2020 | OCTOBER

MACHINE LEARNING REPORT

PREDICTING CS: GO ROUND WINNERS

Authors: Q. Loh, A. Sehnaoui, A. Zayed, F. Mansilla, R. Algahmdi, W. Stahlke IV, T. Bachmann

CONTENTS

EXECUTIVE SUMMARY

ABSTRACT AND OVERVIEW

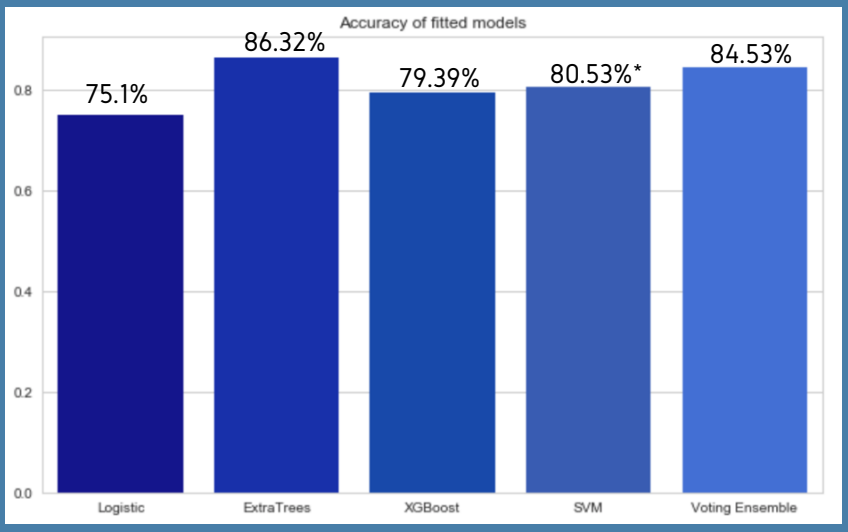
This report will illustrate the approach used for building models to predict round winners in the popular PC game Counter Strike: Global Offensive (hereby referred to as CS: GO). The main objective of the analysis was to find the main parameters/features for highly effective teams and to check which features had a decisive impact on a round's winner.

Consequently, the insights from the performed analysis can help the game developer detect any overpowered guns, improve balance and facilitate understanding about how the game is being played by experienced players. A benefit of this particular dataset is that it recorded gameplays in a high level tournament. By ensuring there are no noob players in the data, we can already exclude a main cause of outlier observations that would skew the analysis with inexperienced and illogical gameplays.

The data mining analysis performed in this study is based on the "Cross-Industry Standard Process for Data Mining" (hereby referred to as CRISP-DM), an iterative methodology which combines business and technical efforts to extract insights from data. After understanding the business purpose, we explored the data and looked for relationships between the variables. In the data preparation phase, the data was cleaned and a set of meaningful features were implemented. Subsequent feature selection and dimensionality reduction strategies concluded this essential stage. In the modelling phase, the eventual techniques for the machine learning models are described. This consists of general methodologies and a brief explanation of the algorithms used. Finally, the report is concluded in the evaluation and deployment phase. Main conclusions are drawn from the analysis and implications for the business setup are discussed.

PROPOSED SOLUTIONS

A model combining a random forest (ExtraTreesClassifier), xgboost and support vector machines was eventually chosen to deliver the most promising fit of the data. An ensemble classifier has the added advantage of the individual weaknesses of each input classifier against each other. When unifying the predictions of each model, three different voting methods have been implemented and compared, namely hard voting, soft voting, and weighted soft voting. The data scientists proposed a hard voting ensemble model for final predictions, which are based on a simple majority voting scheme of the presented classifier algorithms. It is worth mentioning that a variety of pre-processing options have been assessed beforehand to provide optimal input estimators. Finally, we were able to estimate the round winner with an accuracy of 84.53%.



1 BUSINESS & DATA UNDERSTANDING

BUSINESS UNDERSTANDING

In this first chapter, we establish the context and introduce the scenario of this report. Following the CRISP-DM methodology, the A-Team will first provide the business setting for the CS: GO data analysis. We have been mandated to investigate the CS: GO round snapshots dataset available in the kaggle competition with the aim to find the best model to predict which team is going to win the current round. The sponsor is interested in the best possible performance of the model, as he is building a betting office for the gaming industry and would like to have an edge over the competition. Therefore, the motivation of the presented study lies in the following three points.

- Which features are most indicative of which team wins the current round?
- What machine learning models perform best on this dataset?
- What conclusions can be drawn from the analysis?

Given the outlined scenario, the A-Team hired a professional CS player to brief the data scientists about the game purpose and in-game tactics. With an enhanced understanding of each column in the dataset, some potential key features were highlighted which will be discussed in the data preparation section. It is crucial to grasp the importance of dynamics in the game, certain combinations of teams, weapons, and the current map.

DATA UNDERSTANDING

In the next step, the procedure with regards to understanding the data is elaborated on. To begin with, the task at hand can be expressed as a binary classification on the target variable “round_winner”: counter terrorists “CT” or terrorists “T”. The measure of relevance is the accuracy of predictions. This is how results will be evaluated and how all of the tuning and optimizations during training were performed. To conduct such optimizations, the A-Team used Python as a coding language, where Jupyter Notebooks supported the creation of the final markdown .ipynb file. In detail, the task force leveraged on the packages numpy, pandas and sklearn. Moreover, some additional libraries were employed to round up the analysis and further refine the data processing.

An initial exploration of the data was performed to extract meaningful insights from the dataset. Thereby, it was found that the dataset contains snapshots of approximately 700 rounds of a high-level tournament played in 2019 and 2020. The total number of snapshots amounts to 122,411 (82,014 in the training set) that were taken at 20-second intervals.

Each snapshot captures all explanatory features including information about the round itself: time elapsed, weapons used, health, and so forth. The data types present in the data encompass numericals, booleans and string variables.

As shown below in the Data Exploration and Quality Report, no missing values have been detected. Effective value ranges, however, vary highly. For example, any weapon indicator is logically limited by the amount of players per team (max. 5), while the health variable goes from 0 to 500 and the money count even includes values up to 80,000. What became apparent was that every feature seemed to appear twice in total, once per team. The two exceptions to this pattern are the columns: “bomb_planted” and “t_defuse_kits.” The column “bomb_planted” can be linked to the Terrorist team while column “t_defuse_kits” is only applicable to the Counter Terrorist team.To further illustrate the conducted exploration, the table shows summary statistics for some selected variables in the training set (82,014 observations). The data exploration and quality report can be found in the Appendix.

*https://www.kaggle.com/christianlillelund/csgo-round-winner-classification)

2 DATA PREPARATION & PREPROCESSING

A DATA CLEANING

The dataset received for this task was obtained in a very clean state and as previously mentioned, no null values were found. In terms of outliers management, one row was found to be an invalid outlier and therefore removed. The observation stated 4 vs 6 players, which is simply not possible in the game setup. Additionally, the money-related columns exhibited a number of statistical outlier values. However, those represent legitimate observations and thus were kept. It is fairly normal that players in a team will show high dispersion in money due to their historical performance in the current game. This is due to the fact that money is generated by more kills, bombs planted and rounds won.

B FEATURE ENGINEERING & FACTORIZATION

An initial feature correlation task was conducted in order to understand the importance of each attribute with respect to winning and the relationship between the attributes. Given that the majority of our variables are numerical, a correlation analysis can be performed immediately without a lot of transformation. Should the variables be strongly correlated, techniques such as principal component analysis [PCA] will be used. In the Appendix, we illustrate the correlation between the features in a correlation matrix. It was used as a starting point to determine which features will be selected for the modelling phase.

The most important indicators of a round winner were found to be: money, health, type of gun and availability of armor & helmet. Moreover, the correlation between whether or not a bomb was planted and winning the round was very high. It was found that the Terrorist team has a much higher chance of winning the round when the bomb is planted.Feature engineering tasks were needed in order to make it simple for the model to understand patterns, not only between features and round winner, but also between the variables themselves. Factorization of variables & the creation of new variables originated from current ones was conducted in a logical sense. For instance, guns were classified into buckets by their type, likewise with assault rifles, snipers and shotguns. Namely, the folowing features summarizing gun classes were introduced:
ASSAULT / PISTOLS / SHOTGUN / SMG / SNIPER / EXPLOSIVES (for each team)

This factorization has helped the algorithms understand the effect of variables in a much easier way, alternatively reducing the number of relevant variables. Additionally, we know from personal experience that in first-person shooter games, usage of gun class is essential depending on map size. Hence, new features were created with combinations of gun choice with respect to map size. Namely, the following features summarizing further strategies were introduced:
ALL_ASSAULTS / ALL_WEAPONS / LARGEMAPSNIPER / SMALLMAPSNIPER / ALL_ALIVE (for each team); ROUND_COUNT

C FEATURE SELECTION

Once we have defined the variables that could be of interest to train our models with, we will continue with the feature selection. The aim was to get the best possible results from a predictive model on our test dataset by projecting data onto a lower dimensional subspace. Although some of the algorithms such as RandomForest and XGBoost perform some feature selections internally, it is advisable to reduce the input of noise beforehand. We have to bear in mind that less complex models are easier to understand, maintain and translate to the end users. If we pinpoint a reduced set of features to run the models, both computation time and overfitting can be minimized. It will also allow us to have better representation of all the available data. n the next section, we will introduce a benchmark model. The algorithm covers different feature selection techniques that were tested to determine the best results for the model used and facilitate accurate predictions.

2 DATA PREPARATION & PREPROCESSING

FEATURE SELECTION CONTINUED

F-SCORE ANOVA

To avoid a manual selection of those variables, we used the f_classif that computes the Anova F-value for the given set of data. Selecting the top 40 variables yielded a good tradeoff between selecting the most important features (model simplicity) while retaining all the information. This will be further explored in the Dimensionality Reduction section.

CHI-SQUARED

This technique is widely used when dealing with categorical variables, although Python also handles numerical variables internally. ! Assumptions: Data is NOT standardized since the variable domain accepted is [0,]

INFORMATION GAIN

This technique is based on the concept of entropy and how unpredictable an attribute can be. It helped the team select the top features by comparing the importance of excluding or including a variable when predicting the classification.

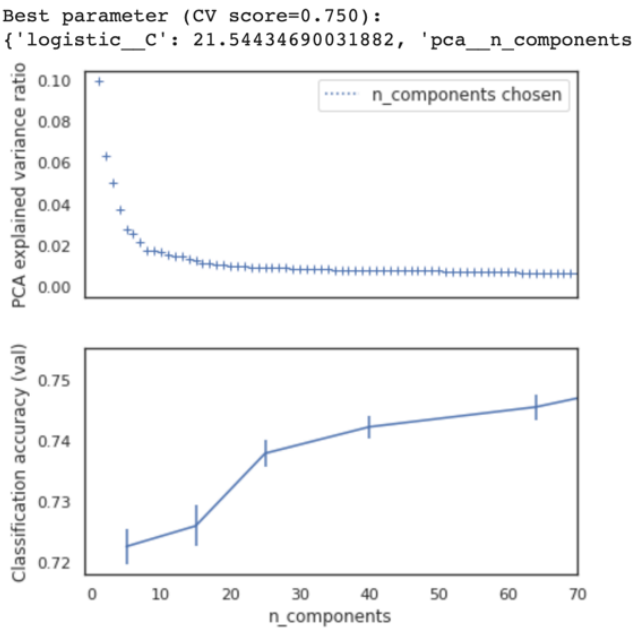
EMBEDDED METHODS

The embedded methods determine which features improve the accuracy of the model while it is being created, with a regularization penalty. Both Lasso and Ridge regression were tested, with a better performance from Lasso compared to the benchmark. To finalize the feature selection, and as a means of optimization of the latter pre-processing strategy, we have identified stable predictions with a negligible offset in accuracy (less variance for a small increase in bias) when applying a Lasso regression model. This additional step allows us to reduce the number of input features from 130 to 106 while improving the generalizability of our models.

For all the models detailed in section 3, the input data has been scaled and feature selection through a Lasso regression on the full training data is performed. Nevertheless, the team has explored additional ways of decreasing the number of input variables for its models. Therefore, the dimensionality reduction is elaborated below.

DIMENSIONALITY REDUCTION !! ASSUMPTION: DATA IS CORRELATED AND STANDARDIZED

When we observe the correlation of the variables we can see that many of them are correlated, therefore PCA technique was performed to create variables that are uncorrelated by definition. The specification of the number of principal components to keep was determined by running a grid search on the benchmark model. If we observe the graph on the right, from 40 components onwards, the increase in accuracy is relatively insignificant, so the team decided to take 40 components, as it captured a vast majority of the variance.



SCALABILITY OF THE MODEL

Depending on the business scenario, we would or would not like to have PCA applied, since it is more difficult to interpret for the end user. In our case, we can legitimately make use of such technique as the sponsor solely asks for an accurate prediction of the target, regardless of the level of interpretability. However, if we were to help the players of Counter Strike select their weapons to obtain the best accuracy, we would not use PCA as their requirements would be, for example, what guns they should use in each game. PCA is not suitable for this task as we enter a sort of black box model where clear variable influence cannot easily be extracted. In that sense, we should be careful when planning to expand our study into other areas.

3 MODELING & EVALUATION

DATASET SPLITTING STRATEGY

In order to deal with a classification task, supervised machine learning algorithms were considered to determine the outcome of the round in Counter Strike. The data available was already divided into train and test sets. A further 80/20 split was performed on the train set to create a validation set, where we compared the performance of different algorithms and quantified whether the selected model would perform well in production. The test set was reserved for a final prediction on the favoured model.

GENERAL MODEL OPTIMIZATION STRATEGY

To perform hyperparameter tuning, the sklearn library offers two interesting approaches: GridSearchCV and RandomizedSearchCV. The former facilitates a fundamental testing of every single combination specified by a set of values to try for each parameter. Quite conveniently, this function already includes a cross validation functionality, which will be thematized at a later stage in the report. With an increasing number of different combinations to try, an exhaustive grid search would extend the runtime by a multiple. Therefore, the team opted for an alternative method and implemented the RandomizedSearchCV. While exploring the same set of parameters, this randomized approach only fits a sample of the actual possible combinations. The size of which is defined by 'n_iter', that was determinative with respect to overall algorithm runtime. The higher we set this value, the more complete our optimization will be. At the same time, it will also drastically increase the computation time. A conclusive assessment of this challenge is discussed in the final chapter.

MODEL EXPLANATIONS

Given the constraints of time for delivery and computation power of the available tools to process different tasks, the A-Team opted for a simple model to be used as a benchmark (Logistic Regression with an accuracy of 75%), where feature selection techniques were compared. In parallel, the team built more complex models to outperform the benchmark model.

01 logistic Regression: Benchmark Model

As a first step, the team built a rather simple model to determine the baseline to beat in further iterations and improvements of the different models. The initial accuracy obtained was 75.0 %.

02 Tree-based models

In the literature, we find that Linear Discriminant Analysis [LDA] performs well when combined with PCA as an additional preprocessing step. The model gives the best possible linear representation between classes by fitting class conditional densities to the data and using Bayes' rule. The LDA does not require much hyperparameter tuning, and we obtained an accuracy on the validation set of 74.44 %.

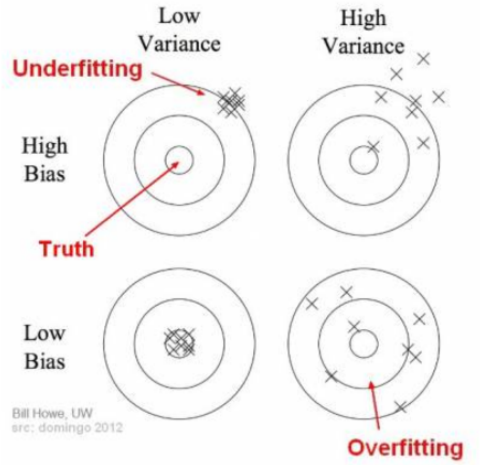
Another way of modelling our data is via classification trees, which allow for a qualitative response and predict the most frequently observed class for the "region" of any new instance. An appealing characteristic of such models is that the split rules found could be converted to direct rules for player/game characteristics. Quite conveniently, we can apply the following models easily to unscaled data. To initiate some tree-based strategies, we first ran a decision tree classifier and achieved an accuracy score on our validation set of approx. 79%. Fitting several trees simultaneously diversified this successful approach and provided even better ways to model the data. A random forest classifier, that averages the probabilistic prediction of the individual tree classifiers, yielded around 83% accuracy score. In order to reduce the risk of overfitting, various strategies were introduced:

1. **Decorrelating trees in random forest**

The A-Team decided to employ a random forest classifier to fit a number of independent classifiers and reduce variance. Only using a subset of all the features at each split allows the algorithm to add randomness and combat overfitting. Moreover, by the nature of the function, we benefit from the fact that every tree is built from a random sample (with replacement) of the actual training data. While the individual trees exhibit high variance, the averaging cancels out much of the noise.

2. **Extremely randomized trees: ExtraTreesClassifier**

To step up our game, we ventured into the deeper grounds of tree-based models and implemented the ExtraTreesClassifier from the sklearn library. An additional factor of randomness allows for another reduction in variance with a trade-off for a somewhat increased bias. The algorithm creates this additional source by randomizing the generation of split rules for each selected feature and only taking the best rules. The described trade-off can be visualized in the below plot



(source: https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229):

In our case, lower variance means more stable predictions coming from a simpler model, while a higher bias would lead to a possibly lower accuracy on the validation data. In conclusion, this extra characteristic fosters a more general modelling of the data and again reduces the risk of overfitting.

As the ExtraTreesClassifier fitted the data very well, we devoted further research into the topic and started to tune its hyperparameters. In the below list we briefly introduce the selection of parameters that has been optimized in a RandomizedSearchCV.

Criterion: Gives a measure the quality of a split / Describes the node purity that can be measured with the 'gini' index or the information gain (labelled cross 'entropy'). Optimized Parameter: 'gini'
N_estimators: Corresponds to the number of trees in the fitted forest.
Optimized Parameter: 1,720
Min_samples_split: Gives the minimum number of samples required to split a node. Optimized Parameter: 2
Min_samples_leaf: Gives the minimum number of samples required to be a terminal node. Optimized Parameter: 1Max_depth: Establishes the maximum depth of the trees, else nodes are expanded until the min_samples_split is reached. Optimized Parameter: 88

TECHNIQUE	F-SCORE 01	IG 02	LASSO 03	CHI-SQUARED 04
N_ESTIMATORS	3430	2290	1720	3430
MIN_SAMPLES_SPLIT	2	2	2	2
MIN_SAMPLES_LEAF	1	1	1	1
MAX_DEPTH	100	31	88	100
CRITERION	ENTROPY	GINI	GINI	ENTROPY
ACCURACY	83.33 %	83.00 %	86.32 %	83.6 %

3.**Boosting: XGBoostClassifier**

As a means of diversifying our modelling approach, Gradient Boosting and AdaBoost algorithms were investigated. Learning from the errors of initial classifications, this family of models fits sequential classifiers aimed at reducing bias. Unfortunately, the additive tree modelling did not achieve the desired result. Therefore, the data scientists at A-Team worked with the xgboost algorithm from a non-sklearn library. It boasts an integrated Variable Importance Functionality and does not require any feature transformation. While XGBoost can be implemented without any preprocessing, the A-Team tried running it with both options, with and without preprocessing. It ended up representing one of our final models. In a subsequent step, the A-Team looked deeper into the algorithm and extracted optimal hyperparameters for the suggested model. Thereby, the following set of parameters has been iterated over:

Colsample_bytree: Part of the feature subsampling parameters. Allowed subsampling ratio of variable per tree, somewhat related to the idea of ExtraTrees. Optimized Parameter: 0.83120
Gamma: Is the minimum loss reduction needed to perform another split. Small gammas make the algorithm more aggressive. Optimized Parameter: 0.28985
Learning_rate: Is the shrinkage step size [0,1] said to prevent overfitting. A large value shrinks the weights of new values after each sequential step and is more conservative. Optimized Parameter: 0.02904
Max_depth: Establishes the maximum depth of the trees, high values tend to overfit. Optimized Parameter: 5
N_estimators: Corresponds to the number of trees in the fitted forest.
Optimized Parameter: 153
Subsample: Gives the random subsample ratio to consider when growing trees for each sequential step. Optimized Parameter: 0.88323

TECHNIQUE	F-SCORE 01	IG 02	LASSO 03	CHI-SQUARED 04
COLSAMPLE_BYTREE	0.92814	0.746798	0.831199	0.746798356
GAMMA	0.099837	0.02904	0.0290418	0.02904
LEARNING_RATE	0.18427	0.28985	0.289853	0.28985
MAX_DEPTH	5	5	5	5
N_ESTIMATORS	180	180	153	153
SUBSAMPLE	0.943976	0.8832290	0.883229	0.8832291
ACCURACY	77.8 %	77.7 %	79.4 %	78.0 %

Due to the increased risk of overfitting in the learning process, the A-Team has mitigated such risk by combining the optimized XGBoost predictions in the final ensemble classification. This concept will be outlined in the last section of this chapter.

03 Support Vector Machines (SVM)

Since the feature space is relatively high, we opted to fine tune a Support Vector Machine algorithm, which tries to maximize a margin classifier. SVM models are prone to overfit if the hyperparameter's tuning is not properly set. We tried to find the balance between the computation cost in time and the choice of parameters that would not cause the model to overfit. The chosen parameters to fine tune were the following:

Regularisation Term (C): Derives the tradeoff between correct classification of the instances and maximization of the margin. A lower value would result in a larger margin with a bigger tendency to underfit our data. Optimized Parameter: 100*

Gamma: Determines how far the influence of an instance reaches. Gamma being large means that the area of influence of the support vectors only includes that support vector, and not even fine tuning C will prevent the model from overfitting. The parameter is suitable for "rbf" kernels. Optimized Parameter: 'scale'

Kernel: The team used predefined kernels and defined a custom one to optimize the accuracy. A grid search determined that the most performant kernels were "rbf" and "poly". Optimized Parameter: 'rbf'

Degree: Of the polynomial kernel. Irrelevant for final model configuration.

TECHNIQUE	F-SCORE 01	IG 02	LASSO 03	CHI-SQUARED 04
KERNEL	RBF	RBF	RBF	POLY
GAMMA	SCALE	AUTO	SCALE	SCALE
DEGREE	-	-	-	4
C	100	100	100	10
ACCURACY	78.13 %	79.12 %	84.16 %	80.4 %

04 Ensemble: Voting Classifier

Even though we performed a thorough grid search within the different algorithms, we wanted to be conservative when it comes to overfitting. Besides, the team was interested in a way to balance out each of the proposed model's individual drawbacks. One way to solve this issue is to use an ensemble model which unites the chosen classification algorithms under one hood. More specifically, a voting classifier, which will "vote" for the class prediction, was employed. In general, there are three techniques for the models to vote:

- **Hard / Majority Voting.** Predict the class with the largest sum of votes from models
- **Soft Voting.** Predict the class with the largest summed probability from models.
- **Weighted Soft Voting.** Assigning customized weights based on our own rationale.

*Optimised value, however a grid search on a subsets of values of C yielded similar results in accuracy for lower C and the team kept C= 10. to avoid overfitting.

JUPYTER NOTEBOOK STRATEGY

The different steps outlined in the report support the data analysis in the associated Jupyter notebook. The overall structure is detailed in this paragraph. First, we set up the file and imported the datasets. After a brief exploratory data analysis, the team cleaned the data, came up with some interesting features and factorized the categorical variables. At this stage, we specified a number of switches that will later allow us to easily modify the input data for our models. The switches include the option of feature scaling and which feature scaling should be followed in the script. Once these different settings are defined, we split the datasets into the introduced train, validation and eventual test sets. A benchmark model was then established to provide a basis to compare the selected feature selection procedure against it. For feature selection, the notebook entails the discussed filtering and embedded methods. In addition, a PCA was paired with an LDA model to discover a way of dimensionality reduction. Next, we compute some selected “challenger” machine learning models. While running the different configurations, we ended the loops with exporting the optimal set of hyperparameters in separate json files. Lastly, we compute the ensemble model specification which concludes the analysis.

4 EVALUATION & DEPLOYMENT

TECHNICAL EVALUATION

In order to assess the outlined models' performances, the team used the accuracy of the predictions as the decisive evaluation criterion. In the following table, we present a quick summary of the models' scores on the validation after applying standardization and the Lasso feature selection. Please note that the first logistic model represents our benchmark and was run on a scaled dataset considering all available features.

Approach	Accuracy
Logistic Model - STD	**0.75096**
Logistic Model - Lasso - STD	**0.75084**
ET Base Model	**0.8585**
ET Randomized Model	**0.8632**
XGBoost Model	**0.79388**
SVM Model	**0.8053**
Voting E-Model 1	**0.8453**

In order to assess the outlined models' performances, the team used the accuracy of the predictions as the decisive evaluation criterion. In the following table, we present a quick summary of the models' scores on the validation after applying standardization and the Lasso feature selection. Please note that the first logistic model represents our benchmark and was run on a scaled dataset considering all available features.

BUSINESS EVALUATION

Due to our models' particularities, we decided to come up with a final assessment taking into consideration the business environment. In the bespoken setting, the A-Team had to further consider the computing cost as a limiting factor. Due to time constraints, the analysis process had to fit a dense time schedule allowing only for a small subset of possible iterations.

For instance, we were pressured to use RandomSearchCV for the sake of reducing the runtime of algorithms. Implementing this tool, the team was even more restricted to a small K when running K-fold cross validation. A potential upside could have been materialized on a higher K to provide a less biased estimator. Eventually, iterations were run with a 5-fold cross validation. When specifying the number of iterations in the RandomSearchCV, we had to reduce the initial value of 200 down to a more feasible value of 50. This led to a decrease in runtime from 200*5 = 1,000 tasks to 50*5 = 250 tasks, significantly shortening the hyperparameter optimization loops by 75%. Considering the fact that such tuning had to be run three times, once for every classifier component in our final ensemble model, the impact of a shorter loop is immense.

In addition, we came up with seven distinct scenarios of data preprocessing to serve as an input to our models. This multiplies the total runtime by a factor of 7, hence making the reduction of iterations even more severe. Implementing all possible scenarios for data pre-processing would have again tripled this estimate, thus we simplified our procedure and only iterated through the chosen subset of scenarios. With average runtimes for the hyperparameter tuning (RandomizedSearchCV 5-fold, n_jobs=-1) for the ExtraTrees classifier of 400 minutes, XGBoost of 20 minutes, and SVM of 1,400 minutes, we estimate our potential runtime of 1,820*7 = 12,740 minutes, i.e. almost 9 full days of running code on our local machines. If we had a server available with exponentially higher computing power, this wouldn't be a concern of ours. Therefore, when implementing the machine learning project in a real-world environment, we highly suggest to run further optimizations and code maintenance on strong machines.

Ultimately, we make the point that in the final iPython notebook the optimization randomized search cross validations are present, yet don't need to be run in order to estimate our final voting classifier. The optimized hyperparameters for each internal classifier are hard-coded so that the final predictions can be obtained in a reasonable amount of time. To sum up, we propose an ensemble model to the sponsor, that enables him to predict the round winner based on an elaborate and balanced model with an accuracy well above a random 50/50 guess. The presented data analysis investigated the business questions proposed in the first chapter and promises a prosperous future for the betting office.

5 APPENDIX

The Appendix includes technical information related to the project. It consists of the following sections:

1. DATA UNDERSTANDING
    - Data Exploration
2. DATA PREPARATION
    - Correlation Analysis

01 DATA UNDERSTANDING

FEATURE	TIME_LEFT 01	CT_SCORE 02	T_SCORE 03	MAP 04	BOMB_PLANTED 05	CT_HEALTH 06
TYPE	DECIMAL	FLOAT	FLOAT	CATEGORICAL	BOOLEAN	FLOAT
MISSING	0	0	0	0	0	0
MISSING %	0 %	0 %	0 %	0 %	0 %	0 %
MAX	175.00	32.0	33.0	TOP: INFERNO	TOP: FALSE	500.00
MIN	0.03	0.0	0.0	COUNT 16,011	COUNT 72,892	0.00
AVG	97.93	6.70	6.77	LEAST: CACHE	LEAST: TRUE	412.11
STD	54.45	4.79	4.82	COUNT 103	COUNT 9,122	132.29

FEATURE	T_HEALTH 07	CT_ARMOR 08	T_ARMOR 09	CT_MONEY 10	T_MONEY 11	CT_HELMETS 12
TYPE	FLOAT	FLOAT	FLOAT	FLOAT	FLOAT	FLOAT
MISSING	0	0	0	0	0	0
MISSING %	0 %	0 %	0 %	0 %	0 %	0 %
MAX	600.00*	500.00	500.00	80,000.00	80,000.00	5.00
MIN	0.00	0.00	0.00	0.00	0.00	0.00
AVG	402.74	314.02	298.71	9,801.42	11,213.00	2.06
STD	139.91	171.08	174.41	11,239.04	12,141.56	1.84

FEATURE	T_HELMETS 13	CT_DEFUSE_KITS 14	CT_PLAYERS_ALIVE 15	T_PLAYERS_ALIVE 16
TYPE	FLOAT	FLOAT	FLOAT	FLOAT
MISSING	0	0	0	0
MISSING %	0 %	0%	0 %	0 %
MAX	5.00	5.00	5.00	6.00*
MIN	0.00	0.00	0.00	0.00
AVG	2.78	1.62	4.27	4.27
STD	2.01	1.61	1.20	1.23

*Single outlier with 't_health'=600 and 't_players_alive' = 6 has been removed

02 DATA PREPARATION & PREPROCESSING

Once the exploration of the data, the A-Team looked into missing values, outliers and non-plausible values in the next phase.

CORRELATION ANALYSIS

The rationale of inclusion/exclusion of variables may be based on the correlation of the features, yet we decided to use all features and select the relevant features of the final training data with filtering and embedded methods. In a following step, one-hot encoding of the categorical variables was performed so that the newly created features and the variable 'map' could easily be displayed.