Tampere University

Juha-Matti Ryttyläinen

DETECTING MATCH STATE CHANGES IN ICE HOCKEY FROM POSITIONAL DATA

Master of Science Thesis
Faculty of Information Technology and Communication Sciences
Examiner: Prof. Joni Kämäräinen
November 2021

ABSTRACT

Juha-Matti Ryttyläinen: Detecting Match State Changes in Ice Hockey From Positional Data
Master of Science Thesis
Tampere University
Master's Degree Programme in Information Technology
November 2021

This thesis aims to determine whether it is possible to utilize machine learning techniques to determine whether an ice hockey match is in a state of active play or not. As an additional requirement, the match state determination should work in real-time. To determine the match state, the method demonstrated in this thesis utilizes position data of players and pucks produced by the Wisehockey platform. The dataset used in the presented experiment consisted of 40 professional ice hockey matches containing the position data for players and pucks. In addition to the position data, the dataset contains all the match state information for the matches, serving as ground truth for machine learning.

The technical problem presented was approached as a classification problem of small time windows of position data. The machine learning model chosen for the experiment was the random forest classifier. The model achieved AUC scores of above 0.89 for all matches in a 10 match test set. However, the method was not robust enough for a real world application but manages to provide insight into the technical problem discussed. From the results in this thesis it seems possible that by improving the method, it is possible to determine the match state accurately enough for real world use. On the other hand, using a different machine learning method entirely might also produce good results.

Keywords: ice hockey, machine learning, positioning, sports analytics

TIIVISTELMÄ

Juha-Matti Ryttyläinen: Ottelun tilan muutosten tunnistus paikkadatasta jääkiekossa
Diplomityö
Tampereen yliopisto
Tietotekniikan DI tutkinto-ohjelma
Marraskuu 2021

Tämän diplomityön tavoitteena oli selvittää onko mahdollista hyödyntää koneoppimismenetelmiä jääkiekko-ottelun tilan määrittämiseen, eli onko ottelu käynnissä vai tauolla. Lisävaatimuksena ottelun tilan määrittelyn piti toimia reaaliaikaisesti. Tässä diplomityössä käytetty menetelmä hyödyntää Wisehockey-järjestelmän tuottamaa kiekkojen ja pelaajien paikkadataa ottelun tilan määrittelyssä. Työssä käytetty datajoukko sisältää kaikki kiekkojen ja pelaajien paikkadatan neljästäkymmenestä ammatilaistason jääkiekko-ottelusta. Paikkadatan lisäksi datajoukko sisältää tiedon ottelun tilasta pelikellon muodossa, jota voitiin käyttää totuutena koneoppimismallia kouluttaessa.

Esiteltyä teknistä haastetta lähestyttiin pienten paikkadatasta koostuvien aikaikkunoiden luokitteluongelmana. Työhön valittiin koneoppimismalliksi satunnaismetsä-luokittelija. Malli saavutti yli 0.89 tarkkuus (area under curve) arvot kaikille kymmenelle ottelulle testijoukossa, mutta mallin virheensietokyky ei silti riittänyt oikeaan sovellus käyttöön. Malli tarjoaa kuitenkin hyödyllistä tietoa kyseisen teknisen ongelman ratkaisemiseen. Työssä esitellyistä lopputuloksista voidaan päätellä, että joko parantamalla esiteltyä mallia tai hyödyntäen toista koneoppimismenetelmää, ottelun tilan määrittely tarpeeksi tarkasti oikeaan sovellus käyttöön on mahdollista.

Avainsanat: jääkiekko, koneoppiminen, paikannus, urheiluanalytiikka

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

AOA	Angle of Arrival
AUC	Area under curve
BLE	Bluetooth low energy
JSONL	JavaScript Object Notation Lines
ROC curve	Receiver operating characteristic curve

1. INTRODUCTION

The rapid increase of sports data collection has brought sports analytics into a new era. Before, sports analytics focused on large aggregate statistics based on statistics gathered by human officials during competition. With the increase in the granularity and availability of sports data, new artificial intelligence and machine learning methods are more commonly applied to sports analytics. The increase in data granularity follows from the increased use of on-body sensors and other technology used to track players. In addition, huge leaps in areas like machine vision during the preceding decades have improved the ability of using video to generate useful sports analytics. Modern sports analytics now combines approaches from multiple fields, like machine vision, statistical learning, and game theory in addition to the aggregate statistics produced in earlier decades. [1]

The topic of this thesis is to apply machine learning methods, to produce match state information in real-time in the sport of ice hockey. Match state information here refers to whether the match is in active play or whether there is a stoppage of play. In other words, whether the match clock is currently running or not. More specifically the goal of this thesis is to test the feasibility of using position data to classify in real-time whether active play is taking place or not. The motivation behind this topic is to be able to generate match clock data based on purely the position data received from players and a puck at an ice hockey rink.

This thesis was commissioned by Wisehockey Oy with the idea that match clock data is needed for producing many ice hockey statistics in the Wisehockey system and occasionally integration with the match clock hardware at a rink is not feasible. The idea is that in those cases the results of this thesis could be utilized to support the production of statistics. To further explain the motivation behind this thesis, two example scenarios are presented. Figures 1.1 and 1.2 show an example of a common face-off event in ice hockey, where the match state starts of as a stoppage of play in Figure 1.1 and then transitions to active play shown in Figure 1.2 after the referee drops the puck.

Figure 1.1. *Match is in a state of stoppage of play before face-off.*

Figure 1.2. *Match is in a state of active play after face-off.*

Another example scenario is presented in Figures 1.3 and 1.4. In figure 1.3 active play is taking place and a player scores a goal, after which the referee blows the whistle and the match transitions into a stoppage of play. As can be seen from the figures, both player and referee behaviour changes after a stoppage of play is called. The players move more slowly and are not pursuing the puck anymore, whereas the referees are not actively avoiding the players and the puck anymore and freely skate in the center.

Figure 1.3. *Match is in a state of active play before goal.*

Figure 1.4. *Match is in a state of stoppage of play after goal.*

The two scenarios presented are prime examples of the classification problem that is the topic of this thesis. In addition to this introduction, this thesis consists of six chapters: Indoor Positioning for Ice Hockey, Wisehockey and the Match Clock, Machine Learning Model for Position Data, Data and Experimental Setup, Results and Conclusions. In the first chapter (after the introduction), the relevant theory and technologies of indoor positioning utilized by Wisehockey is presented. The second chapter introduces the Wisehockey company and system while also presenting the motivation behind the problem this thesis attempts to solve. The third chapter presents the theoretical background behind the machine learning model utilized in this thesis and the methods behind evaluating the performance of the model. The fourth chapter gives a description of the data set and presents the feature selection and model development approaches taken in this thesis. The fifth chapter describes the results of the final model and provides an additional error

case analysis to show where the model makes mistakes. The final chapter discusses conclusions on the approaches taken in this thesis and gives pointers to further improvements and research that could be done on the subject.

2. INDOOR POSITIONING FOR ICE HOCKEY

In this chapter, the background information on indoor positioning and its utilization in ice hockey is presented. The chapter begins with discussing previous work in the area and then provides descriptions of the positioning technologies utilized by Wisehockey to produce positioning data on players and the puck.

2.1 Previous Work

No direct previous work on the subject of match state classification was found in either ice hockey or other sports. This is likely due to there not being a direct application for such classification tasks until now. However, AI and machine learning methods have been increasingly applied to different sports. [1]

With the rapid increase of sports data collection, new techniques have been adopted from machine learning and AI to put the gathered data to use. In addition to the increasing amount of data, the granularity of the gathered data has increased significantly from the aggregate high-level statistics to more fine-grained information like positioning data on individual players. Multiple active research fields of ML and AI have found their way to sports analytics, like machine vision, statistical learning, and game theory. The increasing use of machine learning in sports will have an effect on all areas of sports like coaching, broadcasting, and betting. [1]

Some examples of AI being used in sports include event detection and automatic highlight generation from video data through computer vision. As well as summarizing player information by aggregating their statistics into player vectors utilizing statistical learning. [2]

2.2 Indoor Positioning

Indoor positioning refers to positioning technologies used indoors, where generally GPS or other global positioning technologies work poorly. Multiple different wireless communication protocols are utilized in indoor positioning, some examples being WIFI, ultra-wide band (UWB), and Bluetooth. Indoor positioning systems have many applications in industrial and commercial environments. [3]

The original purpose for wireless communication systems was to transfer information wirelessly, but there is an additional use case of using the properties of the signals themselves to track the position of an object. Generally, with two communication nodes, either of the two nodes can measure the distance to the other node by considering the signal properties. To get from distance measurements to positioning an object, multiple nodes in different known coordinates are needed. With multiple nodes in known coordinates and with distance measurements to the node in an unknown location, the location of the unknown node can be calculated relative to the coordinates of the known nodes. With the distance information between two nodes, it can be determined that the other node has to be located at some point on the surface of a sphere centered in the other node's location such that the distance is the radius of that sphere. Once multiple nodes and distance measurements from different locations are added the location of the unknown node has to be at a point where the sphere surfaces intersect. However, the location of an unknown node can also be determined by other means than the distance from multiple nodes, mainly the direction of the signal. [4]

2.2.1 Angle of Arrival

There are multiple properties of radio signals that enable the measurement of distance or location. The 4 basic properties that can be used are time of flight (TOF), received signal strength (RSS), angle of arrival (AOA), and phase difference between electric and magnetic fields. All methods of distance and location measurement utilize one or multiple of these properties [4]. The most relevant property for the purposes of this thesis is the angle of arrival property, since that is the property utilized by Bluetooth direction finding and the Quuppa Intelligent Locating System [5][6].

The angle of arrival method for distance and location measurement only requires a directional antenna and a radio transmitter. Essentially any type of signal can be used with AOA without any cooperation from the transmitter required. In a pure AOA system, the location and distance are determined by triangulation. As can be seen from figure 2.1 two directional antennas A_1 and A_2 are located at know coordinates $(0, 0)$ and $(d, 0)$. Since the antennas measure the angle of arrival α_1 and α_2 of the signals from the target, the location of the target can be calculated utilizing trigonometry in the following way:

$$x = \frac{d \tan(\alpha_2)}{\tan(\alpha_2) - \tan(\alpha_1)} \tag{2.1}$$

$$y = \frac{d \tan(\alpha_1) \tan(\alpha_2)}{\tan(\alpha_2) - \tan(\alpha_1)} \tag{2.2}$$

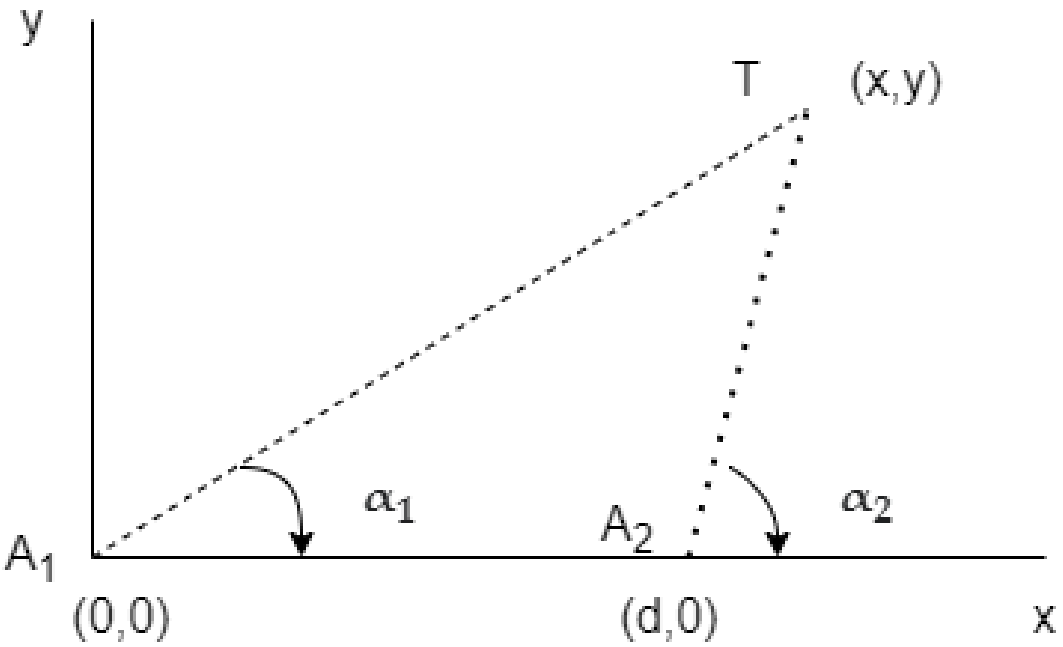

Figure 2.1. *2D triangulation*

Since the angle of arrival cannot be measured exactly there will be some error in the location measurement of the target. The position of the antennas relative to the target is also crucial in reducing the positioning error produced by the AOA method. Once the angle between the antenna and the target starts approaching values greater than 90°, the positioning error starts to rapidly increase [4]. For this reason in addition to noise reduction, it is important to have multiple antennas covering the positioning area from multiple angles to maintain reasonable positioning error.

2.2.2 Bluetooth Low Energy

Bluetooth Low Energy (BLE) is a wireless radio communication technology designed for low-power applications. The frequency band utilized by BLE is the same 2.4GHz band utilized by classic Bluetooth and other radio technologies. BLE is used in different application areas like, data transmission, audio streaming, and location services (positioning). [7]

The most important difference between Bluetooth and BLE is that BLE consumes anywhere from 0.5 to 0.01 times the power of classic Bluetooth. In addition to the power consumption difference, BLE also supports multiple network topologies like broadcast and mesh in addition to point-to-point which is the only topology supported by classic Bluetooth. However, the most relevant feature in the context of this thesis is BLE's support for direction measurements, which classic Bluetooth does not offer. [7]

Bluetooth direction finding feature utilizes angle of arrival (AOA) and angle of departure

(AOD) methods. The AOA method is intended for the use case where a base station needs to locate a person or an object, whereas the AOD method is intended for situations where a person or an object needs to know their location in a building. In other words, the AOA method is used for applications where real-time tracking of objects or people is needed, whereas the AOD method is more like an "indoor GPS". Some common uses cases for the AOA method are asset tracking in industrial environments and player performance tracking in sports. For the AOD method, there are use cases like indoor navigation for people in large indoor areas and navigation for the purposes of autonomous guided vehicles. [5]

2.2.3 Quuppa

Quuppa Oy is a company specializing in real-time locating systems, based in Espoo Finland. The Quuppa Intelligent Locating System is a world-leading location technology platform [6]. The Wisehockey analytics system utilizes The Quuppa Intelligent Locating System to produce position data based on the BLE tags worn by the players and attached to the pucks [8]. the Quuppa Intelligent Locating System combines BLE with direction finding technology (angle of arrival) to produce player and puck positions [6].

In the Wisehockey use case for the Quuppa Intelligent Locating System, Quuppa BLE antennas with direction finding capabilities are installed on top of an ice hockey rink [9]. For the best possible position quality and coverage, from 20 to 28 antennas (or locators) are installed at each rink [8]. The data from the antennas are fed to a software called the Quuppa Positioning Engine, which then produces usable position data for the Wisehockey system [6].

3. WISEHOCKEY AND THE MATCH CLOCK

In this chapter, background information on the Wisehockey company and system is provided. In addition this chapter explains the utilization of the match clock in ice hockey and as such the motivation behind the classification taks presented in this thesis.

3.1 Wisehockey

Wisehockey Oy is a Tampere based company that provides and develops an automatic real-time ice hockey analytics platform. Wisehockey Oy has multiple professional ice hockey leagues as partners and the platform is used in over 1000 professional ice hockey matches each year. The Wisehockey platform provides statistics and data for leagues, TV production, coaching, venues, and betting services. The statistics provided by the platform are mainly calculated from indoor positioning data provided by the Quuppa Intelligent Locating System. [10]

The statistics provided by Wisehockey include the statistics traditionally recorded by humans at the rink like shots, faceoffs, penalties, powerplays, etc. In addition to the previously mentioned statistics, the Wisehockey platform provides more advanced statics, like momentum, player speeds, and player distances traveled. All of the statistics are produced automatically in real-time without requiring humans to track them. A more complete list of example statistics is provided below: [8]

- Player speed, time on ice, and distance traveled on ice
- Puck control
- Faceoffs
- Shot and goal detection
- Shift tracking
- Penalties and powerplays
- Players' +/- statistics
- Momentum

3.1.1 Wisehockey software architecture

Most of Wisehockey's data processing and storing are done in the cloud. Wisehockey has utilized a cloud-native microservice architecture from the beginning of the project, to achieve easy scaling and deployment processes. When data is streamed in real-time from the rinks to the cloud, the raw data stream is stored for development and maintenance purposes, in addition, statistics and video from the rink are processed in real-time to make them available for clients as soon as possible through different Wisehockey interfaces. Wisehockey utilizes a lot of techniques from signal processing and machine learning to filter the raw position data and to produce the statistics described earlier. Figure 3.1 gives a basic overview of the Wisehockey system at a high level.

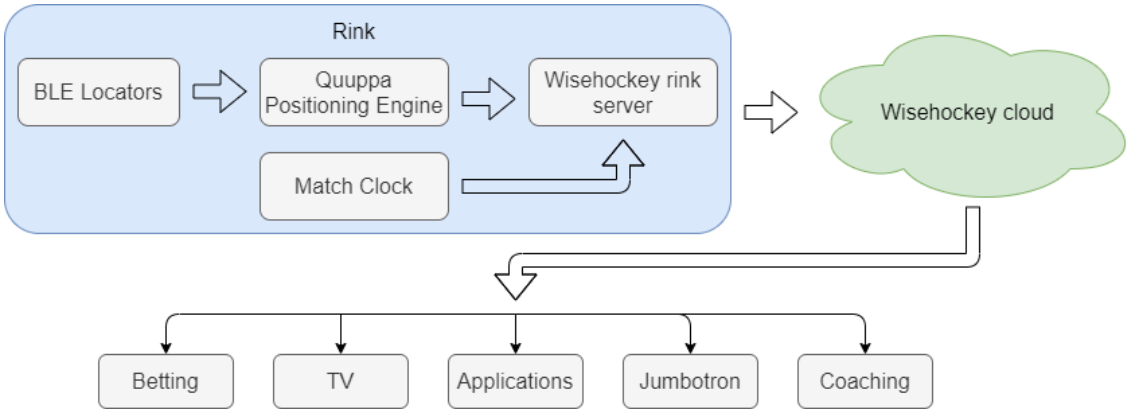

Figure 3.1. *Basic overview of Wisehockey data flow and architecture*

Wisehockey provides statistics to its clients in mainly two ways, through the Wisehockey Webportal and Wisehockey Public API. The Wisehockey Webportal is a web application where users can easily explore statistics related to matches and individual players. In addition to statistics, a video tool called Wiseplayer is also provided through the web application. The Wisehockey Public API is an API that clients can use to fetch data programmatically to be able to integrate Wiseplayer statistics with their own systems. [8]

3.2 Match Clock

Before discussing the match clock, a general picture of how an ice hockey match proceeds in terms of timekeeping has to be given. An ice hockey match commonly consists of three 20 minute periods, with two intermissions between them. If the match is tied after the three regular periods an additional sudden-death overtime period of 5 minutes or 10 minutes is played. Sudden death meaning that the match ends immediately if another team scores. Depending on the match the winner is decided by repeating sudden death overtime periods indefinitely or by a penalty shoot-out competition after a maximum amount of overtime periods is reached. Depending on the ice hockey tournament

the overtime conventions are different, however, more important matches like finals of a tournament tend to forgo the shoot-out competition and instead repeat overtime periods until a goal is made. [11]

In ice hockey, the match clock time is constantly used in the manual tracking of events like shots and penalties. In the same way, some of the automatic statistics produced by Wisehockey require match clock information. For this reason, Wisehockey integrates with the match clock hardware at the rinks to get a live feed of match clock information. [8]

Although the match clock information presented at the scoreboard of rinks varies somewhat, the International Ice Hockey Federation (IIHF) recommends that information like the match clock time, current period number, penalties, and the score among others are shown on the scoreboard. Usually, all the aforementioned scoreboard information is entered into the match clock interface, such that the scoreboard is a part of the match clock system. However, the most important function of the match clock is to track the active play time generally counting down from 20 minutes until it reaches zero at the end of a period, after which it is reset back to 20 minutes for the next period [11]. The match clock system at the rink is presented in figure 3.2. Off-ice officials in a booth next to the rink are responsible for controlling the match clock and inputting all required information into it, through an interface located at the booth. In rinks running televised matches, the TV production is also commonly integrated into the match clock system to be able to show the scoreboard information to viewers.

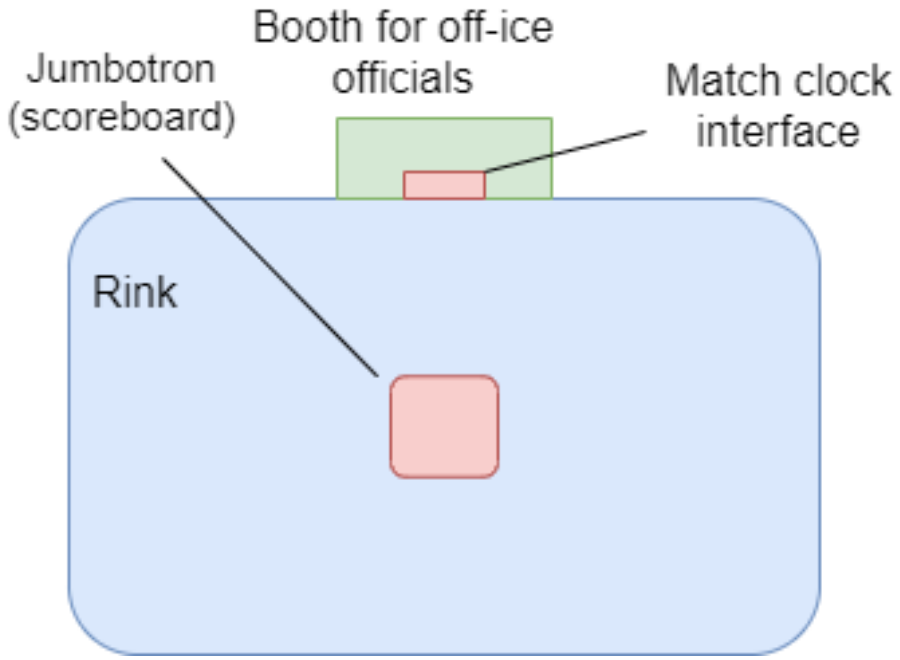

Figure 3.2. *Match clock system at the rink*

Since this thesis was commissioned by Wisehockey Oy, the goal of the experiments in this thesis is to determine the feasibility of replacing match clock integration in situations

where physical integration with the match clock is not feasible, so that the match clock dependent statistics can still be calculated. The idea is that if the match state can be tracked accurately enough based on position data, then the clock data could be generated programmatically in between the stoppages of play. The data for the experiments in this thesis is provided by the Wisehockey platform. Since arenas with the Wisehockey system installed, generally integrate the match clock at the rink to the Wisehockey system, there is a lot of annotated data available to be used for the specific goals of the experiment in this thesis.

4. MACHINE LEARNING MODEL FOR POSITION DATA

In this chapter, the theoretical background on relevant machine learning methods is presented. This chapter covers the machine learning model utilized in this thesis (random forests classifier) and provides a description of the methods used to measure classifier performance.

4.1 Machine Learning Model

The machine learning model utilized in this thesis to classify match states is the random forest or random decision forest classifier. This section aims to give the theoretical background necessary for understanding the operation of the random forest model. There are multiple different implementations of random forest classifiers, however, this section will focus on the techniques used in the random forest implementation of Scikit-learn, which is the implementation used in this thesis. Scikit-learn is a widely used open source Python framework for machine learning. The Scikit-learn framework provides a lot of algorithm implementations for tasks like regression, classification, clustering, and preprocessing among others [12]. The random decision forest and the aforementioned Scikit-learn were chosen, because they have been used before in the Wisehockey project to prototype similar machine learning approaches with good results.

The random forest classifier is an ensemble learning method, meaning that it combines multiple classifiers to produce a classification. In the case of the random forest classifier, these subclassifiers are called decision trees. So before diving into the operation of the random forest model itself, its principal component, the decision tree needs to be explored.

4.1.1 Decision Trees

Decision tree is a tree-like model that is used for modeling decision making. The branches of the tree represent tests of a certain attribute. Decision trees are used as a tool in operations research and as an illustrative model for different systems. In this thesis, only the machine learning applications are explored. Figure 4.1 represents a simple decision tree for choosing a method for commuting to work with two possible branches from each

decision node. Like can be noted from the figure, decisions trees are easy to visually present and for humans to understand.

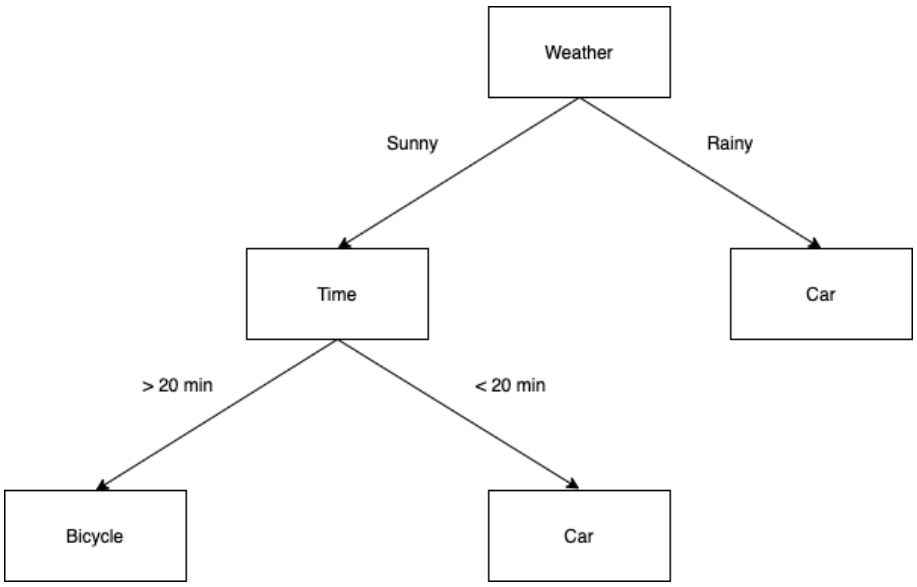

Figure 4.1. *Decision tree for picking a method for commuting*

In machine learning, a decision tree is a supervised learning method, which can be used for either regression or classification. Decision trees used for classification can be called classification trees. In classification trees, the output variable of the tree can only contain values out of a set of discrete values. These discrete output values are called class labels. The specific path from the root of the tree to one of its leaves represents a classification rule.

In machine learning applications specifically, the decision trees are trained on some data set instead of manually setting decision rules. This decision tree approach to machine learning is be called decision tree learning. Classification trees commonly have multiple input variables from which they need to produce the value for the output variable called the class label. The vector of input variables is commonly called the feature vector. There are multiple popular types of decision tree algorithms like CART (classification and regression tree), ID3, C4.5, and C5.0 [13]. The decision tree algorithm utilized by Scikit-learn is the CART algorithm, which will be discussed further [14].

In decision tree learning the feature space is partitioned recursively into regions R_m, where m represents a node in the tree, such that the regions best split up the feature space. The splitting is done by thresholding a feature f with threshold t_m at each node to get a candidate split partitioning the data into $R_m^{left}(f, t_m)$ and $R_m^{right}(f, t_m)$, such that

$$R_m^{left} = \{(x, y) \mid x_f \leq t_m\} \tag{4.1}$$

$$R_m^{right} = \{(x, y) \mid x_f > t_m\} \tag{4.2}$$

Where $x \in \mathbb{R}^n$ is a training vector and $y \in \mathbb{R}^l$ is a label vector. To achieve the aforementioned best split some metric for "best" needs to be defined, this is commonly done by using the converse of the notion of "best candidate split" also called the impurity of the candidate split. Let $Q_m(R_m, f, t_m)$ be the impurity function for node m such that the best split is the candidate split with the lowest impurity value given by Q_m. There are three different commonly used methods for calculating the impurity Q_m: Gini index, misclassification error, and cross-entropy. Since the Gini index is the one used in the final model of this thesis it will be explored further. [13]

Let

$$p_{mk} = \frac{1}{N_m} \sum_{y \in R_m} I(y = k) \tag{4.3}$$

be the proportion of class k observations in node m, where N_m is the amount of observations in node m. Observations in the node m are classified to the class $k(m) = \operatorname{argmax}_k p_{mk}$. In other words the class k with the highest p_{mk} value is chosen as the classification [13]. When the p_{mk} value is calculated for a node m that is a terminal node, then it serves as the probability returned by the classifier for a given class k [14]. $I(y = k)$ here refers to the indicator function returning 1 when $y = k$ and 0 otherwise.

Given the definition for p_{mk}, the Gini impurity of a node m can be defined as

$$Q_m(R_m, f, t_m) = \sum_k^K p_{mk}(1 - p_{mk}) \tag{4.4}$$

In less formal terms the Gini impurity measures how many times a random element from a set would be falsely labeled if it was randomly labeled according to the distribution of labels calculated from a candidate subset. In other words how well does a subset label distribution fit for the superset.

4.1.2 Random Forest

Random forest is a machine learning model that can be utilized for classification and regression. The random forest model works by building multiple de-correlated decision trees during training and then during classification the model outputs the class picked by most of the decision trees within the forest. In other words, the individual classification trees cast a weighted vote on what the output class label should be [13]. Figure 4.2 gives a visual presentation of the random forest classifier. The immediate benefit of the random forest model compared to a single decision tree is that the size of the random forest can be arbitrarily increased to improve accuracy for both training and unseen data [15].

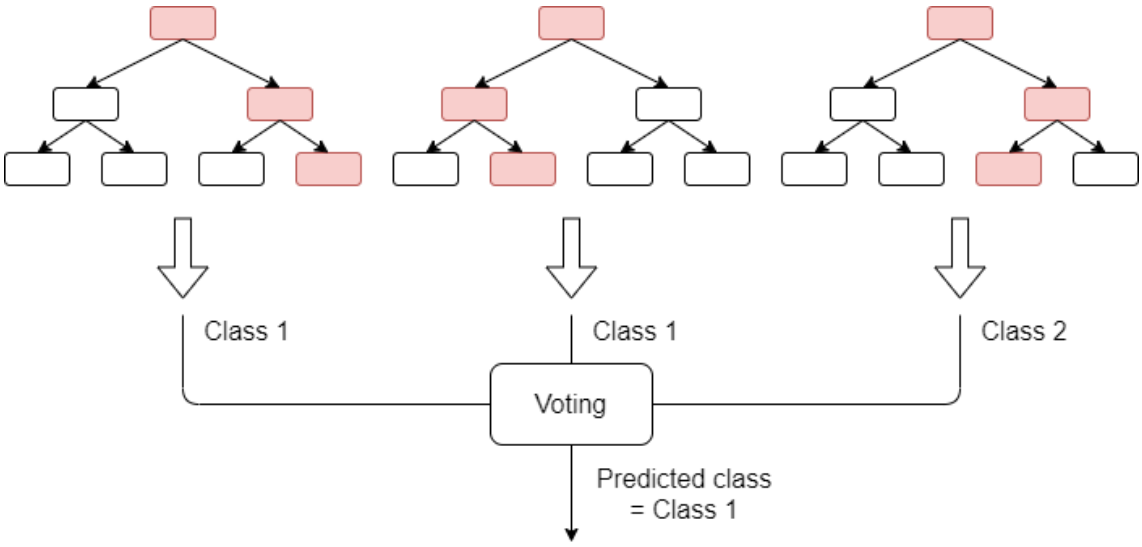

Figure 4.2. *Random forest classifier consisting of three decision trees*

Random forest is an evolution on the bagging method, in which a smaller sample of a training set is sampled with replacement and each sampled subset is used to train a decision tree. Random forest improves on this by sampling a random subset of features, which can be called feature bagging. By sampling a random subset of features the features that are strong predictors will get picked into many of the decision trees. In addition, the random sampling of features causes larger variance between the individual trees, so that they are more decorrelated than in the case if all features were available at every candidate split. For these reasons the random forest method often has a higher classification accuracy than bagging.[13]

4.2 Model evaluation

Model evaluation refers to measuring the performance of a given machine learning model. The aim of model evaluation is to demonstrate the generalizability of a machine learning model to previously unseen data. Assessing the generalizability of a model is crucial

since it works as a guide for picking the right learning model and the right parameters for the given model. [13]

Perhaps the most easily understood metric for classifier performance is classification accuracy. However, using more complex evaluation metrics for classifier performance like ROC curves, in addition to plain classification accuracy is required, because classification accuracy is an inadequate metric in giving a full picture of the performance of classifiers. Using classification accuracy as a metric assumes that the class distribution is balanced and constant, however, classification problems do not commonly have these properties. In fact, classifiers are often used to sift through a large data set looking for anomalies, which would be relatively rare in that data set. In other words, the class specific cost of misclassification is not considered with the classification accuracy metric. [16]

Another important factor in model evaluation is the data used for evaluation. To get the best results the data set should be randomly divided such that there are different subsets for training, model selection/optimization, and model assessment. This can be achieved by splitting the data set into three subsets: training, validation, and test. In a data rich environment, the training data set should only be used for training a model and as such should not be utilized in any performance metrics used to assess the model. The validation data set on the other hand should be used to pick the right model and to optimize the parameters of that model. Lastly, the test data set should be left alone until the final evaluation of the final model. This final evaluation data set gives a more objective measure for the generalizability of the model, since it has not been used in the process of picking or optimizing the model. [13]

This section describes the model evaluation methods utilized in this thesis. The methods described are the receiver operating characteristic curve (ROC), the area under the curve (AUC), and the confusion matrix. These methods are commonly used in the context of evaluating and demonstrating the effectiveness of a classifier.

4.2.1 Confusion Matrix

The confusion matrix is a specific kind of matrix that represents the performance of a machine learning classifier. The main benefit of confusion matrices is that they effectively visualize whether a classifier is confusing two classes. In the case of binary classification, the confusion matrix is a 2 by 2 matrix, where the two columns of the confusion matrix represent the actual class counts and the two rows represent the predicted class counts. [17]

In a binary classifier, there are 4 possible classification outcomes. If both the example and the classification result is positive, it is counted as a true positive (TP). If the same example is classified as negative then it counts as false negative (FN). In contrast, if the

example is negative and classified as positive, then it counts as false positive (FP). If the same sample is classified as negative then it counts as true negative (TN). Given a classifier and a data set, a confusion matrix can be constructed as seen in Figure 4.3. Each cell in the matrix contains the count for the given classification outcome. [17]

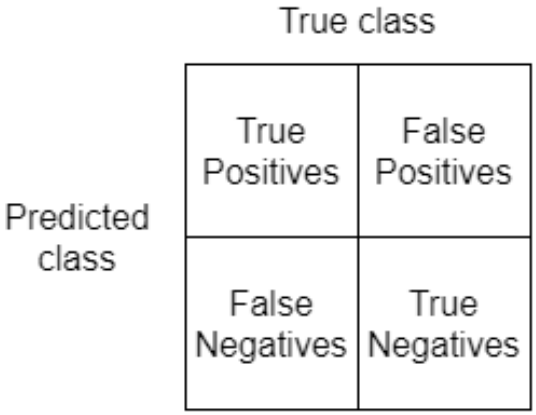

Figure 4.3. *Confusion matrix*

The class balance and class specific classification accuracy can be easily calculated and analyzed from the confusion matrix. Some common metrics calculated from the confusion matrix are

$$Precision = \frac{TP}{TP + FP} \tag{4.5}$$

$$Accuracy = \frac{TP + TN}{P + N} \tag{4.6}$$

$$Recall = \frac{TP}{P} \tag{4.7}$$

$$FP\ rate = \frac{FP}{N} \tag{4.8}$$

$$TP\ rate = \frac{TP}{P} \tag{4.9}$$

where N and P represent the total number of negative and positive class labels respectively. Precision, represented by equation 4.5, represents the ratio between the number of relevant instances and the number of instances classified as relevant. Precision is a very useful measure in the situation where the positive classifications are the interesting ones, but the data set is very unbalanced towards the negative instances. In other words, precision represents the positive predictive value of the classifier. Accuracy, defined by

equation 4.6, is the ratio between the number of true classifications and the number of class labels in total. In other words accuracy represents the predictive value of the classifier considering all classes, not accounting for class balance. Accuracy is perhaps the most familiar metric, because of its intuitiveness and use in everyday life. Recall, represented by the equation 4.7, measures the ratio of the positive classification made and the number of positive class labels. Recall gives a picture of how complete the positive classifications are. In other words, what portion of the interesting (positive) instances did a classifier find. FP rate, defined by equation 4.8 represents the ratio between the number of false positives and the number of negative class labels. FP rate represents the probability of making a false positive classification. TP rate, on the other hand, represented by equation 4.9, is the ratio between the number of true positive classifications and the number of positive class labels, representing the probability of making a true positive classification. The TP rate and FP rate are the measures used in receiver operating characteristic curves. [17]

4.2.2 Receiver Operating Characteristic

Receiver operating characteristic (ROC) graphs are used for visualizing the performance of a classifier. With a simple visualization for the performance of a classifier, it becomes easier to compare different classifiers and their performance for the given problem. ROC graphs are widely used in machine learning research and development. [17]

A ROC graph is a two-dimensional graph that plots the rate of false positives on the x-axis and the rate of true positives on the y-axis. The FP rate and TP rate are calculated as shown in equations 4.8 and 4.9 [17]. The ROC graph can be used in different ways. One way is to plot each classifier to be compared as a single point in the ROC graph as shown in Figure 4.4. This way it is easy to compare the classifiers using the values from the final confusion matrices to calculate the rates after all the test data has been classified by each classifier. The blue line in Figure 4.4 represents a classifier randomly guessing a class. Points below the blue line represent classifiers that work worse than random. However, in a binary classifier worse than random is just a classifier better than random with the classifications inverted, so no point should be under the random line unless an error has been made.

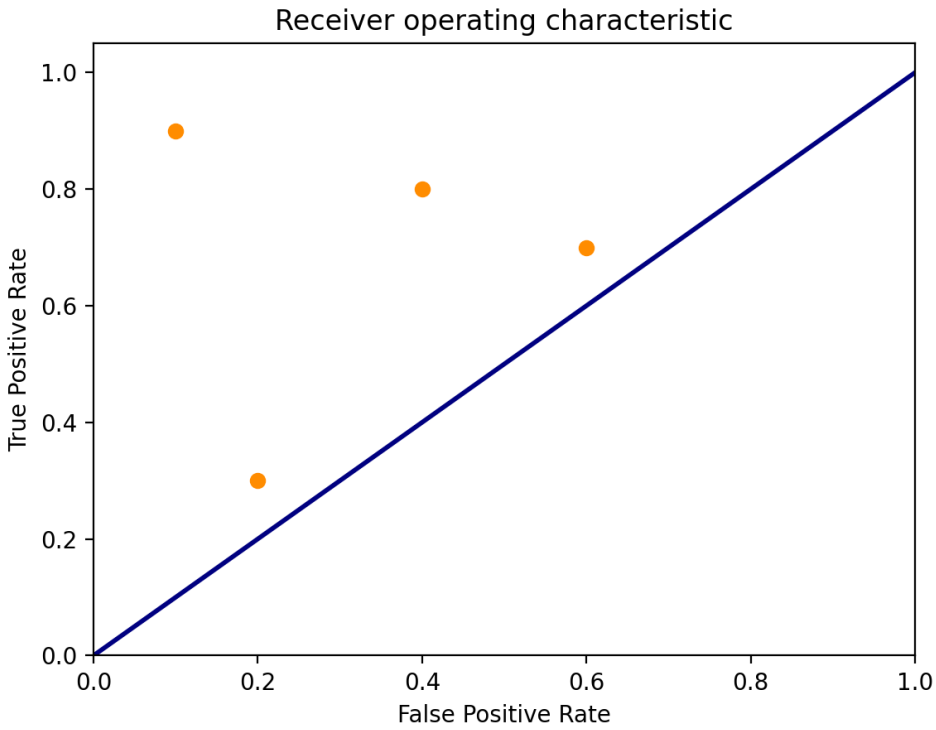

Figure 4.4. *An example ROC graph for comparing classifiers*

Another way to use the ROC graph is to plot a ROC curve. In this case, the probability estimates of the classifier for each classification is considered with different thresholds. Thresholding, in this case, means that the probability score required for positive classification is altered to get each point in the ROC curve. The most basic threshold to use is 0.5, meaning that if the probability score for classification given by the classifier is over 0.5 then the classification is positive and if the score is less than 0.5 the classification is negative. Similar to earlier the FP and TP rates are calculated for each different threshold value resulting in a curve representing the performance of a single classifier with different thresholds [17]. An example of a ROC-curve is given in Figure 4.5.

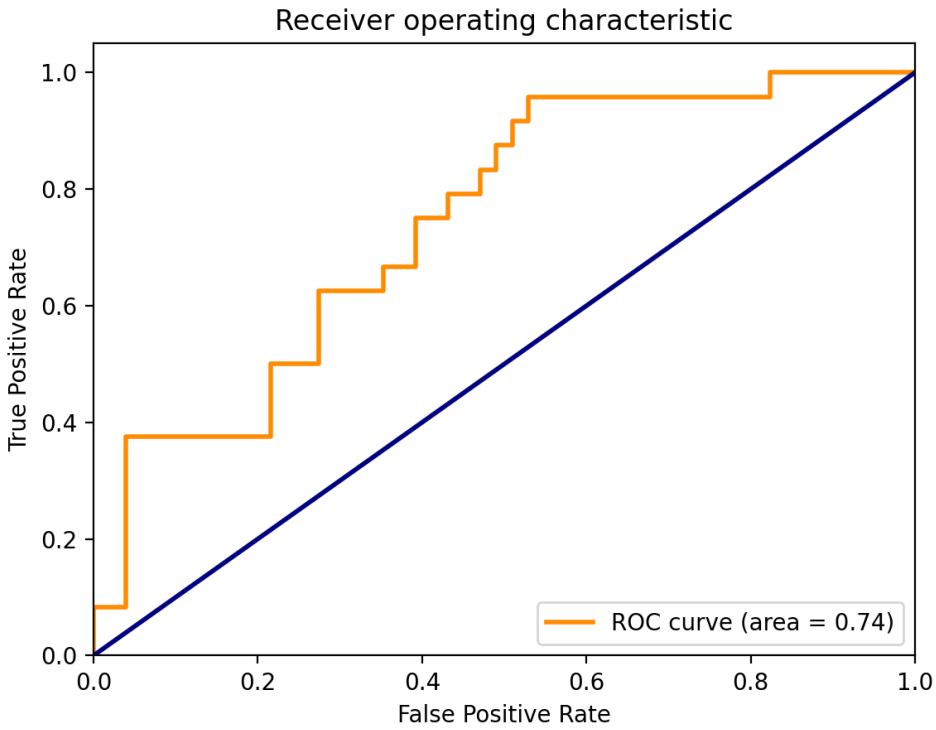

Figure 4.5. *An example ROC curve*

To explain the ROC curve the extreme points on the graph should be considered. The point (0,0) on the graph represents the situation, where the classifier only makes negative classifications, the point (1,1) represents the situation where only positive classifications are made and the point (0,1) represents the situation where all classifications are made perfectly. Generally the better the ROC curve the more it curves towards the top left corner of the graph. Or in other words the higher the area under the curve, the better the result. To easily compare different ROC curves, the curves are commonly reduced to a single scalar value. The aforementioned value is called the area under the curve (AUC). As the name implies, the area under the curve is calculated as the integral of the ROC curve. The AUC will always take on values between 1 and 0 since the highest area that can be covered by a perfect ROC curve is the whole unit square resulting in an AUC of 1. However since curves under the diagonal line which has an AUC of 0.5 are worse than random, AUC values generally take on values between 1 and 0.5. The AUC also has and interesting statistical interpretation of being the probability that a classifier will score a positive instance chosen at random higher than a randomly chosen negative instance. [17]

4.2.3 Feature Importances

Feature importances refer to the relative importance that each feature in a feature vector has on the classification of the target variable. The relative depth of a feature used in a node of a decision tree can be utilized to estimate the importance of a given feature. The higher a feature presents in a decision tree, the more it contributes to the final classification results of said tree. In this sense the higher the feature on a tree, the more important it is. The feature importances of each tree are averaged over multiple random trees to get the feature importances for the whole random forests classifier [18]. Feature importances are commonly represented with a bar chart, like presented in Figure 4.6. The vertical axel represents the percentual importance of a given feature represented by a single bar. The sum of the individual feature importances is one or 100%.

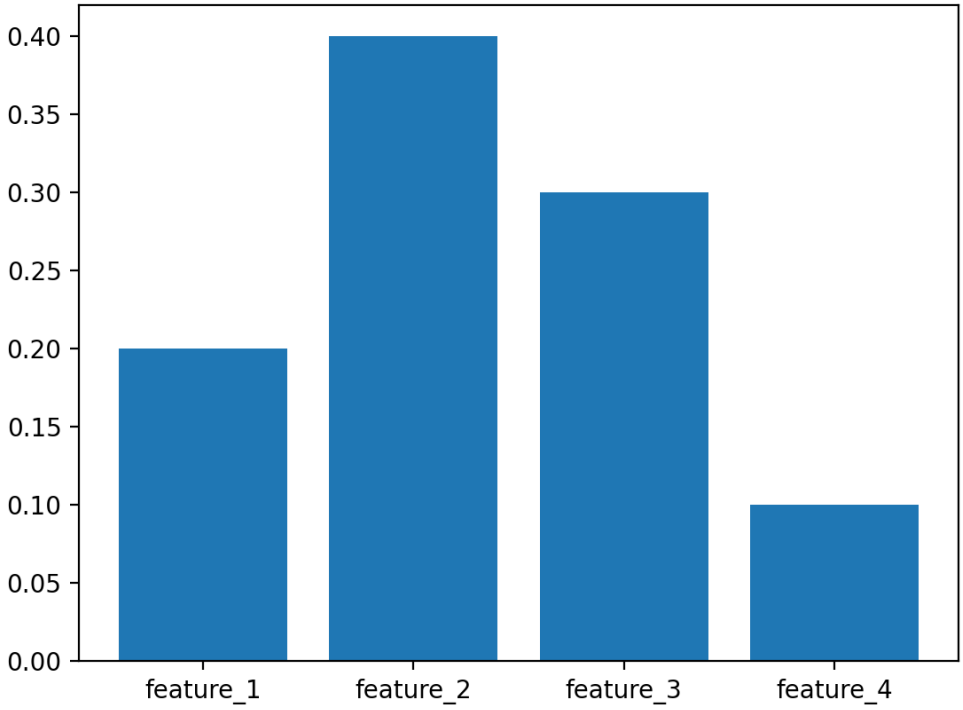

Figure 4.6. *Example feature importance chart*

In this thesis, the feature importances are calculated by the Scikit-learn framework. The impurity based feature importances calculated by the framework are based on purely the training data and as such can not be trusted to give a perfect picture of what the importances are when considering the test data set. In addition to this, the feature importance calculation favors features with higher variability in their values. Meaning that features taking on only a few unique values have smaller feature importances. [18]

5. DATA AND EXPERIMENTAL SETUP

In this chapter, the experiment for finding a model to detect match state changes is presented. A match state change refers to the match state changing from a stoppage of play to active play or vice versa. The more concrete goal of the experiment is to find a model that can classify a given window of time during an ice hockey match to either be active playing time or a stoppage of play, such that the change in the state of the match is temporally close to the real state change. In more formal terms the goal is to find an approximation $\hat{m}(x)$ for a function $m(x) : X \mapsto Y$, where X is the set of possible position data sequences and $Y = \{0, 1\}$ is the set of labels, where 0 represents a stoppage of play and 1 represents active play. Function $\hat{m}$ can be further broken down to functions $e(x) : X \mapsto F$ and $c(f) : F \mapsto Y$, such that $e(x)$ represents the feature extraction function and $c(f)$ represents the classifier, then $\hat{m}(x)$ is given by the composition $\hat{m}(x) = (c \circ e)(x)$. Here F represents the set of possible feature vectors.

This chapter is split into three sections explaining the dataset and preprocessing methodology used, the feature selection performed on the dataset, and the training and model selection. In the first section, the dataset used for the experiment is presented, which consists of 40 professional ice hockey matches with all puck and player position data in addition to match clock data available for each match. The section also explains the preprocessing performed on the position and clock data recorded for each match. The second section on feature selection discusses the features calculated from the dataset to be used as inputs for the learning model. The feature extraction function $e(x)$ mentioned earlier is presented here as well. In the third section, model selection and the training methodology is explained. The approximation of the classifier function $c(f)$ mentioned earlier is discussed in this chapter. This section also discusses the hyperparameter optimization performed on the learning model. All of the algorithms discussed in this chapter were implemented in Python utilizing common open source libraries like Numpy and Scikit-learn [19][12].

5.1 Dataset and Preprocessing

The dataset used in the experiment consists of 40 professional ice hockey matches with all puck position data, player position data, and match clock data included. Most of the

matches were played in different rinks, with measurable differences in rink sizes, although the dataset also contains multiple matches from some rinks. The matches were chosen semi-randomly from a large set of matches, meaning that the matches were picked in order from a random sequence of matches, but all matches with major clock data problems or other major data quality issues were skipped. As explained in section 4.2, it is important in machine learning tasks to use separate datasets for training, validation, and testing. Following this principle, the 40 matches were split into 20 training matches, 10 validation matches, and 10 test matches. Such that training matches were only used for training the classifier, validation matches were used for feature testing and hyperparameter optimization, and the test matches were saved for testing the generalizability of the final model after training and model optimization.

The matches acquired from the Wisehockey system are in a JSONL format, one file per match. JSONL refers to JSON lines, meaning a match data file contains on each row a single JSON object that corresponds to a data point. There are three types of data points: position data point, clock data point, and state change data point. Position data points contain the XY coordinates, a timestamp, and a tag ID. The clock data point contains the time of the match clock and a timestamp. The state change data point contains a timestamp and a value for the game state, either paused or in progress, corresponding to stoppage of play or active play respectively. Examples of the aforementioned data points are shown in Figure 5.1. To be able to actively classify a match in real-time the model cannot utilize a big time window of data. So to teach a classifier, the match data from these large files needed to be broken down further in preprocessing.


```
{
  "tag_id": "example_id",
  "timestamp": "1625980574000",
  "x": -21.91,
  "y": 12.28
}

{
  "timestamp": "1625980574000",
  "match_time": "21:04"
}

{
  "timestamp": "1625980574000",
  "new_state": "PAUSED"
}
```


Figure 5.1. *Different data points contained in each match data file*

As seen in Figure 5.1, the data points themselves don not contain any information on

whether the position was from a puck or a player. This information is of course very important for feature extraction and needs to be added to the data points before the feature extraction phase. The information about which puck or player a given tag belongs to is stored separately in the Wisehockey system. This information is generally referred to as the tag mapping, which is a match specific data structure mapping each player, official, or puck to a specific tag ID. The tag mapping is match specific, because players have to change tags from time to time if the tag runs out of battery and because pucks need to be discarded and replaced as they wear out.

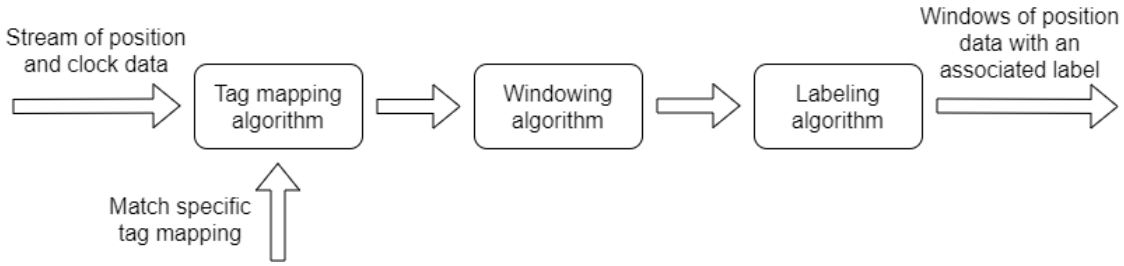

Figure 5.2. *The preprocessing pipeline*

The utilized preprocessing pipeline is shown in figure 5.2. At the start of the preprocessing pipeline, the position and clock data for a match is read from a file in addition to the match specific tag mapping. In the tag mapping algorithm, the relevant tag information is added to each position data point in the data stream. This means that each position data point will contain information on whether the position belongs to a player or a puck and which team in case the position belongs to a player. The modified data stream is then directed to the windowing algorithm.

The windowing algorithm then takes in the data stream and breaks it down to windows of configured size. A window here means an array of data points that all fall inside a specified time period. Before breaking down the data stream the windowing algorithm also sorts the data points, such that they are in chronological order according to their timestamps. The windows are created such that there is a 50% overlap between them. Meaning that the last half of a preceding window contains the same data as the first half of the succeeding window. The reasoning behind the window overlap is to reduce latency in a real-time setting, by effectively halving the buffering required for the algorithm. After the windowing algorithm transforms the position data stream into a stream of windows, the stream proceeds to the labeling algorithm.

In the labeling algorithm, the individual windows are then associated with a class label. The class label is generated by the labeling algorithm by checking for state changes inside the window, such that if an actual state change in the data occurred during the first temporal half of a window, the window would then be labeled corresponding to the state in the rest of the window. Effectively this means that a window is labeled as active play

or stoppage of play based on which state is active for over 50% of the window. After the labeling algorithm, the windows were written to separate files, with the labels recorded in a separate metadata file.

The final configuration for the preprocessing pipeline was to use a 6-second window which then results in a 3-second overlap between the windows. This configuration was reached by testing different window sizes against the validation dataset to determine which gave the best results in a reasonably short window length, usable in real-time classification. After the preprocessing phase, a feature extraction algorithm was applied to each window, producing a feature vector that could be used for classification purposes. The specifics of feature selection and extraction is discussed in the next section.

5.2 Feature Selection and Extraction

In the feature extraction phase, the windows produced by the preprocessing pipeline are turned into feature vectors. Essentially implementing the function $e(x)$ discussed in the beginning of this chapter. The feature extraction algorithm reads a window from file and then proceeds to calculate features from the data points contained in each window. The feature extraction algorithm was designed so that it crudely filtered out position data points outside the rink, such that only positions strictly inside the rink would be considered in the actual feature calculation. After producing the feature vector for a given window, the feature vector was then written to file, such that it could be later accessed in the classifier training process.

The feature selection was done by manually picking features that intuitively encode information about the state of a match. During the feature selection process, the manually picked features were actively tested against the validation data set to try to narrow down the most useful features. The importance of each feature is presented in chapter 6. The final features used in the model are listed below.

- Variance of player x and y positions for each subwindow
- Mean of puck x and y positions for each subwindow
- Variance of puck x and y positions for each subwindow

To try to capture the temporal changes during a time window of multiple seconds the window was further split into multiple subwindows. A simple grid search was executed on the different window and subwindow sizes to find optimal values for each. Since the goal was to be able to classify match state in real-time, the window size had to be small enough to not require a lot of buffering that would not work in a real-time application. The final subwindow configuration was three subwindows of equal size, so in the case of a 6-second window, the subwindows were 2 seconds each. Calculating the above features from at least 2 subwindows (instead of the whole window) significantly improved the end

result presented in chapter 6.

In addition to the features presented, other features were also tested and dismissed. For example:

- Mean of player x and y positions
- Mean of player x and y positions for both teams separately
- Count of players on the ice

Each of these features was not used in the final model for one of two reasons, overfitting to the training data or for having a very low feature importance value. Other features with possibly good prediction capability that were not tried, were the referee position means and variances. These were not tried due to the lack of referee tags in multiple matches in the data set. To summarize, the final feature extraction function $e(x)$ takes in a 6-second sequence of position data and produces a feature vector of 18 values, 6 values for each of the features presented.

The overtime periods caused an extra difficulty in feature extraction due to the count of players on the field changing, from 5 versus 5 to 3 versus 3, not counting the goalkeepers [11]. In addition to the changing player counts, overtimes do not happen in all matches, causing an imbalance in the data set. The imbalance is increased by overtimes commonly being quite short compared to regular play time even when they do happen. When using simple statistical aggregators of player positions like in this thesis the problem is mostly circumvented, but when considering more complicated feature options that capture more details of the player configuration in the rink, the reduced amount of players needs to be taken into account. In addition to overtimes, changing player counts in the rink occur during regular play due to penalties, which causes a similar problem and have to be accounted for.

5.3 Training and Model Selection

The machine learning model chosen for the task was the random forests classifier. The specific implementation used was the random forests classifier provided by the Python machine learning framework Scikit-learn [12]. The choice of random forests classifier was made because of its transparency and low computational requirements. The specific implementation of Scikit-learn was chosen due to the popularity of the library and programmer's familiarity with the library.

The classifier was trained on the feature vectors extracted from the 20 ice hockey matches in the training dataset and then optimized by testing against the 10 matches in the validation dataset. The classifier hyperparameters optimized through grid search were the split quality metric, number of trees, and max features. The hyperparameter max features

refers to the amount of features considered when looking for a split. The best results were achieved by the following configuration:

- Split quality metric function = Gini impurity (default)
- Number of trees = 150
- Max features = square root of total feature count (default)

The hyperparameter optimization did not provide significant improvements to the model over the defaults provided by the library. In fact, only the number of trees parameter was tuned from 100 to 150 during optimization, while the rest of the parameters provided the best results with their default values.

6. RESULTS

In this chapter the results achieved by the methods discussed in chapter 5 are presented by utilizing evaluation methods described in section 4.2. The aim of this chapter is to present the results so that the viability of the tested classifier in a real application can be determined in addition to the overall feasibility of the idea of determining match state with a machine learning approach in real-time.

The chapter is split into two sections. Section 6.1 presents the classifier performance utilizing the model evaluation methods described in section 4.2. In section 6.2, a video analysis is presented to demonstrate the most common error cases of the classifier. The results discussed in this chapter were analyzed and presented utilizing Python and open source libraries like Numpy and Matplotlib [19][20].

6.1 Classifier results

The classifier performed slightly differently on different matches. The ROC curves first discussed in section 4.2 are presented in figures 6.1 and 6.2. The figures show the final ROC curves for the matches in the validation and test data sets respectively. From the ROC curves, it can be seen that both the validation and the test data sets have AUC scores above 0.88 for every match, with most being above 0.95. The accuracies were calculated from all windows in the validation and test data sets utilizing equation 4.6, resulting in accuracies of 92% and 89% respectively. The difference between the validation and test accuracies is likely caused by the hyperparameter optimization performed against the validation data set. It is also possible that the matches in the test data set perform somewhat worse due to rink specific variation or poorer data quality in the matches since the sample size is only 10 matches in each data set.

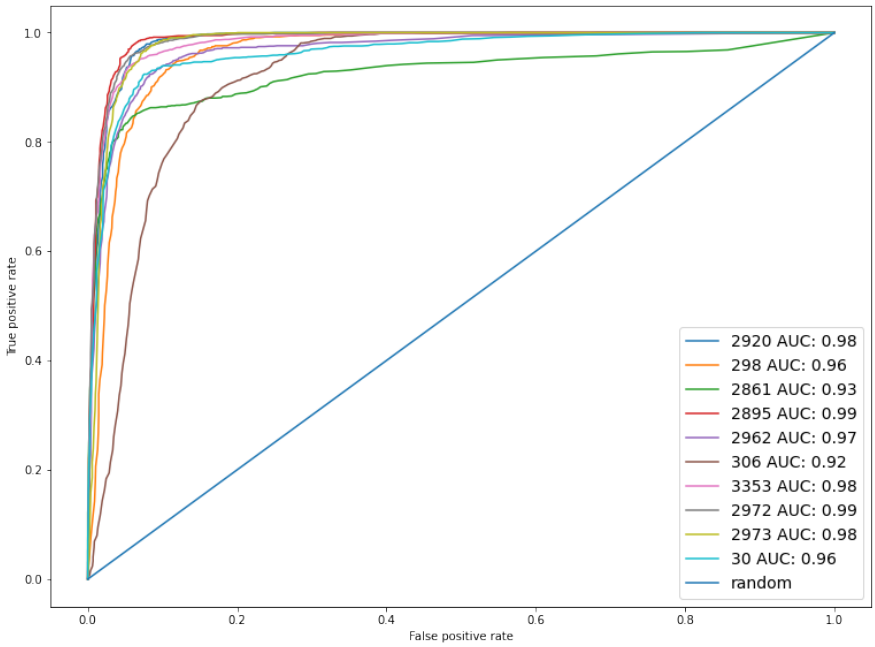

Figure 6.1. *Match specific ROC-curves in the validation data set*

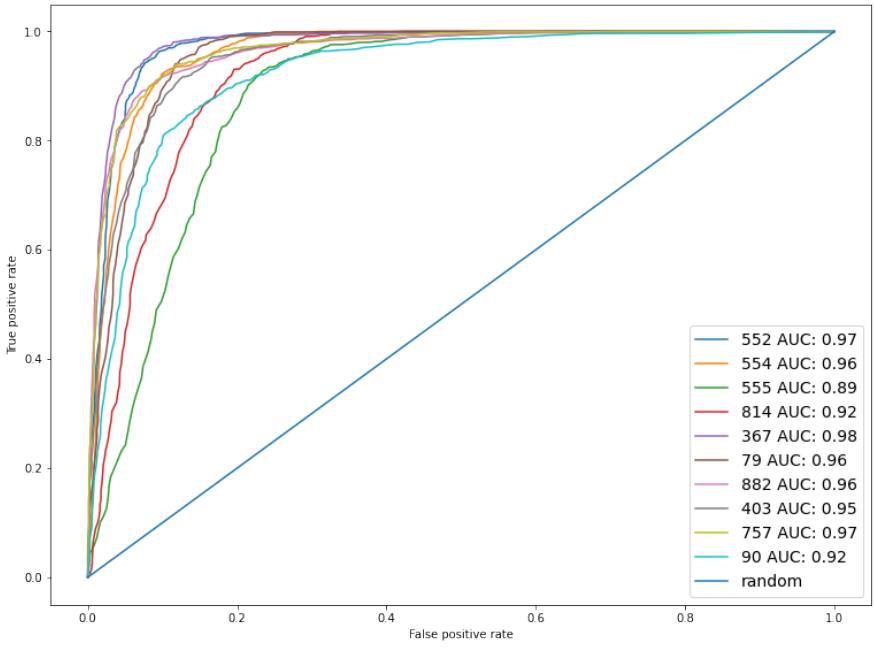

Figure 6.2. *Match specific ROC-curves in the test data set*

Next the confusion matrices first discussed in section 4.2 are presented. The confusion

matrices for both the validation and test data sets are presented in figures 6.3 and 6.4. From the confusion matrices of the validation and test data sets, it can be calculated that the model classified active play with a precision of 93% for the validation data set and 88% for the test data set, implying that there is no great imbalance in the correct classification of either label. This is somewhat surprising given that the stoppages of play during intermissions are easy to classify as stoppages, due to players or pucks being off the ice.

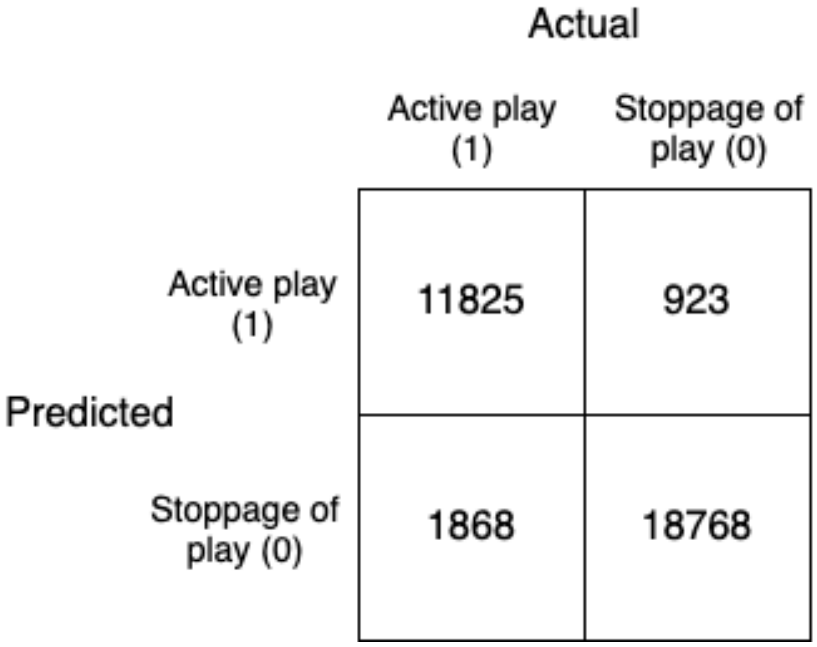

Figure 6.3. *Confusion matrix for all windows in the validation data set*

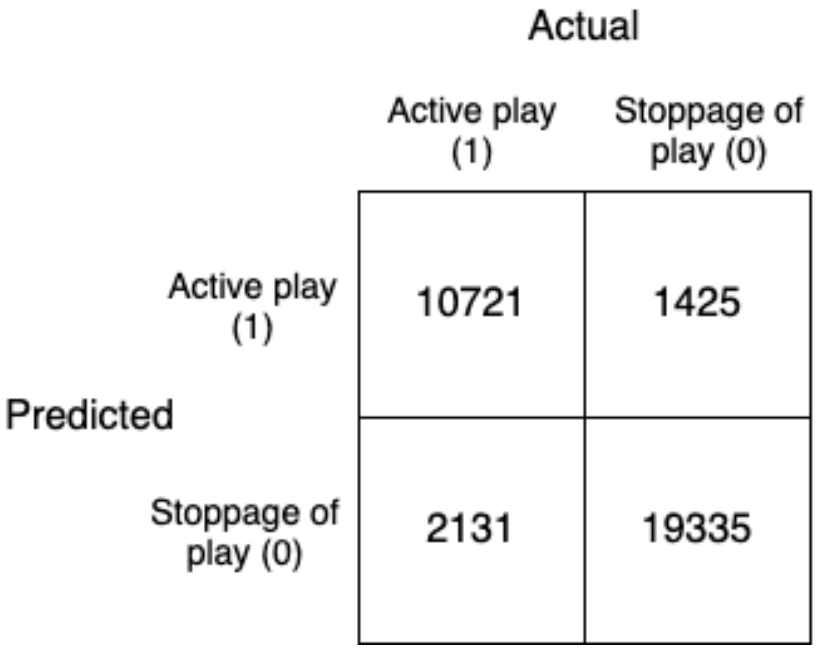

Figure 6.4. *Confusion matrix for all windows in the test data set*

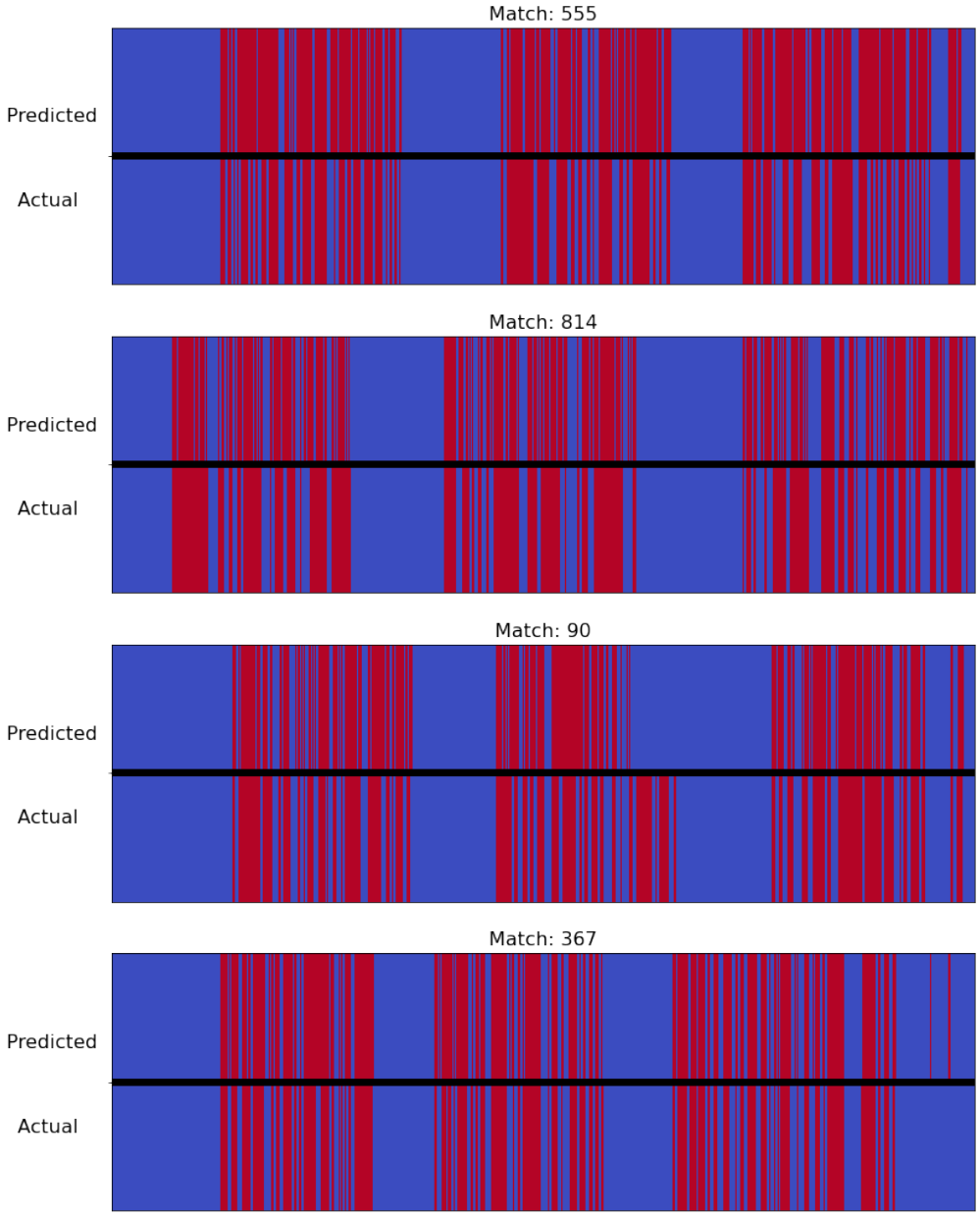

Figure 6.5. *Full match classification examples (red for windows classified as active play and blue for windows classified as stoppages)*

Figure 6.5 illustrates classification results for the full length matches 555, 814, 90 and 367 from the test dataset. The intermissions between periods are clearly visible as the two

large blocks of blue in the middle of the matches. Matches 555 and 814 and 90 were the three worst performing matches and 367 the best performing match in the test dataset as seen in figure 6.2. From figure 6.5, it can be seen that there does not seem to be any large systematic problems with the classification, but instead the classifier miss classifies a few windows here and there. However, in match 90 after the second period there is an error in the ground truth data which causes the additional discrepancy between the actual and predicted windows.

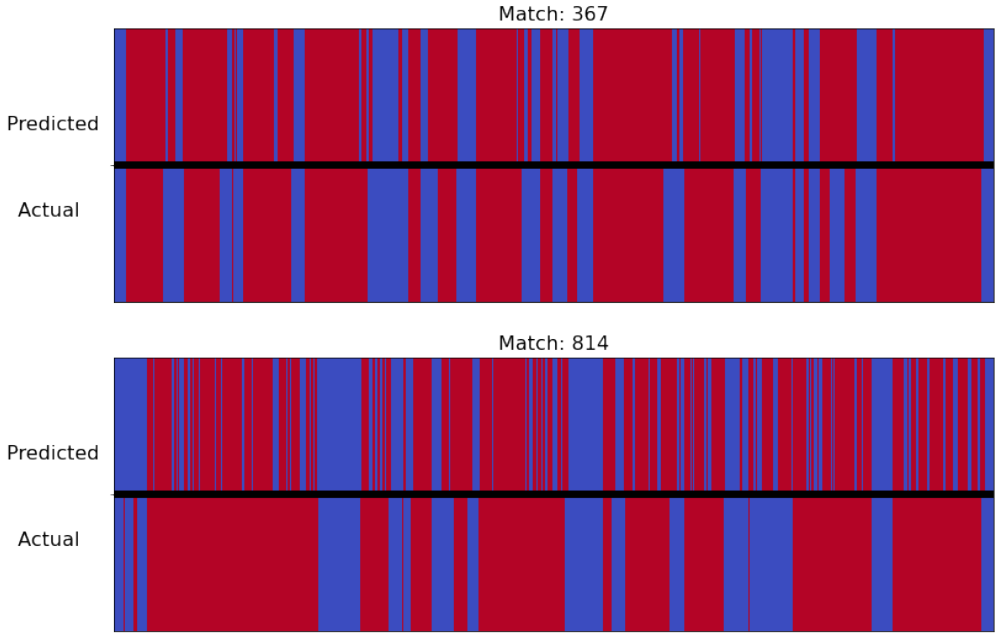

Figure 6.6. *First period classification examples (red for windows classified as active play and blue for windows classified as stoppages)*

Since the matches in the dataset contain a lot of easy to classify pre- and post-match downtime in addition to the intermissions, the results are somewhat skewed and do not give a clear picture of the performance in the non-trivial situations where players are on the ice while the match is not in progress. To give a more clear picture of the performance in these situations, figure 6.6 demonstrates the performance only for the first period of matches 367 and 814 in the test dataset. The classification accuracies of the match segments shown in figure 6.6 are 88% for match 367 and 76% for match 814, whereas the corresponding accuracies for classifying the whole matches were 93% for 367 and 84% for 814. So, for match 814, which was one of the worst performing matches in the test dataset overall, there is an 8 percentage point drop in accuracy when only considering the duration of the first period instead of the whole match.

To give a more clear picture of how well the classifier performs in predicting if active play is taking place, Figure 6.7 shows the seconds of true active play and predicted active play for each mach in the matches of the test data set. From the figure it can be seen that the classifier produces too much play time for some matches and too little for others. However, in the most problematic matches the classifier has predicted a lot more playing time than actually took place.

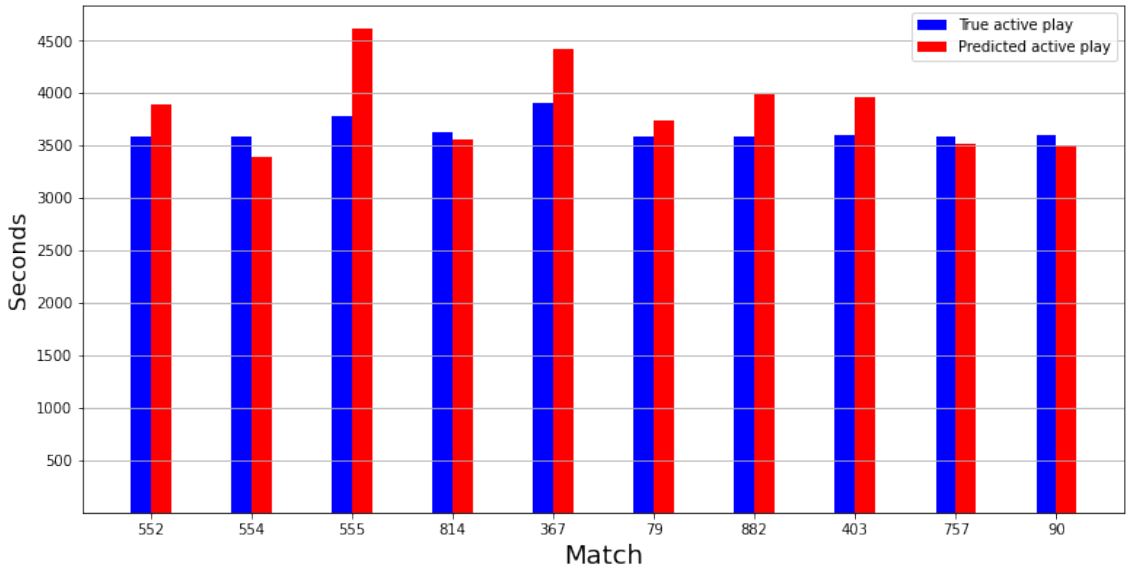

Figure 6.7. *Total seconds of true play vs predicted play for matches in the test data set*

The feature importances calculated for the classifier by the Scikit-learn framework are presented in figure 6.8. In the right-side graph, the features are differentiated by the subwindow index. From the left side graph, it can be seen that all features are important in the classification process, but the variance of puck positions is the most important feature. From the right side graph where the feature importances are differentiated by subwindow index, it can be observed that the features from the third and last subwindow have a higher importance than the features from earlier windows. Since the features are all statistical aggregates, the classification is very much dependent on the functioning of the puck and likely not very sensitive to different player configurations. Meaning that It is feasible for only a few players on the ice to produce a similar player position variance value as more players that are clustered.

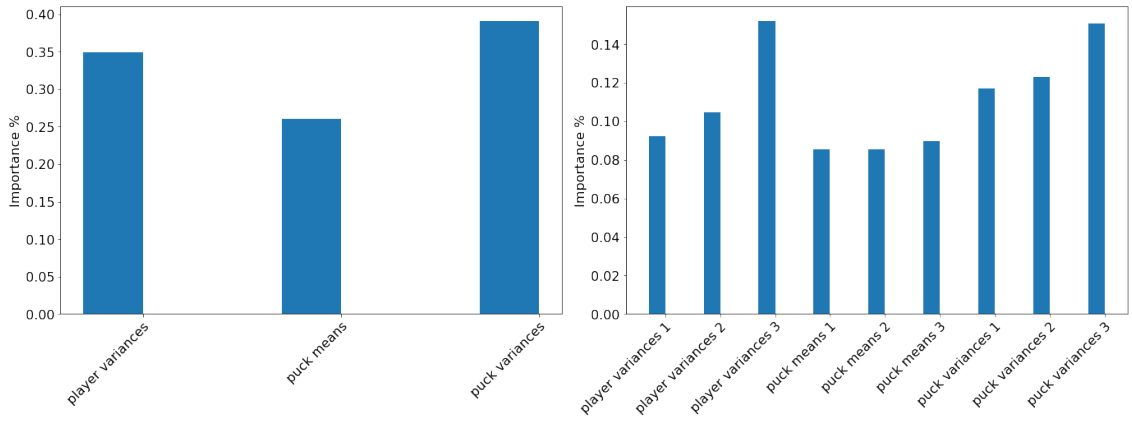

Figure 6.8. *Feature importances (the number indices refer to the subwindow from which a feature was calculated)*

Like explained in section 4.2, with the given feature importances it has to be noted, that the Scikit-learn framework calculates the impurity based feature importances from the training set specifically. Meaning that the feature importances could be somewhat misleading when considering how important a given feature is in the case of the test set [18].

The results presented in this chapter show that a reasonable classification accuracy for real-time match state classification is possible, but the performance of the presented model is likely not good enough for any real world application.

6.2 Error case analysis

To narrow down the errors that the classifier makes, a manual video analysis was carried out. The error cases were picked from match 2972 in the validation set, due to the easy availability of video. It should be noted here that match 2972 is one of the best performing matches in the validation data set, with an AUC of 0.99 and thus the errors demonstrated will be exacerbated in other matches. The collages in figures 6.9, 6.10 and 6.11 demonstrate the most common situations where classification errors are made based on visual analysis. The individual pictures in each collage have an overlay on top of the puck displaying the state determined by the classifier.

In 6.9 the referee has called an icing and is delivering the puck back to the other end of the rink for a face-off, while the players are also getting ready to take face-off positions. The classifier is momentarily mistaking the match state as running, which in this case is a false positive. In this example, the puck is moving fast across the rink causing changes in the position mean and variance of the puck. This issue could perhaps be fixed by having position data on referees and using it as a feature since referees tend to stay away from the puck during actual play, but during short breaks, they often transport the puck to the

face-off site.

Figure 6.9. *Classified match state is running while referee is skating with puck in hand to the other end of the rink after calling icing*

Figure 6.10 demonstrates an erroneous short break during normal play at the middle of the rink. In this instance, the classifier is making a false negative classification. In this situation the puck is quite close to the face-off spot and also quite stationary for a while, which is likely causing the misclassification in this instance.

Figure 6.10. *Classified match state shortly changing to a break during play at the middle of the rink*

In figure 6.11 another erroneous short break is classified during normal play at the other end of the rink. Again the face-off spot is quite close and the movement of the puck

dies down due to the players maneuvering in a small space causing low player position variances, near constant puck position mean, and low puck position variance. Both this and the error case displayed in 6.10 could likely be fixed by a richer feature set, capturing more of the specific configuration of the players in the rink.

Figure 6.11. *Classified match state changing to a break during play at the other end of the rink*

In addition to these clear cases of the classifier performing badly, there are more subtle error cases. In both face-off situations and when a stoppage of play is called there are sometimes significant delays (multiple seconds) before the right classification is made. This issue can likely be improved with the same methods as the more clear error cases, but shortening the classification window length will also likely improve this aspect significantly.

To summarize, the cases where a false positive classification is made are situations where both the players and the puck are still actively moving during a stoppage of play. The false negatives on the other hand are situations where there is relatively little player and puck movement near a face-off spot during active play.

7. CONCLUSION

The topic of this thesis was to test the feasibility of using machine learning techniques to determine the match state in ice hockey. More specifically the goal was to train a classifier that would take in features extracted from player and puck position data and output the match state at given time. As explained before, match state refers to whether active play is taking place or if there is a stoppage. In other words whether the match clock is running or not. This goal was approached in this thesis by training a random forest classifier to classify a feature vector derived from a window of position data. A window here refers to an array of position data falling inside a time window of specified length, such that each window would then be paired with a label of active play or stoppage of play by the classifier.

The motivation for this experiment is to be able to generate clock data, by knowing when the match clock is running or not based only on position data. The idea being that by being able to tell whether the match clock is running or not in real-time, it would be possible to generate the clock data by running and pausing a timer based on the classification result. This is desired in situations where it is impractical to integrate with the match clock system at an ice hockey rink, so that statistics relying on match clock information can still be produced by Wisehockey.

By the results presented in chapter 6 it is clear that the feature set and model used in this thesis is not good enough for real world use, but the overall approach could work if improved upon. The results show promise overall in using a machine learning approach to the given problem. Due to unbalances in the data sets, the ROC curves and AUC values presented for the complete matches are very good. However, a lot of the windows in these matches are easy to classify and don't represent the real technical problem presented. The easy to classify windows being the ones during intermission and before and after the start of the match, when there are no players or pucks on the ice.

Based on the simplicity of the features utilized in classification and the error case analysis presented in chapter 6, it is evident that more rich features could significantly improve the result. Some possible features that could significantly improve the results are features based on referee positions like discussed in the error case analysis section. Other possible features are richer representations of the player configuration on ice, with respect to

the puck, instead of the very simple statistical aggregates used in the experiment.

In addition to different features, different machine learning models could be considered as well. The Wisehockey system has a lot of annotated data for the given problem and only a small part of the available data was used in the experiment. For this reason, a lot more computationally heavy approaches could be attempted, like deep neural networks. An approach like DNN would reduce the need for feature engineering and could potentially work almost end to end, such that the raw positions could almost directly be given as input for classification. However, the raw position data is irregular in the sense that different amounts of entities are on the ice at any given time and different amounts of position data points are received at any given moment, making statistical aggregates the most simple approach in terms of preprocessing and feature extraction. In the end, the reason other machine learning models were not attempted in this thesis was due to the computational limitations of a laptop and also due to time limitations allocated for the research project.

REFERENCES

[1] *Advancing sports analytics through AI research*. Google, Deepmind. 2021. URL: https://deepmind.com/blog/article/advancing-sports-analytics-through-ai.

[2] Tuyls, K., Omidshafiei, S., Muller, P., Wang, Z., Connor, J., Hennes, D., Graham, I., Spearman, W., Waskett, T., Steele, D., Luc, P., Recasens, A., Galashov, A., Thornton, G., Elie, R., Sprechmann, P., Moreno, P., Cao, K., Garnelo, M., Dutta, P., Valko, M., Heess, N., Bridgland, A., Perolat, J., Vylder, B. D., Eslami, A., Rowland, M., Jaegle, A., Munos, R., Back, T., Ahamed, R., Bouton, S., Beauguerlange, N., Broshear, J., Graepel, T. and Hassabis, D. *Game Plan: What AI can do for Football, and What Football can do for AI*. 2020. arXiv: 2011.09192 [cs.AI].

[3] Curran, K., Furey, E., Lunney, T., Santos, J., Woods, D. and McCaughey, A. An evaluation of indoor location determination technologies. eng. *Journal of location based services* 5.2 (2011), pp. 61–78. ISSN: 1748-9725.

[4] Bensky, A. *Wireless positioning technologies and applications*. eng. Second edition. GNSS Technology and Applications Series. Boston, [Massachusetts] ; Artech House, 2016 - 2016. ISBN: 1-5231-1771-0.

[5] *Direction Finding*. Bluetooth SIG, Inc. 2021. URL: https://www.bluetooth.com/learn-about-bluetooth/recent-enhancements/direction-finding/.

[6] *Quuppa technology overview*. 2021. URL: https://www.quuppa.com/technology/overview/.

[7] *Bluetooth Technology Overview*. Bluetooth SIG, Inc. 2021. URL: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/.

[8] Wisehockey Oy. *Real-Time Sport Analytics Platform*. 2020. URL: https://wisehockey.com/wp-content/uploads/2020/02/wh_overview_en.pdf.

[9] *Quuppa Case Study: Liiga & Veikkaus*. 2021. URL: https://www.quuppa.com/case-study-liiga-veikkaus/.

[10] *Wisehockey, Real-Time Hockey Analytics*. URL: https://wisehockey.com/.

[11] International Ice Hockey Federation (IIHF). *2020/21 – 2021/22 Season IIHF SPORT REGULATIONS*. 2021. URL: https://blob.iihf.com/iihf-media/iihfmvc/media/downloads/regulations/2021/2021-iihf-sport-regulations.pdf.

[12] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cour-

napeau, D., Brucher, M., Perrot, M. and Duchesnay, E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[13] Hastie, T., Tibshirani, R. and Friedman, J. *The Elements of Statistical Learning, Second Edition*. Springer Science, 2013. 767 p.

[14] *Decision Trees*. scikit-learn developers. 2021. URL: https://scikit-learn.org/stable/modules/tree.html#tree.

[15] Ho, T. K. Random Decision Forests. (1995).

[16] Provost, F. and Fawcett, T. Analysis and Visualization of Classifier Performance: Comparison under Imprecise Class and Cost Distributions. (1997).

[17] Fawcett, T. An introduction to ROC analysis. *Pattern Recognition Letters 27 (2006) 861–874* (2005).

[18] *Ensemble methods*. scikit-learn developers. 2021. URL: https://scikit-learn.org/stable/modules/ensemble.html.

[19] Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. and Oliphant, T. E. Array programming with NumPy. *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[20] Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.