Decision Support in Racket Games

by

Anju Kumari

A creative component submitted to the graduate faculty

In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Simanta Mitra, Co-major Professor
Gurpur Prabhu, Co-Major Professor
Li-Shan Chou

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this creative component. The Graduate College will ensure this report is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2022

Copyright © Anju Kumari, 2022. All rights reserved.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank my major professor, Dr. Simanta Mitra, my co-major professor, Dr. Gurpur Prabhu, and my committee member, Dr. Li-Shan Chou, for their guidance and support throughout the course of this research.

ABSTRACT

In order to prepare for the next match, players examine enormous amounts of footage throughout practice to determine a potential opponent's patterned behavior and strategic tendencies. However, viewing and analyzing an opponent's previous matches takes a lot of time, so the players often end up analyzing only a few or recent matches of the potential opponent. This clearly isn't optimal in terms of predictive analytics because the match they're looking at may not be a representative match of their opponent. Many important factors that contribute to players' playing style might get overlooked by tennis players in the given timeframe. This may lead to misinterpretation of the opponent's important strategic elements or the opponent's likely playing style in subsequent matches.

The racket games lack the use of technology to assist tennis players in learning the game and help them improve despite its popularity. And, there is an enormous amount of data generated in every professional racket game in the form of vision-based tracking data and also text-based data. Given the abundance of historical tennis data, players' characteristics and match characteristics might all be combined to create a set of labeled training examples.

This project explores the application of supervised machine learning algorithms to provide the decision support for the tennis players by recommending them the shot type and serve direction that the opponent is most likely to hit in a particular context of the game.

The machine learning models like Random Forest, KNN, SVC and also the latest classification algorithms like Catboost are used to categorize the features and predict the outcome.

CHAPTER 1: INTRODUCTION

1.1 Background

Racket sports, such as tennis and badminton, are genuinely global sports, with players originating from over 100 nations. In these sports, a racket is held in one hand and is used to swing a ball or a shuttlecock towards the opposing team's side of the court or field. Due to a combination of frequent changes of direction and rapid sharp movements of the upper and lower limbs, all require a high level of agility, core stability, flexibility, and movement control.

In order to prepare for the next match, players examine enormous amounts of footage throughout practice to determine a potential opponent's patterned behavior and strategic tendencies. However, viewing and analyzing the video recordings requires a lot of time, so the players often end up watching only a few or recent videos of the potential opponent. This clearly isn't optimal in terms of predictive analytics because the match they're looking at may not be a representative match of their opponent. There are several factors which can help the players excel in their upcoming match, such as (a) the serve direction the opponent often plays, (b) type of shots (left or right), or (c) the type of errors the opponent often makes. These important factors might get overlooked by tennis players in the given timeframe. The key strategic aspects of an opponent or the opponent's probable playing approach might get misinterpreted in subsequent games.

1.2 Solution

Given the abundance of historical tennis data, players' characteristics and match characteristics might all be combined to create a set of labeled training examples. These samples could be used by a supervised machine learning algorithm to deduce a function for predicting the playing style of the opponent tennis players.

Since predicting the outcome of matches is difficult due to the numerous elements that influence it, the goal of this project is to train machine learning models using the available datasets to predict the shot type and server direction of tennis players, given the context of the match. In order to achieve this goal, the problem is reduced to subsequent subproblems in order to provide an appropriate recommendation for the tennis players:

1. The shot type of the player,
2. The player's serve direction

The project's objective is to explore how much the machine learning techniques can improve the classification of the serve direction and shot type of tennis players. First the raw dataset is cleaned and processed by removing the null and garbage values then features are extracted. Since there are 32 features, we opted for PCA(principal component analysis) to reduce the dimensionality of the dataset for analysis. Then the reduced features are normalized using

standard scaling and divided into train and test datasets. The train dataset was trained on a Random Forest, a KNN, SVC and Catboost model. Next, the classification accuracy is computed on the test data which is unseen data for the models. Furthermore, hyperparameter selection and tuning are conducted for each ML model in an effort to achieve higher accuracy.

1.3 Organization

This document is organized as follows from here on : Chapter 2 gives an overview of the related work on this topic, and in Chapter 3, our approach using the machine learning models is described. Chapter 4 shows the results, limitations, and result summary of the models used. Chapter 5 provides a summary and future work on the entire project.

CHAPTER 2: RELATED WORK

In this chapter, we will discuss the related work using the tennis dataset. With the recent installation of player and ball monitoring systems in professional sports such as basketball, soccer, and tennis, many studies on how to exploit such data sources have been conducted. Most of this work is done using vision-based tracking data, such as the hawk-eye dataset, which is collected using the hawk-eye technology. This technology is being used in most sports to help with umpiring decisions and to visualize the shot trajectories.

Using the hawk eye dataset, there have been several studies done on predicting the shot type and predicting the serve using the tennis players' style priors. In the paper [1], they have proposed a deep neural network framework to learn the behavioral patterns of tennis players and predict shot type and location. The acquired knowledge from the proposed memory modules can be used to demonstrate player behavior changes while facing different circumstances due to the contingent nature of the proposed framework. [1]. Using Hawk-Eye data from three recent Australian Open Grand-Slam Tournaments, one study[2] presented an approach to predict the most likely serve a player will hit in a particular match-context, against a specific opponent, using the player's style prior. The aim of this paper[6] is to use such high-quality data to model player behavior and forecast the next shot based on the current match environment and prior shot information. Both the shot location (where) and the shot type are included in the forecast (what). In another study [5], they propose that, given the information from the previous shot, the aim of this study is to accurately anticipate where the next shot could be.

This research analyzes match and player characteristics and proposes trained machine learning models for forecasting a player's most likely shot type and serve direction, keeping in mind how innovation has transformed how broadcast spectators, players, coaches, and commentators perceive tennis.

CHAPTER 3: OUR APPROACH

In this chapter, we will discuss the experimental setup of our approach. There are four machine learning models explored in this project and this chapter briefly summarizes how each works. Furthermore, the details about the ML algorithms, data processing, feature engineering using principal component analysis and hyperparameter optimization are explained.

3.1 Experimental Setup

3.1.1 Technologies

The Python programming language was used to implement all of the data processing for the data flow system components. In particular, Python's scientific computing libraries, like NumPy2[21] and Pandas3[22], have made the implementation clear and effective. These two libraries offer a simple user interface for manipulating huge datasets in memory.

The Sklearn library is used for machine learning methods like, randomforest, knn, and SVC which. For CatBoost, Yandex's open-source software package called Catboost[7] is being used in the project to train datasets.

The data processing and model training are performed on the Windows operating system with the help of JupyterLab which is a web-based interactive computing platform. The computer has 28 GB RAM.

3.1.2 Data Flow

There are five steps in the data flow diagram as shown in figure 3.1.2 that must be completed in order to achieve the desired outcome. The following are these steps:

1. Data Preparation
2. ML models Training
3. Evaluate models
4. Hyperparameter Tuning
5. Final classification

For data preparation, the dataset is cleaned and processed using techniques like correlation graph, PCA, and pandas statistics to get rid of garbage value as described in section 3.2. After the data is cleaned, it is saved in a CSV(comma-separated values) file. Each machine learning algorithm reads the dataset from the CSV file and normalizes the dataset to get on the same scale before training. After that, the processed dataset is divided into training and testing sets using random shuffling. To control the shuffling process during the splitting of the dataset, the hyperparameter random state is used to maintain the same dataset proportion while retraining. For this project, 90% of the data is taken as training and 10% of the data is used for testing.

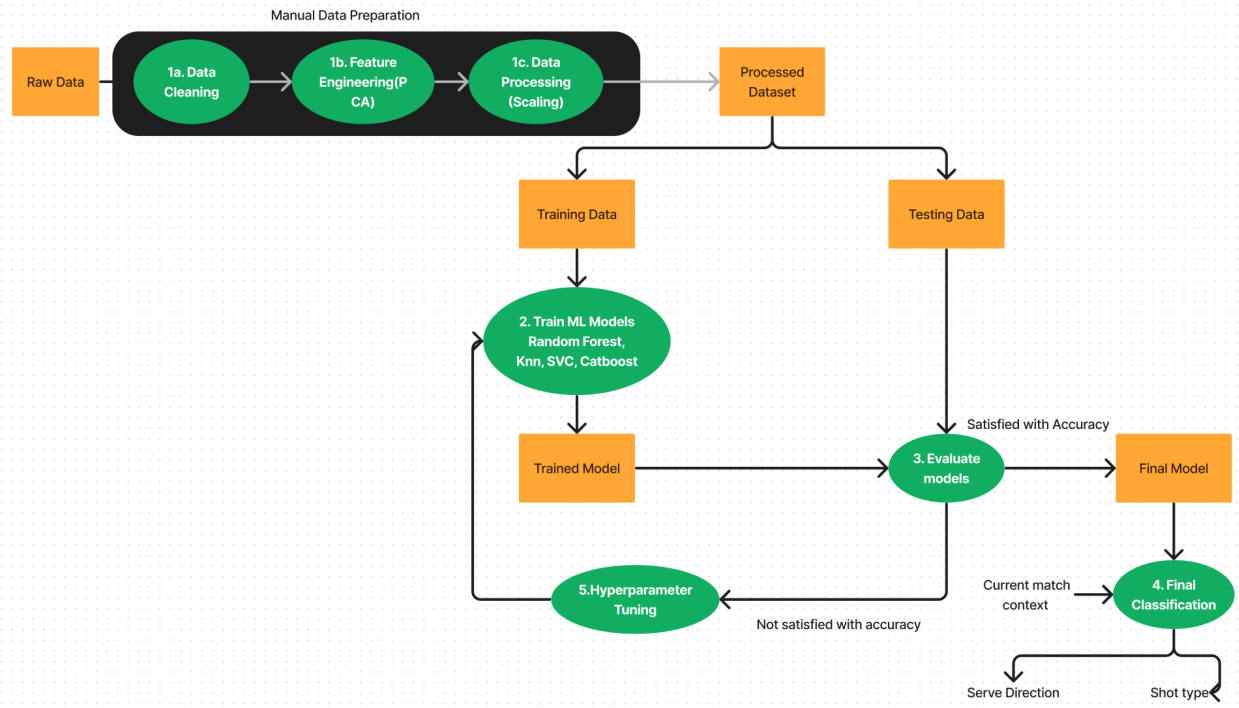

Figure 3.1.2 Data flow diagram

In the next step, the training dataset is then passed to the machine learning models, which learn from the data. Once the training is complete, the trained final model is produced which can be saved using python's pickle library. Finally, the testing dataset is evaluated on this final model which is explained in section 3.3. To achieve higher accuracy, each model has its own set of hyperparameters that are tuned until desired accuracy is achieved or model overfits.

3.1.3 Efficiency of the ML models

There are no restrictions on the efficiency of the system because once the models are trained, the desired classification can be produced in time for an upcoming match with a fair amount of resources (compute power and memory).

3.2 Data Preparation

For this project, a public source for raw tennis players is used which contains the Wimbledon men's single matches and it is saved in a CSV file [19]. The dataset consists of tennis match records which contain shot-by-shot data such as the type of shot, direction of shot, depth of return, sorts of errors, and more features for each point in a match. This is the only publicly accessible data of its kind, a crowdsourced effort to collect detailed shot-by-shot information for professional tennis matches. The rally sequence is exploded into rows, resulting in a dataset that is 875.7 MB in size. 90% of the dataset is utilized for model training, and 10% of the dataset is used for model testing.

The following features are extracted from the historical raw data of the tennis players:

1. Set Number
2. Game Number
3. Points scored
4. Player who is returning (1 or 2) the serve
5. Is the player who is serving the winner of the rally?
6. If the shot fault is a forced error or an unforced error,
7. Serve fault
8. Direction of the serve
9. Depth of the shot
10. Direction of the shot
11. court position of the shot
12. Previous rally sequence count
13. Player's hand

3.2.1 Data Processing

As it is discussed in section 3.2, this data source contains shot-by-shot data for every point of a match, including the type of shot, direction of shot, depth of return, types of errors, and more. However, the raw data contains these details in a coded format like 6b28f2b2b1f1d#. This code contains details about all the shots played in a rally. Each letter and number has its own meaning. For example, in this case, the first numeric character tells us the serve type, and the rest of the coding is for the shot type, shot direction, and shot depth, the next three codes are b28, where b denotes shot type of 'backhand groundstroke', 2 denotes shot direction as' down the middle of the court ', and 8 denotes shot depth as' behind the service line, but closer to the service line than the baseline'.

The following features are extracted from the coded rallies:

1. Serve direction
2. Shot type
3. Shot direction
4. Shot depth
5. Error type
6. Court Position

After feature extraction, coded data is separated into different columns. Then, the data is exploded into rally-length rows with information about each shot in each row.

The next step is to get the information, such as:

1. First Player

2. Second player
3. Match Id
4. Player 1 hand
5. Player 2 'hand
6. Date of the match
7. Point scored by first player
8. Point scored by second player
9. Game won by first player
10. Game won by second player
11. Set won by first player
12. Set won by the second player
13. Previous rally length
14. Is there a second serve (first serve errored)?

- First player, second player, and match id information are extracted from the id attribute given in the raw data in the form "20211121-M-Tour_Finals-F-Daniil_Medvedev-Alexander_Zverev". The id is split with '-' as a parameter and saved it in new columns.
- Points scored by the first and second player were extracted from the variable "Pts" from the raw data. It has data in the form of '0-0'. The Pts is split with '-' as a parameter and stored the first value for the first player and the second value for the second player. Sometimes the data has 'AD' for Pts values. "AD" is taken as 50 points for each of the players in this case.
- First player and second player strong hand information has been extracted from the other charting files by Jeffsackmann's 'charting-m-matches.csv'

- The date of the match is also extracted from the file 'charting-m-matches.csv'.

- The Prev rally count is extracted from the prev row variable 'RallyCount' and stored in the new column 'Prev RallyCount'. This factor is important for analyzing the behavior of the player after long or short rallies.

- The information if the first serve went in or the server played the second serve is extracted and stored in the new column 'First Serve', True/False. This is extracted by analyzing the rally length with the short-coded string stored in the 1st serve. If the length of code is less than the rally count, then 2nd serve is considered.

- Garbage values like "#NAME?" and other random strings are removed from the columns to process one label encoding.

- All this information was extracted from the raw data using a Python script in Jupyter Notebook.
- Before cleaning, the size of the dataset was 573103.
- After data processing and cleaning, the size of the dataset is 2889978.

After cleaning the dataset, it is also important to visualize the dataset to understand how data looks by plotting the graphs. For data visualization, graphs are created using libraries Matplotlib.pyplot[23] and Seaborn[24].

Fig 3.2.1a shows the serve direction of the serve played by different players in various matches. As it can be seen from the graph Figure 3.3.1a that some down the T and out wide are more common serve direction than the body type serve direction among the tennis players. This gives the general idea about potential opponents since the data displays the player-by-player serve direction for the upcoming match. However, there are many other game-related factors, like the player's hand or the previous rally count, that might affect a player's tendency to hit in a specific serve direction. This data is utilized in this research to predict the opponent's most likely serve direction in a particular match scenario.

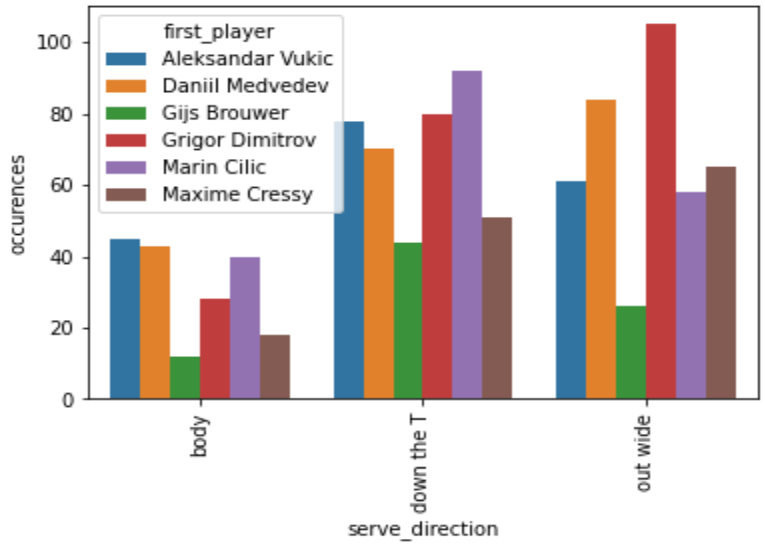

Figure 3.2.1a Serve direction of the shots played by different players in various matches

Fig 3.2.1b shows the shot types played by different players in various matches. As it can be seen in the Figure 3.3.1b, some of the shots like backhand groundstroke, backhand lob, backhand slice are very common among the players while some of the shots like backhand half-volley, forehand

half-volley are not very common among the players. This also gives the general idea about potential opponents since the data displays the player-by-player shot type for the match.

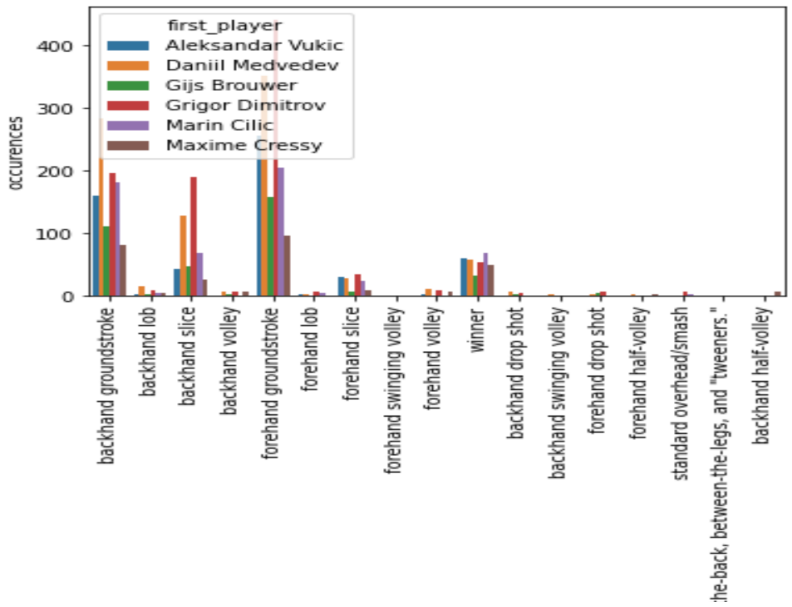

Figure 3.2.1b Shot types played by different players in various matches

However, similar to serve direction, there are many other players' characteristics and game-related factors, like the player's hand or the previous rally count, that might affect a player's tendency to hit in a specific shot type. This information is utilized in this study to predict the shot type that opponent is most likely to hit in a particular match scenario.

3.2.2 Feature Engineering

In the dataset, there are 32 features available, but not all of them are important for training the model. For example, the match_id or date is not needed to decide the shot type of a player. This is intuitive and can be understood by looking at the data. However, some of the features are so closely related to the outcome that you are not so sure about how much importance they hold in order to get the outcome with good accuracy.

In this kind of situation, principal component analysis (PCA) was used to do the feature engineering. For example, rally count, previous rally count, and players' handness. It is hard to decide which is more important and which one to discard. As it is known, machine learning models require data to be trained, and if there is less data and more features, then there is a chance for the model to overfit. To overcome this, it is necessary to reduce the dimension of the

dataset so that it contains the majority of the information even if some of the variables or features of the dataset are lost. [18]

It is important to note that the data preparation is performed for both the classification models separately. All the figures used in this section are examples from serve direction classification.

There are basically three steps to performing PCA:

1. **Standardization (Dataset scaling)**

   This phase is used to normalize the range of continuous initial variables so that they all contribute equally to the analysis. Mathematically, this can be achieved by subtracting the mean and dividing by the standard deviation for each value of each variable, so that all the variables are transformed to the same scale.

$$z = \frac{value - mean}{standard\ deviation}$$

2. **Covariance Matrix Computation**

   The goal of this phase is to figure out how the variables in the input data set differ from the mean in relation to one another, or even if there is a relationship between them. because variables might be highly connected to the point where they include redundant data. The covariance matrix is computed in order to find these correlations. It's the sign of the covariance that counts; if it's positive, the two variables are correlated. If the value is negative, then they are inversely correlated, i.e., one increases while the other decreases.

   Figure 3.2.3a is an example of the correlation diagram in case of serve direction classification which denotes a close association and a more distant correlation between the features. This heat map (covariance matrix) shows that ranking features based on their correlation with the other features would be an easy way to choose them. The heatmap determines how much two variables are linearly dependent on one another by looking at their correlations and comparing each feature to every other feature of the dataset. Evaluating this linear relationship between the features also helps in order to understand the data. However, features overlap with one another when utilized in a machine learning system. For instance, combining two variables that are useless separately may be beneficial. As a result, we choose the subset of features that perform well rather than assessing each feature individually with the help of principal components.[18]

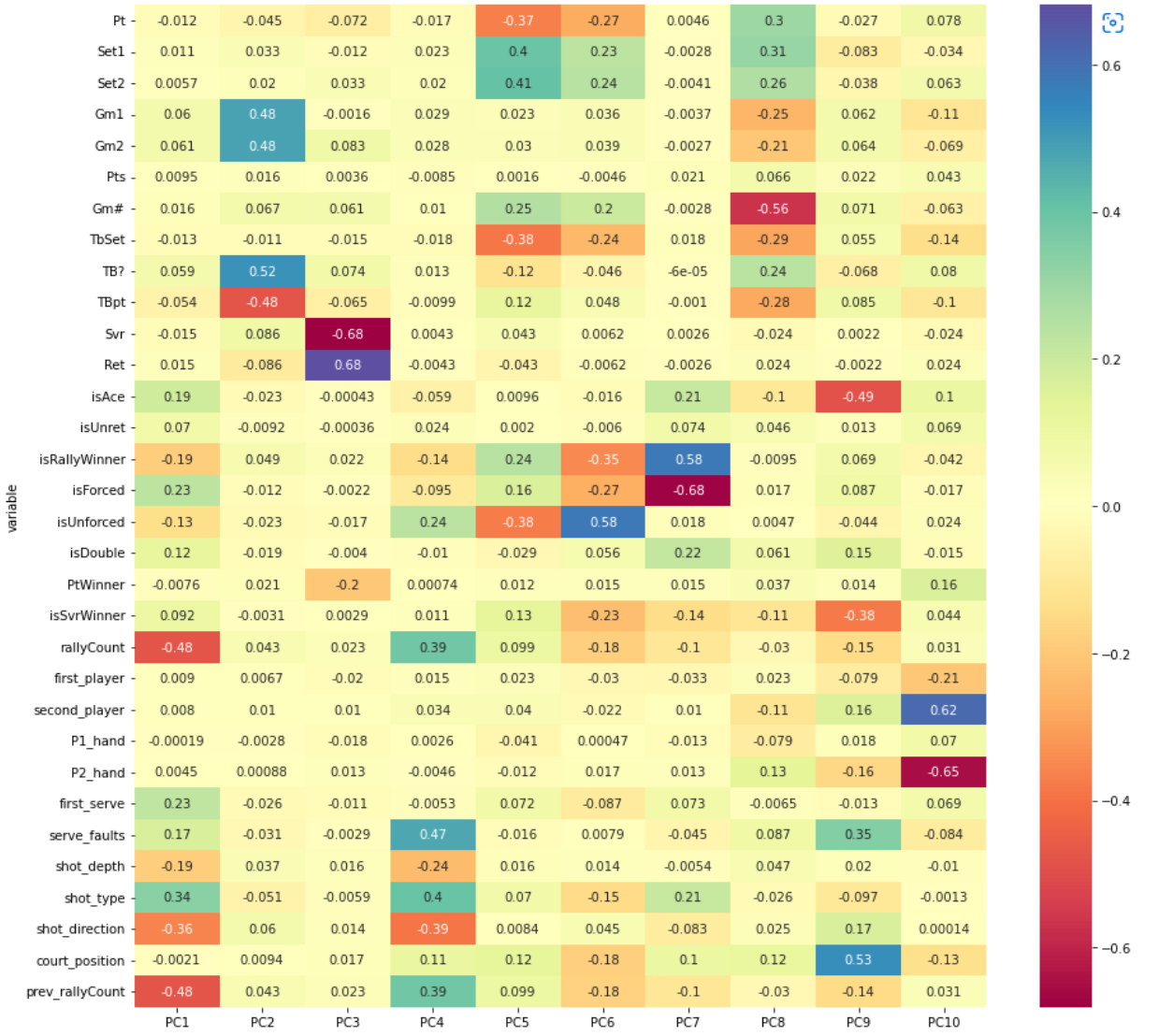

Figure 3.2.3a Covariance matrix between variables and PCA

The color or brightness of a heatmap is inversely related to the correlation coefficient. In the data, the colors red and blue denote a more distant association and a close correlation, respectively.

3. Principal Component

In this phase, the principal components of the dataset are discovered by computing the eigenvectors and eigenvalues from the covariance matrix. Principal components are new variables that are created by combining or mixing the basic variables in a linear way. The new variables, i.e., principal components, are uncorrelated as a result of these combinations, and the majority of the information from the initial variables is squeezed or compressed into the first components, then the maximum remaining information in the second, and so on. The principal components for our dataset are as shown in the

Fig.3.2.3b PCA. Fig.3.2.3c PCA shows the top 10 features after dimensionality reduction. These 10 features contain most of the information in the dataset. As it can be seen in figure 3.2.3b that , these features have an explained variance ratio of at least 5.

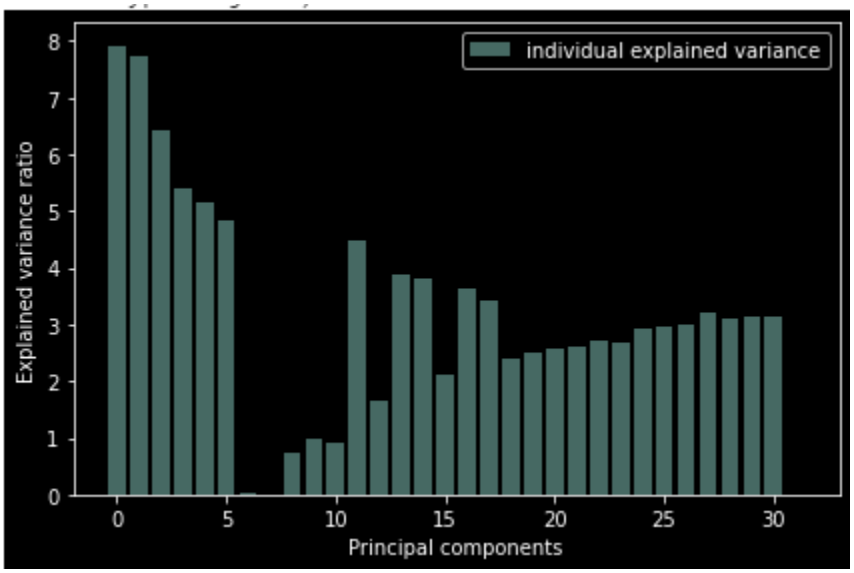

Figure 3.2.3b Principal component Analysis of 31 features

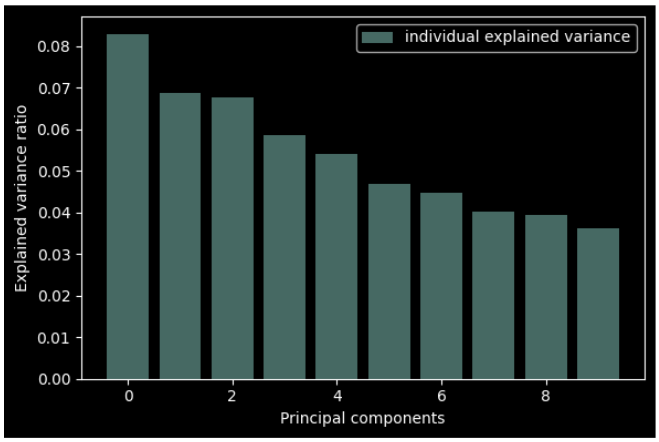

Figure 3.2.3c Principal component Analysis of 10 features

Figure 3.2.3d shows that most of the features are red which demonstrates the strong relationship between the top 10 PCAs which are selected because they are closely related and have most of the information of the dataset.

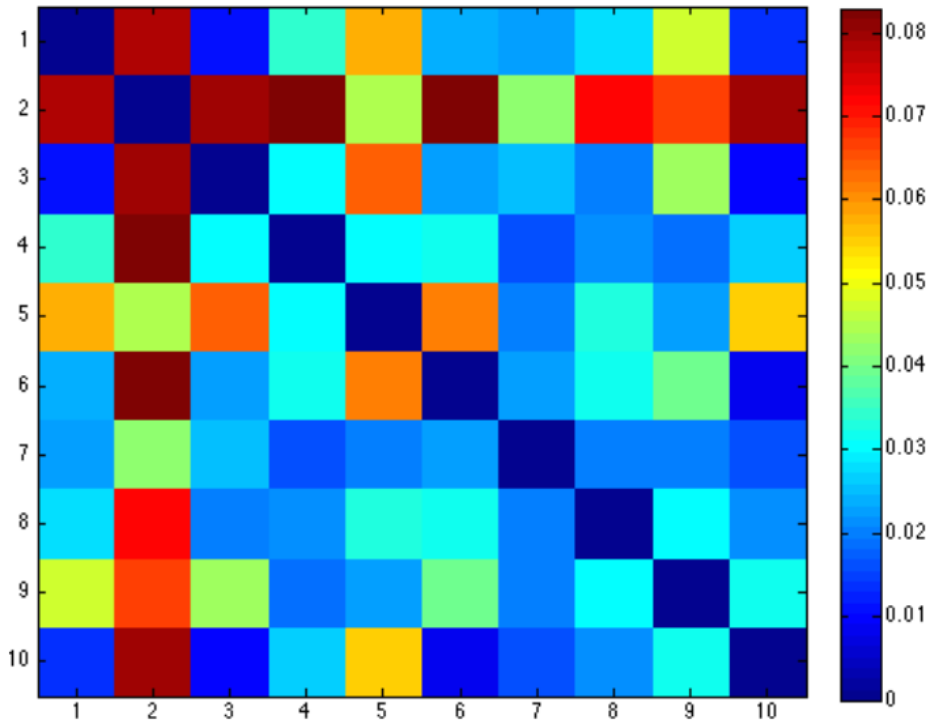

Figure 3.2.3d Correlation between top 10 features after PCA

3.3 Machine Learning

This section gives an overview of the machine learning and the ML algorithms that are used in this project for training the models on the tennis dataset.

3.3.1 Overview of ML

Machine learning is a subset of Artificial Intelligence (AI) algorithms that learn from data as shown in Figure 3.3: AI vs ML vs DL. ML learns from the structured data based on the extracted features that are fed to them. However, there is another subset of machine learning which is called deep learning(DL). DL uses multilayer neural networks to learn from vast amounts of data and has the potential to adapt on its own to extract features and learn from it as shown in fig 3.3b [17].

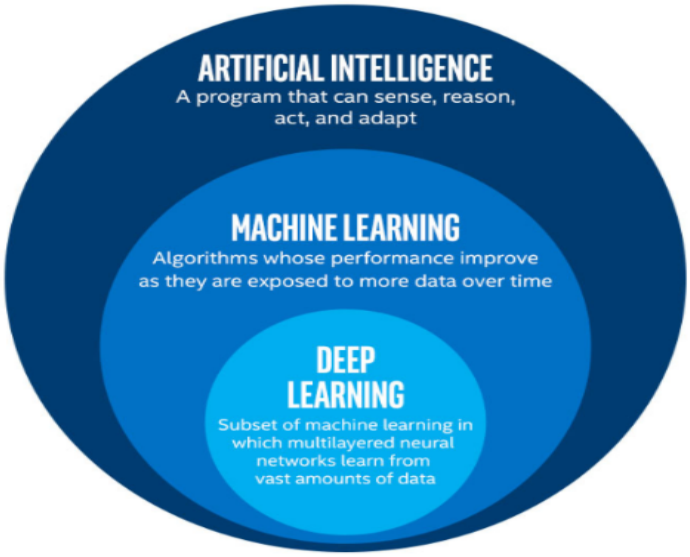

Figure 3.3: [AI vs ML vs DL]

This project explores supervised machine learning, which infers a function from a set of labeled training instances, where a labeled instance is a pair consisting of an input vector and the desired output value. In the case of tennis, input vectors i.e the set of training examples are created using historical tennis data which can contain numerous match and player characteristics. The shot type and serve direction are the output values for a particular match's input vector.

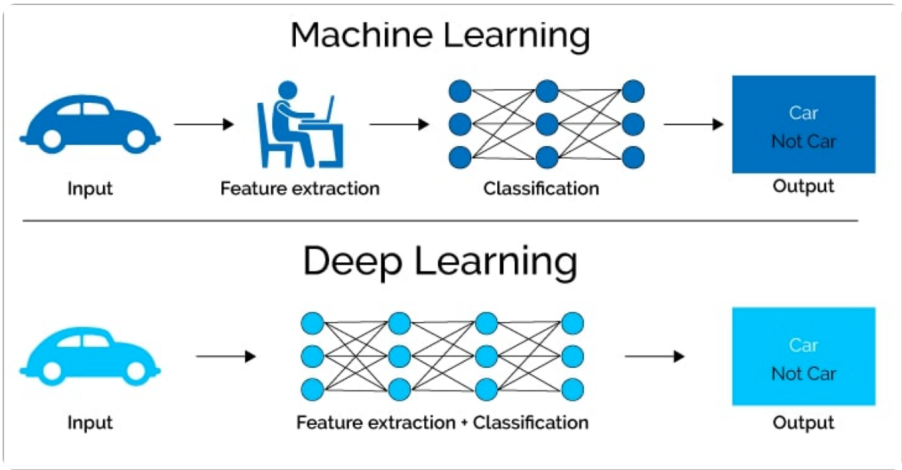

Figure 3.3b: ML vs DL

3.3.2 Machine Learning Models

The field of machine learning classification is vast, and depending on the dataset, a variety of classification algorithms can be utilized. The dataset used in this project is categorical and labeled, based on that, the machine learning algorithms that are used in these projects are described further.

3.3.2.1 Random Forest Algorithm:

Random Forest is a widely used algorithm in machine learning because of the ease of implementation and high accuracy it can achieve. It is a supervised learning model which trains numerous decision trees where a decision tree[15] is a supervised machine learning algorithm that can be used to classify or predict data based on how queries from the past have been answered. [9]

It works by constructing a forest of decision trees and the output of a random forest is obtained by merging the outputs of each tree for that data point. To split the nodes in a decision tree, it chose the best features from all the random features. While building several decision trees, depending on the hyperparameter of the model, extra randomness is provided in choosing the subset of the features for the tree. As a result, the trees become more diversified, resulting in a better model[20].

This model takes the number of decision trees as a parameter, which signifies the number of decision trees you want to build before taking the decision. The greater number of trees makes the algorithm slower but increases the accuracy. For this project, I have trained the random forest model with 50 and 100 numbers of trees with max depth of 7, 10, and 22. And the maximum accuracy is achieved with 50 decision trees and 22 depth. Better accuracy can be achieved with more trees and depth which was not explored in this project because of memory consumption constraint.

3.3.2.2 K Nearest Neighbor (KNN):

KNN is also one of the most popular algorithms in machine learning because of its ease of implementation and performance. This is also known as the "lazy learning algorithm" because in this algorithm, the classification of the new dataset is made based on the behaviors of its K-nearest neighbors.[10]

This algorithm works by plotting the training dataset and then locating the new dataset in that plot. Then the distance of the new dataset from all the training datasets... is calculated, sorted, and stored in an array. The top k elements in the distance sorted array are the nearest neighbors in the new dataset. Then the labels of those neighbors only are checked to determine the label of the new dataset by using the mode function. [12]

Since this algorithm takes the distance between two variables, it can fail when the variables have different scales because the relative distance between the points changes, which can lead to ambiguous results. This kind of issue can be solved by normalizing all the features to the same scale. In this way, each feature is given an equal weight in distance computations when all features are scaled to a single scale.[12]

In this project, the euclidean distance is used as this is the default metric that sklearn provides and the KNN model has been trained with 3, 5, and 10 nearest neighbors. With 3 and 5 neighbors, the classification accuracy is almost the same, but with 10 the accuracy increases as the algorithm has many labels to decide the label for the new dataset. The dataset was first scaled and then trained with different k values. It was discovered that training the dataset with a k value greater than 10 was making the model overfit and the accuracy was optimal at k = 10 for the available dataset.

3.3.2.3 SVC (Support Vector Classifier)

An SVC is a supervised machine learning classification algorithm. It can also be used for regression problems. The goal of the support vector classifier algorithm is to find a hyperplane in an N-dimensional space where N is the number of features that distinguish a hyperplane between data points so that there is maximum possible margin between them. It performs powerful data transformations based on the kernel function you specify, and then attempts to optimize the separation boundaries between your data points based on the labels or classes you define.[13]

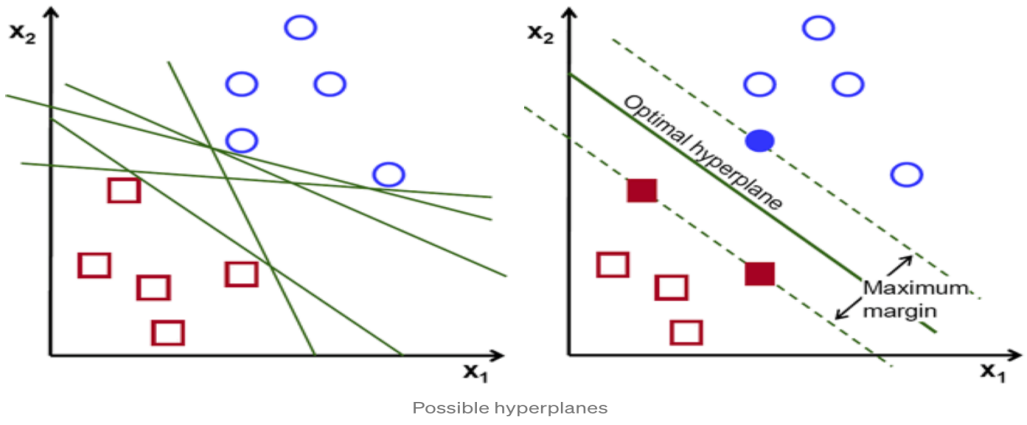

Figure 3.3.3a: SVC: Possible Hyperplanes

There are numerous hyperplanes possible to select to split the two groups of data points. The main goal is to discover a plane with the greatest margin, or the greatest distance between data

points from both classes. Maximizing the margin distance gives some good margin range, making it easier to classify future data points. [13]

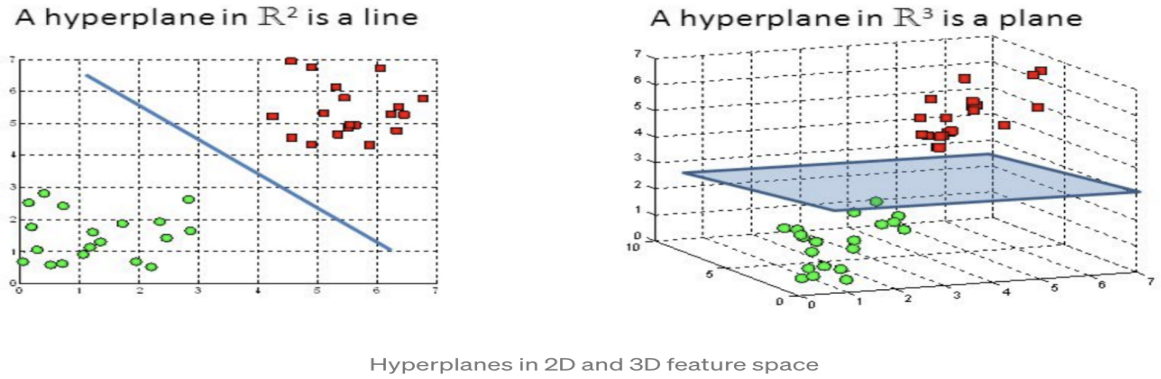

Figure 3.3.3b: SVC: Hyperplanes in 2D and 3D feature space

Now, it is necessary to understand what this hyperplane does. Hyperplanes are decision boundaries that assist in data classification. Different classes can be assigned to data points on either side of the hyperplane. The hyperplane's dimension is also determined by the number of features. If there are only two input features, the hyperplane is just a line. The hyperplane becomes a three-dimensional plane when the number of input features reaches three. When the number of features exceeds three, it becomes harder to describe.[14]

Support vectors are nothing but the data points that are closer to the hyperplane and have an influence on the hyperplane's position and orientation. The margin of the classifier' can be maximized using these support vectors. The hyperplane's position will be altered if the support vectors are deleted. These are the points that are used in constructing our SVM (Support Vector Machine).

SVM does not enable multiclass classification in its most basic form. It allows for binary classification and the splitting of data items into two groups. In order to support multilabel, the same method is applied after breaking down the multi classification problem into several binary classification problems. For instance, three SVM can be used to classify data if there are three classes: red, blue, and green. In this case, the red-blue line will only attempt to maximize the separation between blue and red points. Green points are unrelated to this. Likewise with red-green and blue-green.[14]

There are two techniques to achieve multi-label classification:

1. One-to-One: In this technique, the goal is to map data points to a high-dimensional space in order to achieve mutual linear separation between classes where the multiclass problem is broken down into several binary classification problems. Each pair of classes has a binary classifier.[14]

2. One-to-Rest: This technique is similar to One-to-One except in this one, the breakdown is set to a binary classifier per class.[14]

For this project, the classification problem is multi-labels so a One-to-One technique is used. There are 3 types of labels for server direction and 18 types of labels for shot types. So, a multi-label SVC algorithm is chosen to predict the serve direction and shot type with linear kernel since the used dataset is categorical.

3.3.2.4 Catboost

CatBoost uses a new open source library that successfully handles categorical features. It is a decision-tree-based machine learning approach based on gradient boosted decision trees. Gradient boosted decision trees return a classification model in the form of an ensemble of weak classification models, most commonly decision trees. During training, a series of decision trees are constructed one after the other. Each succeeding tree is constructed with reduced loss than the prior trees. It usually outperforms random forests.[7]

With the help of hyperparameters, CatBoost can halt training earlier than required by the training parameters if overfitting takes place. For instance, it is possible to halt it before a certain number of trees have been built.

In this project, the Catboost model is trained with a parameter of max-depth which determines the maximum depth of decision trees used while training. The maximum accuracy is achieved with a max depth of 10. Better accuracy can be achieved with more trees and depth which is not explored in this project because of memory consumption constraint.

3.3.3 Hyperparameter Optimization

Hyperparameters are parameters in machine learning models that are not optimized during training. These parameters are optimized using the validation in order to maximize model performance.

The hyperparameters used in this project for Random forest, KNN, SVC and Catboost are shown in Table 3.4. Random forest models are tuned by varying the number of decision trees and their depth. Some of the combinations of decision tree and depth are 50 trees and 10 depth, 100 tree and 22 depth and so on until the model overfits.

KNN models can be tuned by varying the number of closest neighbors. For instance, the closest neighbor can be 3, 5, 10 and so on until the model overfits.

The Catboost model can be fine-tuned by varying the number of decision trees, their depth and learning rate of the model. For instance, the learning rate in this project is 0.055, the number of trees is 100, and there are 5, 10, and 15 depth levels.

In the case of SVC model, when the number of features increases, it classifies the data by mapping the dataset into higher dimensional space such that data is linearly separable. However, mapping the data into higher space is computationally expensive. So, the computation cost can be reduced with the help of kernel function. In this project, three kernel functions linear, rbf (Radial Basis Function) and polynomial, are explored. There is also a kernel coefficient called gamma which is used with non-linear kernel functions such as rbf and poly. It tries to perfectly fit the training examples with the hyperplane. The default value for gamma is 'scale'.[21] and the default kernel is rbf.

Algorithms	**Hyperparameters**
Random Forest	**n_estimators** = number of trees in the forest max_depth = max number of levels in each decision tree
KNN	**k**=number of closest neighbors
SVC	**Kernel** = linear, rbf and poly **gamma** = 'scale'
Catboost	**Max_depth** = depth of each tree **n_estimator** = The max trees that can be constructed to solve machine learning problems. **Learning_rate** = It regulates how much the weights of the model are adjusted in relation to the loss function, which assesses how far an estimated value is from its actual value.

Table 3.4 Hyperparameters of the models

3.4 Evaluate classification

We have developed the classification model to provide the decision support for the tennis players in tennis by recommending to them the server direction and shot played by the potential opponent based on their historical data. Now, when the machine learning models are trained, it can be evaluated on the new dataset which has the same features as the training or testing data set. The curated list of features are discussed in section 3.2 data preparation.

CHAPTER 4: RESULTS

In this chapter, we will compare the four models mentioned in the previous chapter based on accuracy of the shot type and serve direction classification, i.e., how well they performed with unseen data. For this purpose, the dataset is divided into two categories: training and testing. Note that the test data has not been utilized for any other reason up to this point and is thus deemed a reliable approximation for new and upcoming matches.

4.1 Model Comparison

The machine learning models are trained separately for classification of **serve direction** and **type of the shots** played by the players in a given scenario of the match such as previous rally count, set number, game number, players hand and opponent etc. In this section, we will discuss the accuracy achieved in each of the classification models on their respective hyperparameters.

Without using these machine learning models , if we try to calculate the probability of shot type or serve direction using the dataset, as can be seen in table 4.1, the shot type classification accuracy is 28.60% and serve direction accuracy is 44.19%.

Variables	**Probability**
Serve Direction classification	44.19
Shot type classification	28.60

Table 4.1 Probability of serve direction and shot type without ML

Using a machine learning approach, we will show in this project that these classification probability can be improved up to 87% in Serve direction classification and up to 72% in shot type classification.

4.1.1 Serve Direction Classification Result

In this subsection, we will discuss the performance of the machine-learning models used in this project on different hyperparameter selection for predicting the serve direction in the tennis match by the players.

After training the machine learning models like random forest, KNN, SVC and CatBoost, it is found that the highest test accuracy is achieved with the KNN algorithm at 87.95% when trained with closest neighbors of 10 as shown in table 4.1.1a and in figure 4.1.1b. The test accuracy achieved by random forest is 84.79% when trained with 50 decision trees with max depth of 20.

After that it is found that increasing the number of decision trees or increasing the max_depth overfits the random forest model. This overfitting can be overcome by either reducing the number of features or with a large dataset, i.e. future work is required.

CatBoost achieved a max test accuracy of 87.93% with max depth of 10 which is very close to KNN. However the training time of KNN is very low in comparison with Catboost. SVC achieved a max test accuracy of 87.33% with the polynomial kernel.

It's interesting to note that the KNN machine learning model excels with categorical datasets like the tennis dataset and also requires less training time. This results shows that text based tennis data with shot by shot information works great to build classification models with test accuracy upto 87.95%.

Algorithms	**Hyperparameters**	**Accuracy (Test)**	**Accuracy (Train)**	**Training time**
Random Forest	n=100, max_depth=22 n = 50, max_depth=25 n = 50, max-depth = 22 **n = 50, max_depth = 20** n = 50, max_depth = 18 n = 50, max-depth = 10 n = 100, max_depth = 10 n = 100, max_depth = 7	84.81 84.71 84.78 **84.79** 84.77 82.37 82.24 75.89	88.55 90.03 88.50 **87.61** 86.79 82.52 82.40 75.73	1 hr 42 min 36 sec 50 min 33 sec 50 min 20 sec **46 min 22 sec** 45 min 26 sec 28 min 47 sec 57 min 37 sec 41 min 59 sec
KNN	**n = 10** n = 5 n = 3	**87.95** 87.57 87.08	**90.50** 91.70 93.03	**21 min 34 sec** 14 min 29 sec 10 min 31 sec
SVC	**kernel = poly** Kernel = rbf Kernel = Linear Kernel = rbf with gamma coefficient	**87.33** 86.57 81.63 85.92	**87.37**	**1 hr 29 min 42 sec**
Catboost	**Max_depth = 10** Max_depth = 5 Max_depth = 3	**87.93** 86.90 86.04	**87.68**	**2 hrs 11 min 21 sec**

Table 4.1.1a Accuracy of serve direction classification

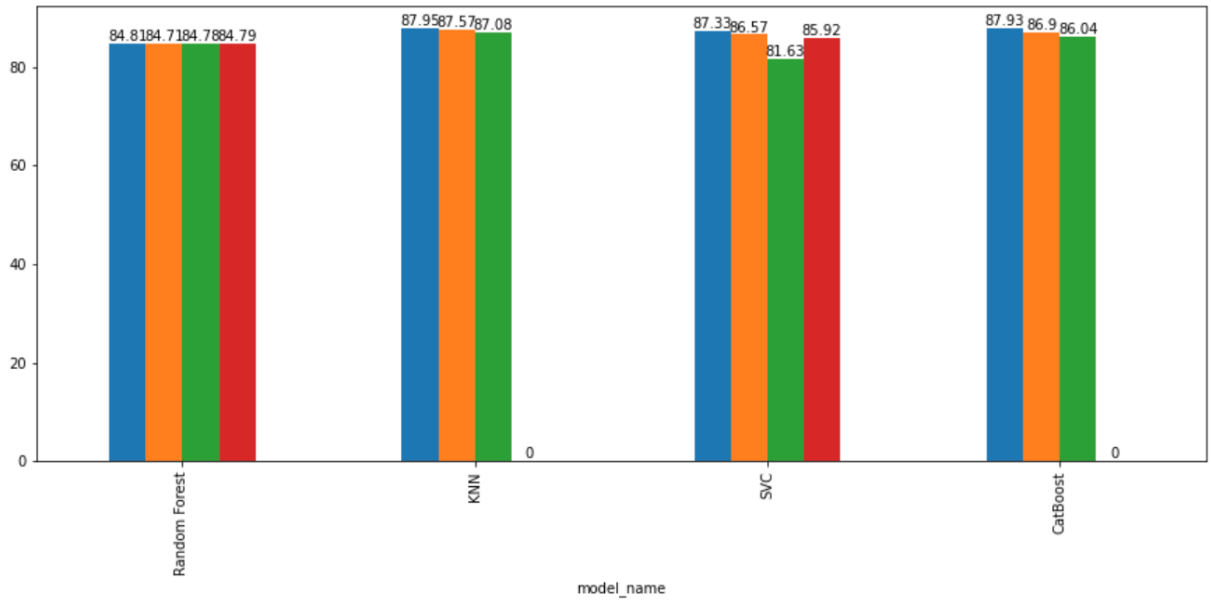

Figure 4.1.1b: Test accuracy of the models used in serve direction classification.

The algorithms like SVC and catboost can be explored more for better classification accuracy. Some of the entries in Table 4.1.1a are missing due to time constraints of this project.

4.1.2 Shot Type Classification Results

In this section, we will discuss the performance of the machine-learning models used in this project on different hyperparameter selection for predicting the type of the shot most likely played by the tennis players in the given context of the match to help tennis players.

After training the machine learning models like random forest, KNN, SVC and CatBoost, it is found that the highest test accuracy is achieved with the Random forest algorithm at 71.48% when trained with max depth of 25 and 50 decision trees as it is shown in table 4.1.2a and in figure 4.1.2b.The test accuracy of random forest start decreasing on further training which shows the overfitting of the model. The test accuracy achieved by KNN is 61.86% when trained with 10 neighbors and SVC achieved 60.99% test accuracy with rbf kernel and kernel coefficient of gamma='scale'.

Catboost algorithm took more than 6 hours time for each run in shot type classification. And highest test accuracy of 57.73% is achieved with max depth of 10. Overall, KNN takes comparatively less time to train in this case of tennis dataset.

Due to time and memory constraint, shot type classification could not be trained with various possibilities of hyperparameters and kernels. The result in Table 4.1.2a shows that data flow

built in this project for the classification of shot type can be improved using a larger dataset. It can be further improved if provided with better computational resources.

Algorithms	Hyperparameters	Accuracy (Test)	Accuracy (Train)	Training time
Random Forest	n = 100, max_depth = 25 n = 100, max_depth = 22 **n = 50, max-depth = 25** n = 50, max-depth = 22 n = 100, max_depth = 7	69.50 70.50 **71.48** 70.85 68.69	78.50 75.95 79.94 76.53 74.56	1 hr 28 min 59 sec 1 hr 24 min 30 sec 44 min 27 sec 42 min 14 sec 37 min 35 sec
KNN	**n = 10** n = 5 n = 3	**61.86** 59.71 59.85	**67.86** 70.59 73.37	**16 min 48 sec** 10 min 54 sec 7 min 43 sec
SVC	Kernel = rbf with gamma coefficient Kernel = Linear Kernel = poly Kernel = rbf	**60.99** 57.81 60.12 59.15	60.27	More than 6 hours 2 hrs 16 min 21 sec
Catboost	**Max_depth = 10** Max_depth = 3	**57.73** 54.41		**More than 6 hours**

Table 4.1.2a Accuracy of shot type classification

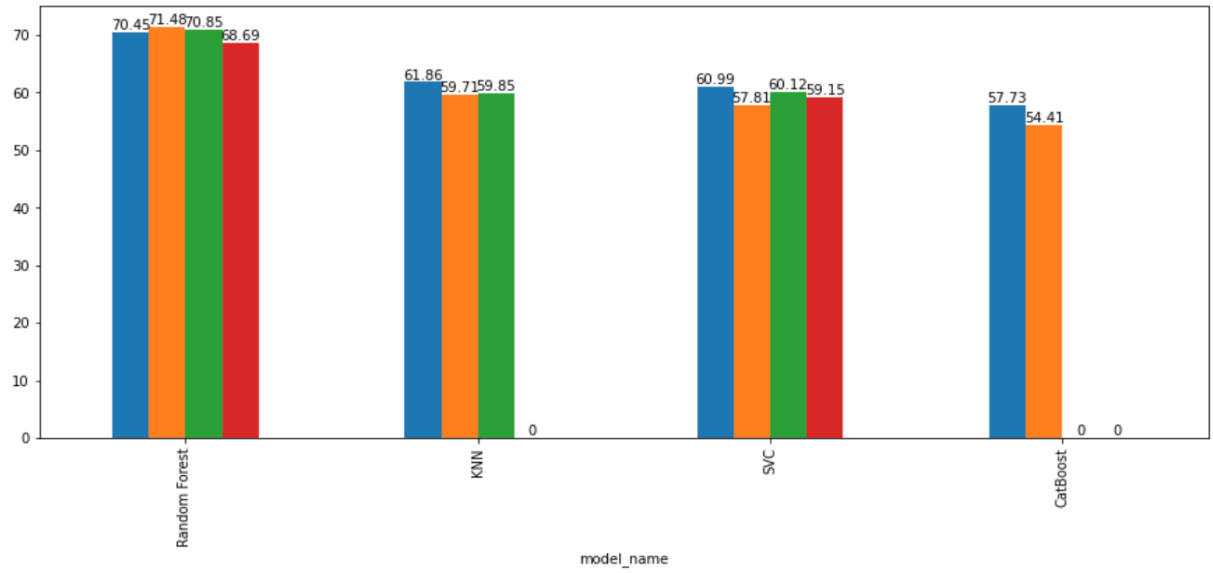

Figure 4.1.2b: Test accuracy of the models used.

There are 18 shot types and 487 players in the dataset. And there are various factors like high variability in the shots played by the players in order to win the match, varying caliber of the players etc, could affect dataset consistency and therefore, the learning rate of the machine learning algorithm. These factors are our hypothesis for getting less accuracy in case of shot type classification than serve direction using the same dataset.

4.2 Results Summary

In this project, the application of machine learning algorithms to this problem has been explored. It is observed that the accuracy that is achieved by taking into account the previous context of the match, players' characteristics like playing hands and other characteristics like serve fault, serve direction, court position etc., is much higher than what can be achieved using the simple probability method to analyze the behavior of the players.

In serve direction classification, KNN outperformed other machine learning algorithms used in this project, both in terms of accuracy and training time. Catboost and SVC performed worse in terms of training time. The accuracy achieved by Catboost and SVC is close to KNN. Random forest models started overfitting after max depth of 25.

In shot type classification, the highest accuracy of 71.48% is achieved using Random forest algorithm with 50 numbers of trees and 25 max depth. The model overfits on further increasing hyperparameters. CatBoost and SVC took more than 6 hours to train and accuracy is low as well. KNN achieved 61.86 but the training time is lowest compared to other models used in this project.

The fact that the models produced accuracy levels of up to 87.93% without overfitting reflects the quality of the dataset used.

The primary task of extracting tennis match features from raw historical data has been done in this project by using principal component analysis and also by finding the correlations between all the different features of the match.

The project's findings, such as data preparation methods and accuracy achieved with different hyperparameters, can help a broader range of sports with similar datasets into the application of ML-based algorithms provided by this project. The proposed models can also be simply modified to anticipate other aspects of the match, like court surface type, and features can also be augmented.

4.3 Limitations

There are several limitations of this exploratory study. Some of these have to do with the dataset (the availability of, the information in the dataset etc.) and on our selection of features from the dataset, whereas some have to do with our choice of Machine learning models. There are also limitations due to the time available to do the study and also availability of computational resources.

All the features used in this project represent the qualities of the tennis players and not the conditions of the match. Weather conditions such as temperature, wind, etc. may favor a specific playing style; for instance, different temperatures have an effect on how the ball spins and bounces. There are additional factors, such as the tennis court's surface since there are three main types: grass, clay, and hard courts. Each type of court surface has an impact on the playing style. Adding match-specific features to the model could help reduce model bias even more. Another feature like the locations of the shots played by the players can be included in the dataset. This kind of information will boost the recommendation system for tennis players to get more insightful information.

Another possibility is that the classification model can take into account the sequential aspect of the match and not just the previous shot played. For training the machine learning model on a sequential dataset, deep learning models like LSTM, RNN, and BLSTM can be used.

The hyperparameters for SVC machine learning model can be explored more for better results. In this project, the kernel like rbf, poly and linear with kernel coefficient gamma=’scale’ are used. Other kernels like sigmoid, precomputed can be explored if given more computational resources.

CHAPTER 5: SUMMARY AND FUTURE WORK

In this project, we developed the decision support system for the tennis players which will help them in improving their match by recommending them the serve direction and the shot type. The data flow created to get the curated list of features with the help of techniques like principal component analysis from raw tennis data are done efficiently in this project. It is found that with the help of machine learning models the historical dataset of tennis players can be trained to achieve the accuracy upto 87.95% to predict the desired recommendations.

However, there is a high scope of this project to scale and future work is required to process location-based data. Adding match-specific features to the model could help reduce model bias even more, for instance, weather conditions (temperature, wind, etc.) may favor a specific playing style.

Another possibility would be to extend this project to utilize the sequence aspect of the match and training the deep learning models like LSTM, RNN, and BLSTM.

Every player has their own style of playing, so the direction of serve of type of shots cannot be purely defined. Keep this in mind, unsupervised learning algorithms can be applied to explore the dataset and learn about shot types and serve direction.

In terms of our classification model, it would be very interesting to further investigate and validate models that required more computational power than it was available for this project. If given more computational resources, the model can be optimized with the help of hyperparameters.

REFERENCES

1. *Fernando, T., Denman, S., Sridharan, S., & Fookes, C. (2019). Memory augmented deep generative models for forecasting the next shot location in tennis. IEEE Transactions on Knowledge and Data Engineering, 32(9), 1785-1797*
2. *Wei, X., Lucey, P., Morgan, S., Carr, P., Reid, M., & Sridharan, S. (2015, August). Predicting serves in tennis using style priors. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 2207-2215).*
3. *Whiteside, D., & Reid, M. (2017). Spatial characteristics of professional tennis serves with implications for serving aces: A machine learning approach. Journal of sports sciences, 35(7), 648-654.*
4. *Martin, C., Bideau, B., Touzard, P., & Kulpa, R. (2019). Identification of serve pacing strategies during five-set tennis matches. International Journal of Sports Science & Coaching, 14(1), 32-42.*
5. *Wei, X., Lucey, P., Morgan, S., & Sridharan, S. (2013, November). Predicting shot locations in tennis using spatiotemporal data. In 2013 International Conference on Digital Image Computing: Techniques and Applications (DICTA) (pp. 1-8). IEEE.*
6. *Wei, X., Lucey, P., Morgan, S., & Sridharan, S. (2016). Forecasting the next shot location in tennis using fine-grained spatiotemporal tracking data. IEEE Transactions on Knowledge and Data Engineering, 28(11), 2988-2997.*
7. *https://catboost.ai/en/docs/*
8. *sklearn.svm.SVC — scikit-learn 1.1.1 documentation*
9. *sklearn.ensemble.RandomForestClassifier — scikit-learn 1.1.1 documentation*
10. *sklearn.neighbors.KNeighborsClassifier — scikit-learn 1.1.1 documentation*
11. *https://scikit-learn.org/stable/modules/tree.html*
12. *https://www.analyticsvidhya.com/blog/2021/08/how-knn-uses-distance-measures/*
13. *https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47#:~:text=Support%20vectors%20are%20data%20points,help%20us%20build%20our%20SVM.*
14. *https://www.baeldung.com/cs/svm-multiclass-classification*
15. *https://scikit-learn.org/stable/modules/tree.html*
16. *https://towardsdatascience.com/artificial-intelligence-vs-machine-learning-vs-deep-learning-2210ba8cc4ac*
17. *https://levity.ai/blog/difference-machine-learning-deep-learning*
18. *https://builtin.com/data-science/step-step-explanation-principal-component-analysis*
19. *Dataset https://github.com/JeffSackmann/tennis_MatchChartingProject*
20. *https://builtin.com/data-science/random-forest-algorithm*
21. *https://numpy.org/*
22. *https://pandas.pydata.org/*
23. *https://matplotlib.org/stable/tutorials/introductory/pyplot.html*
24. *https://seaborn.pydata.org/*

APPENDIX

1. **Set, Gm, Pts:** Tennis uses a hierarchical scoring system, with a match being made up of sets, which are made up of games, which are made up of single points. There are six games in each set, and a set can reach 6-6.
2. **Serve:** There are four types of serve direction.
3. **Rally :** In tennis, a series of back-and-forth shots between players throughout a point is referred to as a rally. A serve and return of serve initiate a rally.
4. **Rally Sequence**: Each shot after serve is followed by continuous return strokes until a point is scored to finish; it is referred to as a rally sequence.
5. **Serve Direction:** It is the direction you want your serve to land in your opponent's court. For example, serving down the middle of the court is called down the T.
6. **Shot Type:** It is the kind of shot you want to hit to return the ball so that it reaches the opponent in that particular style. For instance, the backhand shot, as the name implies, is the opposite of the forehand and is played by the opposite hand by swinging the racket away from the body.
7. **Serve Depth:** It is one of the criteria used to evaluate the quality of the shot and serve in tennis. For instance, within the service boxes, behind the service line but closer to the service line than the baseline, closer to the baseline than the service line. The better the serve depth, the lower the probability of the opponent returning the shot.
8. **Court_position:** The positioning on the court that enables you to react as quickly as possible to your opponent's shots; for instance, baseline smash
9. **Prev_rallyCount:** It is the count of how long the previous rally was
10. **P1 and P2 hand:** Players' strong hand
11. **Unforced Error:** When a player slightly leans to the left in the middle of a point, the opponent notices it in the corner of his or her eye and alters a shot at the last second in an effort to gain the upper hand. It is called an unforced error.
12. **Serve-faults:** If the server(player) swings and misses the ball, the serve is a fault.

Seed Questions:

1. Where did I find this dataset? Why did I choose this dataset?
2. What is overfitting of the data? Why is it important to reduce it?
3. How is Catboost different from Random Forest?
4. How do hyperparameters help in improving the accuracy of the model?