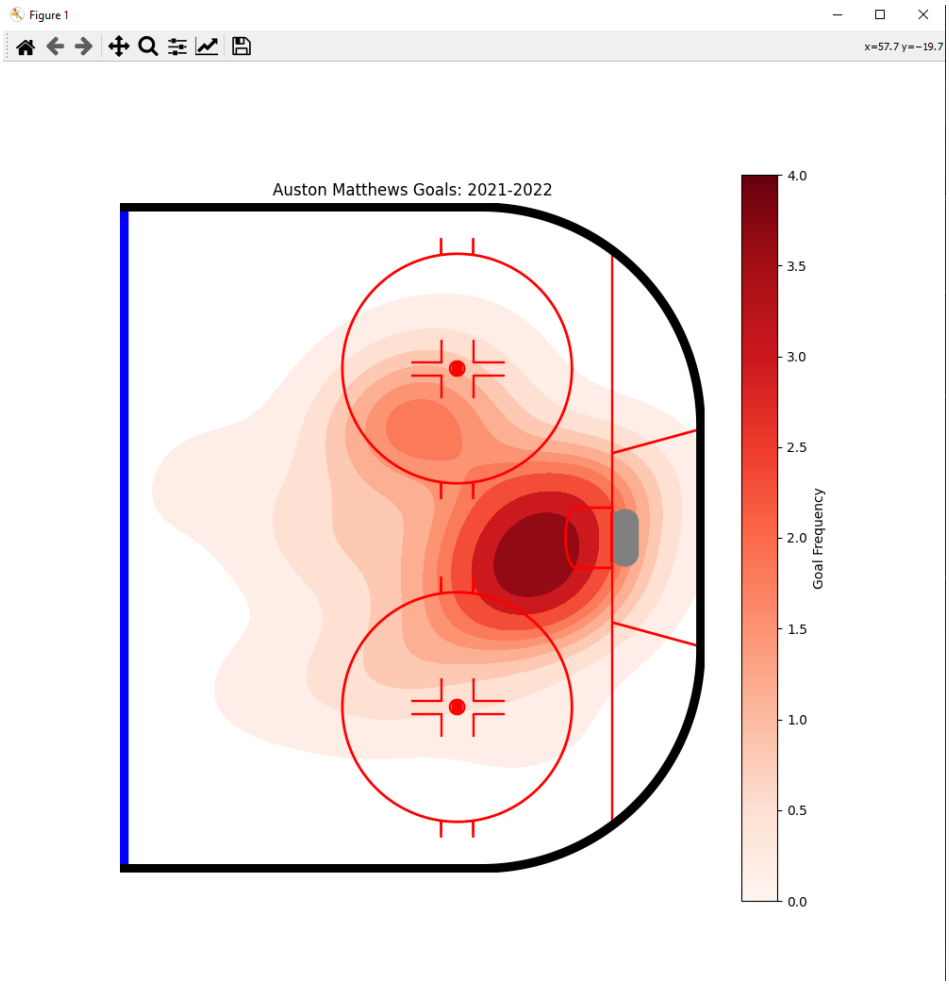

Figure 8 - Goal heat map

```
Auston Matthews is # 1 for goals with a percentile rank of: 96.7741935483871
Auston Matthews is # 5 for assists with a percentile rank of: 83.87096774193549
Auston Matthews is # 1 for shots with a percentile rank of: 96.7741935483871
```

Figure 9 - Ranking statistics

Chapter 5

Conclusions

After referring back to the requirements of the software development, each one has been satisfactorily met with self-testing supporting this and having this backed up by feedback from users. The artefact successfully retrieves statistics for every player on the Toronto Maple Leafs active roster and these statistics are available to be displayed to the user through an interactive user interface which allows the user to create data visualisations by selecting which statistics they want to measure and display. A ranking system was also implemented using the available statistics to display who the most valuable players are on the roster. The final requirement of having the data be easy to understand to users of varying hockey knowledge was completed and this is supported by the user testing where three users with different levels of hockey knowledge all confirmed that the application was useful for displaying hockey data and contextualising the data through the use of the ranking system and the visualisations, with the heat map aiding potential strategy creation and coaching (if they were in the position to) due to an understanding where players regularly score from and where their weakest areas for shooting and scoring are.

While the second objective of creating an application that displays hockey statistics and visualisations was successful, the first objective of researching and potentially implementing machine learning applications for the data analysis was only partially successful. Thorough research was performed as evidenced in the literature review; however, the implementation was not successful and ultimately was not included in the final version of the artefact. To say the implementation was not successful does not mean that there was not a machine learning application but that the machine learning model had an accuracy that was too low for it to be a reliable form of data analysis, coming in with an accuracy of no higher than 62% on each test, despite a vast array of statistics to train the model. An issue with attempting to train machine learning models to predict outcomes in sports games with many concurrent events at any given time such as football (soccer) and ice hockey is that there are elements which are not recorded in statistics, these can include player motivation and player fatigue. Other models have attempted to overcome this challenge by creating their own accounting of fatigue with Tax and Joustra (2015) being one of these where they accounted for playing time in the previous game as well as the number of days since the last game. However, despite attempting to mitigate this flaw their model still only achieved an accuracy of 55%. Due to requiring the artefact to offer

reliable statistics the machine learning element was removed with the focus becoming solely on recorded historical statistics to inform the users, as the machine learning model would draw (often) incorrect conclusions about a player's future performance, which would make the application unreliable for its purpose of analysing a player data in the hopes of using this information for strategy, whether it be for fantasy league trades or coaching decisions in the National Hockey League.

Overall, the project was successful in meeting its objectives as testing proved that the artefact could be used to analyse National Hockey League statistics for the purpose of informing strategic decisions for coaching and fantasy league trading. The use of machine learning within ice hockey data analysis was also thoroughly explored with a model being successfully trained albeit with underwhelming accuracy, despite this the insights are still useful due to the model still predicting the majority of teams correctly, therefore making the machine learning stage a partially successfully. The use of the Cleanroom methodology particularly aided the project as the only issues presented by user testing were minor interface issues that did not impact the use of the application. This ensured that the project was completed in a timely manner with no major application issues to resolve when the project was drawing to a close.

Chapter 6

Reflective Analysis

While the application was largely successful in achieving the requirements, there were hindrances that impacted the ease of use of the application. One of these hindrances was using Tkinter, the built-in python library for developing user interfaces. Issues started to arise immediately after the creation of a basic window that displayed the player dataframe using the Tkinter function Treeview, this solution to displaying the dataframe in a clear way worked well initially, however as soon as any additions were made to the Tkinter window they either would not display or they would interfere with the dataframe display by removing it from the Treeview frame. Additional tabs to facilitate additional pages in the window to display the other dataframes would not display due to Treeview displaying over any tabs thereby making the tabs inaccessible. Once the Pandasgui window was made the initial Treeview frame became redundant as the Pandasgui offered visualisation tools as well as the dataframe display with the ability to choose a column to rank the players by. Also, when buttons were added to open another window containing the other dataframes (working around the tabs issue) the original Treeview frame would clear itself even if another frame was declared for the new windows. Other than the display issues with Treeview, Tkinter only creates a barebones and simplistic interface, something that was noted by Participant one in the testing of the application, if the project was to be undertaken again the choice of library for creating a user interface would be changed in order to make the artefact appear more professional, something that is considered desirable by the participants of the testing. The largest issue with the user interface is that each part of the analysis takes place in different windows which are operated by different libraries, this prevents the artefact from being able to take place in a single window. Had time not been a constraint, the interface would operate in a single window with different tabs for dataframe viewing, data mapping (using graphs) and heat map analysis of player shots. Selecting a dataframe or player (depending on the tab) would be accessed by using a dropdown box where a user could scroll and make their selection with the current tab changing to match their selection, a larger testing group would also be used and at regular intervals to ensure an application is developed that suits its intended audience.

A positive reflection on the artefact is that all requirements were met from displaying statistics and visualisations to ensuring that the data was displayed in a way that is easy to understand for less experienced users. Participants two and

three were entirely inexperienced with ice hockey statistics and they commented on their ability to interpret the data easily despite not having the context of where the data came from, specifically mentioning the heat map for the assistance in understanding. Participant one who had extensive experience with ice hockey, NHL as well as ice hockey statistics was pleased with the statistics on view and the visualisations for quick comparisons between players, showing that the aim of appealing to users from a wide range of hockey experience backgrounds (as mentioned in Design) had been achieved. The machine learning aspect to the project was still successful as a model was trained and predictions were created based on statistics taken from the NHL, despite the machine learning code not being utilised in the finished project it could still be used in the artefact if the artefact was to be updated as a working software component. Time management was also executed effectively as production of the artefact was completed within a reasonable timeframe, due, in large part, to the Cleanroom methodology which utilised self-testing throughout the project rather than towards the end, this allowed for issues to be dealt with immediately and features to be removed that were no longer required such as the machine learning element of the project. Had the Waterfall methodology been used the project would have been at a disadvantage due to the small testing group and the potential for large issues to have been brought to light during testing when there would have been little time to solve these issues, this could have potentially led to a non-working artefact which would have resulted in a failed project. The use of GitHub Projects was also a contributing factor to the success of the project, much more so than the Gantt chart as tasks to be completed could be easily tracked without having to overhaul the timeline in which other tasks had to be completed and a complete re-evaluation of timeframes for different stages of development.

References

Cohen, A. (2021). NHL Starts 2021 Season With Puck and Player Tracking in All Arenas. [online] Sporttechie.com. Available at: https://www.sporttechie.com/nhl-starts-2021-season-with-puck-and-player-tracking-in-all-arenas/[Accessed 19Feb. 2022].

Cullen, F.T., Myer, A.J. and Latessa, E.J. (2009). Eight Lessons from Moneyball : The High Cost of Ignoring Evidence-Based Corrections. [online] Victims and Offend-ers. Available at: https://www.tandfonline.com/doi/full/10.1080/15564880802612631?needAccess=true[Accessed 19Feb. 2022].

Using Machine Learning Algorithms to Identify Undervalued Baseball Players Ta-tsuya Ishii (twishii). (n.d.). [online] Available at: http://cs229.stanford.edu/proj2016/re-port/Ishii-UsingMachineLearningAlgorithmsToIdentifyUndervaluedBaseballPlayers-report.pdf.[Accessed 20 Feb. 2022].

Hamitlon, M., Hoang, P., Layne, L., Murray, J., Padgett, D., Stafford, C. and Tran, H. (2014). Applying Machine Learning Techniques to Baseball Pitch Prediction. Pro-ceedings of the 3rd International Conference on Pattern Recognition Applications and Methods. [online] Available at: https://www.scitepress.org/Pa-pers/2014/47639/47639.pdf.[Accessed 20 Feb. 2022].

Online MBA. (2019). How Data Analytics Have Changed Baseball Forever. [online] Available at: https://onlinemba.montclair.edu/data-analytics-have-changed-baseball-forever/[Accessed 20 Feb. 2022].

Gu, W., Foster, K., Shang, J. and Wei, L. (2019). A game-predicting expert system using big data and machine learning. Expert Systems with Applications,[online] 130, pp.293–305. Available at: https://reader.elsevier.com/reader/sd/pii/S0957417419302556?token=883698FAA5A6547597C73C857FC4D35693F5C70D708ACA08F2D60ED2BCB01A073327033833B2F0E04F990CBE59841D19&originRegion=eu-west-1&originCreation=20220223224850[Accessed 21Feb. 2022].

Liu, G. and Schulte, O. (2018). Deep Reinforcement Learning in Ice Hockey for Context-Aware Player Evaluation. [online] Available at: https://www.ijcai.org/pro-ceedings/2018/0478.pdf.[Accessed 21Feb. 2022].

Bunker, R. and Susnjak, T. (2019). The Application of Machine Learning Tech-niques for Predicting Results in Team Sport: A Review. [online] Available at: https://arxiv.org/pdf/1912.11762.pdf.[Accessed 23 Feb. 2022].

Hockey Analytics. (2016). ELO Ratings for the NHL. [online] Available at: http://hockeyanalytics.com/2016/07/elo-ratings-for-the-nhl/[Accessed 23 Feb. 2022].

Schulte, O., Khademi, M., Gholami, S., Zhao, Z., Javan, M., Desaulniers. (2017). A Markov Game model for valuing actions, locations, and team performance in ice hockey. [online] Available at: https://link.springer.com/content/pdf/10.1007/s10618-017-0496-z.pdf[Accessed 23 Feb. 2022].

NHL (2021). NHL, Amazon Web Services to debut advanced stats during playoffs.[online] NHL.com. Available at: https://www.nhl.com/news/nhl-aws-to-debut-ad-vanced-stats-during-playoffs/c-324660432[Accessed 24 Feb. 2022].

Kirovska, N. and Koceski, S. (2016). *USAGE OF KANBAN METHODOLOGY AT SOFTWARE DEVELOPMENT TEAMS*.

[online] *http://eprints.ugd.edu.mk/14949/1/030302.pdf*. Available at:

http://eprints.ugd.edu.mk/14949/1/030302.pdf [Accessed 21 May 2022].

Wilson, J.M. (2003). Gantt charts: A centenary appreciation. *European Journal of Operational Research*, [online] 149(2), pp.430–437. doi:10.1016/s0377-2217(02)00769-5. Available at: https://www.sciencedirect.com/science/article/pii/S0377221702007695?casa_token=RqYyGvEhjNkAAAAA:llD1uvrlNz2pu5EZDz8YOUhjB3u3TAVN_3_u5tydPtPyM0Nw3USKl6GHy3aheBxq_veegSpf#aep-section-id16 [Accessed 21 May 2022].

Mingwei, Z. (2020). Discussion on the relationship between clean room and traditional software engineering methods and practices. *Proceedings of the 2nd International Conference on Artificial Intelligence and Advanced Manufacture*. [online] Available at: https://dl.acm.org/doi/epdf/10.1145/3421766.3421874 [Accessed 21 May 2022]

Stancin, I. and Jovic, A. (2019). An overview and comparison of free Python libraries for data mining and big data analysis. *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. [online] doi:10.23919/mipro.2019.8757088. Available at: https://ieeexplore.ieee.org/abstract/document/8757088?casa_token=oPuOvO

ZWVU8AAAAA:ZrI3vBT8yfqmdSzmUYRXP9wwLNruN1mJJFyaW6-WOv8B6UR-uoEFUVA-F3kJb_fgoMhNvCfAKio [Accessed 22 May 2022]

Song, Y.-Y. and Lu, Y. (2015). Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, [online] 27(2), pp.130–5. doi:10.11919/j.issn.1002-0829.215044. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4466856/ [Accessed 22 May 2022]

Parmar, C., Barry, J.D., Hosny, A., Quackenbush, J. and Aerts, H.J.W.L. (2018). Data Analysis Strategies in Medical Imaging. *Clinical Cancer Research*, [online] 24(15), pp.3492–3499. doi:10.1158/1078-0432.ccr-18-0385. Available at: https://aacrjournals.org/clincancerres/article/24/15/3492/80892/Data-Analysis-Strategies-in-Medical-ImagingData [Accessed 22 May 2022]

Azzam, T., Evergreen, S., Germuth, A.A. and Kistler, S.J. (2013). Data Visualization and Evaluation. *New Directions for Evaluation*, [online] 2013(139), pp.7–32. doi:10.1002/ev.20065. Available at: https://onlinelibrary.wiley.com/doi/epdf/10.1002/ev.20065 [Accessed 22 May 2022]

Tax, N. and Joustra, Y. (2015). *Predicting The Dutch Football Competition Using Public Data: A Machine Learning Approach*. [online] ResearchGate. Available at: https://www.researchgate.net/publication/282026611_Predicting_The_Dutch_Football_Competition_Using_Public_Data_A_Machine_Learning_Approach [Accessed 24 May 2022].

Appendices

```
import pyspark
from pyspark.sql import SparkSession
import numpy as np
import pyspark.sql.functions as func
from os import environ
from pyspark.ml.stat import Correlation
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml import Pipeline
from pyspark.ml.feature import StringIndexer
from pyspark.mllib.evaluation import MulticlassMetrics
from pyspark.ml.classification import LinearSVC
from pyspark.ml.classification import MultilayerPerceptronClassifier
import os
import sys
import itertools
```

Figure 1 - Machine learning file imports

```
os.environ['PYSPARK_PYTHON'] = sys.executable
os.environ['PYSPARK_DRIVER_PYTHON'] = sys.executable

spark = SparkSession.builder.config('spark.driver.memory','32G').config('spark.ui.showConsoleProgress', 'false').getOrCreate()
spark.sparkContext.setLogLevel("ERROR")
df = spark.read.csv("skaters.csv", inferSchema=True,header=True)
names = df.schema.names

stringIndexer = StringIndexer(inputCol="team", outputCol="team_index").fit(df)
df = stringIndexer.transform(df)
test, training = df.randomSplit([0.3, 0.7], 20)
va = VectorAssembler(inputCols = ["icetime", "games_played", "gameScore", "season"], outputCol='sensors')
test = va.transform(test)
training = va.transform(training)

df_classifier = DecisionTreeClassifier(featuresCol="sensors", labelCol="team_index")
df_model = df_classifier.fit(training)
df_predictions = df_model.transform(test)
#df_predictions.show()
predict_accuracy1 = MulticlassClassificationEvaluator(labelCol="team_index", metricName="accuracy")
my_eval = MulticlassClassificationEvaluator(labelCol='team_index')
weightedPrecision1 = predict_accuracy1.evaluate(df_predictions, {predict_accuracy1.metricName: "weightedPrecision"})
weightedRecall1 = predict_accuracy1.evaluate(df_predictions, {predict_accuracy1.metricName: "weightedRecall"})
FalsePos = df_predictions.where((df_predictions["prediction"] == 1.0) & (df_predictions["team_index"] == 0.0)).count()
TrueNeg = df_predictions.where((df_predictions["prediction"] == 0.0) & (df_predictions["team_index"] == 0.0)).count()
print("Error rate", 1 - my_eval.evaluate(df_predictions), "Specificity %",TrueNeg / TrueNeg + FalsePos, "Sensitivity", weightedRecall1)
```

Figure 2 - Machine learning file code

```
import requests
import pandas as pd
import tkinter as tk
from tkinter import *
from tkinter import ttk
from pandasgui import show
import matplotlib.pyplot as plt
import requests
from matplotlib import image
from hockey_rink import NHLRink
from PIL import ImageTk, Image
from matplotlib import cm
from matplotlib.patches import Circle, Rectangle, Arc, ConnectionPatch
from matplotlib.patches import Polygon
from matplotlib.collections import PatchCollection
from matplotlib.path import Path
from matplotlib.patches import PathPatch
import seaborn as sns
```

Figure 3 - Main file imports

```python
def heat_map(inname):
    fig=plt.figure(figsize=(10,10))
    plt.xlim([0,100])
    plt.ylim([-42.5, 42.5])
    Austonx = []
    Austony = []

    seasons = ["20212022"]
    print("Loading heat map...")
    for season in seasons:
        tor_schedule = requests.get("https://statsapi.web.nhl.com/api/v1/schedule?season="+season+"&teamId=10&gameType=R")
        schedule = tor_schedule.json()
        schedule = schedule["dates"]

        game_ids=[]
        for game in schedule:
            game_data=game["games"]
            game_data=(game_data[0])
            status = game_data["status"]
            status = status["abstractGameState"]
            if status == "Preview":
                continue
            else:
                id = game_data["gamePk"]
                game_ids.append(id)

        for ids in game_ids:
            url = requests.get("https://statsapi.web.nhl.com/api/v1/game/" + str(ids) + "/feed/live")
            content = url.json()

            event = content["liveData"]
            plays = event["plays"]
            all_plays = plays["allPlays"]
            for i in all_plays:
                result = i["result"]
                event = result["event"]
                if event=="Goal" or event == "Shot" or event=="Missed Shot":
                    team_info = i["team"]
                    team = team_info["triCode"]
                    players = i["players"]
                    for m in players:
                        playertype = m["playerType"]
                        if playertype == "Scorer":
                            scorer = m["player"]
                            n = scorer["fullName"]
                            if n == inname:
                                coord = (i["coordinates"])
                                x = int(coord["x"])
                                y = int(coord["y"])
                                if x < 0:
                                    x = abs(x)
                                    Austonx.append(x)
                                    y = y*-1
                                    Austony.append(y)

                                else:
                                    x=x
                                    Austonx.append(x)
                                    y=y
                                    Austony.append(y)
                else:
                    continue
```

Figure 4 - heat map function

```
rink = NHLRink()
ax = rink.draw(display_range="ozone")
hb = ax.hexbin(Austonx, Austony, gridsize=25, cmap='Reds')
ax.clear()
ax = rink.draw(display_range="ozone")
cb = fig.colorbar(hb, ax=ax, label='Goal Frequency')
kde = sns.kdeplot(
    Austonx,Austony, shade = True, shade_lowest = False, alpha=1, cmap="Reds"
)


plt.title(inname + " Goals: 2021-2022")
plt.show()
```

Figure 5 - heat map display

```
df = pd.DataFrame(columns = ['Name', 'Games Played', 'Time On Ice', 'Assists', 'Goals', 'Shots', 'Rank'])

goalie_df = pd.DataFrame(columns = ['Name', 'Games Played', 'Power Play Saves', 'Shots Against', 'Goals Against', 'Rank'])

individual_df = pd.DataFrame(columns = ['Name', 'Games Played', 'Time On Ice', 'Assists', 'Goals', 'Shots'])

root = tk.Tk()
root.title("Player Data")

root.geometry("800x800")
root.pack_propagate(False)
root.resizable(0, 0)
```

Figure 6 - dataframe creation and Tkinter setup

```
def goal_rank(goal_list, goal, name):
    goal_list.sort()
    i = 0
    for l in goal_list:
        if l < goal:
            i += 1
        else:
            percentile = i / len(goal_list) * 100
    pos = len(goal_list) - i
    print(name, "is #", pos, "for goals with a percentile rank of:", percentile)

def assist_rank(assist_list, assists, name):
    assist_list.sort()
    i = 0
    for l in assist_list:
        if l < assists:
            i += 1
        else:
            percentile2 = i / len(assist_list) * 100
    pos = len(assist_list) - i
    print(name, "is #", pos, "for assists with a percentile rank of:", percentile2

def shot_rank(shot_list, shots, name):
    shot_list.sort()
    i = 0
    percent = 0
    for l in shot_list:
        if l < shots:
            i += 1
        else:
            percent = i / len(shot_list) * 100
    pos = len(shot_list) - i
    print(name, "is #", pos, "for shots with a percentile rank of:", percent)
```

Figure 7 - Ranking functions

```python
def get_goalie_ids():
    url = "https://statsapi.web.nhl.com/api/v1/teams/10/roster"
    url = requests.get(url)
    url = url.json()
    roster = url["roster"]
    goalie_list = []
    for i in roster:
        person = i["person"]
        position = i["position"]
        name = person["fullName"]
        ids = person["id"]
        if position["abbreviation"] == "G":
            goalie_list.append(ids)
    return goalie_list

def get_player_ids():
    url = "https://statsapi.web.nhl.com/api/v1/teams/10/roster"
    url = requests.get(url)
    url = url.json()
    roster = url["roster"]
    id_list = []
    for i in roster:
        person = i["person"]
        position = i["position"]
        name = person["fullName"]
        ids = person["id"]
        if position["abbreviation"] != "G":
            id_list.append(ids)
    return id_list
```

Figure 8 - Player ID retrieval

```python
def individual_player_data(get_player_ids, individual_df, get_goalie_ids):
    n = input("Enter player name: ")
    goal_list = []
    shot_list = []
    assist_list = []
    for x in get_player_ids():
        link = "https://statsapi.web.nhl.com/api/v1/people/" + str(x) + "/stats?stats=statsSingleSeason&season=20212022"
        name_link = "https://statsapi.web.nhl.com/api/v1/people/" + str(x)
        name_link = requests.get(name_link)
        name_link = name_link.json()
        people = name_link["people"]
        for l in people:
            fullname = l["fullName"]
        if fullname != n:
            link = requests.get(link)
            link = link.json()
            stats = link["stats"]
            for i in stats:
                splits = i["splits"]
                for i in splits:
                    stat = i["stat"]
                    goals = int(stat["goals"])
                    goal_list.append(goals)
                    assists = int(stat["assists"])
                    assist_list.append(assists)
                    shots = int(stat["shots"])
                    shot_list.append(shots)
            continue
        else:
            link = requests.get(link)
            link = link.json()
            stats = link["stats"]
            for i in stats:
                splits = i["splits"]
                for i in splits:
                    stat = i["stat"]
                    goal = int(stat["goals"])
                    goal_list.append(goal)
                    toi = stat["timeOnIce"]
                    time = ''.join(x for x in toi if x.isdigit())
                    time = int(time[:-2])
                    assist = int(stat["assists"])
                    assist_list.append(assist)
                    shot = int(stat["shots"])
                    shot_list.append(shot)
                    games_played = int(stat["games"])
                    new_row = {'Name':fullname, 'Games Played':games_played, 'Time On Ice':toi, 'Assists':assist, 'Goals':goal, 'Shots':shot}
                    individual_df = individual_df.append(new_row, ignore_index=True)
    show(individual_df)
    heat_map(n)
    goal_rank(goal_list, goal, n)
    assist_rank(assist_list, assist, n)
    shot_rank(shot_list, shot, n)
```

Figure 9 - Individual player analysis

```python
def get_player_data(get_player_ids, df):
    for x in get_player_ids():
        link = "https://statsapi.web.nhl.com/api/v1/people/" + str(x) + "/stats?stats=statsSingleSeason&season=20212022"
        name_link = "https://statsapi.web.nhl.com/api/v1/people/" + str(x)
        name_link = requests.get(name_link)
        name_link = name_link.json()
        people = name_link["people"]
        for l in people:
            fullname = l["fullName"]
        link = requests.get(link)
        link = link.json()
        stats = link["stats"]
        for i in stats:
            splits = i["splits"]
            for i in splits:
                stat = i["stat"]
                goals = int(stat["goals"])
                toi = stat["timeOnIce"]
                time = ''.join(x for x in toi if x.isdigit())
                time = int(time[:-2])
                assists = int(stat["assists"])
                shots = int(stat["shots"])
                games_played = int(stat["games"])
                try:
                    rank = 23 / goals + shots + assists
                except: # exception for the division by zero error for when there's no stats
                    rank = 0
                new_row = {'Name':fullname, 'Games Played':games_played, 'Time On Ice':toi, 'Assists':assists, 'Goals':goals, 'Shots':shots, 'Rank':rank}
                df = df.append(new_row, ignore_index=True)
    return df
```

Figure 10 - Roster Analysis

```
def get_goalie_data(get_goalie_ids, goalie_df):
    for x in get_goalie_ids():
        link = "https://statsapi.web.nhl.com/api/v1/people/" + str(x) + "/stats?stats=statsSingleSeason&season=20212022"
        name_link = "https://statsapi.web.nhl.com/api/v1/people/" + str(x)
        name_link = requests.get(name_link)
        name_link = name_link.json()
        people = name_link["people"]
        for l in people:
            fullname = l["fullName"]
        link = requests.get(link)
        link = link.json()
        stats = link["stats"]
        for i in stats:
            splits = i["splits"]
            for i in splits:
                stats2 = i["stat"]
                pp_saves = int(stats2["powerPlaySaves"])
                games = int(stats2["games"])
                shots_against = int(stats2["shotsAgainst"])
                goals_against = int(stats2["goalsAgainst"])
                shutouts = int(stats2["shutouts"])
                pps_percentage = float(stats2["powerPlaySavePercentage"])
                shs_percentage = float(stats2["shortHandedSavePercentage"])
                ess_percentage = float(stats2["evenStrengthSavePercentage"])
                try:
                    ranks = 23 / pps_percentage + shs_percentage + ess_percentage + games
                except:
                    ranks = 0
                new_row = {'Name':fullname, 'Games Played':games, 'Power Play Saves':pp_saves, 'Shots Against':shots_against, 'Goals Against':goals_against, 'Rank':ranks}
                goalie_df = goalie_df.append(new_row, ignore_index=True)
    show(goalie_df)
```

Figure 11 - Goalkeeper analysis

```
df = get_player_data(get_player_ids, df)

t = Label(root, text = "NHL Player Analysis", font = "Helvetica 12 bold", bg="white")
t.pack()
t2 = Label(root, text = "", bg = "white")
t2.pack()
t3 = Label(root, text = "All data is stored in pandas dataframes. Select either to analyse a roster or an individual player.", font = "Helvetica 12 bold", bg = "white")
t3.pack()

btn = Button(root,
        text ="Analyse individual player data",
        command = lambda : individual_player_data(get_player_ids, individual_df, get_goalie_ids))
btn.pack(pady = 10)
btn.place(x=450, y=350)

btn2 = Button(root,
        text ="Analyse player roster data",
        command = lambda : show(df))
btn2.pack(pady = 10)
btn2.place(x=450, y=300)

btn3 = Button(root,
        text ="Analyse goalie roster data",
        command = lambda : get_goalie_data(get_goalie_ids, goalie_df))
btn3.pack(pady = 10)
btn3.place(x=450, y=400)

frame = Frame(root, width=300, height=200)
frame.pack()
frame.place(anchor='center', relx=0.2, rely=0.8)

img = Image.open("NHL-Logo.png")
img = img.resize((300, 200), Image.ANTIALIAS)
img = ImageTk.PhotoImage(img)
pic = Label(frame, image = img)
pic.pack()

root.mainloop()
```

Figure 12 - Function calls, button creation and finished Tkinter setup

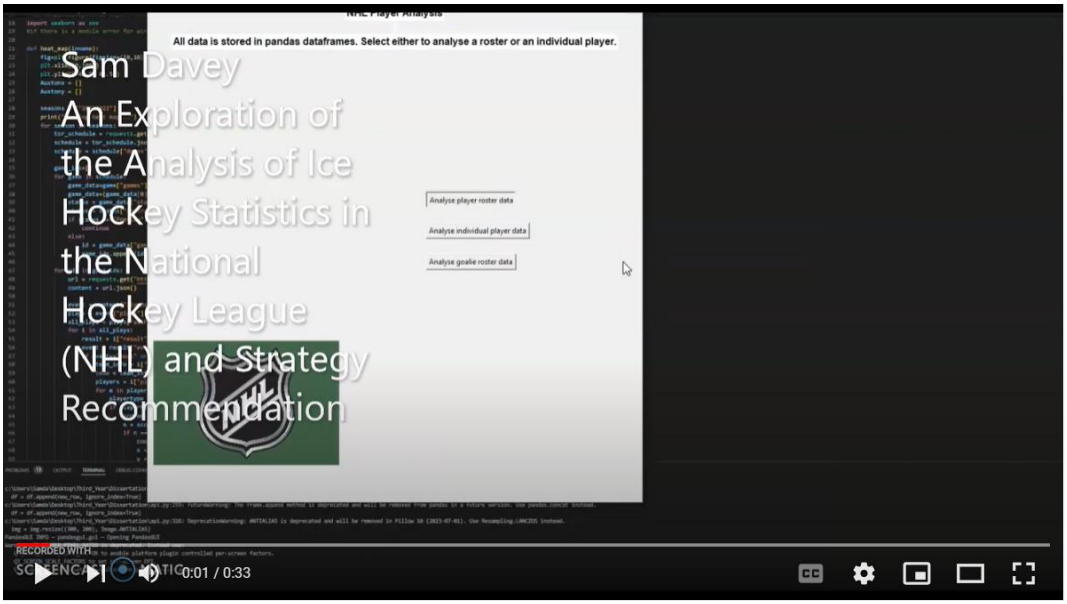

Figure 13 - Demonstration - https://www.youtube.com/watch?v=20nYPku2_9I