Machine Learning

Professor: Angel Castellanos Gonzalez

Group Assignment: CS-GO

Group B

Nicolas Boer
Eleonora Jimenez
Laura Frazer
Stefano Magnan
Alberto de Roni
Rebecca Rosser

Table of Contents

Guidelines

Details

- This is a binary-class classification task (the target variable is round_winner)
- I will use accuracy to evaluate your predictions, so I recommend you use the same metric
- for the training process.
- You can apply any of the algorithms we have explained in class to address the task:
- Regression, Decision Trees, Naive Bayes, KNN...
- You can use R or Python
- You can use Dataiku for Data Cleaning and Feature Engineering
- This is a group assignment

Recommendations

- Look to the problem description. It contains valuable information on the dataset and the task
- Feature Engineering is important for this dataset. It includes categorical and numerical values, so, you should properly deal with each of them: remove outliers, remove useless features, look for NAs and missing data and impute them, create new features if you consider them useful, reduce the dimensionality if it makes sense.
- You can use whatever library you want. Nevertheless, I would recommend you review caret (for R) and Sklearn (for Python) documentation. These libraries provide you with almost anything you need to solve the task.

Submission

You must send me:

- The Markdown, Notebook, or code you have implemented to generate your results. This markdown must provide a comprehensive description of the machine learning pipeline: ideas that you have tested, the problems that you have faced, and you should explain and justify any decision you take.
- Final report in a separate file where you summarize all your findings, the insights you could have extracted from your data, the conclusions of the different steps of your analysis, as well as the final conclusions from your results.
- The CSV with your predictions on the test set.

Evaluation

For grading you work, I will consider the 3 following aspects:

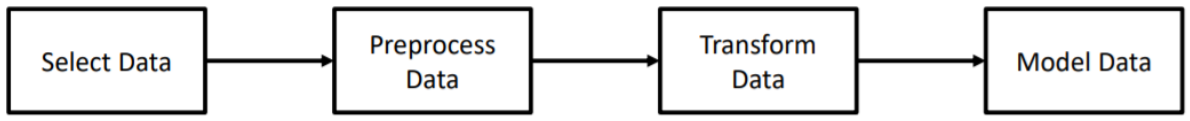
- The Machine Learning pipeline in your markdown: 40%
- The report summarizing your work, findings, and conclusions: 40%
- The final accuracy in the competition: 20%

Introduction

The "business" context on which this analysis is based is the popular online game Counter-Strike Global Offensive (CS-GO), a game where players are divided in two teams, Terrorists and Counter-Terrorists. The objective of Terrorist players is to plant a bomb or kill all the players of the enemy team, while the objective of Counter-Terrorists is to defuse the bomb or kill all the enemies as well. The game is played in 30 rounds of 1 minute and 55 seconds and the team that first wins 16 rounds wins the match.

The objective of this analysis is to predict which team is more likely to win a specific round based on a dataset. This dataset contains over 120,000 snapshots captured every 20 seconds across hundreds of games played at high-level tournaments between 2019 and 2020.

To approach this problem from a ML perspective, we followed the Machine Learning Pipeline methodology and completed the following steps:



We performed the following activities:

1. **Select data**: we uploaded the data inside our environment and did an EDA to better understand both the final objective and the variables available to reach it.
2. **Preprocess data**: after having studied the "business context", which includes the rules and how the game works, we performed some data cleaning activities.
3. **Transform data**: after having cleaned the data, we applied some transformations on the original dataset, based on both business intuition and statistical techniques.
4. **Model data**: after creating new datasets with new features, we finally trained different machine learning algorithms on them, to find the one that provides the best accuracy score.

We found the model that maximizes our chosen metric, accuracy. Then we use this model to predict the values of our target variable (which one of the teams is going to win) in the test set.

Select Data

Original dataset

To resolve this Machine Learning challenge, two datasets were provided, train and test sets. These datasets come from a randomized split of one Kaggle dataset that can be found at this link (https://www.kaggle.com/christianlillelund/csgo-round-winner-classification).

The full dataset consists of over 120,000 snapshots that register the situation in a precise moment of the game during various rounds of a series of competitive tournaments played in 2019 and 2020.

The target variable of our dataset is Round Winner, which indicates which team between CT and T is going to win a specific round.

The other variables present in the dataset measure various characteristics of the game for each one of the two teams at the moment were the snapshot was taken. Some examples are the Time Left to the end of the round, if a bomb was planted or not, and a series of characteristics of the CT and T teams (eg.the amount of players alive per team). In this report we will refer to “pairs” of variables to these variables that relate to the same characteristic (such as health) but is separated for CT and T teams.

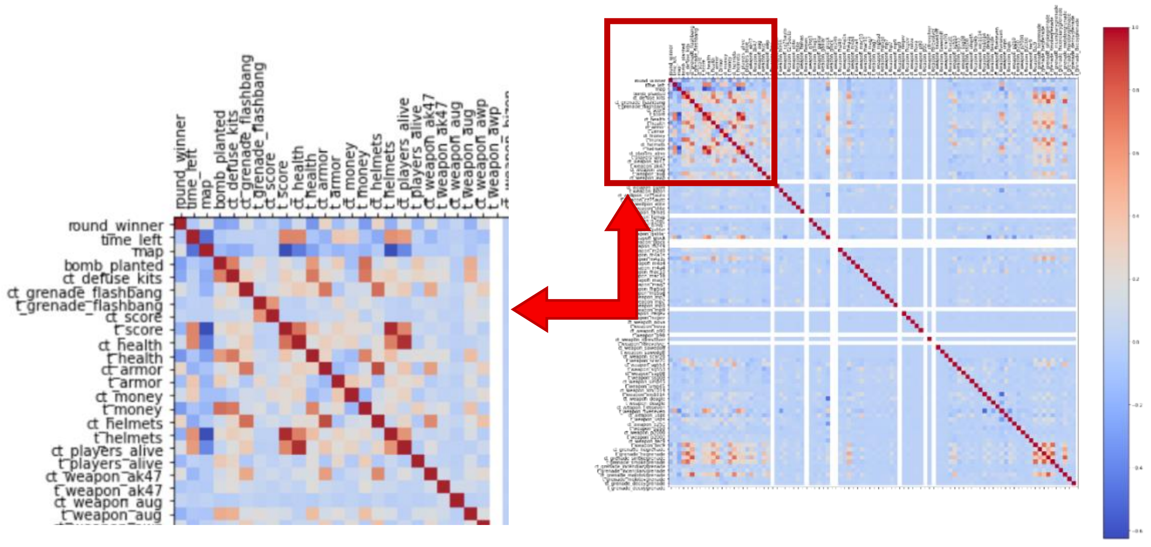


Figure 1: Showcases the same variable for Terrorist and Counter-Terrorists – an event we refer to as "pair" of variables

Exploratory Data Analysis

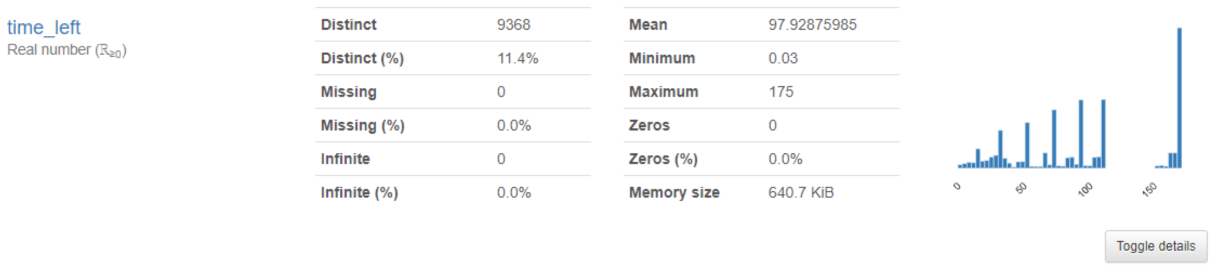
To perform the Exploratory Data Analysis, we used the Pandas Profiling Report, in order to automatically have the distributions and statistical characteristics of all the variables present in the dataset. The report was saved in “.html” and exported from the notebook due to performance purposes (CS_GO_Report.html of 120MB).

First thing to notice is that we have 3 categorical variables, Round Winner: encoded ac ‘CT’ and ‘T’, Map: which is valorized with the name of the map for that match, and Bomb Planted: which is encoded as True or False (Boolean). These variables needed to be transformed into numerical features so algorithms could process them.

Looking at the remaining numerical variables, we noticed that most of them present a skewed distribution. This is partly due to the great amount of 0 values present in these variables.

In addition, some variables representing weapons seem to have little or no discriminate power at all, having mostly 0 values. Nonetheless, we decided to keep them in our dataset because we planned to perform feature engineering on the variables, by combining them to extract more information.

In addition, we noticed that in some variables, such as time_left, there seem to be errors because they are not compliant to the rules of the game. Approximately 10% of the data has a value greater than the maximum value the official rules of the game allow. In this case, given the amount of data affected by this, we assume that it is not an error in the data and that for highly competitive games there could be some exceptions in the amount of time assigned for a round.



time_left
Real number ($\mathbb{R}_{\geq 0}$)

Distinct	9368	Mean	97.92875985
Distinct (%)	11.4%	Minimum	0.03
Missing	0	Maximum	175
Missing (%)	0.0%	Zeros	0
Infinite	0	Zeros (%)	0.0%
Infinite (%)	0.0%	Memory size	640.7 KiB

Figure 2: time_left description (Pandas Profiling Report)

Additionally, we want to include all time_left data in our model because the test data is characterized by this pattern as well. As such, we want to model our algorithms to take this factor into account.

For other variables, such as t_players_alive, we found some values that are clearly against the rules, (e.g. 6 players for one team, when the max allowed is 5). Given the rarity of these cases we can assume they are errors in the data, that we will remove in the Data Cleaning step.

From this EDA we came out with the following conclusions:

- The information in this dataset is dispersed in many different variables, so we need to combine them through "business sense" or in a statistical way
- Most of the variables present skewed distributions, so we need to take this into account in the modelling part
- There is a need for transforming categorical variables into numerical on the Preprocess step

Preprocess data (Data Cleaning)

In this section we will explain all the data transformations done.

Outliers identification and exclusion

As we have seen during the EDA, some variables present some data that is explicitly against the rules of the game. To identify and remove them from the dataset, we defined a numerical threshold for each variable, that identifies the number over which the value is considered an error.

Then, we loop all the columns of the dataset, comparing each value of each column with the parameter of that variable and identified the rows that have at least one of these errors.

Finally, we remove from the original dataset these rows by dropping them from the dataset. We can say that this approach is based on business acumen rather than statistics.

Transform Data

Categorical variable transformation

To be used in machine learning models, categorical variables must be coded in a numerical way so the computer can process them. We performed the following transformations:

- Round Winner: we substituted 'CT' and 'T' to 0 and 1 respectively.
- Bomb Planted: we substituted False and True (Boolean) to 0 and 1 respectively.
- Map: We utilized a One Hot Encoder, to transform this variable into columns to have the same number of columns as the number of distinct maps, with a 1 in the corresponding value for each row. This last transformation is done on the datasets after having performed feature engineering, just before the modelling phase.

Delta

Looking at the original data, we realized that most of the variables are organized in pairs, meaning that the same information is repeated for both the CT and T teams.

We had the intuition that the delta of the values of each pair of variables (e.g. CT and T health, CT and T players alive) can have a higher predictive power than the two variables taken separately. The intuition behind this is that, what most influences the outcome of a round is not the absolute number of the two teams taken separately (e.g. number of players alive in that precise moment). Instead, the most influential aspect is how much more a team has of that measure compared to the opposite team.

To extract this information, we take the difference combining the values of each pair of variables into a single column. This is also a form of dimensionality reduction. This feature engineering step is crucial in our dataset because the majority of the ML algorithms we have seen, such as Decision Trees, have problems in combining information from variables that alone are not good at predicting the target variable.

To mathematically prove this intuition, we calculated the correlation between each pair of variables, the delta, and the target variable. The correlation between the delta variable and the target one was always higher than the respective correlation between the original variables and the target taken separately.

The output of this step is a dataframe containing the original variables that are not divided into CT and T, plus the delta of all the other pairs of variables. We called this dataframe 'delta_df'.

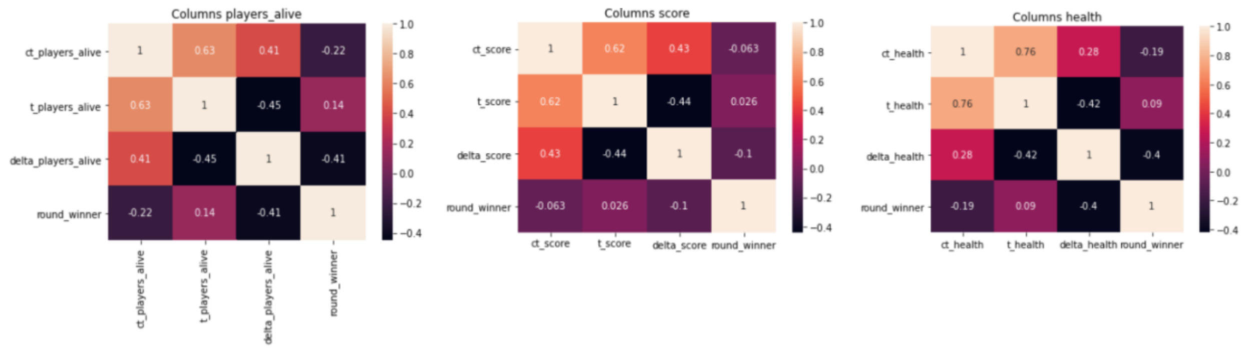


Figure 3: correlation between variables and their delta with the target variable

Weapon Categorization

During the EDA, we saw that the weapon variables are many and some of them seem to provide little discriminatory power to predict the target variable. Therefore, we wanted to group them with business criterion.

After having researched the characteristics of each weapon according to the game, we created a weapon categorization dataset that we used to group the variables belonging to each category into a single variable. We did this, separately, for both the CT and T variables. This grouping was performed using both the average and the mean of the values of the original variables. After having created the grouped variables, we took the delta between the corresponding pairs of CT and T, as we did in the previous step.

Let us illustrate this with an example:

> There are many weapons listed under the category of 'Rifle': weapon_awp, weapon_ssg08, weapon_scar20, weapon_g3sg1, etc. These weapons would be grouped under the category 'Rifle' for both Terrorists and Counter-Terrorists, separately. Then, the number of weapons inside the 'Rifle' category would be added and the average computed for both Terrorists and Counter-Terrorists. Lastly, the difference between the average number of rifles for Terrorists and Counter-Terrorists is computed to obtain the delta.

The output of this step is a dataframe "weapcat_deltaa_df" containing the delta of the original variables grouped by category, plus the original variables that are not divided into CT and T.

Principal Component Analysis

Given the high number of features present in the original data (~100), and to a lesser extent, in the dataset obtained as output of the feature engineering step (~58), we performed a PCA. The purpose of the PCA was to combine the variables and be able to explain the majority of the variance of the target variable with less features (e.g. selecting only the amount of features that explain ~90% of the variance).

However, a word of caution is needed before performing PCA. While performing PCA we are assuming two things:

- **Normal distribution**: as we saw in the EDA, the distribution of variables in the original data is very skewed. Similarly the transformed data is not normally distributed (even if the data is a bit more symmetrical). Therefore, when looking at and interpreting the results of the analysis, we need to take into account that the data does not fulfill the assumption of being normally distributed. .
- **Standardized scale**: We will standardize the variables to have the same scale before performing the PCA.

Taking into consideration the PCA assumptions, we performed the following trials:

1. Perform PCA on weapon variables (standardized)
2. Perform PCA on all variables (standardized)
3. Perform PCA on all CT and T variables separated (standardized)
4. Perform PCA on pairs of CT and T (standardized)
5. Perform PCA on delta of weapon variables (standardized)
6. Perform PCA on delta of weapon variables + variables without delta (standardized)
7. Perform PCA on weapon variables grouped by category (standardized)
8. Perform PCA on weapon variables grouped by category + variables not grouped (standardized)
9. Perform PCA on pairs of CT and T for weapon variables grouped by category for CT columns and T columns (standardized)
10. Perform PCA on delta of weapon variables grouped by category + variables not grouped (standardized)
11. Perform PCA from time left to players alive (19 variables) and for the grenades related variables (standardized)

In all these analyses, we first standardized the input variables, then extracted the Principal Components that explain at least 90% of the variance of the original variables. After doing this, we saved the results in a dataframe and calculated the correlation of the PCs extracted with the target variables to understand the predicting power of these PCs. Additionally, we calculated mutual information scores between the PCs and the target variable.

The results of the PCA were not satisfactory at all. Both the number of dimensions reduced and the predicting power of the PCs obtained were not good enough. We can conclude the following:

- In all the analyses, only one PC stands out with a correlation to the target variable greater than 0.4, which means that we will need many PCs to describe at least 90% of the variance.
- We could link the disappointing results to the fact that the data was not fulfilling the standard PCA assumption of normally distributed variables.
- Additionally, transforming "business knowledge" into features may bring better results than performing PCA alone.

To wrap up, we decided to use the dataset obtained from PCA N°10 (PCA on delta of weapon variables grouped by category + variables not grouped) in the modelling phase because it showed the most promising results of all the other analysis.

Evaluation of Feature Engineering

After having created these new datasets with new features, we evaluated the prediction ability by means of two methods: correlation and mutual information.

We found out that the most promising results were obtained using the following datasets:

- Original Data with Map as Dummy: N° of features = 104
- Deltas ('delta_df'): N° of features = 59
- Weapons category grouped + deltas ('weapcat_deltaa_df'): N° of features = 24
- Weapons category grouped + deltas + PCA ('pca_weapcat_deltaa_df'): N° of features = 18

Model data

First, we defined a baseline by selecting the most relevant feature identified by the decision tree: armor. Then we made the difference between teams on that features (ct_armor – t_armor) and we defined a threshold

1. Using delta_armor (difference between ct_armor and t_armor) we defined a threshold = 0
2. If the difference is > threshold, the winner is predicted to be CT
3. If the difference is < threshold, the winner is predicted to be T

The Baseline accuracy: 0.7172

Now, that we had a baseline we took a systematic approach to assess the different algorithms to the different datasets taking as metric the accuracy.

Listed below, is the comprehensive list of models we decided to use:

- Logistic regression model
    - Pure logistic regression
    - Logistic regression + cross-validation + 'Ridge' regularization
    - Logistic regression + cross-validation + 'Lasso' Regularization
- Decision Tree Classifier models
    - One decision tree + GridSearchCV
    - Bagging
    - Random Forest Classifier + RandomizedSearchCV
    - XGBoost model + RandomizedSearchCV

- Support Vector Machines (SVM) model
    - SVM + RandomizedSearchCV + kernel=linear
    - SVM + RandomizedSearchCV + kernel=rbf
    - SVM + RandomizedSearchCV + kernel=poly

As we have limited computational resources to process all these algorithms on all the datasets, we decided to compare the performance of the fastest algorithms on all the selected dataframes (Logistic regressions and Decision Tree Classifiers except XGBoost).

We identified that the 'delta_df' dataframe was providing the best accuracy across the different algorithms. Therefore, we decided to try XGBoost and SVM (with RandomizedSearchCV) to increase the accuracy.

We tried different paid and free tools to try to decrease the processing time, by parallelizing the computations:

- Group B individual computers: we used all our laptops
- Google Colab: The free version provides a 'simple' 2 cores + 4gb ram
- AWS Sagemaker: We use it with free version, it provided a 'simple' 2 cores + 4gb ram
- Paperspace: We managed to setup a server with 12 cores + 32gb of ram

Even if using all these resources, while for XGBoost we managed to achieve some results, for SVMs models we could not properly tune the necessary hyper parameters needed to achieve better results, because of the excessive computational power needed to fit the models and run the Grid or Randomized Search for the various combination of parameters (e.g. with a Intel i7 processor + 24gb ram it took + 24hs to compute and we close it).

As one could expect by doing this parallelizing of jobs, we cannot have all the results in one Jupyter Notebook. We run the algorithms with the full train set and got the accuracy using the attribute "best_score" of the RandomizedSearchCV function.

The following table summarizes the results:

Models and performance

	Dataset			
Model	**original df (accuracy)**	**delta_df (accuracy)**	**weapcat_deltaa_df (accuracy)**	**pca_weapcat_deltaa_df (accuracy)**
Logistic regression models				
Pure logistic regression	0.7500	0.7484	0.7384	0.7461
Logistic regression + cross-validation + 'Ridge' regularization	0.7460	0.7459	0.7459	0.7460
Logistic regression + cross-validation + 'Lasso' Regularization	0.7506	0.7491	0.7506	0.7463
Decision Tree Classifier models				
One decision tree + GridSearchCV	0.8040	0.7863	0.7736	0.7493
Bagging	0.8460	0.8406	0.8191	0.7892
Random Forest Classifier + RandomizedSearchCV	0.8543*	0.8547	0.8289*	0.7990*
XGBoost model + RandomizedSearchCV	n.a.	0.8227	n.a.	n.a.
Support Vector Machines (SVM) models				
SVM 'default'	n.a.	0.7211	n.a.	n.a.
SVM + RandomizedSearchCV + kernel=linear	n.a.	n.a.	n.a.	n.a.
SVM + RandomizedSearchCV + kernel=rbf	n.a.	n.a.	n.a.	n.a.
SVM + RandomizedSearchCV + kernel=poly	n.a.	n.a.	n.a.	n.a.

** Using GridSearchCV*

Baseline accuracy: 0.7172

The Random Forest with the 'delta_df' dataframe held the best results with an accuracy on the train set of: 0.8547

The next step was to use that model and those transformations made in the ‘delta_df’ dataset on the test_set to make the final predictions.

Conclusion

After performing all the steps in the machine learning pipeline, we come up with the following final thoughts on the whole exercise:

- Define a structured approach and structured document since the beginning, not to have results of the algorithms in different versions of the notebooks and provide clarity to each step (e.g. get lost in the numbers).
- Make a ‘baseline’ with the ‘simplest’ model you can do, so you know how much you improve with the different dataset and models.
- Make iterations since the beginning, start fast and agile, make a simple model then test the original dataset and build from there.
- Spend time in the EDA to understand what type of data the dataset provides and related the data to the constraints of the different algorithms (e.g. PCA).
- Get as much ‘business knowledge’ you can get and built new features out of them. These features proved to be very important in explaining the target variable.
- Do not fall in love with a dataset nor with a model. Test and challenge all the models and datasets.
- Consider the difference between GridSearchCV and RandomizedSearchCV before computing the algorithms.
- Bear in mind the computational requirements of some algorithms (e.g. SVM) to make the retro-planning from the deadline.
- Always copy() the original dataframe and WIP dataframes or set different names for the variables, to be able to change the code quickly without re-running from the beginning.