ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Εφαρμογές της υπολογιστικής γεωμετρίας στην ανάλυση ποδοσφαιρικών αγώνων και πρόβλεψη αποτελέσματος αγώνων με χρήση διαγραμμάτων Voronoi

Απόστολος Σ. Θεοδώρου

Επιβλέποντες: Ιωάννης Εμίρης, Καθηγητής, ΕΚΠΑ
Ιωάννης Χαμόδρακας, Εργαστηριακό Διδακτικό Προσωπικό (ΕΔΙΠ), ΕΚΠΑ

ΑΘΗΝΑ

Δεκέμβριος 2024

NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

BSc THESIS

Applications of Computational Geometry in Football and Result Prediction Using Voronoi Diagrams

Apostolos S. Theodorou

Supervisors: Ioannis Emiris, Professor, UoA
Ioannis Chamodrakas, Member of the Laboratory Teaching Staff, UoA

ATHENS

December 2024

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Εφαρμογές της υπολογιστικής γεωμετρίας στην ανάλυση ποδοσφαιρικών αγώνων και πρόβλεψη αποτελέσματος αγώνων με χρήση διαγραμμάτων Voronoi

Απόστολος Σ. Θεοδώρου
Α.Μ.: 1115201500046

ΕΠΙΒΛΕΠΟΝΤΕΣ: Ιωάννης Εμίρης, Καθηγητής, ΕΚΠΑ
Ιωάννης Χαμόδρακας, Εργαστηριακό Διδακτικό Προσωπικό (ΕΔΙΠ), ΕΚΠΑ

BSc THESIS

Applications of Computational Geometry in Football and Result Prediction Using Voronoi Diagrams

Apostolos S. Theodorou
S.N.: 1115201500046

SUPERVISORS: Ioannis Emiris, Professor, UoA
Ioannis Chamodrakas, Member of the Laboratory Teaching Staff, UoA

ΠΕΡΙΛΗΨΗ

Στις μέρες μας το ποδόσφαιρο, ιδίως το επαγγελματικό, δεν αποτελεί μόνο έναν τρόπο ψυχαγωγίας που απευθύνεται σε ποδοσφαιριστές και φιλάθλους, αλλά συνιστά επίσης πεδίο τόσο οικονομικής δραστηριότητας όσο και επιστημονικού ενδιαφέροντος. Η επιστήμη της υπολογιστικής γεωμετρίας χρησιμοποιείται ευρέως από πολλές ομάδες για την ανάλυση δεδομένων των ίδιων των ομάδων αλλά και των αντιπάλων τους. Σκοπός της παρούσας εργασίας είναι η παρουσίαση μερικών από τις εφαρμογές της υπολογιστικής γεωμετρίας στην ανάλυση ποδοσφαιρικών αγώνων. Επιπρόσθετα η δημιουργία ενός μοντέλου αναγνώρισης και πρόβλεψης αποτελέσματος αγώνα, το οποίο θα βασίζεται στο διάγραμμα Voronoi.

Η εργασία διαρθρώνεται με τον εξής τρόπο. Αρχικά θα παρουσιάσουμε δύο θεμελιώδη εργαλεία της υπολογιστικής γεωμετρίας, το κυρτό περίβλημα και το διάγραμμα Voronoi και θα ανατρέξουμε στη βιβλιογραφία, εντοπίζοντας διάφορες χρήσεις τους στην ανάλυση αγώνων και δεδομένων. Στη συνέχεια θα παραθέσουμε το σχετικό επιστημονικό υπόβαθρο των πιθανοτήτων και της μηχανικής μάθησης που απαιτείται για τη δημιουργία των δικών μας μοντέλων. Αφού αναφέρουμε άλλες έρευνες στον τομέα της πρόβλεψης αγώνων ποδοσφαίρου και τα αποτελέσματά τους, θα κατασκευάσουμε ένα δικό μας μοντέλο, σκοπός του οποίου είναι η αναγνώριση του αποτελέσματος αγώνων που έχουν ολοκληρωθεί, με βάση τα διαγράμματα Voronoi των τελικών φάσεών τους. Έπειτα, εφαρμόζοντας κάποιες τροποποιήσεις στον αλγόριθμο, θα δημιουργήσουμε ένα μοντέλο που προβλέπει το αποτέλεσμα μελλοντικών αγώνων. Τέλος, θα παρουσιάσουμε τα αποτελέσματα των προσπαθειών μας, σχολιάζοντάς τα και θα συζητήσουμε ορισμένες βελτιώσεις που δύνανται να πραγματοποιηθούν μελλοντικά.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Υπολογιστική Γεωμετρία, Μηχανική Μάθηση

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Διάγραμμα Voronoi, Ταξινόμηση, Πρόβλεψη Αποτελέσματος, Ποδοσφαιρική Ανάλυση

ABSTRACT

Nowadays football, especially when played on a professional level, is not just a way of entertainment that brings players and fans together, but also a field of economic activity and scientific interest. Computational geometry is widely utilized by many teams in order to analyze their data and data that concern their opponents. This thesis aims to present some of the applications of computational geometry in the analysis of football matches. Moreover another purpose is the construction of a reckoning and prediction model of outcomes of football matches, based on the Voronoi diagram.

This thesis is organized as described below. Firstly we will present two fundamental tools of computational geometry, convex hull and Voronoi diagram and we will detect several uses of them in football analysis in the literature. Secondly we will expand on the related scientific field of probabilities and machine learning, which is necessary for the construction of our own models. Before building our first model, whose purpose will be the reckoning of completed matches based on the Voronoi diagrams of their highlights, we will examine the scientific research on football prediction and the results so far. Then we will modify our algorithm to create a second model, which will predict outcomes of future matches. Lastly, we will demonstrate and comment on the results of our attempt and will discuss some improvements that could take place in the future.

SUBJECT AREA: Computational Geometry, Machine Learning

KEYWORDS: Voronoi Diagram, Classification, Match Prediction, Football Analysis

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Indisputably, modern football is not just a game of skill, but also a strategy game. It is not unusual for underfunded teams with a dubious reputation (underdogs) to outperform historical teams with universally recognised players. This fact clearly illustrates that except for players' individual abilities, choosing the particular style of play, which combines most effectively each player's virtues, contributes to a great extent to a team's success.

Despite the fact that football is a team sport, collaboration and team tactics have not always been an integral part of it. A typical example is that in England, where modern football originated, passing used to be a very rare phenomenon until the end of the 19th century. At that time, many Scottish players migrated south to join English football clubs due to professionalism, introducing other styles of play. Apart from professionalism, the rapid spread of football all over the world, especially across Europe and South America, led to the enrichment of the sport with numerous new tactical variations.

Furthermore, in this day and age, the increasing TV coverage, combined with the introduction of new technology in athletics, opened the way for a more assiduous and systematic study and analysis of the applied tactics and playing systems. In this analysis, the concept of space plays a crucial role. For instance, choosing a proper formation that helps the players minimize empty space when defending or create empty space near the opponent's goal area through unmarking movements and deliberate passing patterns is one of the most important decisions a coach has to make when planning a match. Computational geometry has a significant contribution to football analysis, as it provides the necessary tools to model space and draw useful conclusions regarding the team's strategy and tactics.

2. BACKGROUND AND RELATED WORK

In this chapter we will provide some fundamental knowledge from the field of computational geometry about some concepts that are widely used in football analysis, such as the convex hull and the Voronoi diagram. After the establishment of that basic knowledge with definitions and a short reference to relevant algorithms, we will inspect the corresponding bibliography to understand the exact manner in which these concepts are being put into practice when analyzing a football match. Then, we will present some necessary background material regarding machine learning algorithms and techniques, in order to discuss the work that has been done in the direction of predicting the outcome of football matches.

In more detail, in section 2.1 we define the concepts of polygonal line, polygon, and convexity so that we are able to define the concept of the convex hull. Following this, we briefly mention the known algorithms for constructing the convex hull of a set of points in the plane. In section 2.2 we approach the concept of the Voronoi diagram, while also making a reference to the most popular construction algorithms. In section 2.3 we focus on the use of the previous concepts in football analysis by examining related papers. In section 2.4 we briefly present some machine learning algorithms, as a prerequisite knowledge for the next section. Finally, in section 2.5 researches and related work on attempting to predict the outcomes of football matches are discussed.

2.1 Convex Hull

2.1.1 Definition

Consider a finite sequence of line segments $(e_1, e_2, ..., e_n)$, where each segment e_i is defined by an ordered pair of points. The sequence is called closed if the ending point of the segment e_i, for $i = 1, 2, ..., n-1$, coincides with the initial point of the next segment e_{i+1} and the ending point of segment e_n coincides with the initial point of e_1. Every (closed) sequence of line segments is also called (closed) polygonal line. [16]

Consider a finite closed sequence of line segments. The region of the plane that is interior of the sequence is called a closed polygon, or more simply a polygon. [16]

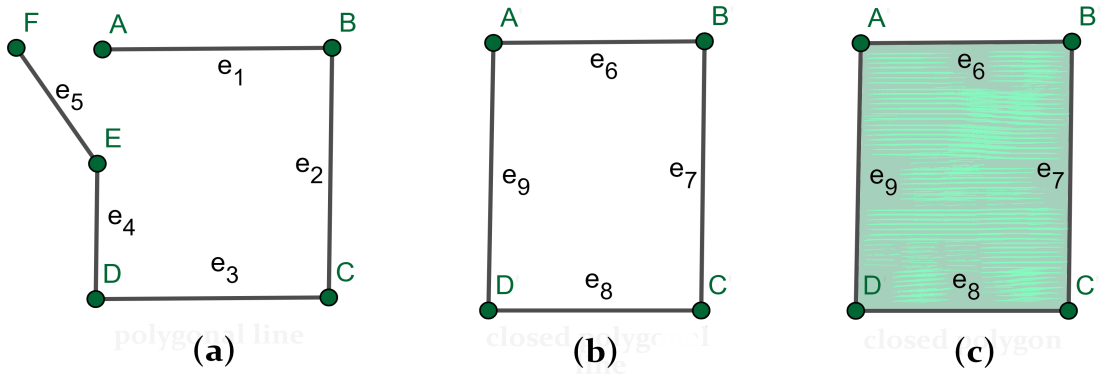


Figure 2.1: (a) A polygonal line, (b) a closed polygonal line, (c) a closed polygon

Any object, or set of points P, is called convex if and only if for any pair of points $p, q \in P$, the line segment pq is completely contained in P. [11]

The convex hull of a set of points P in the plane is the convex polygon of the minimum area that contains all the points in P. [16]

An alternative definition of the convex hull of a finite set of points P in the plane is the following: it is the unique convex polygon whose the vertices are points from P and that contain all points of P. [11]

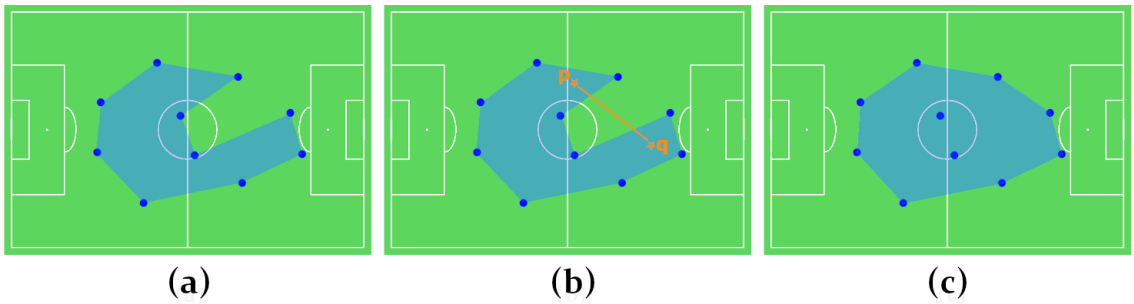


Figure 2.2: (a) One possible non-convex closed polygon whose vertices are the positions of players on the football pitch at a specific moment. (b) Two points that belong to the closed polygon and their line segment violates the property of convexity. (c) The convex hull of position of the players on the pitch

2.1.2 Construction Algorithms

The first algorithm for the construction of the convex hull of a finite planar set was suggested in 1972 by Graham [34]. His algorithm, known as the Graham scan, uses a backtracking technique and has time complexity $\mathcal{O}(n \log n)$. The algorithm initially determines a point that lies on the interior of the convex hull as a reference point and an arbitrary half-line, so that each point in the set can be expressed in polar coordinates. Then the points

are ordered in terms of increasing angle and some points that clearly lie in the interior are deleted. In the last step, triplets of consecutive points are examined, and depending on the angle they form, it is either confirmed that the first point is an extreme point of the convex hull or the middle point is deleted, as it lies in the interior of it. The algorithm ends when all the points have been examined.

The second algorithm was suggested in 1973 by Jarvis [27]. He used a simple wrapping technique, known as gift wrapping or Jarvi's march, which computes the convex hull of a given set of points in $\mathcal{O}(nh)$ time, where h is the number of vertices the convex hull consists of. Starting with the leftmost point of the set, which definitely belongs to the convex hull, the algorithm gradually adds vertices in a clockwise direction, by checking that the edge created due to the last addition has all the remaining points lying on its right side. Gift wrapping is an output-sensitive algorithm, meaning that its running time depends on the size of its output. In the worst case, when all the points are vertices of the convex hull, the algorithm's complexity becomes $\mathcal{O}(n^2)$. However, most of the times, only a small subset of the points belongs to the convex hull, so in practice, gift wrapping tends to be faster than Graham's scan.

In 1977 Preparata and Hong approached the problem with the "divide and conquer" technique [18]. They developed an algorithm that recursively subdivides the initial set of points in two subsets, computes each subset's convex hull, and then merges the two convex hulls by tracing two tangents, common to both hulls. Their algorithm has $\mathcal{O}(n \log n)$ time complexity.

Another algorithm based on the "divide and conquer" technique, was published independently later that year and in 1978 by Eddy and Bykat respectively [13] [6]. The algorithm got the name QuickHull from the famous QuickSort algorithm due to the similarity they have. It works mainly by computing triangles and excluding points that lie in their interior from the vertices of the convex hull. The mean time complexity of this approach is also $\mathcal{O}(n \log n)$, but it can go up to $\mathcal{O}(n^2)$ in the worst case, where the points are not distributed randomly enough.

In 1979 Andrew published his algorithm, which is a variation of the Graham's scan, where the points are sorted in lexicographical order [1]. Thus, the time complexity is the same as Graham's scan, $\mathcal{O}(n \log n)$. The algorithm starts from the leftmost point and computes the upper half of the convex hull moving in a clockwise direction and is completed when the lower half of the convex hull has been computed, beginning from the rightmost point and ending at the leftmost one.

An incremental algorithm was presented in the '80s by Kallay and was later improved by Edelsbrunner [29] [14]. The algorithm, named beneath-beyond, presorts the points along their x- coordinate and processes them in increasing order. The triangle defined by the first triplet of points is the initial convex hull. In every next step of the algorithm, the next point of the set is added to the convex hull and the edges that are visible from that point are deleted. The hull is growing iteratively until the last point is added. The initial pre-processing step of the sorting is the most expensive operation. Therefore, its time complexity is $\mathcal{O}(n \log n)$. This is the upper bound in both average and worst case.

In 1986 Kirkpatrick and Seidel presented the first optimal output-sensitive algorithm with time complexity $\mathcal{O}(n \log h)$. They achieved this by using a variation of the ”divide and conquer” paradigm, which they called ”marriage before conquest” [32]. While in the traditional ”divide and conquer” technique the original problem is firstly divided into subproblems, then these subproblems are solved and ultimately, their solutions are combined to form the final solution, in Kirkpatrick and Seidel's approach the last two steps are reversed. It is first determined how the solution, that has not been computed yet, will be combined and then the subproblems are actually solved. In that way, their algorithm manages to exclude redundant points in the second step, thus minimizing the input size of the subproblems that are being solved in the last step.

A decade later, in 1996, Chan came up with another optimal output-sensitive algorithm, which runs in $\mathcal{O}(n \log h)$ time as well, by expanding the idea of gift wrapping, combining it with other known convex hull construction algorithms, like Graham's scan [8]. The basic idea is to partition the initial set of points into smaller groups at first and compute each group's convex hull with an algorithm of $\mathcal{O}(n \log n)$ complexity, producing some convex polygons. Then those polygons are wrapped together with Jarvi's technique in a bigger convex polygon, which is the convex hull of the whole set.

2.2 Voronoi Diagram

2.2.1 Definition

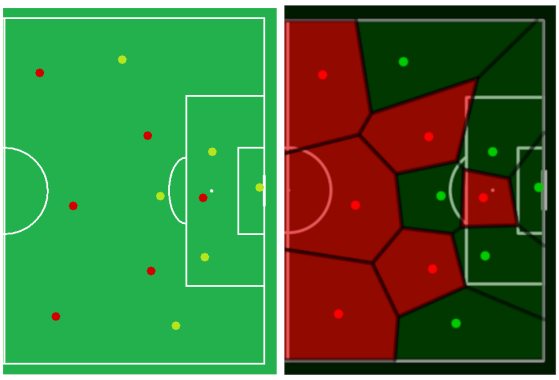
Denote the Euclidean distance between two points p and q by $dist(p, q)$. In the plane we have:

$$dist(p, q) := \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$$

Let $P := \{p1, p2, ..., p_n\}$ be a set of n distinct points in the plane. These points are the sites.

We define the *Voronoi diagram* of P as the subdivision of the plane into n cells, one for each site in P, with the property that a point q lies in the cell corresponding to a site p_i if and only if $dist(q, p_i) < dist(q, p_j)$ for each $p_j \in P$ with $j \neq i$. [11]

In other words, a cell contains all the points in the plane which lie closer to the cell's site than any other site.



(a) **(b)**

Figure 2.3: (a)The positions of defenders with green and attackers with red on the pitch at a specific moment. (b) The generated Voronoi diagram of the image on the left. The plane is the one half of the whole football pitch and the sites of the Voronoi diagram are the positions of the players.

2.2.2 Construction Algorithms

The simplest way to generate the Voronoi diagram of a given set of sites is to compute the cell of each site as the intersection of the half-planes created by the mid-perpendicular lines between the site and each one of the other sites. The total cost of the computations is $\mathcal{O}(n^2 \log n)$.

A more efficient incremental algorithm was suggested in 1976 by Sibson and Green [21]. Their algorithm starts with defining a ”window”: a rectangle that contains all the points in the dataset. (Initially start with 2 points) Then, new points arrive in random order. The first point to arrive acquires the whole window as its cell. From the second point and beyond, each point that is added lies inside an already existing cell. In order to create the cell of the new point three steps are to be made. First, the cell where the point lies is separated by the mid-perpendicular line of the site and the new point. After that, the new point will gain territory from the neighboring cells. This is computed once again by finding the mid-perpendicular line between the new point and the site of each neighboring cell. In the last step, the sectors of the edges which were cut by the border of the new cell and lie in the interior of the cell must be removed from the diagram. Those three actions combine the incremental step that updates the diagram for every new point that is added. The algorithm runs in $\mathcal{O}(n^2)$.

Another approach to address the Voronoi problem is based on the divide and conquer technique. The initial set of points P is vertically divided into two subsets A and B. What follows is the recursive computation of the Voronoi diagram for each of the subsets separately. Then, the two diagrams are combined into one, producing the Voronoi diagram of the whole set of points. It is proven that a monotone chain of edges exists, such that the Voronoi diagram of A coincides with the part of the Voronoi diagram of P that lies on the

one side of the chain and the Voronoi diagram of B coincides with the part of the Voronoi diagram of P that lies on the other side of the chain. So the third step of the algorithm is to compute this specific chain. That can be done in linear time.. Lastly, the edges of the Voronoi diagrams of each subset that cross the chain must be cut, so each Voronoi diagram of the subsets lies entirely on the one side of the chain. This approach achieves the optimal time complexity which is $\mathcal{O}(n \log n)$.

Fortune's algorithm is one of the most well-known methods of the generation of the Voronoi diagram of a given set of points, and it is also optimal. It is a plane sweep algorithm in which a horizontal line scans the plane from top to bottom. Apart from the horizontal line, another curve, which is synthesized by parts of different parabolas, spreads across the plane. When the sweep line comes across a new site, a parabola which contains all the points lying closer to the site than the sweep line is created. Each parabola corresponds to a specific site. The beach line passes, for each x-coordinate, through the lowest point of all the parabolas. As the sweep line continues downwards, the parabolas of the different sites expand, with their intersections on the beach line tracing out the edges of the Voronoi diagram. When a parabolic arc shrinks to a point and disappears, while the sweep lines moves downwards„ this point is a vertex of the Voronoi diagram. In addition to optimal time complexity, Fortune's algorithm also achieves optimal space complexity $\mathcal{O}(n)$[20].

2.3 Applications of Computational Geometry in Football Analysis

Nowadays, the concepts of computational geometry discussed above are widely incorporated into contemporary performance analysis software, used by elite teams worldwide. That sophisticated software supports managers in tactical performance analysis and decision making. In this section, we will focus on the academic research in the field.

Let's begin with the convex hull. This notion is mainly connected to the detection of the team's formation, a major tactical decision, and tactical behaviour while attacking or defending. Shaw and Glickman used the convex hull in their analysis of team strategy based on the relative players' position, in order to recognize the team's formation consistency [46]. They pointed out the similarity of the shapes of the convex hulls of the team captured in different instants of time, concluding that each player's movement is not independent, but highly correlated to his playmates' positions, so that the team's formation is maintained during the game. Another study [5] also confirmed this result based on the teams' convex hull. In the second study, the shapes of the convex hulls observed during a match were categorized in clusters. Although both teams presented a variety of different shapes during the match, those shapes were consistent to a great extent.

Hendriks used the convex hull to study the reaction of the team right after the ball possession is lost and concluded that there is a rapid decrease of the area covered within the first 5 seconds [23]. This result agrees with a significant number of other studies that correlate the wider convex hull area with attacking and the narrower area with defending.

Bauer and Anzer used the convex hull to calculate the effective playing space, one of

the parameters that contribute to the detection of counterpressing in professional football matches [3].

Lastly, in another study [39] researchers used the convex hull of the two opponents to calculate the surface area covered by each team, as well as players spread as functions of time. Then, they studied the median frequencies of the produced time series to describe team tactics and patterns of play. In contrast to Hendrix' research findings, their results indicated that collective movements in football are slow and may last more than a minute. An additional conclusion was that frequency values correlated with ball possession, with higher values indicating rapid ball possession changes and lower values indicating that a team maintained ball possession for longer periods of time.

Conflicting results are not surprising, as each study is based on different data, mined from different leagues. While football rules and principles are the same, the style of play and applied team tactics may vary greatly depending on the league examined or even among teams of the same league.

We move on to the Voronoi diagram. The Voronoi diagram has also been tightly connected to a team's spatial configuration. Rosen used the Voronoi diagram as a tool to visualize and observe the collective motion behaviour of the players and the ball in a football match [45].

However, the most common use of the diagram is the measuring of space domination, a critical factor for the outcome of the game. The space a team controls coincides with the sum of the area of the Voronoi cells, whose sites are the team's players. In the majority of the studies, researchers endeavour to correlate this value with other variables, in order to evaluate the effectiveness of specific actions or applied tactics.

This technique has been widely applied in RoboCup, an international robotics competition, where teams consisting of robotic players compete in football matches. Prokopenko, Wang, and Obst developed a mechanism for their team which contributed to the selection of the players' actions, depending on a dynamic tactical scheme [41]. In this way, they attempted to dominate over the opponent teams more by tactical flexibility rather than by excelling in their robots' technical abilities. In addition, other researchers [28] [10] exploited the Voronoi diagram to determine a strategically advantageous positioning for their robotic agents.

The notion of the Voronoi diagram has also been used in another robot football competition, called Mirosot. More specifically, Law attempted to create a form of spatial perception, so that the robots of his team could ”intuitively” cooperate on the spur of the moment, without playing or moving based on a pre-programmed strategy. However, his efforts were impeded by the high processing times that the image processing techniques demanded. He solved this problem by basing his analysis on Voronoi diagrams instead of pitch images, which require only positional data but still maintain all the useful information about spatial distribution [35]. The conclusions of this research were similar to Kim's results. Kim [31], due to a lack of tracking data from real matches, used the Voronoi diagram with data extracted from a popular electronic football game. His observations focused on teams' domination, excess areas, and defending strategy.

Of course, the Voronoi diagram could not be absent from studies with real world matches data. Ueda, Masaaki, and Hiroyuki examined the relationship between the dominant region calculated with the Voronoi diagram and offense–defense performance [51]. Rein, Raabe, Perl, and Memmert chose the Voronoi diagram as the most appropriate model to measure space dominance, in order to evaluate changes in space control caused by passing [44]. Those measures helped them categorize passes by their effectiveness. Their finding was that passes made from the midfield and attacking field tend to be more effective, as they usually lead to a larger gain in space dominance for the attacking team.

Despite its extensive usage in football analysis, the Voronoi diagram has received a lot of criticism from several researchers [15] [19] [7], who have argued about the efficiency of the diagram regarding the calculation of the dominance area. The main weakness pointed out is that the diagram disregards some important parameters, such as the players' speed, motion, reaction time, and distance from the ball, that may have a decisive role in which player will reach the ball first. For this reason, some studies have adapted variations of the classic Voronoi diagram, such as the weighted Voronoi diagram, while others have developed other more sophisticated models [15].

2.4 Machine Learning

In this day and age, machine learning is one of the most evolving fields of artificial intelligence and computer science in general. Its applications are permeated every aspect of modern life, from spam filtering in e-mail services to cancer detection and autonomous driving. The strength of machine learning lies in its ability to turn information and data, which have rapidly increased in the last decades, into knowledge, and discover hidden patterns in them.

Traditional software engineering is based on rules, that derive from well-established knowledge, to solve problems. For example, in order to decide whether a number is odd or even, we would use the rule of the remainder of the division by two. The rule implies that if the remainder of the division of the given number by two equals zero, then the number is even. Otherwise, the number is odd. The simplicity of this problem renders this rule easy to understand. However, in other cases formulating the rules, that would lead to the solution of more complicated problems, is a much more challenging task. For instance, how easily could someone define the rules to classify an e-mail as spam?

The innovative approach of machine learning is that it combines the provided data with given answers to find out the rules that govern a specific problem. Then, these extracted rules can be applied to any other data concerning the same problem and lead to a solution. This capability makes machine learning techniques highly recommended and appropriate for that kind of problems, whose rules are difficult to be explicitly described.

2.4.1 Probability Theory

In this section, we will present some fundamental parts of the probability theory, on which depend some of the algorithms we will use in this thesis. Firstly, we are going to describe four useful discrete probability distributions.

2.4.1.1 Discrete Probability Distributions

A discrete probability distribution expresses the relationship between a discrete random variable and the probabilities of its possible outcomes. The four most commonly used distributions for classification are Bernoulli, Binomial, Categorical (or Multinulli), and Multinomial. Apart from the above, another famous distribution is the Poisson distribution.

The Poisson distribution describes the probability of the number of times isolated events occur during a period of time. For example, the distribution could describe how many times a lighting will strike during a thunderstorm. It is based on the Euler's number:

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots \simeq 2,7183$$

Raising e to z we get the following series:

$$e^z = \sum_{n=0}^{\infty} \frac{z^n}{n!} = \frac{z^0}{0!} + \frac{z^1}{1!} + \frac{z^2}{2!} + \frac{z^3}{3!} + \cdots = 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \dots$$

In order to turn this series into a probability distribution, the sum of terms should equal one. So we multiply with e^{-z}:

$$e^{-z} \cdot e^z = e^{-z}(1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \dots) = e^{-z} + e^{-z}z + e^{-z}\frac{z^2}{2!} + e^{-z}\frac{z^3}{3!} + \cdots = 1$$

If we set z equal to the expected or average number of event's occurrences, then the terms $e^{-z}, e^{-z}z, e^{-z}\frac{z^2}{2!}, e^{-z}\frac{z^3}{3!}, \dots$ express the probability of $0, 1, 2, 3, \dots$ occurrences of the event. So the only requirement to calculate the probability of any number of occurrences is to know the average number of the occurrences of the event. [38]

In Bernoulli distribution, the discrete random variable is binary, which means that it can have only two possible outcomes. A typical example is the flip of a coin. If the outcomes are expressed as 1 and 0, then the probabilities of each event are the following:

- $P(x = 1) = p, \quad 0 \leq p \leq 1$
- $P(x = 0) = 1 - p$

The binomial distribution is the repetition of consecutive independent Bernoulli trials. A classical case is the repetitive flip of a coin. The distribution estimates the number of successes in a fixed number of trials.

The categorical distribution is being used in cases where an event has at least three possible outcomes. For instance when we roll a die there are six possible outcomes. The distribution counts the probability $p_1, p_2, \ldots, p_k$ of every possible outcome, where $p_1 + p_2 + \ldots + p_k = 1$

The multinomial distribution models multiple independent events with at least three possible outcomes. For example when we roll the same die twice and we want a specific outcome, like four the first time and six the second one, then this probability is calculated using the multinomial distribution.

2.4.2 Supervised Learning - Classification Algorithms

In this section we will outline some of the most common supervised learning algorithms in machine learning. In supervised learning, the goal is to find a mapping from input data to their output. If the output is a continuous number, then we are referring to a regression problem. If the output is a category, then it is a classification problem. In order for the problem to be solved the machine must pass through a learning process with examples. Those examples will train the model to correctly classify the given input data into their classes.

During training, the provided data should be as general as possible. Otherwise, we are running the risk of overfitting, a problematic situation where the model has been excessively adapted to the given examples and can not accurately classify new, previously unknown data. Another reason why our training data should be general is that in supervised learning our data introduce some bias to the model. In other words, our model will imitate what it has been taught. This means that both the reliability and the impartiality of data are critical for the success of the model. Last but not least, the amount of data plays a crucial role, as the more data a model has been trained on, the less likely it is to deal with new input data for which it has not come across similar training data yet.

2.4.2.1 Multinomial Logistic Regression

The multinomial logistic regression is a generalization of the logistic regression for multiclass classification problems. In the binary logistic classification algorithm, a sigmoid function is being applied to the input data and the output determines the class of every point depending on whether it lies above or below a certain threshold. One way to implement a multinomial logistic regression model is as an independent set of binary logistic regression models. In this case, models and classes have an one to one correspondence, as every model separates the members of a specific class from members of all the other classes. The procedure to classify an unknown point to a category is the following: the

point is being given as input to every model. Then, each model calculates the probability of the point to belong to the corresponding class. The class with the larger probability is the final prediction of the multinomial logistic regression model. Another approach that results in smaller standard errors is to use one class as a pivot and regress all the other classes against the pivot one. Of course, there are other methods which also formulate the multinomial logistic regression based on binary logistic regression models.

2.4.2.2 Support vector machines (SVM)

Support vector machine (SVM) is another widely used algorithm suitable for classification. The algorithm's purpose is to map the training set's examples on a multidimensional space and then find a boundary that separates the different classes. Having found this boundary, the algorithm is able to categorize new data depending on the side of the boundary they fall into. The real challenge, though, is the determination of the optimal boundary, among the plethora of possible boundaries that can be chosen. The optimal boundary is the one that is the furthest away from the closest point of the training set. If the training data can be distinctly separated by a straight line, then the optimal boundary, also called decision boundary, has equal distance from the closest points of each class. These points are called support vectors. Between the lines that pass through the support vectors and are parallel to the decision boundary there is a margin in which there are no points from either class. When this margin is wider the algorithm can more easily identify the class in which an unknown point belongs. Thus, the algorithm selects the decision boundary that maximızes this margin.

While the classical SVM algorithm supports binary classification, the same principals can be also applied to multiclass classification problems. The basic idea is to break the initial problem of multiclass classification into smaller binary classification problems and combine their solutions. There are two methods one can employ. The first one is to consider only the points of two classes at a time. Thus, a decision boundary will be created for every possible pair of different classes. For example, if the training data belong to three classes, A, B or C, then three decision boundaries would be formed. One to separate classes A and B, another one for the separation of classes A and C and a third one to separate classes B and C. The second method is to have a classifier for every class. The classifier of each class separates the points of the class from all the other points that be long to other classes. In that way, in the previous example with the three classes A, B and C three classifiers would be formed. Generally, the first approach generates m(m-1)/2 decision boundaries while the second generates m decision boundaries, where m is the number of different classes.

2.4.2.3 Naive Bayes

Naive Bayes algorithm is a probabilistic approach to deal with the classification of categorical data, based on Bayesian probability theory. Bayesian theorem allows us to calculate

the probability of an event to occur, given that another event has occurred. For instance, we could determine the probability of a team to win a match, given that the opponent team has less players on the pitch, due to a sent off. Bayes theorem's strong assumption is that all the features are independent and all contribute equally to the occurence of the event. This is quite a naive assumption to make, which is rarely met in real world case scenarios. For example, as discussed above, the researcher in football analysis has proved a strong relationship between successful passing and ball possession (or goal scoring). Despite this obvious drawback Naive Bayes tends to achieve good accuracy in many classification problems, like document classification[50], spam message classification[40] and traffic risk management[9]. It remains to be seen if it will have a good prediction rate in our problem as well.

2.4.2.4 Decision trees

The decision tree is a very popular and powerful tree-based algorithm used for supervised machine learning classification problems. It utilizes a tree-like structure with each branch representing an if/else rule. Inner nodes contain features of the data, the edges possible values of those features and the leaves contain the categories of the data. The procedure to classify a new unknown element is the following: the tree is being traversed beginning from the root until the leaves, following the a path. The path is being determined by applying the conditions of each node on the element's feature values. This path ends to a leaf, which represents the prediction of the category of the element. Decision tree algorithm is similar to ”divide and conquer” methods, as in every step the initial set is split in two subsets.

It is obvious by the number of different combinations of the features and the nodes, that there are many ways to construct a decision tree. The way the features are assigned to nodes is based on several criteria, such as maximizing the information gain and minimizing the entropy. Entropy is a metric to measure the uncertainty in data. The more mixed a dataset is the higher the entropy. In other words, in a set that consist of elements from many different classes, with each class having the same number of representatives, predicting the class of a random element would be a tough task. On the other hand, in a dataset in which the majority of items derive from only one class, then a random element will belong to that class most of the times. The splitting of the data stops either when there are no more features left to be assigned to nodes or all the subsets that have been created contain data of the same class.

Decision trees have several advantages. Among them is the fact that they can be utilized both for numeric and categorical data. In addition, they require less or no data preparation, like outliers removal, normalization and missing values handling. Moreover a major advantage is their effectiveness with non-linear data. Last but not least, the greatest advantage of decision trees, to which they owe their popularity, is their simplicity. Decision trees are so simple that they can be easily described using natural language and therefore, they can be understood by humans.

2.4.2.5 Random forest

Random forest algorithm extends the idea of the decision trees, combining many of them for a single prediction. It is a special case of the ensemble classification methods, which utilizes decision trees as classifiers. Each tree makes a prediction and the most popular prediction is being returned as the prediction of the random forest classifier. In case of numerical data, the returned prediction is the mean value of the individual predictions. Random forest cures the overfitting problem that might appear in a single decision tree, as it consults many decision trees, which are , relatively, uncorrelated one another. The uncorrelation is being ensured by the way each decision tree is constructed. In opposition to the construction of the trees in the traditional decision tree algorithm, in the random forest case, the features that composite each node are chosen randomly. Apart from that, the trees use different subsets of the initial dataset as training examples. The more uncorrelated each tree is, the better the random forest prediction model performs. Also, it has been theoritically proven that the generalization error in random forest converge to a limit when the number of trees is large enough. This is the reason why random forests don't suffer from overfitting when more decision trees are added to the ensemble model. On the contrary, this addition tends to limit the generalization error.

2.4.2.6 k-Nearest Neighbors

Arguably the k-nearest-neighbors algorithm is one of the simplest classification methods in machine learning. As a nonparametric method, it does not require any assumption for the training data. This is a highly convenient characteristic that makes the algorithm suitable for any dataset, without any prerequisite knowledge about the data's underlying distribution or specific properties. The basic idea behind knn algorithm is so intuitive that it could be summarized in just an old saying, "tell me who your friends are, and I will tell you who you are". Every query point is assigned to the class to which its nearest neighbor (if k=1) or neighbors (if k>1) belong. The nearest neighbor is the point of the dataset that lies closer to the query point. In order to define the nearest neighbor a distance metric should be applied. Depending on the type of variables, some commonly used metrics are Euclidean or Manhattan distance for numeric data and Hamming distance for text data. In cases when the proximity and the density of the training data are not the only parameters, but there are other factors which also play a role in the classification, such as size, velocity, or range, then weight assignment to each point of the training set is considered a useful method to address this kind of problems. Lastly, the choice of hyperparameter k, which represents the number of nearest neighbors that will decide the class of a query point, can have an effect on the quality of the classification. On the one hand, even though larger values of k have been observed to address more efficiently the issue of outliers, it may lead to underfitting. On the other hand, smaller values of k tend to lead to overfitting. While there is no golden rule for the optimal selection of the k value, several heuristics have been proposed. The only non-negotiable property of k is that it must be an odd number, due to the "majority voting" nature of the algorithm. Although knn have several strengths,

it also has some weaknesses. The major drawback is the high computational cost of the classification, which may be a prohibitive factor for large datasets with multidimensional points. In those cases, it is critical that data reduction techniques are performed and a trade off between speed and is made.

2.4.2.7 Neural networks

Neural networks or artificial neural networks (usually also referred to as deep learning) are an old concept in artificial intelligence. They were first introduced by McCulloch and Pitts in 1943. However, their capabilities were really highlighted during the last decade, significantly boosted by the increased processing power of modern GPUs. These networks have an amazing ability to recognise patterns in data of large datasets that a human could not perceive.

The fundamental building blocks of neural networks are the neurons. The neurons are computational units that receive a vector of input values, perform some computations on them and produce a single output value. In more detail the neuron computes the dot product of the input values x_1, x_2, x_n with a set of weights w_1, w_2, w_n and then adds a constant b to the sum, called bias.

$$z = b + \sum_{i=1}^{n} w_i \cdot x_i$$

Then a function f, called activation function, is applied to z, producing the final output of the neuron: $f(z) = a$

Activation functions are non-linear, differentiable functions that map the weighted sum z to another range that usually is more suitable for the application. Some popular activation functions are: - Sigmoid - tanh - ReLU - Softmax - ELU - Swish

A neural network consists of layers, at least two, with each layer containing at least one neuron. The two necessary layers are the input layer and the output layer. The input layer's neurons receive the initial input of the network and the output layer's neurons produce the final output values. The layers in between are called hidden layers. Although hidden layers are optimal, they greatly contribute to the building of more complex networks which can better deal with non-linear datasets with many features. The number of hidden layers must be carefully chosen, as too many layers can lead to overfitting. Since there is no concrete rule to determine the optimal number of hidden layers, a trial and error approach is required so that a balance can be achieved.

2.5 PCA

Despite the vast processing capabilities that we nowadays enjoy the curse of dimensionality still occurs in the field of machine learning. The curse of dimensionality springs from

the high number of dimensions of data. Sometimes the number of features can be comparatively large, even surpassing the number of samples of the dataset. This is a deeply problematic situation, considering that for each added feature, an exponentially larger number of training data are required in order the model to be sufficiently trained. Even worse, there are particular methods, like k-nn, that cannot work at all in high dimensions, because samples are so scattered that the concept of proximity stops making sense. For all the aforementioned reasons, the dimensions of data should be limited to an extent that a good balance between accuracy and speed is achieved.

The research in this field has been encouraging, as it has been observed that in many real world problems, only a small subset of features is enough to make accurate predic tions. Now the question that arises is: how will this subset be determined? The features that should be chosen are those that maintain the information which best describes or separates the classes of the samples, in the case of classification. Various techniques have been developed for this task. Principal component analysis (PCA) is one of the most widely used.

PCA can be described as a sequence of five steps. Firstly, the values of the features should be standardized so that they equally contribute to the analysis. The importance of this step can be easily understood with an example. Imagine that in our attempt to predict the outcome of a football match, we consider two variables, the number of red cards and the number of successful passes. The maximum difference in red cards can not exceed a specific number, as, according to the current regulation, a game ends if a team fields less than seven players. Thus, the maximum difference in red cards is four. For every red card one team receives, the opponent team gains great advantage. On the other hand, a difference of four successful passes between the two teams is negligible, because each team attempts hundreds of passes during the game. If the standardization did not take place then those two facts would have the same impact on the prediction of the outcome.

The second step of PCA is to compute the covariance matrix. This is a square symmetric matrix that contains all the possible pairs of features and their covariance. Simply stated, the covariance of two variables expresses the relationship between them. When the covariance is positive, then if the one variable increases, the other variable increases as well. On the contrary, when the covariance is negative, then if the one variable increases, the other decreases.

The computation of the covariance matrix is a preparation step for the next task, which is the determination of the principal components. This is done by calculating the eigenvectors and the eigenvalues of the covariance matrix. Each eigenvector corresponds to one principle component and the eigenvalue expresses the amount of variance the corresponding component carries. The eigenvector with the highest eigenvalue is the first principal component, that carries the most information about the dataset. The eigenvector with the second highest eigenvalue is the second principal component and so on.

Since the principal components have been determined what follows is the selection of the number of principal components to be maintained. The more principal components are discarded, the more dimension reduction is conducted. The number of principal compo-

nents that are kept is the new dimensions of the data. The decision on the number of components depends on both the wanted extent of the reduction of the dimensions and the amount of information that is acceptable to be lost. In the majority of the problems two or three components are enough in order for more than 90% of the variance to be captured. In the last step. the data must be reoriented from their initial axes to those that represent the chosen principal components.

2.6 Model evaluation

In our thesis we will test several different classifiers in order to estimate the significance of every attempt, with an ulterior purpose to predict the outcome of a match and even further of the whole championship. Apparently, the classification model with the most successful prediction rate on the training phase will be the one used to classify the unknown dataset with the unlabeled data. In order to distinct the most efficient model, we must be able to compare the efficiency of one model with another one. Moreover, it would be beneficial to have a tool which would help us understand if a modification we made on a specific model's parameters had a positive or negative impact on its performance.Finally, a comparison between traditional benchmarks and new approaches is necessary for the further advancement of the field of football prediction and machine learning in general. For these reasons, it is of critical importance to define and use some evaluation metrics. The metrics we will be relying on for the evaluation of our models are the following: accuracy, precision, recall, F1 score and Mathews Correlation Coefficient. Before we analyze the above metrics, we will describe a useful structure that is associated with most of them, called confusion matrix.

2.6.1 Confusion matrix

A confusion matrix is a two dimensional $n \times n$ square matrix, where n is the number of classes that exist in the dataset. The rows of the matrix represent the actual classes, while the columns represent the predicted classes. In the main diagonal lie the number of data that were correctly predicted by the model. The higher the diagonal elements, the more accurate the model. On the other hand, elements that do not lie on the diagonal correspond to wrong predictions. As an illustration, let's inspect the following confusion matrix of three classes.

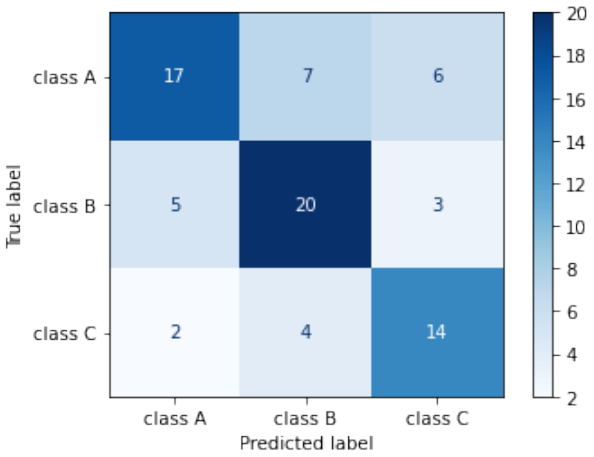


Figure 2.4: The confusion matrix of a classification problem with three classes

The element in first row and first column (17) is the number of instances that were correctly predicted to belong to class A. Next to it, the second element of the first row (7) is the number of instances that were predicted to belong to class B, but the class they really belong to is A. The first element of the second row tells us that there were 5 elements of class B that were misclassified to class A.

2.6.2 Accuracy

Accuracy is a simple and easily comprehensible metric that expresses the percentage of the correct predictions. In binary classification where there are two classes, one class could be identified as positive and the other one as negative. In this case, the confusion matrix would be the following:

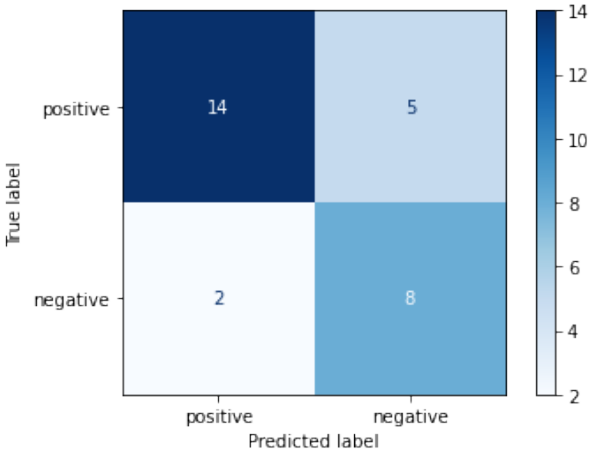


Figure 2.5: The confusion matrix of a binary classification problem

and the accuracy would be calculated with the following type

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

In multiclass classification the accuracy is calculated by summing all the diagonal elements and dividing by the total number of predictions made.

Accuracy is inappropriate for imbalanced dataset. Imagine the spam filtering problem. Suppose that a training dataset consist of one hundred message samples, of which five are spam messages. An inefficient model that would detect none of the messages as spam would have 95% accuracy rate.

In those cases, it is preferable to use other kind of metrics like the following.

2.6.3 Precision

Precision expresses what percentage of instances that were classified as positives (or negative) and were truly positive (or negative), in the case of binary classification.

In multiclass classification the precision expresses the percentage of the instances that were classified as items of a certain class, and indeed belong to that class. In the example given in the section of the confusion matrix with the three classes A, Band C, the precision of each class would be the following:

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

The precision is increased if the false positives are decreased.

2.6.4 Recall

Recall expresses what percentage of all the instances belonging to a certain class has been correctly classified.

$$Recall = \frac{TP}{TP + FN}$$

The recall is maximized when the false negatives are minimized.

2.6.5 F1 score

F1 score is a combination of precision and recall. In more detail F1 score is the harmonic mean of those two metrics and is calculated by the following type

$$F_1\ score = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

F1 score usually ranges somewhere in between precision and recall. Thus this metric is very useful when a balance must be achieved between false positives and false negatives reduction.

2.6.6 Mathew's Correlation Coefficient (MCC)

A slightly different metric that involves all the elements of the confusion matrix is the so called Mathew's Correlation Coefficient. The type of this metric is the following one

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}}$$

Mathew's Correlation Coefficient ranges between -1 and 1. Negative values are an indication of extended misclassification while values closer to one is a sign of an accurate prediction model. Lastly, values around zero are also considered inadequate as they show that the model has similar accuracy with a classifier that would randomly categorize the input data.

2.7 Related research on football outcomes prediction

Football is not a new sport. The attempts to make match predictions based on scientific methods rather than pure intuition began a long time ago. At first, researchers approached football prediction with a more statistical perspective. In 1956 Moroney modeled the goals that a team scores in a football game as a Poisson distribution [38]. Although the predicted values were not far from the actual ones, this piece of information disregards the dynamics of the two teams. In more detail, this model would suggest that a team has the same probability of scoring zero, one, two, three, etc. goals in any match, independently of who the opponent might be. In 1971 Reep, Pollard and Benjamin made their own analysis suggesting that the Negative Binomial distribution fits better than Poisson to the number of goals scored at a match by individual teams [43]. About a decade later, Maher questioned the rejection of the Poisson in favour of the Negative Binomial distribution [36]. He claimed that if the different attacking and defending qualities of each team are taken into account, then a customized Poisson distribution to every match can fit the data of the scores reasonably well.

Although traditional statistical methods, such as Poisson models[2] [26], are still utilized , in the modern era, machine learning techniques and neural networks constitute the dominant trends in the field of football prediction. Some research of the last ten years that is worth mentioning is the following.

In 2013 Sujatha, Godhavari, and Bhavani developed an artificial neural network to predict the outcome of four matches between the Bundesliga teams Borrusia Dortmund and Bayern Munich in the season 2011-2012 [48]. They used the results from the previous meetings of the two teams during the period 2005 to 2011 and some other factors, including but not limited to transfer money spent, UEFA's point system, league rank., as input. The artificial neural network achieved high accuracy in the number of goals, but it performed less adequately in outcome prediction in comparison with other methods.

In 2014 a group of researchers [25] tried to predict 20 English Premier League matches played in season 2014-2015 using logistic regression and an artificial neural network. The input of the models was both direct statistics like home and away goals, corners, shots etc., and some performance indexes for the manager and the players. While logistic regression yielded higher accuracy than artificial neural networks, it could only predict win or loss results. This could result in lower accuracy in larger data sets in which more draws were observed.

Another attempt to predict the outcome of football matches with the use of neural networks was made in 2020 [42], but in this case, it focused on matches between national teams. More specifically, the predictions were made for the group stage matches of FIFA world cup 2018. The model's input consisted of ten features. The accuracy of the model in the group stage was 63,3%. According to the author, a drop of the accuracy was observed in the knock out matches.

An interesting experiment was conducted by Kampakis and Adamides in 2014 [30]. The two researchers combined historical features with almost two million tweets in an attempt to predict English premier league matches. Twitter proved to be a valuable source of information as it can provide data which is not available in historical statistics, such as the fans' emotions for the game, discussions about the outcome, and players' injuries. Three different datasets were used in this study: one with historical data, another one with data retrieved from Twitter, and a third one with combined data. It is worth noting that the model tested on the Twitter dataset outperformed the model that used historical data as features. Not surprisingly, the mean accuracy was raised even higher with the combined dataset, going up to 69,6%.

In 2017 Hijmans and Bhulai made predictions for the national team of the Netherlands using generalized boost models, naive bayes and k-nn classification [24]. They used a dataset that consisted of various variables that belong to the following categories: type of game (e.g. friendly, qualification etc.), squad attributes, individual player attributes, and the team's form. The best performance was observed with the generalized boost models method, which predicted 60,22% of the matches correctly.

In 2018 Esme and Kiran used the k-nn algorithm for the prediction of 153 games of the second round of the Turkish super league [17]. Their predictions were highly based on historical statistical data as well as betting odds. The model they created performed quite similarly to the bookmakers' predictions in the full time result case. However, in the double chance betting, in which the bettor can make two predictions simultaneously, the model outperformed the bookmakers' accuracy by 7,84%. In total, the full time result accuracy

of their model was 57,52%. Their result is consistent with previous studies [49], which suggest that with open data engineering, it might be possible to beat the bookmakers.

Admittedly, betting and profit-making are two of the basic reasons why research on football match predictions has been greatly boosted in our time. Another study that proves that is the one carried out by Stübinger, Mangold, and Knoll in 2019 [47]. The researchers utilized machine learning algorithms to predict football outcomes, in order to generate positive returns based on a successful betting strategy. Their input data was derived from the top five European leagues during a period of eleven seasons and regarded players' characteristics and skills. Their predictions and betting strategy were focused on goals difference. When the predicted difference was over two goals, then the match was considered safe for betting. Else no trading was conducted. Among all the applied methods, the combination of all of them was proven the most beneficial, yielding 81.77% total accuracy.

As it was expected, the spread of COVID-19 greatly affected all kinds of football leagues and associations around the world competitively and economically, causing many match postponements and cancellations. The impact of the virus on sports events is also being reflected in the field of the related research, which was also impeded, due to reasons related both to public health and lack of data. However, in the following years, football leagues and the research around football came back decisively.

In 2021 Kozak and Głowania applied heterogeneous ensembles of classifiers to predict the results of Bundesliga matches based on teams' positions and points in the league table [33]. They compared their method to single classifiers and other ensemble methods and concluded that their approach gives the best results. The yielded accuracy was 56%. The main drawback of the suggested model seemed to be its poor performance in draw predictions.

The same year Beal, Middleton, Norman, and Ramchurn combined statistical data and sports journalists' articles to predict match outcomes of the English premier league [4]. Their method turned out to produce more accurate predictions than other benchmarks based on text-only data, statistical-only data, and bookmakers' odds. The achieved accuracy was 63,19%, which was 6,9% higher than the traditional methods' accuracy.

In a study of 2022 [22], researchers evaluated several combinations of features with different classifiers to determine the one that gives the highest predictive score. The set of classifiers used included Logistic Regression, SVM, Random Forest, K-NN, and Naive Bayes. The features derived from two seasons of the English Premier League, specifically seasons 2011-2012 and 2012-2013. The researchers initially conducted two tests, with different features selected in each test, considering only Manchester United club's matches, the team which admittedly had the best performance among all during those two seasons (2nd place in 2011-2012 and champion in 2012-2013). In both tests, the majority of classifiers had a fair prediction rate. Random Forest and Naive Bayes in particular achieved 79,17% and 80% accuracy in the first and the second test respectively. However, in the experiments that involved the whole set of teams that participated in the league, a significant drop in accuracy was observed. An exception to this fact was the k-nn method, whose accuracy rate scaled up to 83,95% with a specific set of features.

Duarte also conducted a study in 2022 on predicting the English Premier League Champion of the season 2021/22 [12]. He incorporated the last three seasons of the league, using the first two for training in order to predict the third season's final ranking table. Among several classifiers whose accuracy was tested, the k-nn was evaluated as the most powerful model for the purpose of the study, with 75% to 85% accuracy, depending on the data set it was tested on. The model's accuracy in not previously known data (72,37%) was good enough for the researcher to correctly predict the top six clubs, not in the correct order though, as well as the teams that were relegated, with their exact positions.

Last but not least, the most relevant research to our thesis is the work of Malamatinos, Vrochidou, and Papakostas, also conducted in 2022 [37]. In their study, they tested five different machine learning models with data derived from six seasons (2014-2020) of the Greek Super League, in order to predict the outcome of the matches of the season 2021/22. They also conducted a comparison with the results from two other data sets with English Premier League and Dutch Eredivisie matches. The Greek Super League proved to be the most predictable league with a maximum accuracy of 67,73%. We will use this result as a benchmark for our own research since we will attempt to predict the outcomes of the same matches.

3. OUTCOME OF COMPLETED MATCHES RECKONING PROCESS

In this chapter we will present both the process and the results of our attempt to reckon the outcome of completed matches of the greek Super League championship of the season 2021-2022 based only on the Voronoi diagrams of their final attempts (highlights). In section 3.1 we will describe the dataset we worked upon, providing information such as the way the data were extracted, how they were divided into classes and some statistics of the league. Then we will thoroughly analyze the procedure of estimating matches outcomes. In more detail, in section 3.2 we will cover the preprocessing of the input data and the training of the model. Following this, in section 3.3 we will discuss the evaluation metrics we based upon to determine the most suitable algorithm for our problem. In section 3.4 we will show the results of the classification model. In the last section of this chapter we will compare three different strategies on how to use the output of the classification model in order to determine the final outcome of a match.

3.1 Dataset

The dataset used in this thesis consist of 1862 png colored images with dimensions 600×400 pixels. Each image depicts the Voronoi diagram of an attempt made on a match of the greek Super League championship of the season 2021-2022. We were able to collect data for 178 out of 182 matches of the main season. Due to lack of official tracking data of the players, the whole dataset was created from scratch with a procedure that involved the following steps:

1. We wrote a python program using the Tkinter and MplSoccer libraries which creates a football pitch diagram where the user is able to import players' positions by clicking on the corresponding point of the diagram. The user also determines for every player involved if he belongs to the attacking or the defending team and if he is the holder of the ball. The program receives the aforementioned data as input and produces the Voronoi diagram of the pitch, with sites the players' positions.

2. We found online, publicly available on Youtube, videos with the highlights of almost every match of the season. For every attempt included in them, we paused the video the moment when the attacker makes the last touch with the ball.

3. Then we enter the positions of the players shown in the paused video to the diagram we created with the python program and receive the Voronoi diagram of this attempt as the output of the program

4. Last but not least, we rename the output png file with a name of the following format:

   $$class - homeTeamCode - awayTeamCode - attackingTeamCode - minute.png$$

   where

- *class*: the category to which we classify the attempt based on its level of success for the attacking team. We chose to divide the attempts into five classes:
  (a) 1: poor attempt
  (b) 2: mediocre attempt
  (c) 3: good attempt
  (d) 4: save or off the post
  (e) 5: goal
- *homeTeamCode*: a three letter code name for the home team
- *awayTeamCode*: a three letter code name for the away team
- *attackingTeamCode*: a three letter code for the team which attempted the attack
- *minute*: the minute that the attempt took place

It can be easily noticed that the methods used to obtain the input data and classify them into classes have two major disadvantages. Firstly it is obvious that the players' positions in the pitch and consequently in the Voronoi diagrams are lacking preciseness. The fault in the accuracy occurs not only due to the manual introduction of the data in the diagram which involves human error, but also because not all the pitches of greek Super League comply with the international standardization of the pitch dimensions. As far as the classification of the attempts is concerned, no one can deny that, apart from the attempts which belong to the goal category, the characterization of an attempt for example as mediocre and not good and vice versa involves a lot of subjectivity.

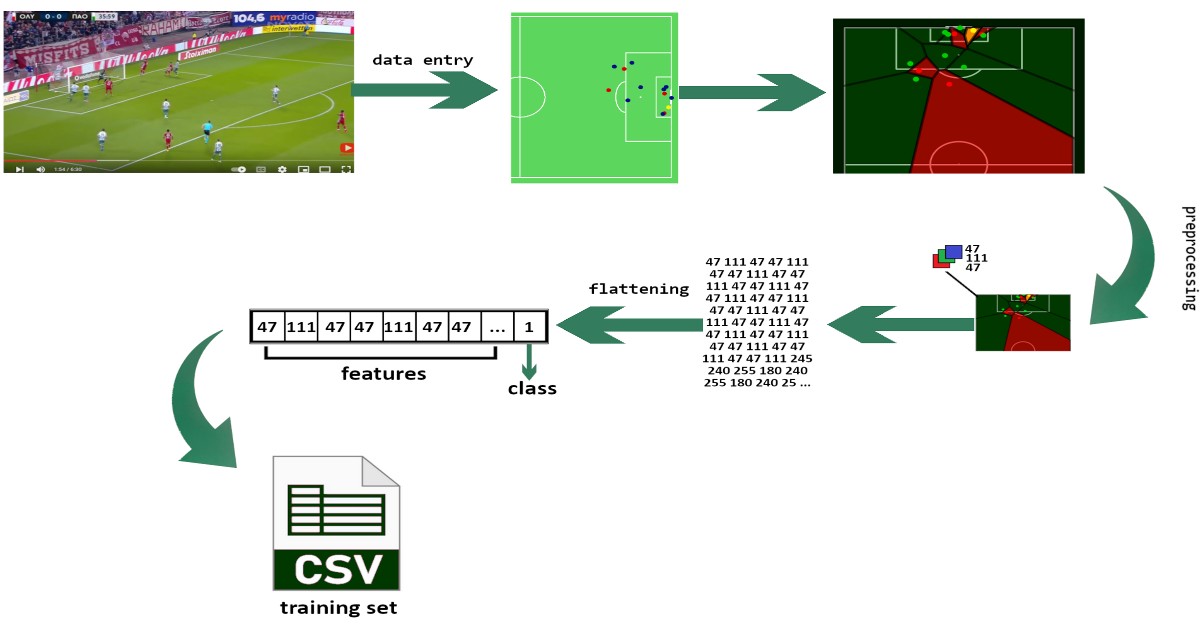


Figure 3.1: The process of the creation of the dataset

3.1.1 Statistics

The statistics provided below concern the first half of the 2021-2022 season of the greek Super League. Based on them we will make an effort later in this thesis to interpret the predictions of machine learning model that we will build in order to reckon the outcomes of the matches played during the second half of the season. Apart from that we also aim to predict the final ranking of the championship utilizing these statistics.

The first table presents the number of attempts per class and the total sum of the labels of the attempts that each one of the fourteen teams of the championship has made in the first half of the season (13 matchdays).

Attempts Per Category																	
		Home								Away							
#	Teams	num. of match.	Total Atte.	1	2	3	4	5	Labels Sum	num. of match.	Total Atte.	1	2	3	4	5	Labels Sum
1	aek	6	35	5	6	11	1	12	114	7	41	4	7	12	5	13	139
2	apo	5	16	2	5	2	4	3	49	6	19	3	5	5	3	3	55
3	ari	6	49	15	12	8	7	7	126	7	30	3	8	11	3	8	98
4	ast	6	36	9	5	10	4	8	105	6	22	2	6	6	5	3	67
5	atr	6	45	9	6	10	12	2	149	7	22	6	6	3	0	7	62
6	ion	6	29	2	3	9	6	8	99	7	24	2	7	6	3	6	96
7	gia	6	33	4	10	6	3	10	104	5	17	1	3	2	3	8	65
8	lam	6	24	2	4	6	6	6	82	6	22	5	4	7	2	4	62
9	ofi	7	34	4	4	12	5	9	113	6	27	3	6	5	7	6	51
10	oly	7	63	14	9	19	11	10	183	6	36	4	3	11	4	14	129
11	pan	7	30	3	7	5	9	6	98	6	23	5	3	6	3	6	71
12	pao	7	43	1	6	12	6	18	163	6	34	11	10	6	3	4	81
13	the	6	49	6	13	8	10	12	156	7	40	5	4	14	5	12	135
14	vol	7	33	3	7	7	9	7	109	6	25	2	1	7	3	12	87

Table 3.1: The attempts of the first 13 matchdays per class

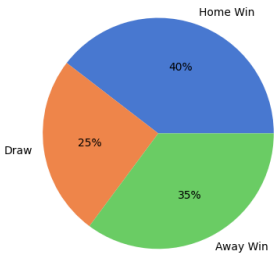


Figure 3.2: The outcomes of the matches of the first half of the season

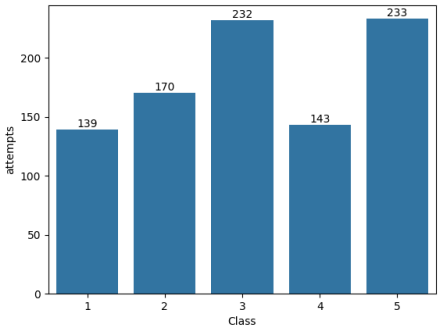


Figure 3.3: The distribution of attempts per class for the first half of the season

3.2 Preprocessing

The proprecessing of our input data aims mostly to the reduction of their dimensions and to a lesser extent the increase of their purity. At first we resize the original images from 600×400 pixels to 240×160 pixels. This reduction in size does not imply reduction of the precision of the model, as our trials indicated. As a second step, we cropped the images removing the white frame which surrounds the Voronoi diagram as it carries no piece of information. Thus the dimensions of the final input images are 148×120 pixels. Then we flatten the image. The last preprocessing action before emboding the image to the training set is to apply scaling. In our case scaling the data does not play a decisive role to the improvement of our model, since all the features are of the same type, pixel values in a range from 0 to 255, and measured at same scales. However we implemented this technique because it has become a standard in machine learning and does not drastically increase the computational cost.

3.3 Evaluation metrics to determine best classification algorithm

All the classification algorithms described in section 2.4.2 were tested in order to find the most suitable for the classification of the attempts into the five classes. To achieve this goal it is essential to determine a common subset of data, upon which the algorithms will be tested, as well as objective evaluation metrics. For the first part of this task we randomly selected 50 matches and applied the 5-fold cross validation technique. While the majority of the classification algorithms examined are deterministic, which means that given the same training data they will produce the same model, some of them, such as random forest or decision tree classifiers have some randomness in the building phase of the model. This characteristic makes 5-fold cross validation a top-quality choice, as its

repeatability ensures an overall view of the performance of those randomized algorithms and does not rely on only one model, which may show a misleading precision.

For the problem of the preferred evaluation metric, plenty of options were examined. Except for the widely used metrics presented in section 2.6 we also considered to introduce a new metric which is the difference of the predicted label from the actual one. The logic behind this metric is that while the traditional metrics receive a prediction as either correct or false, this one provides a meaningful range that indicates how far the prediction has fallen. To explain the idea better, if a successful attempt in which a goal was scored (class no 5) was predicted as a goalkeeper's save (class no 4) then the distance would be 1. If the same attempt had been predicted as a poor one (class no 1), the distance would have been 4. The sum of the distances of all predictions is the final sum. The lower the final sum the higher the model's performance. On the first view the distance metric seems to provide a better understanding of the model performance. However, the major disadvantage of this metric is that in a balanced dataset it tends to favor models which mostly predict the class that lies in the middle. Indeed, from a mathematical point of view, classifying an attempt as a good one (class no 3) is the safest answer, because in worst case its distance with the real label will be 2. Especially in datasets in which the middle class is prominent it is highly likely that the preferred models based on the distance metric will perform poorly with the minority classes. Taking these facts into consideration we turned down the idea of the distance metric.

Reconsidering the traditional metrics, a possible way to determine the most efficient metric for us is to analyze how the results of the attempts classification process will be handled. This analysis, presented in detail in the next section, leads to the conclusion that although it would be favorable to build a model which would be able to classify correctly the attempts of all the classes, the most valuable class for us is class no 5 (attempts in which a goal has been scored). There are two requirements that the desired model should fulfill. Firstly the attempts of the class no 5 to be correctly classified as such. Secondly the attempts which belong to other classes to be misclassified as little as possible as goal attempts. Therefore the most suitable metric is the precision of the class no 5, which is not only increased when the attempts of the fifth class are correctly classified but is also decreased when the attempts of other classes are misclassified as goals.

The following tables present the performance of each classification algorithm. Apart from different algorithms we also tested different scaling techniques as well as different image sizes. Smaller size image were tested five times using 5-fold cross validation and larger images were tested three times using 5-fold cross validation. All tests used the same training and testing sets and the same equipment. The results presented are the average values of all the trials:

240×160 images with MinMaxScaler								
	Metric	Random Forest	Decision Tree	SVC	Logistic Regression	Multinomial Naive Bayes	kNN	Neural Network
Average of 5 trials	Average f1-score	0.237	0.202	0.215	0.240	0.233	0.200	0.228
	Average MCC	0.084	0.026	0.071	0.069	0.068	0.017	0.068
	Average Precision	0.284	0.214	0.266	0.252	0.278	0.220	0.240
	Average Recall	0.256	0.220	0.244	0.252	0.242	0.214	0.246
	Goal Precision	0.328	0.272	0.298	0.364	0.280	0.280	0.336
	Goal Recall	0.388	0.276	0.352	0.368	0.510	0.136	0.342

Table 3.2: The average score of the seven classifiers on different metrics with images of 240×160 pixels size scaled with MinMaxScaler

240×160 images with MaxAbsScaler								
	Metric	Random Forest	Decision Tree	SVC	Logistic Regression	Multinomial Naive Bayes	kNN	Neural Network
Average of 5 trials	Average f1-score	0.216	0.208	0.193	0.263	0.229	0.169	0.224
	Average MCC	0.080	0.021	0.070	0.095	0.066	-0.023	0.070
	Average Precision	0.262	0.220	0.304	0.272	0.272	0.18	0.242
	Average Recall	0.246	0.224	0.242	0.272	0.240	0.180	0.250
	Goal Precision	0.320	0.262	0.362	0.388	0.270	0.202	0.354
	Goal Recall	0.344	0.258	0.300	0.396	0.500	0.102	0.340

Table 3.3: The average score of the seven classifiers on different metrics with images of 240×160 pixels size scaled with MaxAbsScaler

600×400 images with MaxAbsScaler								
	Metric	Random Forest	Decision Tree	SVC	Logistic Regression	Multinomial Naive Bayes	kNN	Neural Network
Average of 3 trials	Average f1-score	0.231	0.198	0.215	0.262	0.226	0.193	0.218
	Average MCC	0.082	0.010	0.076	0.093	0.061	0.005	0.063
	Average Precision	0.267	0.207	0.266	0.272	0.272	0.202	0.233
	Average Recall	0.250	0.220	0.248	0.272	0.236	0.210	0.240
	Goal Precision	0.350	0.257	0.318	0.388	0.266	0.254	0.287
	Goal Recall	0.367	0.267	0.354	0.404	0.490	0.136	0.273

Table 3.4: The average score of the seven classifiers on different metrics with images of 600×400 pixels size scaled with MaxAbsScaler

There is no doubt that the Logistic Regression algorithm outbalances the other six algorithms. Especially combined with the max abs scaler it dominates in almost all the used metrics, with only exception the recall of class no 5 where Multinomial Naive Bayes can predict one out of two goals correctly. As we observe the size of the image does not play a decisive role in the performance of the model, so we will use the 240×160 pixel images to train our final model for time efficiency and scalability to larger datasets. As far as the scaler is concerned, the max abs scaler seems to enhance in some extent the scores of the Logistic Regression algorithm, so we will choose this one over the min max scaler. Some other alternatives, apart from the Logistic Regression, would be the Support Vector Classifier, a Neural Network and the Random Forest. On the opposite Decision Tree algorithm and Multinomial Naive Bayes fall behind. Lastly, the k-Nearest Neighbors algorithm should be definitely avoided for this problem as it even presents a negative Matthews correlation coefficient in some cases.

3.4 Model tuning using class weights

Tuning a model in machine learning is about experimenting with the values of the hyperparmeters in order to achieve an even better prediction rate. As one can notice the dataset we will use to train and test our model is slightly imbalanced with classes no 3 and 5 being dominant. Therefore our attempts to boost the performance of our model will focus on the modification of the weight of each class. We also observed during the determination

process of the most suitable classifier that the default lbfgs solver was unable to converge. However the amount of time that the other three solver demanded was prohibitive. Thus experimenting with the solver hyperparameter is not an option.

There is no denying that the combinations of the weights are countless and the attempt to examine as many of them as possible seems tempting. However due to limited time and computational power it is essential to make a selection of only a few weight values for every class. We have performed an exhaustive search of the 1024 models made of combinations with the following weight values:

- $\{0.9, 1.0, 1.2, 1.4\}$ for w_1
- $\{0.9, 1.0, 1.2, 1.4\}$ for w_2
- $\{0.9, 0.8, 0.7, 0.6\}$ for w_3
- $\{1.0, 1.2, 1.4, 1.6\}$ for w_4
- $\{0.9, 1.0, 1.15, 1.25\}$ for w_5

Due to the mass of the tested combinations of class weights, it is not worth considering to present the full list of them in the results. It would be serviceable though to provide a subset of them which, we hope, will enlighten the purpose and the usefulness of this experimental process.

The first row of the table will be our reference point. As we have slightly modified only the weight of the third class to 0.9 the metrics observed are the closest we can to the original non modified model. We will compare the rest of the models with this one to see what influence the changes in weights have on the performance of the model.

In rows 2 and 3 we reduce even more the the weight of the third class leaving the weights of the other classes untouched. The initial reduction leads to a noticeable raise of the metrics that concern the goal class, with little effect on the average precision and recall of the model. The second reduction though has clearly a negative impact on the performance of the model, which is mostly reflected on the Matthews's correlation coefficient metric.

In the next two rows we modify the guide model by gradually increasing the weight of the fifth class. The aim of this action is to boost mainly the goal related metrics, with the hope that this will also benefit the general performance of the model. Indeed, by increasing the fifth weigth by 15% we observe higher goal precision and recall as well as a higher Matthews's correlation coefficient and at the same time the average precision and recall are maintained at the same level. But once again at the second increment all the metrics are going downwards.

In rows from 6 to 8 we are attempting to modify two class weight at once. We chose to enhance the two less represented classes, which are the first and the forth one. At first by equally increasing their weights by 20% we notice a discouraging reduction of the performance for almost every metric. However by increasing their weight even more we

get a model with average metrics as good as those of the guide model and even better goal related metrics. Lastly, when we increase the weight of the fourth class to 1.6 in the next row we discern mixed changes to the metrics of the model, with goal related being increased, average precision and Matthews's correlation coefficient being decreased and average precision remaining unchanged.

In rows 9 and 10 we display two combinations of weights that did not work while in the last three rows three successful combinations are presented. With the combination of the 11th row the maximum goal recall was achieved. The model with the class weights of the 12th row has the highest Matthews's correlation coefficient, average precision and average recall metrics. Last but not least, the model with the last combination of class weights demonstrates the best performance as far as goal precision is concerned. Taking everything into consideration we decided to adjust our final model using the last combination not only because we have selected the goal precision as the most suitable metric of evaluation but also because the rest of the metrics are fairly high in comparison to the rest combinations.

Experimental Tuning of the Model using Class Weights										
	weights					metrics				
#	w_1	w_2	w_3	w_4	w_5	MCC	Average Precision	Average Recall	Goal Precision	Goal Recall
1	1.00	1.00	0.90	1.00	1.00	0.15	0.33	0.31	0.38	0.41
2	1.00	1.00	0.80	1.00	1.00	0.15	0.31	0.30	0.44	0.50
3	1.00	1.00	0.60	1.00	1.00	0.10	0.29	0.26	0.39	0.41
4	1.00	1.00	0.90	1.00	1.15	0.16	0.33	0.31	0.46	0.50
5	1.00	1.00	0.90	1.00	1.25	0.13	0.32	0.29	0.42	0.45
6	1.20	1.00	0.90	1.20	1.00	0.14	0.31	0.29	0.38	0.36
7	1.40	1.00	0.90	1.40	1.00	0.15	0.32	0.30	0.42	0.45
8	1.40	1.00	0.90	1.60	1.00	0.14	0.30	0.30	0.44	0.50
9	1.00	1.20	0.80	1.00	0.90	0.08	0.28	0.26	0.36	0.36
10	1.40	1.00	0.90	1.20	1.00	0.12	0.30	0.29	0.33	0.32
11	0.90	1.20	0.60	1.60	1.15	0.15	0.32	0.30	0.48	0.59
12	0.90	1.40	0.70	1.40	1.15	0.20	0.39	0.34	0.44	0.55
13	1.00	1.00	0.60	1.20	1.25	0.19	0.36	0.33	0.50	0.55

Table 3.5: Comparison of the performance of models with different class weights

3.5 Strategies of reckoning the final outcome

Until now we have decided which is the most efficient classification algorithm, we have used this classifier to train and build our model and we have tuned that model experimenting with the class weight parameter in order to boost its performance. The final step to complete the process of reckoning the final outcome of any completed match of the season is to determine the way in which the predicted classes of the attempts will be handled. We have developed three different strategies for that task, which are described below. In order to make our strategies clear and better understood we provide one simple example for every one of them.

3.5.1 Absolute Strategy

The first strategy is the most simple and obvious one. Its name is inspired by the absolute trust we show to our model. We assume that the model has made 100% accurate predictions. Thus it is enough just to count the attempts that were predicted to belong to class no 5 not only to define the winner but even the accurate score of the match.

3.5.1.1 Absolute Strategy Example

Let us assume that our model has classified seven attempts as $[2, 3, 5, 4, 5, 5, 1]$. The three first attempts belong to the home team and the rest four to the away team. Our absolute strategy suggest that whatever the model predicted is true. So it suggests that the home team scored one goal in its third attempt and the away team scored two goals in its second and third attempt respectively. Consequently the winner is the away team with score 1-2.

3.5.2 Probabilistic Strategy

This strategy is more sophisticated in comparison to the first one. In this case we accept the indisputable fact that our model is far away from being perfect and we try to deal with its inaccuracy using probabilities. As a preparatory step we need to detect the most probable real labels for every predicted class. This information can be directly extracted by the confusion matrix of the classification. The following confusion matrix was formed when trying to predict the matches of the last three matchdays (11^{th}-13^{th}) of the first half of the greek Super League by using the first ten matchdays of the season as a training set.

Let us take the class no 2 as an example. Examining the 3^{rd} row of the matrix we observe that the most probable label when our model predicts an attempt to belong to the 2^{nd} class is 3. The second higher probability is the real label to be the same with the predicted one. And the third more likely real label is 5. Making a list of the highest probabilities of the real labels for an attempt predicted to be in the 2^{nd} class we get [3, 2,5]. Similarly working for the rest of the classes we get the following lists with the most probable labels.

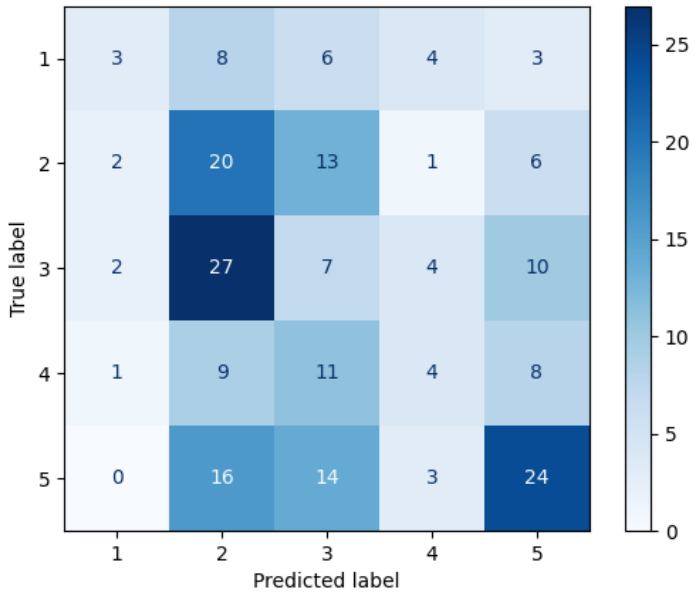


Figure 3.4: The confusion matrix of the predictions of the attempts from matchdays 11, 12 and 13, made by a model which was trained on the first ten matchdays

Most probable real labels			
predicted label	probable real labels		
	1st	2nd	3rd
1	1	2	3
2	3	2	5
3	5	2	4
4	4	3	1
5	5	3	4

Table 3.6: Comparison of the performance of models with different class weights

Now that we have access to that information we are able to describe the probabilistic strategy in more detail. The main idea is to create all the combinations of probable labels by replacing each predicted label with the three most probable ones. Thus 3 to the power of the count of attempts of the match different combinations are produced. Every combination can be interpreted as either a home win, a draw or an away win. Each combination ”votes” for the most probable outcome according to its own labels. By summing up the

number of votes for each outcome we can determine the outcome of the match as the one which collects the most votes. We can also estimate the probability of each outcome as the quotient of the number of votes in favor of this outcome to the total number of votes.

One detail that must be clarified here is that each vote has not the same weight. That occurs because as we noticed in the confusion matrix, the three most probable real labels for each predicted label occur with different frequencies. Therefore combinations are divided into three categories, depending on their composition. There are combinations that consist mainly of true labels with the highest probability, those which consist mainly of true labels with the second higher probability and those whose true labels have the least high probability. The votes of the combinations of the first category have an extra weight while the votes of the combinations of the last category have less influence on the determination of the final outcome.

3.5.2.1 Probabilistic Strategy Example

In this example we will introduce only two attempts, one for each team so that the number of combinations ($3^{numberofattempts}$) remains manageable. Let us assume that the model predicted one mediocre attempt for the home team and a very good attempt (goalkeeper's save or of the post attempt) for the away team. So our list of predictions will be $[3, 4]$.

We observe in the confusion matrix above that the most probable real labels when the predicted label is 3 (mediocre attempt) are 5, 2 and 4. So the first attempt produces the following combinations: $[5]$ $[2]$ $[4]$

Now we examine the second attempt. The predicted label is 4. The most probable real labels are, according to the confusion matrix, 4, 3 and 1. So the algorithm of the probabilistic strategy will execute the following steps:

Triple the already existing combinations

We will get:
$[5]$ $[2]$ $[4]$ $[5]$ $[2]$ $[4]$ $[5]$ $[2]$ $[4]$

To the first $number\_of\_combinations\_before\_tripling$ *add the first most probable real label. To the next* $number\_of\_combinations\_before\_tripling$ *add the second most probable real label and to the last* $number\_of\_combinations\_before\_tripling$ *add the third most probable real label*

We will get:
$[5, 4]$ $[2, 4]$ $[4, 4]$ $[5, 3]$ $[2, 3]$ $[4, 3]$ $[5, 1]$ $[2, 1]$ $[4, 1]$

We have produced all the possible combinations of the three most probable real labels of each predicted attempt. We can now examine these combinations to predict the final outcome and its probabilities. We notice there are three combinations that indicate a home win (1st, 4th and 7th), six combinations vote for draw (2nd, 3rd, 5th, 6th, 8th, 9th) and no

combination that suggests an away win.

If the combinations had the same weights then our final prediction for the outcome would be a draw with probabilities of home win, draw and away win being $33\%, 66\%$ and 0% respectively. However we argued that some combinations have a higher chance to be real than others. As a typical example of this consider the first and the last combination. The first one consist of the two most probable real labels of each attempt, while the last one consist of the two less probable choices. Although in this example with only two attempts it is not very obvious, in the long run with more attempts, combinations of higher chance to get real tend to stack up to the beginning of the list and those with the lesser chance to show up in reality lie to the end of the list. Therefore we enhance the first items of the list with a weight of 1.33, we leave the middle of the list untouched and we downgrade the last items, giving them a weight of 0.66. As a result in this example we have three combinations which vote for home win with weights 1.33 and 1.0 and 0.66, six votes for draw with weights 1.33, 1.33, 1, 1, 0.66 and 0.66 and no combination votes for away win. So the final prediction will be still a draw and also the probabilities happen to be the same.

3.5.3 Cumulative Strategy

Our third and last strategy is quite simple. The predicted labels of the attempts of every team are summed up and then we compare the difference of the two sums. If this difference is lower than a threshold we have defined, then the outcome is considered a draw. Else if the difference is above this threshold then the team with the greater sum stands out as the winner. Depending on the difference of sums we also estimate the goal difference of the match. In order this strategy to be as efficient as possible it is of crucial importance to determine a good threshold. Once again we will trace back to our data.

The following table contains the difference of the sum of the labels of the attempts of the two opponent teams. The columns represent the goal difference of the match. The table includes the matches played in the first half of the season.

Although the mean values of the sums of differences of the labels suggest that we should define the minimum threshold to be between 7 and 8, we ran an exhaustive search for the determination of the best values of thresholds for the first half of the season. The search indicated that setting the minimum threshold lower, at 3, lead to a better differentiation between draws and short goal difference wins. Therefore 3 was qualified as preferable minimum threshold. Similarly the thresholds to discrete the wins with medium goal difference (1-2 goals) from the wins with short goal difference (0-1 goals) and the wins with wide goal difference (2-4 goals) from those with medium goal difference were set to 14 and 21 respectively.

Difference of the sum of labels per goal difference							
	Matches	Draw	1 Goal	2 Goals	3 Goals	4 Goals	5+ Goals
1	aek-ion	-	-	-	14	-	-
2	ari-ofi	14	-	-	-	-	-
3	oly-atr	22	-	-	-	-	-
4	pan-ast	2	-	-	-	-	-
5	pao-apo	-	-	-	-	15	-
6	the-gia	-	8	-	-	-	-
7	vol-lam	-	2	-	-	-	-
8	ari-gia	-	-	-	-	-	16
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
86	oly-ion	-	17	-	-	-	-
87	gia-pan	-	-	-	8	-	-
88	vol-ari	-	0	-	-	-	-
	Mean	7.38	8.92	11.82	20.13	24.0	16.0

Table 3.7: The difference of the sum of the real labels of the attempts of the opponents of each match classified according to the goal difference of the match

3.5.3.1 Cumulative Strategy Example

This example is really simple and sort. Assume that our list of predicted labels consist of twelve attempts, of which the first seven belong to the home team and the remaining five to the away team
$[3, 4, 2, 2, 4, 5, 3, 1, 1, 5, 5, 2]$

The cumulative algorithm will calculate the two sums of the predicted labels of the attempts, one for each team. The sum for the first team is $homeTeam = 23$ and for the second team is $awayTeam = 14$. Consequently the difference of the two sums will be $homeTeam - awayTeam = 23 - 14 = 9$ in favor of the home team. We defined our minimum threshold to be 3, so the observed difference is above it. This leads us to the conclusion that our prediction for the outcome of the match is a home win, despite the fact that our model has initially predicted two goals for the away team and one goal for the home team.

Now that we have thoroughly covered the entire process which will be utilized to reckon

completed matches we are able to proceed to the results and note the performance of each strategy.

3.6 Results

The following table presents the number of correct predictions of outcomes that every strategy achieves per matchday. In every matchday seven matches are played, with only exception the 19th in which the match between Asteras Tripolis and Appolon Smyrnis was not covered by the television due to snowfall. The first thirteen matchdays were used as a training set, so the predictions start from the 14th matchday.

The highest prediction rate, which is 58%, is being recorded by the probabilistic strategy. The cumulative strategy is the second best, being 53% accurate. The absolute strategy takes the third place, being only 1% less accurate than the cumulative one. An observation that can be made is that the highest accuracy for every strategy in a single matchday is six out of seven correct predictions and this rate has been only reached once for every strategy. Another observation is that there are some matchdays in which all of the applied strategies performed poorly. The most typical example is the 20th matchday, in which the cumulative strategy was unable to predict any of the outcomes correctly. Attempting to analyze the reasons why all of the tested algorithms failed, we notice that this was indeed a special matchday during which none of the ”big four” teams of the greek Super League (Olympiakos, Panathinaikos, AEK and PAOK) was victorious.

Motivated by this analysis, we should mention that the prediction rates observed do not fulfill our expectations. It may be true that the tested strategies would outperform a naive, dummy model which would randomly predict an outcome of a match to be one of the three possible choices, home win, away win and draw. However nothing could be further from the truth than suggesting that every outcome is equally possible in a match. In the majority of the matches one of the two teams is odds-on to win the match and the other one is the outsider. In the case of the greek Super League we claim that there is a noticeable difference in the quality and the dynamics among the teams. Thus a naive algorithm which would take advantage of this knowledge could propose that whenever one of the mentioned above ”big four” teams plays with an opponent that does not belong to that group, the team will be victorious. When two teams of this group play against each other then the outcome will be a draw. Lastly when none of the opponent teams belong to the ”big four” group the algorithm arbitrarily predicts the home team as a winner. This simple algorithm applied to the second half of the league would outperform all of the tested strategies, predicting correctly 53 out of 90 matches, which is equivalent to 59%.

Number of outcomes that were correctly predicted using each strategy				
Matchday	Num. of matches	Correctly Predicted		
		Absolute	Probabilistic	Cumulative
14	7	6	5	4
15	7	3	5	5
16	7	3	6	5
17	7	3	4	4
18	7	2	5	3
19	6	5	4	4
20	7	2	2	0
21	7	5	4	6
22	7	4	4	3
23	7	4	4	5
24	7	3	4	4
25	7	4	2	2
26	7	3	3	3
Total		47	52	48
Accuracy		52%	58%	53%

Table 3.8: Strategy comparison for completed matches

Our failure in achieving a higher rate than the one a simple algorithm that supports the favorite brings off leads us to the pursuit of new ideas which could at least surpass the upper bound of 60%. Such an idea would be to substitute the model we have built using the attempts off all the teams with fourteen new models, one model for every team of the league. This alternative approach has both pros and cons. One the one hand, each model will be customized to a single team, which may contribute to a deeper learning of the way the team attacks and scores. For example, one team may be very efficient in scoring from close distance while another may excels in scoring by long shots. So for the first team a Voronoi diagram in which the player who holds on to the ball is outside the boundary of the penalty area will probably correspond to a poor chance, but for the later one it may indicate a save or a goal. On the other hand this technique significantly reduces the number of

diagrams upon each model will be trained.

In practice, in order to save time and computational resources, we developed four customized models to compare with the central model we have demonstrated so far. The teams we selected to build models for are Olympiakos, Panathinaikos, Volos and Apollon Smyrnis. Their place in their final ranking of the season were 1st, 5th, 10th and 14th respectively. Due to the variance in their performances we support the view that every team of the league can be represented to a decent extent by one of the four selected teams. The customized models have not been tuned. Therefore the following table demonstrates the number of correct predictions and the prediction rates of the central model, in which ll the teams contribute, with and without tuning, against those of the customized models:

Strategy	num. of matches	central with tuning		centr. without tuning		customized models	
		#	%	#	%	#	%
absolute	51	28	55%	28	55%	24	47%
probabilistic	51	31	61%	32	63%	30	59%
cumulative	51	27	53%	28	55%	29	57%

Table 3.9: Comparison of 'central' models with customized models

As we notice the idea of introducing a customized model for each team instead of a central model in which matches and attempts of all the teams contributes to its training does not serve our purpose. The majority of the customized models are less accurate than both the tuned and not tuned central model. Even if the customized models based on the cumulative strategy slightly outperform the central models of the same strategy, their prediction rate is still lower than all the models which utilize the probabilistic method to interpret the predictions for the attempts. An unexpected fact that comes to light by this experiment is that the tuning of the model using class weights seems to have a negative impact on the accuracy of the model. We checked whether this condition is confirmed when examining the total number of the matches of the second half of the league using the probabilistic strategy, which give the highest rate, and the results are the following:

Probabilistic Models 90 matches		
	Not tuned	Tuned
Correct predictions	53	52
Accuracy	59%	58%

Table 3.10: Comparison of the performance of the model with and without tuning

It might be harsh to support the view that the tuning actually downgrades the quality of the model, since the difference of the two models is only one correct prediction, but certainly its importance in our case is not critical.

4. OUTCOME OF FUTURE MATCHES RECKONING PROCESS

In this chapter we will concern ourselves with future matches. To clarify the term 'future', we refer to matches which of course have been completed at least two years sooner than the time these lines were written. However all the data used to predict the outcome of the matches were available before the day the matches took place. Thus the exact same predictions could have been made from a week to a day before the events of the matches had actually taken place.

Our dataset for the current process will be the table presented in the statistics section of chapter three, namely the number of attempts per class every team had made until the 13^{th} matchday. Note that no Voronoi diagrams will participate in the estimation of the future outcomes. This time, instead of classifying the attempts made during the matches, we will use a different approach. First we calculate the mean number of attempts per match per class made by every team. This computation is performed twice for every team, once for the home matches and once for the away matches. For example, the champion of the examined season Olympiakos made on average 9 attacks when played home, which are further analyzed into 2 poor, 1.29 mediocre, 2.71 good, 1.57 very good attempts and 1.43 goals scored and 6 attacks in its away matches with 0.67 poor, 0.5 mediocre, 1.83 good, 0.67 very good attempts and 2.33 goals respectively.

Having this piece of information for every team of the season, we randomly generate the number of the attempts for both opponents for every class. We achieve this using the normal distribution with mean value the mean number of attempts of the specific class. The standard deviation is set to 1. Then the lists of attempts of both teams are being interpreted using one of the three strategies described in chapter 3 in order to determine the final outcome of the match. Because the dataset and the matches we will make predictions for are the same as in the previous process, the most probable real labels defined for the probabilistic strategy and the thresholds defined for the cumulative strategy remain the same. Due to the introduction of randomness in the process, we test every strategy three times to get more representative results.

Absolute Strategy				
Matchday	Num. of matches	Correctly Predicted		
		1st try	2nd try	3rd try
14	7	3	5	1
15	7	4	5	3
16	7	5	2	2
17	7	2	2	3
18	7	3	3	1
19	6	0	1	1
20	7	3	2	3
21	7	1	4	4
22	7	2	3	6
23	7	3	2	3
24	7	2	2	4
25	7	2	1	2
26	7	1	1	4
Total		31	33	37
Accuracy		34%	37%	41%

Table 4.1: Absolute strategy performance on predicting future matches

Probabilistic Strategy				
Matchday	Num. of matches	Correctly Predicted		
		1st try	2nd try	3rd try
14	7	4	4	3
15	7	2	3	4
16	7	3	5	4
17	7	3	5	2
18	7	3	1	1
19	6	3	3	3
20	7	3	2	4
21	7	3	4	5
22	7	4	4	3
23	7	4	5	4
24	7	4	2	5
25	7	3	2	5
26	7	3	2	2
Total		42	42	45
Accuracy		47%	47%	50%

Table 4.2: Probabilistic strategy performance on predicting future matches

Cumulative Strategy				
Matchday	Num. of matches	Correctly Predicted		
		1st try	2nd try	3rd try
14	7	2	2	4
15	7	3	5	2
16	7	5	5	6
17	7	4	4	4
18	7	3	2	2
19	6	3	3	0
20	7	1	3	2
21	7	2	4	4
22	7	4	2	3
23	7	6	2	4
24	7	5	4	3
25	7	1	3	2
26	7	2	4	3
Total		41	43	39
Accuracy		46%	48%	43%

Table 4.3: Cumulative strategy performance on predicting future matches

Similarly to the prediction of completed matches the probabilistic strategy tends to show more potential in predicting future matches. It also seems to be more stable, achieving prediction rates in a range between 47% and 50%, in comparison to the other two strategies in which more intense fluctuations are observed in their performances. However more testing has to be conducted in order to verify the correctness of this consideration. Although it is highly likely that better values of accuracy will be presented if more tests are done, all the applied strategies are far from the desirable prediction rates that we had set as a target.

Despite the fact that the achieved accuracy did not satisfy our expectations, the ability to predict the final ranking of the teams at the end of the 26th matchday, given the ranking of the first half of the season is remarkable.

Real Ranking			Average Ranking of 9 trials			
Teams	Real Rank Position	Real Points	Average Rank Position	Average Points	Position Range	Points Range
oly	1	65	1	61.78	1	55-71
the	2	53	2.67	48.78	2-4	44-54
aek	3	46	2.44	49.89	2-3	45-53
ari	4	45	5.44	40.89	3-8	36-49
pao	5	42	5.44	39.78	4-9	33-44
gia	6	40	5.78	39.33	4-8	31-42
ofi	7	37	6.22	37.22	4-9	31-42
ast	8	35	10.56	25.22	8-13	17-32
pan	9	32	11	26.44	8-13	20-32
vol	10	30	9.22	30.56	7-13	25-37
ion	11	26	8.67	30.11	6-12	21-39
atr	12	23	10.78	26.56	8-13	19-33
lam	13	18	12	24	10-14	18-29
apo	14	13	13.67	14.56	11-14	13-26

Table 4.4: The real ranking and the average ranking of the 9 trials

The above table includes plenty of information such as the average rank position and the average points every team collected in the 9 simulations. The two last columns represent the maximum and minimum rank position every team occupied and the minimum and maximum points it collected. Observing the two ranking we feel that the following remark are worth mentioning:

- In all of the conducted tests, Olympiacos was correctly predicted as the champion of the season. This success was quite expected, as Olympiacos was 8 points ahead of the second best A.E.K. on the first half of the season.
- PAOK, indicated as 'the' in the rankings, occupied the 2nd position, covering a 5 points difference from the second best of the first half of the season, AEK, which eventually finished 3rd. This ranking reversal was predicted by four out of nine simulations. In another four simulations AEK and PAOK maintained their initial positions,

2nd and 3rd respectively. Finally there was one simulation where AEK occupied the 2nd and PAOK the 4th place.

- Apart from the collected points of Asteras Tripolis, indicated as 'ast', the real points and positions of the rest teams are lying between the predicted ranges resulted by the simulations. While some ranges are wider than others, running multiple tests could be used as a technique to predict the final position and the total collected points of the teams of the league with a certain confidence interval.

- Although half of the final rank positions of the teams were predicted correctly, we can convert the average ranking table to the original ranking table with only three moves. The tables below are the original ranking table with the average predicted ranking with its points rounded in the nearest integer. In order to convert the second table into the first one we have to do the following:

  1. swap the 2nd with the 3rd row
  2. swap the 11th with the 12th row
  3. Move rows 11 and 12 three positions above

  This fact evinces that generally the dynamics of the teams are decently reflected in the predicted ranking.

- One last notice is that comparing the rankings of the first half of the season with the final ranking, little differences are to be observed. This clearly indicates that the Greek Super League is of the season 2021-2022 was a quite predictable league and that our strategy to base the predictions of the outcomes of the second half of the season to the first half of the season was a right choice.

Real Ranking		
Rank Position	Teams	Points
1	oly	65
2	the	53
3	aek	46
4	ari	45
5	pao	42
6	gia	40
7	ofi	37
8	ast	35
9	pan	32
10	vol	30
11	ion	26
12	atr	23
13	lam	18
14	apo	13

Average Ranking of 9 Trials		
Rank Position	Teams	Points
1	oly	62
2	aek	50
3	the	49
4	ari	41
5	pao	40
6	gia	39
7	ofi	37
8	vol	31
9	ion	30
10	atr	27
11	pan	26
12	ast	25
13	lam	24
14	apo	15

5. CONCLUSION AND FUTURE WORK

5.1 Conclusion

In the current thesis we discussed about the importance of the Voronoi diagram in analyzing football matches, as well as how Voronoi diagrams in combination with machine learning techniques could be utilized as an initiative approach to predicting football matches based on images rather than on statistical data. For this purpose we designed and implemented several classification models in order to become competent in reckoning the outcome of completed matches, whose final attempts are in our possession in form of Voronoi diagrams. Then we tested various methods we had developed for the reckoning of the outcome of completed matches, in order to predict future matches, this time using statistical data instead of Voronoi diagrams.

While the significance of the Voronoi diagram in football analysis is indisputable, its use in the prediction of outcome of football matches turned out to be inefficient. The highest accuracy we achieved was 59% in reckoning completed matches. Applying the same methods to generated data based on the attempts of the teams in the first half of the league led to an also inadequate accuracy rate of 50% for the prediction of the matches of the second half of the league. This prediction rate is noticeably lower than the 67,73% of accuracy that has been observed in the literature for the same dataset [37].

Although the algorithms developed in this thesis did not surpass previous research efforts, there are signs that they might be useful in computing possible ranges that concern the final rank positions of the teams and their collected points.

5.2 Future Work

There is no denying that many improvements could be made to the proposed methodology which could result in a more efficient prediction model. First and foremost it would be highly beneficial to gain access to players tracking data captured through wearable gps or camera technology. The effect of this would be the production of Voronoi diagrams of superb quality. This would also eradicate the problem of manual data entry which is both inaccurate and time consuming. Another possible way to improve our dataset would be to include data from more seasons. It would be interesting to study whether the additional training data would enhance the quality of the model or if the changing compositions and applied tactics of the teams would eventually have a negative impact on it.

Apart from improving the dataset, splitting the diagrams into two categories, goal and not goal and performing a binary classification instead of multinomial could also be examined as an alternative mode of action. Besides it is a matter of debate whether the division of the dataset in many not objectively distinct categories (mediocre attempt, good attempt, very good attempt)is advantageous for the model or not.

Moreover a useful suggestion to improve the effectiveness of the model would be to introduce data related to the skills of the player who performs the attempt. Although judging a player's skills involves some subjectivity, there is no doubt that features like shot accuracy could play a crucial role in how the attempt will end.

Last but not least, if extensive testing of the rejected algorithms or even testing of non-examined classification algorithms were to happen, as well as further tuning, the results could be surprising.

Python scripts and code execution instructions

The python scripts used by the author for the purpose of this thesis are available on github: https://github.com/ApostolosTheodorou/Football_Predictions_using_Voronoi_Diagrams

The following instructions may be helpful in the execution of the code.

Evaluation of different classification algorithms

To run an evaluation of the tested classifiers navigate to the *code/classifier_selection* directory and run:

-$ python3 evaluateClassifier.py

Estimated time for 1 fold: 12 min.

If you want to run multiple folds create more folds directories with the structure of the existing fold and name them Fold-2, Fold-3 etc. Then change the number of folds in file *evaluateClassifier.py* line 16.

Predictions of future or completed matches

To run a prediction for future or completed matches move the *savedModels* and *matchdays* directories in the *code/prediction* directory and run:

-$python3 driver.py -ma <future>/<completed> -f <first matchday of predictions> -l <last matchday of predictions> -mo <model path> -s <absolute>/<probabilistic>/<cumulative>

Example:
-$python3 driver.py -ma future -f 20 -l 26 -mo ./savedModels/RandomForest_oly_1_13_14_26.skops -s probabilistic

New models training

To train a new model run the following code:

-$python3 train_model.py -p <path to matchdays directory> -t <team> -c <classifier> -str <starting training set matchday> -etr <ending training set matchday> -ste <starting test set matchday> -ete <ending test set matchday> -s <model's name>

where:

- team is a three character string with possible values: All (for all teams to contribute to the building of the model), aek, ion, ari, ofi, oly, atr, pan, ast, pao, apo, the, gia, vol, lam

- classifier possible values: Decision Tree, SVC, Random Forest, Nearest Neighbors, Naive Bayes, Neural Network, Multinomial Logistic Regression
- arguments refering to matchdays can receive integer values from 1 to 26
- in the name of your model do not include file extension (.skops file extension is added by default)

Example: *-$python3 train_model.py -p ./matchdays -t oly -c 'Random Forest' -str 1 -etr 13 -ste 14 -ete 26 -s RandomForest _oly_1_13_14_26*

ABBREVIATIONS - ACRONYMS

GPU	Graphics Processing Unit
SVC	Support Vector Classifier
SVM	Support Vector Machine
k-NN	k-Nearest Neighbors
PCA	Principal Component Analysis
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
MCC	Mathew's Correlation Coefficient

THE TEAMS

oly	Olympiacos F.C.
the	P.A.O.K. F.C.
aek	A.E.K. F.C.
ari	Aris F.C.
pao	Panathinaikos F.C.
gia	PAS Giannina
ofi	O.F.I. F.C.
ast	Asteras Tripoli F.C.
pan	Panetolikos F.C.
vol	Volos NFC
ion	Ionikos F.C.
atr	Atromitos F.C. Athens
lam	PAS Lamia 1964 F.C.
apo	Apollon Smyrni

REFERENCES

[1] A.M. Andrew. Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters*, 9(5):216–219, 1979.

[2] Giovanni Angelini and Luca De Angelis. Parx model for football match predictions. *Journal of Forecasting*, 36(7):795–807, 2017.

[3] Pascal Bauer and Gabriel Anzer. Data-driven detection of counterpressing in professional football: A supervised machine learning task based on synchronized positional and event data with expert-based feature extraction. *Data Mining and Knowledge Discovery*, 35(5):2009–2049, 2021.

[4] Ryan Beal, Stuart E Middleton, Timothy J Norman, and Sarvapali D Ramchurn. Combining machine learning and human experts to predict match outcomes in football: A baseline model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 15447–15451, 2021.

[5] Murilo José de Oliveira Bueno, Maisa Silva, Sergio Augusto Cunha, Ricardo da Silva Torres, and Felipe Arruda Moura. Multiscale fractal dimension applied to tactical analysis in football: A novel approach to evaluate the shapes of team organization on the pitch. *Plos one*, 16(9):e0256771, 2021.

[6] A. Bykat. Convex hull of a finite set of points in two dimensions. *Inf. Process. Lett.*, 7(6):296–298, 1978.

[7] Fabio Giuliano Caetano, Sylvio Barbon Junior, Ricardo da Silva Torres, Sergio Augusto Cunha, Paulo Régis Caron Ruffino, Luiz Eduardo Barreto Martins, and Felipe Arruda Moura. Football player dominant region determined by a novel model based on instantaneous kinematics variables. *Scientific Reports*, 11(1):18209, 2021.

[8] Timothy M Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete & Computational Geometry*, 16(4):361–368, 1996.

[9] Hong Chen, Songhua Hu, Rui Hua, and Xiuju Zhao. Improved naive bayes classification algorithm for traffic risk management. *EURASIP Journal on Advances in Signal Processing*, 2021(1):30, 2021.

[10] HesamAddin Torabi Dashti, Nima Aghaeepour, Sahar Asadi, Meysam Bastani, Zahra Delafkar, Fatemeh Miri Disfani, Serveh Mam Ghaderi, Shahin Kamali, Sepideh Pashami, and Alireza Fotuhi Siahpirani. Dynamic positioning based on voronoi cells (dpvc). In *RoboCup 2005: Robot Soccer World Cup IX 9*, pages 219–229. Springer, 2006.

[11] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry Algorithms and Applications*, pages 2–8. Springer, 3rd edition, 2008.

[12] Ryan Duarte. Utilizing machine learning techniques in football prediction. 2022.

[13] William F. Eddy. A new convex hull algorithm for planar sets. *ACM Transactions on Mathematical Software*, 3:398–403, 1 1977.

[14] Herbert Edelsbrunner. *Constructing Convex Hulls*, pages 139–176. Springer Berlin Heidelberg, Berlin, Heidelberg, 1987.

[15] Costas J Efthimiou. The voronoi diagram in soccer: a theoretical study to measure dominance space. *arXiv preprint arXiv:2107.05714*, 2021.

[16] I.Z. Emiris. *Computational geometry: a modern algorithmic approach (in Greek)*. Klidarithmos, 2008.

[17] Engin Esme and Mustafa Servet Kiran. Prediction of football match outcomes based on bookmaker odds by using k-nearest neighbor algorithm. *International Journal of Machine Learning and Computing*, 8(1):26–32, 2018.

[18] S. J. Hong F. P. Preparata. Convex hulls of finite sets of points in two and three dimensions. *Communications of the ACM*, 20:87–93, 2 1977.

[19] Javier Fernandez and Luke Bornn. Wide open spaces: A statistical technique for measuring space creation in professional soccer. In *Sloan sports analytics conference*, volume 2018, 2018.

[20] Steven Fortune. A sweepline algorithm for voronoi diagrams. In *Proceedings of the second annual symposium on Computational geometry*, pages 313–322, 1986.

[21] Peter J Green and Robin Sibson. Computing dirichlet tessellations in the plane. *The computer journal*, 21(2):168–173, 1978.

[22] Usman Haruna, Jaafar Zubairu Maitama, Murtala Mohammed, and Ram Gopal Raj. Predicting the outcomes of football matches using machine learning approach. In *Informatics and Intelligent Applications: First International Conference, ICIIA 2021, Ota, Nigeria, November 25–27, 2021, Revised Selected Papers*, pages 92–104. Springer, 2022.

[23] Kees Hendriks et al. Football data analysis. *Technische Universiteit Eindhoven*, 2016.

[24] Abel Hijmans and S Bhulai. Dutch football prediction using machine learning classifiers. *Research Paper Business analytics*, pages 1–24, 2016.

[25] Chinwe Peace Igiri and Enoch Okechukwu Nwachukwu. An improved prediction system for football a match result. 2014.

[26] Tugbay Inan. Using poisson model for goal prediction in european football. 2021.

[27] R. A. Jarvis. On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters 2*, pages 18–21, 12 1972.

[28] Steffen Kaden, Heinrich Mellmann, Marcus Scheunemann, and Hans-Dieter Burkhard. Voronoi based strategic positioning for robot soccer. In *Proceedings of the 22nd International Workshop on Concurrency, Specification and Programming. Warsaw, Poland*, volume 1032, pages 271–282. Citeseer, 2013.

[29] Michael Kallay. The complexity of incremental convex hull algorithms in rd. *Information Processing Letters*, 19(4):197, 1984.

[30] Stylianos Kampakis and Andreas Adamides. Using twitter to predict football outcomes. *arXiv preprint arXiv:1411.1243*, 2014.

[31] S Kim. Voronoi analysis of a soccer game. *Nonlinear Analysis: Modelling and Control*, 9(3):233–240, 2004.

[32] David G Kirkpatrick and Raimund Seidel. The ultimate planar convex hull algorithm? *SIAM journal on computing*, 15(1):287–299, 1986.

[33] Jan Kozak and Szymon Głowania. Heterogeneous ensembles of classifiers in predicting bundesliga football results. *Procedia Computer Science*, 192:1573–1582, 2021.

[34] Graham R. L. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters 1*, pages 132–133, 1 1972.

[35] James Law. Analysis of multi-robot cooperation using voronoi diagrams. In *3rd International RCL/VNIItransmash Workshop on Planetary Rovers, Space Robotics and Earth-Based Robots*, 2005.

[36] Michael J Maher. Modelling association football scores. *Statistica Neerlandica*, 36(3):109–118, 1982.

[37] Marios-Christos Malamatinos, Eleni Vrochidou, and George A Papakostas. On predicting soccer outcomes in the greek league using machine learning. *Computers*, 11(9):133, 2022.

[38] M. J. Moroney. *Facts from figures*, pages 96–98, 101–104. Penguin Books, 1956.

[39] Felipe Arruda Moura, Luiz Eduardo Barreto Martins, Ricardo O Anido, Paulo Régis C Ruffino, Ricardo ML Barros, and Sergio Augusto Cunha. A spectral analysis of team dynamics and tactics in brazilian football. *Journal of sports sciences*, 31(14):1568–1577, 2013.

[40] Bin Ning, Wu Junwei, and Hu Feng. Spam message classification based on the naïve bayes classification algorithm. *IAENG International Journal of Computer Science*, 46(1):46–53, 2019.

[41] Mikhail Prokopenko, Peter Wang, and Oliver Obst. Gliders2014: Dynamic tactics with voronoi diagrams. In *RoboCup 2014 symposium and competitions: Team description papers, Joao Pessoa, Brazil*, 2014.

[42] Md Ashiqur Rahman. A deep learning framework for football match prediction. *SN Applied Sciences*, 2(2):165, 2020.

[43] Charles Reep, Richard Pollard, and Bernard Benjamin. Skill and chance in ball games. *Journal of the Royal Statistical Society: Series A (General)*, 134(4):623–629, 1971.

[44] Robert Rein, Dominik Raabe, Jürgen Perl, and Daniel Memmert. Evaluation of changes in space control due to passing behavior in elite soccer using voronoi-cells. In *Proceedings of the 10th international symposium on computer science in sports (ISCSS)*, pages 179–183. Springer, 2016.

[45] Emil Rosén. Analysis and visualization of collective motion in football: Analysis of youth football using gps and visualization of professional football, 2015.

[46] Laurie Shaw and Mark Glickman. Dynamic analysis of team strategy in professional football. *Barça sports analytics summit*, 13, 2019.

[47] Johannes Stübinger, Benedikt Mangold, and Julian Knoll. Machine learning in football betting: Prediction of match results based on player characteristics. *Applied Sciences*, 10(1):46, 2019.

[48] K Sujatha, T Godhavari, and Nallamilli PG Bhavani. Football match statistics prediction using artificial neural networks. *International Journal of Mathematical and Computational Methods*, 3, 2018.

[49] Niek Tax and Yme Joustra. Predicting the dutch football competition using public data: A machine learning approach. *Transactions on knowledge and data engineering*, 10(10):1–13, 2015.

[50] SL Ting, WH Ip, Albert HC Tsang, et al. Is naive bayes a good classifier for document classification. *International Journal of Software Engineering and Its Applications*, 5(3):37–46, 2011.

[51] Fumiya Ueda, Honda Masaaki, and Horino Hiroyuki. The causal relationship between dominant region and offense-defense performance-focusing on the time of ball acquisition. *Football Science*, 11:1–17, 2014.