TECHNIUM

Technium Vol. 15, pp.60-74 (2023)
ISSN: 2668-778X
www.techniumscience.com

A framework for applying the Logistic Regression model to obtain predictive analytics for tennis matches

Georgios Friligkos[1], Evi Papaioannou[2], Christos Kaklamanis[2]

[1] University of Patras, Greece and UBS, Zurich, Switzerland
[2] University of Patras and CTI "Diophantus", Greece
friligkos@ceid.upatras.gr, papaioan@ceid.upatras.gr, kakl@ceid.upatras.gr

Abstract. In this work, we apply the Logistic Regression (LR) model for predicting the outcome of tennis matches and providing win probability for participating/competing players. Prediction models are classified as machine learning methods, which generate predictions for future scenaria exploiting existing data collections. With the objective to maximize the accuracy of our predictions, we apply the Logistic Regression model under various parameter configurations seeking for an optimal combination of independent variables (features). We present and discuss promising results obtained via the holdout-validation method and the cross-validation method. Furthermore, as a proof of concept of our approach, we designed and implemented a web platform, where users can obtain real-time predictions for tennis matches. Our LR-based framework combined with Artificial Intelligence (AI) can serve as a paradigm in the field of sports analysis for predicting outcomes and evaluating athletic performance.

Keywords: Logistic Regression, prediction of tennis match outcomes, machine learning, artificial intelligence, python, postgreSQL, Azure web app.

1. Introduction

Humans have always been obsessed with predicting and therefore controlling/harnessing the future [12]. This inherent desire to anticipate forthcoming events has driven us to develop various tools, techniques, and models to gain insights into what lies ahead. From ancient civilizations using sky patterns to forecast weather and harvests to the sophisticated algorithms employed in modern financial markets, the quest to forecast future outcomes overwhelms our history and shapes our contemporary world.

In recent years, predictive analytics [11] has emerged as a promising domain within data science, offering a systematic approach to making predictions based on historical data. Among the various fields where predictive analytics has been applied, sports stands out as an arena where forecasting outcomes has long been a fascination. Tennis, a sport known for its intricate blend of skill, strategy, determination and athleticism, is no exception. Fans, enthusiasts, and professionals alike have sought to gain an edge by predicting the outcomes of tennis matches.

To make a good forecast, one crucial ingredient is a wealth of data about what has already transpired. With the advent of the internet, the World Wide Web, and the popularity of internet-connected devices, data has become more abundant and accessible than ever before. The digital age has introduced an era of unprecedented data collection, offering a treasure of information waiting to be utilized. The question that arises, then, is not merely about the availability of data but rather how to make effective use of this

vast web of information, and in this pursuit, we now have machines as well as algorithms and methods so that we can process these data in an efficient and quick way. These techniques fall under the rapidly growing field of machine learning. Machine learning [3], as we know it today, represents a significant milestone in the quest to harness data for predictive and analytical purposes. Its origins can be traced back to the mid-20th century when the concept of artificial intelligence (AI) was taking shape. One of the pioneering moments in the history of machine learning was the development of the Perceptron [10] by Frank Rosenblatt in 1957. The Perceptron was a simple algorithm designed to mimic the human brain's neurons and was capable of making binary decisions based on input data. However, it wasn't until the 21st century that machine learning truly began to flourish, owing to several key factors, such as, data availability, computing power, algorithmic innovations and open source community.

To leverage machine learning, models are being used. Several have been proposed in the literature, each with its unique strengths and applications. Among these models, Neural Networks (NN) [3], often inspired by the structure of the human brain, consist of interconnected layers of artificial neurons. These models are capable of learning complex patterns and representations from data through a process known as learning. On the other hand, the Random Forest (RF) [2] model is an ensemble learning technique that combines multiple decision trees to generate predictions. Renowned for its robustness and versatility, Random Forest excels in handling noisy or missing data, while also effectively capturing both linear and non-linear relationships. Logistic Regression (LR) [8], the central focus of this paper, is a well-established method of machine learning used for binary classification tasks. It finds applications in various other domains [4], such as marketing, natural language processing, and social sciences, whenever there's a need to predict binary outcomes or probabilities. It has a solid foundation in statistical theory and has been widely employed in sports analytics due to its ability to provide insights into the factors that influence match results. Logistic Regression proves highly advantageous when dealing with problems characterized by multiple features. Its capacity to assess the impact of each characteristic on the probability of victory or defeat makes it an invaluable tool for tennis match prediction. Its simplicity and efficiency render it a preferred choice, and its capability to handle complex scenarios with multiple features further enhances its suitability for in-depth and comprehensive analysis.

In this paper, we utilize and compare both holdout-validation and cross-validation methods [7] to assess the performance of our Logistic Regression model in predicting tennis match outcomes. These methods enable us to rigorously test the model's ability to generalize from the training data to unseen data, ensuring that our predictions are robust and reliable. Holdout-validation involves the division of the data-set into two subsets, typically a training set and a testing set. The model is then trained on the training set and evaluated on the testing set, providing a clear measure of its predictive performance. On the other hand, cross-validation takes this process a step further. It divides the data-set into multiple subsets or folds. The model is trained and tested iteratively on different combinations of these folds, resulting in a more comprehensive evaluation. The combination of holdout-validation and cross-validation offers a comprehensive evaluation strategy for Logistic Regression models, addressing issues related to robustness, generalization, overfitting detection, and performance consistency across various data partitions.

The rest of the paper is structured as follows. In Section 2 we describe the Logistic Regression model in depth. In Section 3, we apply the Logistic Regression model on tennis data and present the results. In Section 4 we illustrate our proof-of-concept web platform. In Section 5, we introduce the technologies and software employed in this paper. We conclude in Section 6 discussing limitations and future plans.

2. The Logistic Regression model

In what follows we present the comprehensive definition of the Logistic Regression model. Additionally, we delve into the theoretical foundations and outline the critical aspects of this model.

2.1. Definition

Logistic Regression [8] is a statistical machine learning model used for binary classification tasks. LR is a supervised learning algorithm that aims to predict the probability of a binary outcome based on one or more input independent variables (features). The core idea behind Logistic Regression is to model the

relationship between the input features and the binary outcome by transforming a linear combination of these features into a probability value using the logistic sigmoid function [1]. This transformation ensures that the predicted probabilities fall in the range of 0 to 1, making it suitable for interpreting the likelihood of the positive class, which typically represents the event or outcome of interest in binary classification tasks. The model learns a set of coefficients (also known as weights) associated with each input feature during the training process. These coefficients are optimized to minimize the difference between the predicted probabilities and the actual binary labels in the training dataset. Logistic Regression is widely employed in various domains, including finance, healthcare, and sports analytics, due to its simplicity, interpretability, and effectiveness in binary classification tasks.

2.2. Logistic Sigmoid function and linear combination

The logistic Sigmoid function [1], often denoted as $\sigma(z)$, is a crucial component of the Logistic Regression model. It transforms the linear combination of input features (z) into a value between 0 and 1, which can be interpreted as the probability of the positive class (class 1) in a binary classification problem. The formula for the logistic Sigmoid function is:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{1}$$

Where e is the base of the natural logarithm (approximately equal to 2.71828). The Sigmoid function takes any real-valued number (z) and maps it to a value between 0 and 1. The linear combination (z) is the weighted sum of the input features (x) and their respective coefficients (w). It represents the log-odds of the probability of the positive class. The formula for z in Logistic Regression is:

$$z = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n \tag{2}$$

Where w_0 represents the intercept term and is commonly referred to as bias in machine learning. Bias adds a fixed parametric value to the model's equation, giving it a prejudiced way to influence predictions independently of the feature values. $w_1, w_2 \ldots w_n$ are the coefficients (also known as weights) associated with each feature. $x_1, x_2 \ldots x_n$ are the input independent variables (features).

2.3. Error, cross-entropy loss and optimization algorithms

In Logistic Regression, the goal is to find the best set of coefficients (w) that minimizes the error between the predicted probabilities ($\sigma(z)$) and the actual binary outcomes (0 or 1). The most common measure of error in Logistic Regression is the cross-entropy loss [1], also known as log loss. The cross-entropy loss function quantifies the dissimilarity between the predicted probabilities and the true labels. The formula for the cross-entropy loss is:

$$L(y, \hat{y}) = -\sum_{i=1}^{N} y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i) \tag{3}$$

Where y is the vector with the actual categories (0 or 1). $\hat{y}$ is the vector with the predicted probabilities for each category. N is the count of observations (e.g., tennis matches) in the training data. The goal during the training process is to find the set of coefficients (w) that minimizes the overall cross-entropy loss across the entire dataset.

This is typically achieved using algorithms like gradient-descent (GD), which is a widely used optimization algorithm in machine learning and deep learning. Gradient-descent is employed to find the minimum (or maximum) of a real-valued function, often referred to as the cost or loss function. In the context of Logistic Regression, GD is used to minimize the cost function by adjusting the model's coefficients iteratively. GD starts with initial parameter values and then calculate the gradient (slope) of the

cost function at the current point. Afterwards it adjusts parameters in the opposite direction of the gradient to move toward the function's minimum and finally repeats the above steps until the cost function reaches a minimum or until convergence criteria are met.

2.4. Performance metrics

In this section, we delve into the study and analysis of performance metrics, including accuracy, recall, precision, F1 score, and AUC-ROC, to evaluate the efficacy of Logistic Regression model.

Starting with accuracy, is a measure of how well the model correctly predicts both positive and negative outcomes. Accuracy is defined as the ratio of correctly predicted instances (both true positives and true negatives) to the total number of instances. Recall, also known as sensitivity or true positive rate, measures the model's ability to identify all positive instances correctly. Recall is defined as the ratio of true positives to the total number of actual positive instances. Precision on the other hand quantifies the model's ability to make positive predictions correctly. Precision is defined as the ratio of true positives to the total number of predicted positive instances. F1 score is the harmonic mean of precision and recall. It provides a balanced measure of a model's performance by considering both false positives and false negatives. F1 score is particularly useful when there is an imbalance between positive and negative classes. Finally, AUC-ROC stands for area under the receiver operating characteristic curve and measures the model's ability to distinguish between positive and negative classes across various probability thresholds. The ROC curve plots the true positive rate against the false positive rate. The AUC-ROC score quantifies the area under this curve, with a higher score indicating better discrimination between classes.

These metrics help evaluate how well a predictive model performs in different aspects, such as correctness, sensitivity, precision, balance, and class separation.

2.5. Evaluation techniques

In what follows, we present a concise overview of two prominent model evaluation techniques: holdout-validation and cross-validation. These methods are essential for assessing the performance and generalization ability of Logistic Regression model.

Holdout-validation [7], also known as train-test split, is a model evaluation technique where the dataset is divided into two parts, a training set and a testing set. Typically, a larger portion of the data [1] (e.g., 70-80%) is allocated for training, and the remaining portion (e.g., 20-30%) is reserved for testing. The model is trained on the training set and then evaluated on the separate testing set, allowing to measure its performance on unseen data. Holdout-validation offers advantages in terms of simplicity, making it easy to implement and understand. It is computationally efficient, making it suitable for handling large datasets, and it serves as a useful method for conducting initial model assessments. However, the holdout-validation model has its limitations, including potential variance in performance evaluation due to random splits, the exclusion of a portion of data from training, which can reduce model effectiveness, and limitations in detecting over-fitting, particularly when the testing set is small.

Cross-validation [7] on the other hand is a more robust evaluation technique where the dataset is divided into multiple subsets or folds. The model is trained and tested iteratively on different combinations of these folds. A common type of cross-validation includes k-fold cross-validation, in which the dataset is divided into k subsets, each containing approximately an equal number of samples. These subsets are referred to as folds. The model is trained and tested k times. In each iteration, one of the k folds is used as the testing set, while the remaining k-1 folds collectively form the training set. After each iteration, the model's performance metrics (e.g., accuracy, precision, recall) are recorded. This step results in k different performance measurements, one for each fold. The k performance measurements are typically averaged to obtain a single evaluation metric for the model. This aggregated metric provides a more reliable estimate of the model's performance than a single train-test split. Cross-validation presents several advantages. It provides more consistent results by reducing the impact of random data partitioning. Cross-validation offers a truer estimate of real-world performance and is particularly effective when dealing with datasets containing limited samples. However, cross-validation has its drawbacks. It demands

more computational resources and time since the model needs to be trained multiple times. Additionally, interpreting results can be more challenging due to the multiple evaluations conducted.

3. Applying the Logistic Regression model to tennis data

In this section, we delve into the practical application of the Logistic Regression model to tennis data. Our objective is to both train and rigorously evaluate the model's performance using two distinct evaluation techniques: holdout-validation and k-fold cross-validation. Through these methods, we aim to assess the model's predictive capabilities and its ability to generalize effectively to real-world tennis match outcomes.

3.1. Data collection, feature selection and inter- feature correlations

For data collection, we utilize the python programming language to parse the official ATP Tour website, `https://www.atptour.com/en`, which encompasses men's professional tennis. The matches and statistical data collected for the players are exclusively sourced from tournaments falling under the following categories: ATP 250, ATP 500, ATP 1000, and Grand Slam events. Data is gathered on a daily basis and stored in an Azure PostgreSQL server. A diverse set of independent variables (features) has been curated for each player, ensuring a comprehensive representation of their performance and form.

In our dataset, we store 36 distinct features for each player, capturing various aspects of their performance. To refine the model and enhance its predictive capabilities, we conducted a rigorous feature selection analysis. This process involved assessing the importance of each feature and evaluating inter-feature correlations. Through this comprehensive analysis, we identified the most influential features that significantly contribute to predicting tennis match outcomes. As a result, we opted to utilize a refined set of 17 carefully chosen features per player, as depicted in Table 1 below.

Table 1. Features used in LR model

Features	Description
PLAYER1_RANK_COMPARED_TO_OPPONET	Index for the ranking position of player 1 in relation to the opponent.
PLAYER1_MATCHRECORD_OVERALL_W_L_CAREER_INDEX	Index for the overall win-loss record of player 1 in his career.
PLAYER1_MATCHRECORD_SURFACE_W_L_CAREER_INDEX	Index for the win-loss record of player 1 on a specific surface during his career.
PLAYER1_DOUBLEFAULTS_ACES_RATIO_ON_SURFACE*	The ratio between double faults and aces on a specific surface for player 1.
PLAYER1_SERVE_1ST_ON_SURFACE	The percentage of first serves by player 1 on a specific surface.
PLAYER1_SERVE_1ST_POINTS_WON_ON_SURFACE	The percentage of points won by player 1 with his first serve on a specific surface.
PLAYER1_SERVE_2ND_POINTS_WON_ON_SURFACE	The percentage of points won by player 1 with his second serve on a specific surface.
PLAYER1_BREAK_POINTS_SAVED_ON_SURFACE	The percentage of break points saved by player 1 on a specific surface.
PLAYER1_BREAK_POINTS_CONVERTED_ON_SURFACE	The percentage of break points converted into breaks by player 1 in the game on a specific surface.

Continued on next page

Table 1 continued from previous page

Features	Description
PLAYER1_SERVICE_GAMES_WON_ON_SURFACE	The percentage of games won by player 1 on his serve on a specific surface.
PLAYER1_TOTAL_SERVICE_POINTS_WON_ON_SURFACE	The percentage of points won by player 1 on his serves on a specific surface.
PLAYER1_SERVE_1ST_RETURN_POINTS_WON_ON_SURFACE	The percentage of points won by player 1 with the return of the opponent's first serve on a specific surface.
PLAYER1_SERVE_2ND_RETURN_POINTS_WON_ON_SURFACE	The percentage of points won by player 1 with the return of the opponent's second serve on a specific surface.
PLAYER1_RETURN_GAMES_WON_ON_SURFACE	The percentage of games won by player 1 on the return on a specific surface.
PLAYER1_RETURN_POINTS_WON_ON_SURFACE	The percentage of points won by player 1 on the return on a specific surface.
PLAYER1_TOTAL_POINTS_WON_ON_SURFACE	The overall percentage of points won by player 1 on a specific surface.
PLAYER1_LOST_WON_SETS_RATIO_YTD*	The ratio between the lost and won sets of player 1 for the current year.

All values of the aforementioned features range from 0 to 1 and are expressed as decimal numbers with 2 decimal places. For example, 0.55. The exception is for features marked with * that may have values greater than 1. This occurs in cases where a player exhibits highly negative statistics, but even in such cases, the value remains within the range of single-digit. Due to this range of values, additional normalization is not required as there is no need to balance the scales of variables or address issues related to data asymmetry. In the event of missing values for any variable, a value of -1 is assigned, which is not possible to otherwise appear within this specific set of features.

3.2. Optimizing feature composition using binomial-coefficient analysis

All possible combinations of features were examined and evaluated to find the optimal composition of features that maximizes prediction accuracy. The binomial coefficient [9] formula was utilized to calculate the potential combinations.

$$C(n,k) = \frac{n!}{(n-k)!\,k!} \tag{4}$$

where n the available features and k the number of features to be selected each time. For each individual combination of features, the model was trained, and accuracy was calculated to enable a comparative evaluation and selection of the optimal composition of independent variables.

3.3. Hyperparameter tuning

Hyperparameter tuning [5] is a critical step in the development of machine learning models, aimed at optimizing their performance. In this work, we employed grid search cross-validation to systematically explore the hyperparameter space for Logistic Regression. The hyperparameters under consideration were C, penalty, and solver.

The C parameter controls the regularization strength in Logistic Regression. It determines the trade-off between fitting the training data perfectly and preventing overfitting. The penalty parameter specifies the type of regularization to be applied to the Logistic Regression model. We evaluated two options: L1 (Lasso) and L2 (Ridge) regularization. L1 encourages sparsity in the model by effectively driving some of the coefficients associated with less influential features to zero, simplifying the model and emphasizing the importance of the most relevant variables in the prediction process, while L2 regularization penalizes large coefficients to prevent overfitting. The solver parameter dictates the optimization algorithm used to train the Logistic Regression model. We explored four solver options: 'liblinear', 'lbfgs', 'newton-cg', and 'sag'.

After a grid search and cross-validation process, we identified the optimal hyperparameter configuration that yielded the best model performance on our dataset. The selected configuration is as follows. Solver: 'liblinear' , Penalty: 'L2' and C: 1.0.

3.4. Presentation and comparative assessment of results

The results presented are based on a sample of 630 tennis matches, all of which were conducted, and their statistics collected, within the timeframe from June 25, 2023, to September 10, 2023. Based on feature analysis, the feature with the highest importance was PLAYER1_RANK_COMPARED_TO_OPPONENT, which is consistently utilized in model training. For the remaining 16 features included in Table 1, we systematically explore all possible combinations using the binomial-coefficient formula. We present the results for these combinations, evaluated under both assessment methods holdout-validation and k-fold cross-validation. In the training and evaluation of the LR model, we leverage the python language and advanced libraries, such as Scikit-learn.

3.4.1. Holdout-validation results. We present the results obtained through holdout-validation. In Table 2 below, column 1 displays the count of independent variables (features) involved in the model training, while column 2 indicates the total number of feature combinations. Column 3 reports the execution time in minutes. In Column 4, the average value of the performance metric accuracy is presented. Column 5 showcases the highest achieved accuracy.

Table 2. Holdout-validation results

#Features(k)	#Combinations $C(16, k)$	Execution-time (min)	Average-accuracy	Maximum-accuracy
k=2	120	0.06	0.6619	0.6984
k=3	560	0.26	0.6657	0.7089
k=4	1820	0.83	0.6685	0.7142
k=5	4368	2.03	0.6708	0.7142
k=6	8008	3.95	0.6730	**0.7195**
k=7	11440	5.67	0.6751	**0.7195**
k=8	12870	6.42	0.6769	0.7142
k=9	11440	5.80	0.6786	0.7142
k=10	8008	4.21	0.6802	0.7089
k=11	4368	2.36	0.6821	0.7089
k=12	1820	1.01	0.6841	0.7089
k=13	560	0.32	0.6866	0.7037
k=14	120	0.07	0.6897	0.7037
k=15	16	0.01	0.6931	0.6984
k=16	1	0.00	0.6984	0.6984

In Figures 1 and 2, we provide visual representations illustrating the density of accuracy and kernel density within our dataset for one specific value k=7, corresponding to the instance with the highest accuracy.

Technium Vol. 15, pp.60-74 (2023)
ISSN: 2668-778X
www.techniumscience.com

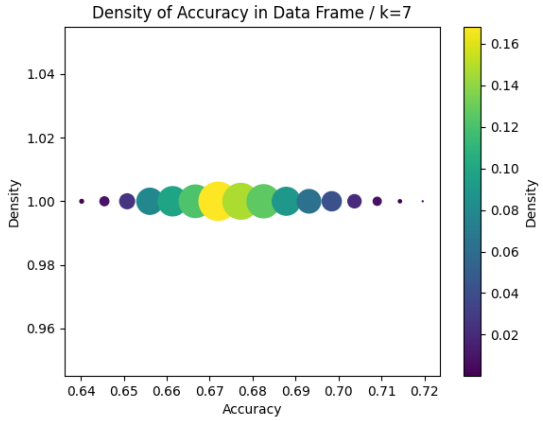

Figure 1: Density of accuracy for k=7

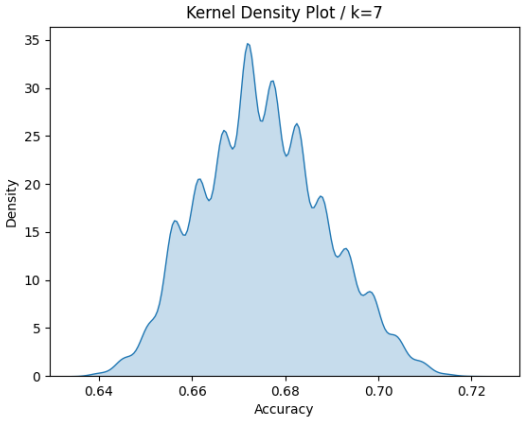

Figure 2: Kernel density for k=7

Now we illustrate (Figures 3 and 4, respectively) the fluctuation in average accuracy and in maximum accuracy across the values of k to visualize the model's performance sensitivity to different feature selections.

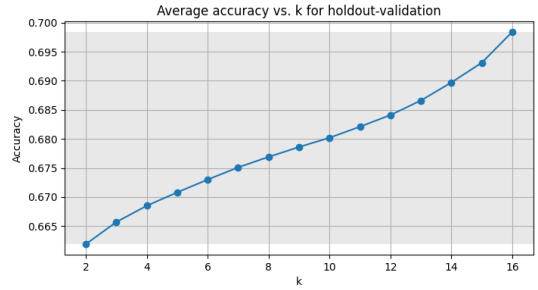

Figure 3: Fluctuation of average accuracy

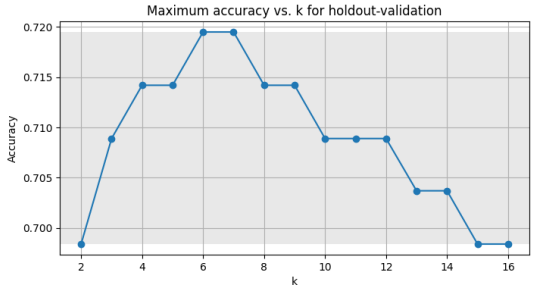

Figure 4: Fluctuation of maximum accuracy

3.4.2. 5-fold cross-validation results. In Table 3, we now present the results obtained from 5-fold cross-validation, a commonly chosen number of folds [6] in cross-validation techniques.

Table 3. 5-fold cross-validation results

#Features(k)	#Combinations $C(16, k)$	Execution-time (min)	Average-accuracy	Maximum-accuracy
k=2	120	0.11	0.6543	0.6667
k=3	560	0.51	0.6558	0.6714
k=4	1820	1.74	0.6570	0.6730
k=5	4368	4.07	0.6582	0.6793
k=6	8008	7.56	0.6595	0.6809
k=7	11440	11.06	0.6608	0.6809
k=8	12870	12.84	0.6619	**0.6825**
k=9	11440	11.75	0.6630	**0.6825**
k=10	8008	8.66	0.6639	0.6809
k=11	4368	4.89	0.6646	0.6793
k=12	1820	2.09	0.6651	0.6778
k=13	560	0.66	0.6654	0.6761
k=14	120	0.14	0.6657	0.6730
k=15	16	0.02	0.6658	0.6730
k=16	1	0.00	0.6650	0.6650

Technium Vol. 15, pp.60-74 (2023)
ISSN: 2668-778X
www.techniumscience.com

In Figures 5 and 6, We provide visual representations illustrating the density of accuracy and kernel density within our dataset for one specific value k=9, corresponding to the instance with the highest accuracy.

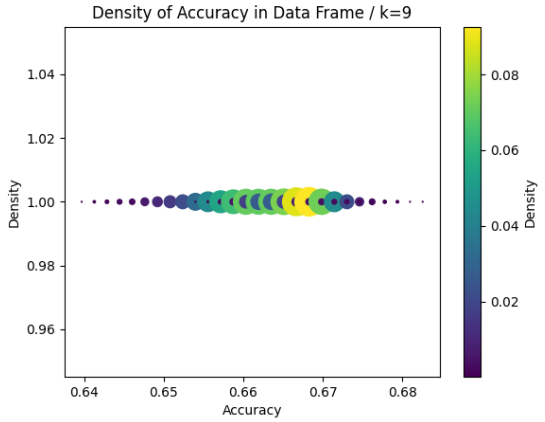

Figure 5: Density of accuracy for k=9

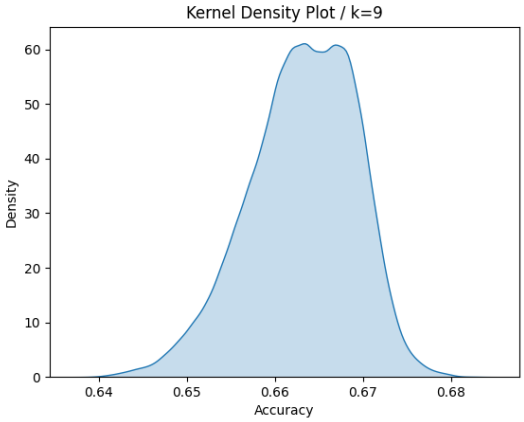

Figure 6: Kernel density for k=9

Now we illustrate (Figures 7 and 8, respectively) the fluctuation in average accuracy and in maximum accuracy across the values of k (number of features) to visualize the model's performance sensitivity to different feature selections.

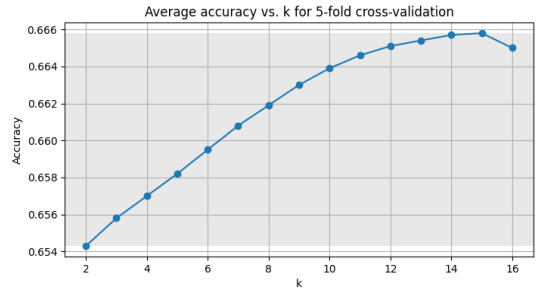

Figure 7: Fluctuation of average accuracy

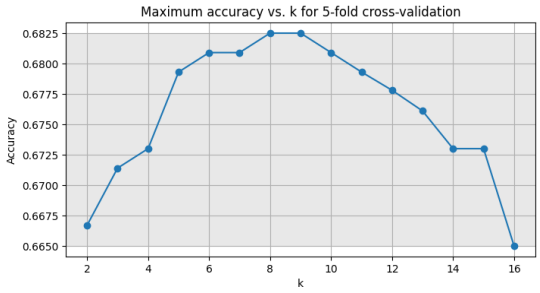

Figure 8: Fluctuation of maximum accuracy

3.4.3. Comparative analysis of model performance. After running the Logistic Regression model using both the holdout-validation and 5-fold cross-validation methods on tennis data, we observed that the accuracy was higher with the holdout method compared to cross-validation. However, it's important to note that the difference in accuracy between the two methods may not always be indicative of the model's true performance. While cross-validation may require more execution time and resources (based on the results), it offers a robust evaluation of the model's capabilities, providing a comprehensive understanding of its performance across different scenarios and its ability in helping prevent over-fitting or under-fitting.

4. Our proof-of-concept web platform

In this section, we showcase our proof-of-concept web platform, Tennis Crystal-Ball, which is accessible at the following URL: `https://tennis-crystal-ball.azurewebsites.net/`. This proof-of-concept web platform was developed in order to enable users to practically exploit the Logistic Regression model for obtaining predictions and win probabilities for upcoming tennis matches. We also provide a high-level overview of all the functionalities offered by the web platform.

Starting with the basic functionalities, users can click on the 'Upcoming Matches' option on the left menu to view the tennis matches scheduled for the day. The matches are displayed in the platform's results panel.

Technium Vol. 15, pp.60-74 (2023)
ISSN: 2668-778X
www.techniumscience.com

After exploring the upcoming matches, users can also review the head-to-head (H2H) history between two players for an upcoming match. Let's assume the user is interested in examining the upcoming match between players Carlos Alcaraz and Hubert Hurkacz. In this case, the user should select both players from the drop-down menu options and then click the "View H2H" button. All the matches played between these two players are displayed in the results panel, as depicted in Figure 9

Year	Event	Surface	RND	Winner	Loser	Result
2022	ATP Masters 1000 Miami	Outdoor Hard	SF	Carlos Alcaraz	Hubert Hurkacz	765 762

Figure 9: H2H results (only 1 match in this scenario)

At this point users can execute the Logistic Regression model by clicking the "Execute Logistic Regression" button. The composition of the features used for training the model is optimized to maximize the accuracy of the model on the specific dataset. In panel 1 of Figure 10, we present the results of the metrics: accuracy, recall, precision, f1, and auc_roc. In panel 2, we provide a summary of the Logistic Regression model.

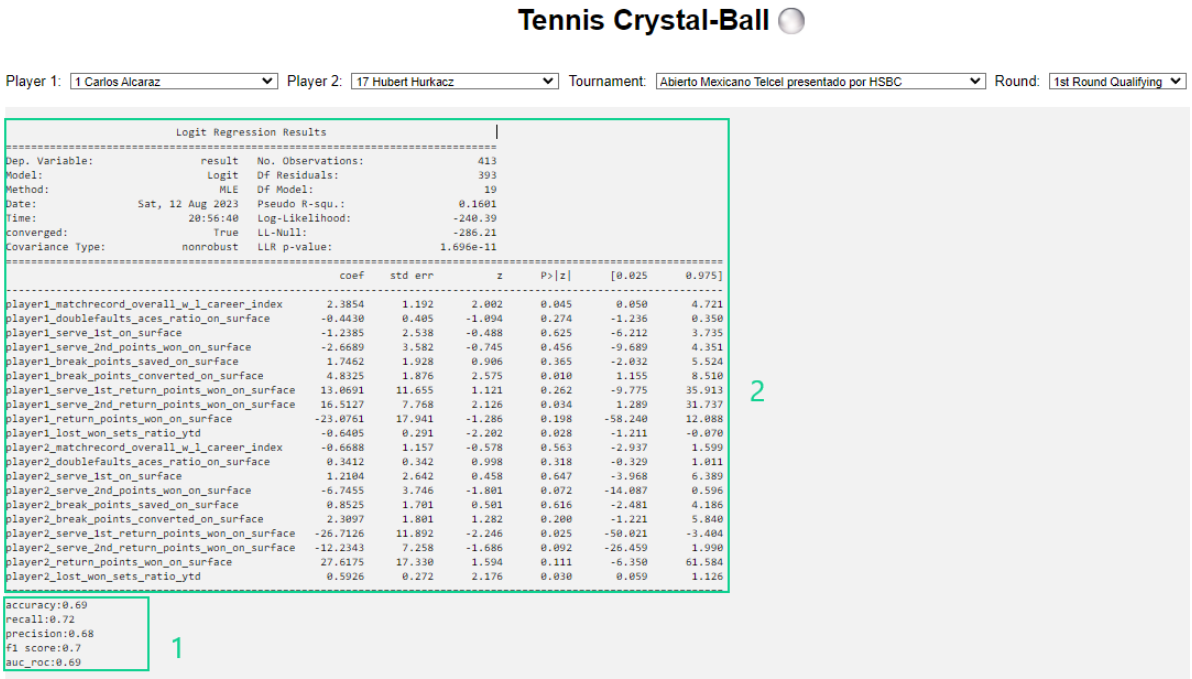

Figure 10: Logistic Regression results

For accessing win probability information for a specific match of interest, users need to complete the four drop-down menus with player names, tournament name, and match round. Afterward, they can click the "Calculate Win Probability" button. In this scenario, the Logistic Regression model predicts a 68% win probability for Player 1 (Carlos Alcaraz) and a 32% win probability for Player 2 (Hubert Hurkacz). These results are displayed in Panel 1, as depicted in Figure 11.

Technium Vol. 15, pp.60-74 (2023)
ISSN: 2668-778X
www.techniumscience.com

Figure 11: Win probability results for an upcoming match

Users can refresh the results panel by clicking the ”Refresh” button to request predictions for other matches. At this point, users have completed the basic functionalities of the platform. Moving on to the advanced functionalities, the web platform also offers users the additional functionality of selecting independently which features they wish to include in the Logistic Regression model’s training, as depicted in Figure 12.

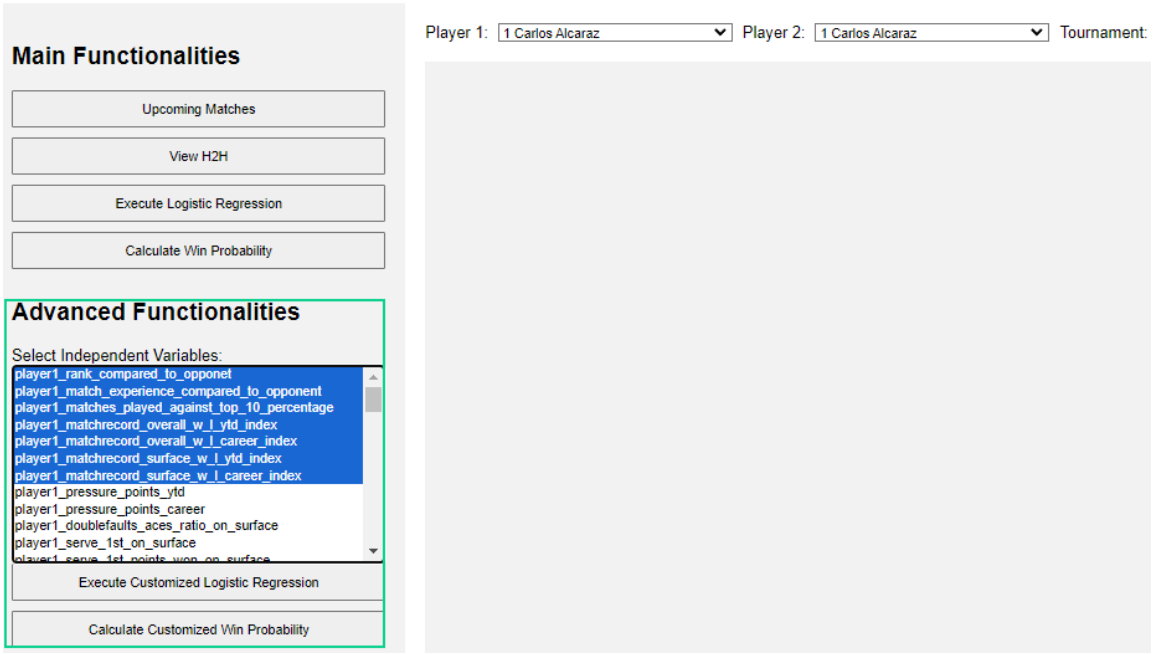

Figure 12: Customized selection of independent variables highlighted in blue

Subsequently, based on this personalized (customized) model, users can request win probability predictions for specific matches of interest. The user-selected independent variables are highlighted in blue. Users can now click the ”Execute Customized Logistic Regression” button, and the model is trained using the chosen independent variables. The results are displayed in the platform’s results panel, as depicted in Figure 13 below. In Panel 1, the number of observations included in the training dataset is displayed. In this specific case, the dataset consists of 413 matches. Panel 2 presents the independent variables selected by the user for training. Panel 3 displays performance metrics, including accuracy, recall, precision, F1 score, and auc_roc.

Technium Vol. 15, pp.60-74 (2023)
ISSN: 2668-778X
www.techniumscience.com

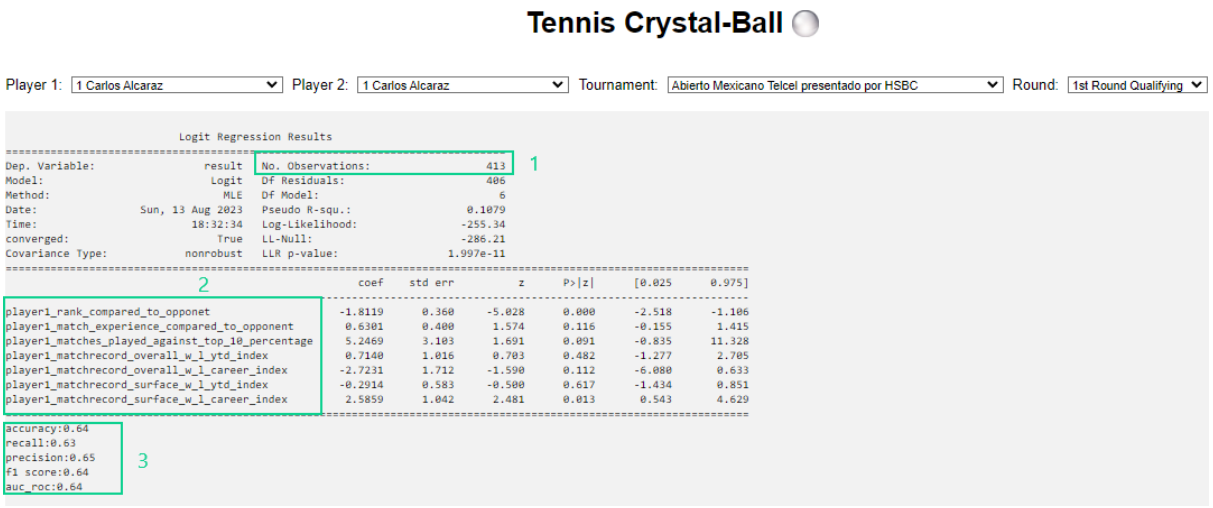

Figure 13: Customized Logistic Regression results based on user's selection

Users can utilize the personalized trained model to predict upcoming tennis matches. In this scenario, the user chooses to obtain predictions for the same match as described in use case 1. In panel 1 of Figure 14, the new probabilities generated by the user's personalized trained model are presented, allowing the user to compare the results.

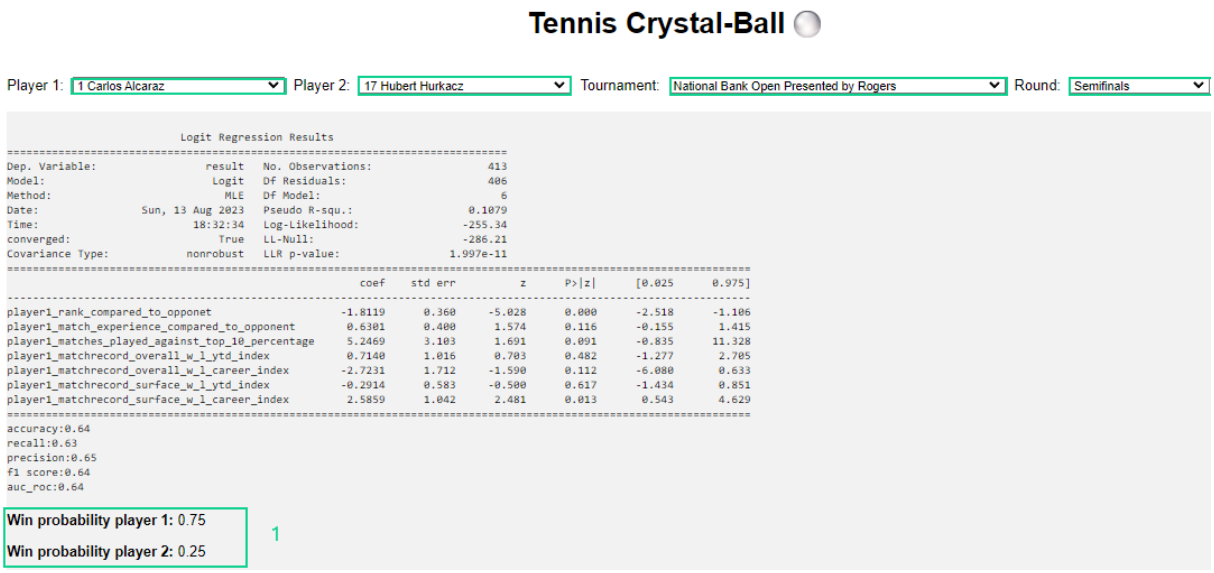

Figure 14: Customized win probability results

Users can press the "Refresh" button to update the results panel and request predictions for other matches based on their customized model. Finally, the user can click the "Old Matches" button to examine all the past matches that are part of the dataset. Furthermore, users have the option to select one of the past matches and recalculate the win probability based on the data available at the time of the match. Users always has the option to press the "Refresh" button to clear the results panel and drop-down selections. At this point, all the functionalities provided by the platform, are completed.

5. Underlying technologies and software

In this section, we provide an overview of the technologies and software that support the entirety of this work. Comprising four distinct yet interconnected parts, this infrastructure plays a pivotal role in the successful execution of our project. Each part represents a crucial phase in our journey, from the initial data sourcing to the development of our web platform for user interaction. By unraveling the technological intricacies of each phase, we aim to provide a comprehensive understanding of the tools and technologies utilized throughout this work.

In the initial phase of our work, we employed python [20] to parse valuable tennis data from the official ATP Tour website `https://www.atptour.com/en` in order to build our database. Our data collection process included the daily execution of python scripts that utilized advanced libraries, such as, beautiful-soup, pandas, selenium, JSON and others. This approach ensured the continuous maintenance of our dataset with up-to-date and accurate, as well as targeted, information.

Moving on to the database part, we utilized the open-source relational database PostgreSQL [19], version 14.7, known for its robust data management and storage capabilities, in order to build our own database. Additionally, we employed Azure PostgreSQL server, creating a dedicated read-only user, ”read_only_user,” as depicted in Table 4 to grant direct read access to the database for interested parties.

Table 4. Database Connection Parameters

Spec-type	Value
Host:	tennis-world.postgres.database.azure.com
Port:	5432
User:	read_only_user
Data Base:	tennis
Password:	12345678

To facilitate access, the user’s IP address must be added to the Azure server’s firewall rules, a configuration that can be managed through the pgAdmin tool upon request.

Transitioning to the machine learning part, for the implementation of the Logistic Regression model, we employed the python [20] programming language, leveraging essential libraries, including Scikit-learn, Statsmodels, Matplotlib, and NumPy. These libraries facilitated the development and evaluation of the model’s predictive capabilities.

we implemented our proof-of-concept web platform for practical application of the Logistic Regression model. Specifically, we utilized the Azure Web App [14] service under the free tier, configured with a runtime stack of node.js version 18. For the deployment process, we opted for the local git option. To support the execution of python files within the application, we configured the extensions with python version 3.6.4.4. These technical details, as depicted in Table 5, ensured the seamless operation of our web platform.

Table 5. Technical specifications

Spec-type	Value
name:	Tennis-Crystal-Ball
domain:	tennis-crystal-ball.azurewebsites.net
git clone url:	https://george_4030@tennis-crystal-ball.scm.azurewebsites.net/Tennis-Crystal-Ball.git
deployment provider:	Local-Git
virtual IP address:	20.105.224.29

Furthermore, our web platform was built using technologies such as node.js [18], express.js [13], JavaScript [17], HTML [16], and CSS [15]. Hosting of the developed web platform is provided by the Azure web app service, ensuring its accessibility to users. The architectural diagram of our platform is depicted in Figure 15 below.

Technium Vol. 15, pp.60-74 (2023)
ISSN: 2668-778X
www.techniumscience.com

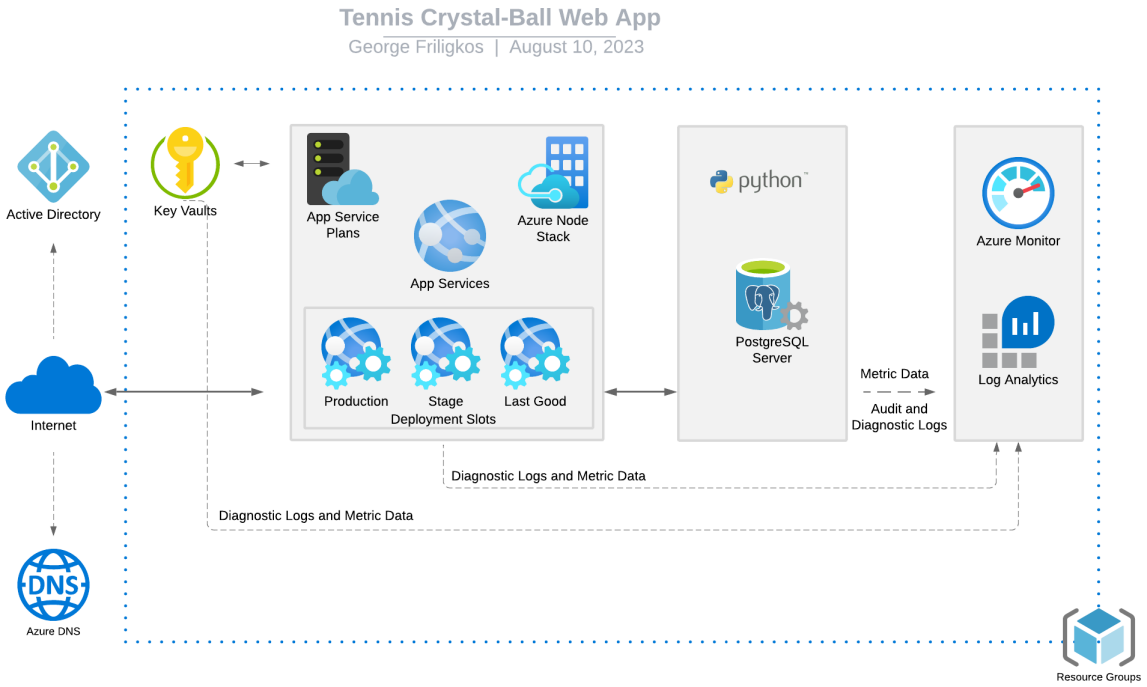

Figure 15: Architectural diagram

6. Concluding remarks, limitations and future plans

In this work, we designed and developed a reliable and robust tennis match prediction engine, despite the inherent limitations and challenges of the prediction process itself. Our approach and models used have demonstrated accuracy and reliability. However, it's important to acknowledge that predictions may not always be infallible (as exceptions do exist). For instance, unforeseen upsets in tennis matches, where a lower-ranked player defeats a higher-ranked one, can be challenging to predict accurately. Therefore, it is crucial for users to exercise caution when relying solely on predictions for decision-making, as blindly following predictions can lead to overconfidence and financial risks in the context of betting, as well as overlooking other important factors in coaching and tennis-related decisions.

In this work, a significant amount of features were developed and utilized for players, indicating their competitive state and performance level, both overall in their career and recently (their current form) before the upcoming match. However, what the model cannot capture and include for training is the human factor. One such human-related aspect is the physical condition of athletes, i.e., whether there are underlying injuries that may not prevent players from participating in a match but may significantly affect their performance within it. Another human-related aspect is the psychological condition of the athletes, including factors from their personal life, family, and social environment that can negatively influence the players' psychology and consequently their performance and thus the outcome of the match. To address such limitations of the model, artificial intelligence (AI) can perhaps be exploited. Specifically, the use of AI for predicting the psychological state of players in sports matches is an exciting prospect that can add valuable elements to the prediction model and constitute a significant improvement.

To advance in this direction, several key steps are taken into account. Firstly, it's essential to gather data from player's social media platforms, with caution, which may encompass a wide range of content such as tweets, Facebook posts, comments on photos, and various online expressions. It's important to cross-check the data for accuracy and reliability. Natural Language Processing (NLP) is then applied to analyze this text data from social media, helping to decipher player sentiments, reactions, and potentially noteworthy expressions. The next critical phase involves developing a model capable of computing a guess of the emotional state of players based on their social media text. This entails identifying both positive and negative emotions and evaluating the intensity of these emotions. Finally, the results of the emotional state analysis are integrated into the tennis match prediction model. By doing so, the model becomes more adept at considering the psychological dispositions of the players and their potential

influence on their performance

This significant addition demonstrates the potential of machine learning (ML) and artificial intelligence (AI) to be successfully applied in the world of sports. With further research and improvements, the accuracy and reliability of the Logistic Regression model can be further enhanced, leading to valuable predictions for the world of tennis.

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. ISBN: 978-0387-31073-2.

[2] Leo Breiman. "Random Forests." In: *Machine Learning* 45 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016, pp. 96–161, 326–366. ISBN: 978-0262035613.

[4] David W. Hosmer Jr., Stanley Lemeshow, and Rodney X. Sturdivant. *Applied Logistic Regression*. Wiley, 2013. ISBN: 978-1-118-54835-6.

[5] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren (Editors). *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2019. ISBN: 978-3-030-05317-8.

[6] Bruce G. Marcot and Anca M. Hanea. "What is an optimal value of k in k-fold cross-validation in discrete Bayesian network analysis?" In: *Computational Statistics* 36.3 (2021), pp. 2009–2031. DOI: 10.1007/s00180-020-00999-9.

[7] Andreas C. Müller and Sarah Guido. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media, 2016, pp. 251–302. ISBN: 978-1449369415.

[8] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012, pp. 1–32, 245–280. ISBN: 978-0-262-01802-9.

[9] Kenneth H. Rosen. *Discrete Mathematics and Its Applications*. 7th. New York: McGraw-Hill, 2012. ISBN: 978-0073383095.

[10] Frank Rosenblatt. "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain." In: *Psychological Review* 65.6 (1958), pp. 386–408. DOI: 10.1037/h0042519.

[11] Eric Siegel. *Predictive Analytics: The Power to Predict Who Will Click, Buy, Lie, or Die*. Wiley, 2013. ISBN: 978-1118356852.

[12] Nate Silver. *The Signal and the Noise: Why So Many Predictions Fail – but Some Don't*. Penguin Press, 2012. ISBN: 978-0-14-312508-2.

[13] Express.js. *Express.js*. 2023. URL: https://expressjs.com/.

[14] Microsoft Azure. *Azure Web App*. 2023. URL: https://azure.microsoft.com/en-us/services/app-service/web/.

[15] Mozilla Developer Network (MDN). *CSS (Cascading Style Sheets)*. 2023. URL: https://developer.mozilla.org/en-US/docs/Web/CSS.

[16] Mozilla Developer Network (MDN). *HTML (Hypertext Markup Language)*. 2023. URL: https://developer.mozilla.org/en-US/docs/Web/HTML.

[17] Mozilla Developer Network (MDN). *JavaScript*. 2023. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[18] Node.js Foundation. *Node.js*. 2023. URL: https://nodejs.org/.

[19] PostgreSQL Global Development Group. *PostgreSQL: The world's most advanced open-source relational database*. 2023. URL: https://www.postgresql.org/.

[20] Python Software Foundation. *Python Language Reference, version 3.10*. 2023. URL: https://docs.python.org/3.10/reference/index.html.