History Battles Classification

Fèlix Fernández Peñafiel[a,1]

[a] *1523257*

Abstract—Warfare has been an ever-present part of humanity, either for religious, political or social purposes. Victory for one side means defeat for another, at times a costly one. Due to the stakes at hand, a way to predict, beforehand, the outcome of a battle, may help an army's commander make better decisions on whether to engage in battle or not. That is why this project will try to classify the outcome of battles, using different machine learning techniques. Two of the tested techniques yield results above 75% F1-Score, far exceeding the values expected of a random classifier.

Keywords—*Classification, Battle, History*

Contents

1. Introduction

Warfare has been a pivotal force shaping human history, with battles often determining the fates of nations, civilizations, and cultures. Historically, military strategy has relied on intuition, experience, and sheer chance to predict the outcome of conflicts. In modern times, advances in data science and machine learning have opened up new opportunities to analyze historical data and identify patterns that can help predict outcomes.

The goal of this project is to use machine learning techniques to classify the outcomes of historical battles. By analyzing key factors such as troop strength, commanders, terrain, and weather conditions, this project seeks to develop predictive models capable of determining whether an attacker or defender is likely to emerge victorious. The insights gained could help historians, strategists, and enthusiasts better understand the dynamics of warfare.

Through preprocessing, feature engineering, and the application of algorithms such as Logistic Regression, Random Forest, Support Vector Machines, and Gradient Boosting, the project evaluates the feasibility of predicting battle outcomes with a high degree of accuracy. The results provide a glimpse of the potential of machine learning to transform the analysis of complex historical events.

2. The Data

The project is based on the Historical Military Battles dataset, found in Kaggle, originally created by the US Army, although it has later been cleaned up. This dataset contains 660 battles, from 1600 to post-WWII conflicts.

The dataset contains numerous CSV files, some containing the different attributes of each battle, others that explain the different values that these attributes can take up. It must be noted that while some of these attributes are absolute, like the force's strength of each contender, are absolute, and thus not open to interpretation, others, regarding the attacker's advantage in a specific field, are relative, subject to the view of the dataset's creators. This is what makes it impossible to add new battles to the dataset, which has been tried but the accuracies suffered a huge drop.

2.1. Importing and Modifying the Dataset

As stated, the Historical Military Battles dataset is formed by many datasets, all with different attributes, and only sharing a battle's identifier, which will be the bridge to combine the datasets. In order to do that, the datasets needed to be cleaned, a task done in the file *History_Battles_Dataset_Creation.ipynb* in the Github repository.

First of all, each individual dataset had to be filtered in order to eliminate useless variables such as the battle's name. After this, most datasets could be merged into one, such as the *Weather* and *Terrain* datasets.

The *Belligerents* and *Commanders* datasets needed more filtering, since they had multiple rows per battle. Starting with the first one, the dataset was separated into attackers and defenders into different datasets, and all attributes within them were renamed, to make clear if these pertain to the attacker or the defender. After this, *Belligerents* was ready to be combined with the others.

The only individual dataset left to merge was *Commanders*. As *Belligerents*, this had multiple rows pertaining to one battle. However, while *Belligerents* had only two rows per battle (the attacker and defender), *Commanders* could have more, up to 5 rows per battle (3 defenders, 2 attackers). After some coding, the *Commanders* dataset was readjusted, so each battle would only fill one row, with attributes specifying the commander number, and whether why were the attacker or defender. After this, the dataset was merged with the others.

2.1.1. Data Manipulation

As we will see later on, the obvious feature to predict was *wina*, a binary attribute, the outcome of the battle. This attribute had a problem though, the sheer number of NaNs, 219 out of 660 battles. Due to the importance of this feature, I've decided to not impute it, as I do with others later on, but to manually search each battle and impute myself the result (using Wikipedia). [1]

Going through the dataset, I've noticed Napoleon Bonaparte is sometimes referred to Bonaparte, and others as Napoleon I. I've decided to unify these into Napoleon. There might be other instances of other commanders that are referenced with two different names throughout the dataset, but checking every single commander and comparing it with every single other is a task too costly for the gain, since I expect that these instances are very low in number.

Lastly, I decided to add a "Missing" value to fill every space in the commander attributes that was NaN. Filling it with imputing techniques might make sense with other variables but not this one, where the variance is too high.

2.2. Attempt to Increase the Dataset

All throughout the project's realization, I was aware that the main issue of the dataset was its size. 660 battles, the length of the dataset, did not seem enough to achieve high levels of accuracy, due to the low training size the models would have at their disposal.

Due to this, after I tested my first models with the original dataset, I tried to incorporate ancient history battles into the already existing dataset, since the dataset already contained modern ones, with a scrapper.

This posed three problems. The first was that most variables of the dataset could not be retrieved with a scrapper, like *inita*, which

[1] For clarification, indecisive battles have been classified as victories for the defender.

measures the attacker's relative strength in initiative, with values from 0 to 2. This all had to be done manually, which leads to my second and third problems.

The second problem I found soon enough was that a large part of my variables were interpretative, like *inita*. This, although trained military personnel, like the ones who created this dataset, might be able to identify which value was best, I could only try to do my best given my understanding of history, which was obviously not good enough.

And the third problem was the time it took to include each battle, since it had to be done manually. I only tried adding 42 battles before I checked how the results would change, which took me about half a day.

After including 42 battles (Alexander III's and the Punic Wars, until to the Battle of Zama in 202 BC), I decided to merge this dataset with the already existing one, and try what results would the models obtain. When I saw the scores dropped by 10% for about every model I had tried, I decided to forgo my intentions on aggrandizing the dataset.

I have included the scrapper and CSV file of this attempt in the Github repository, although they have no impact on the end result of the project.

3. Preliminary Data Analysis

The dataset offers a display of variables that could be considered the target, such as the battle casualties. However, I've decided to settle with *wina*, a binary variable that predicts if the victor is the attacker (1) or the defender (0). As stated before, I've manually imputed this feature so that there are no NaNs.

Before anything else, we must check the dataset's balance, since we will have to take this into account when applying the different moethods of classification:

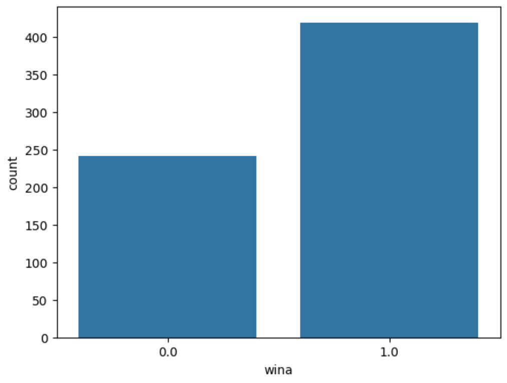

Figure 1. Our dataset is imbalanced, something to take into consideration for our models.

Part of selecting the target variable, is eliminating other possible variables that could influence our prediction when it is not wanted, since the objective is for the prediction to be made before the battle begins. Following our example, it would make no sense if battle casualties were included in our models, for it is a post-battle feature.

After this, I've also decided to eliminate the features which contain over 100 NaNs (around 15% of the dataset). The quantity of NaNs in the dataset is considerable, and I tried setting the threshold to 50 NaNs, but it yielded worse results. Even with these restrictions, the models' performance are hurt by the quantity of NaNs to fill, but I felt 100 was a good threshold to balance performance and the provided data. I also manually imputed the dataset with 0s in the variables where it meant the particular field favoured neither the attacker nor defender (in all these features, 0 was also the most repeated value by a large margin).

The dataframe is ready now to be subjected to a heatmap, looking at the correlation of each variable with the target, yielding the highest score to *inita*, the attacker's relative initiative advantage. Indeed, we can see that clearly here:

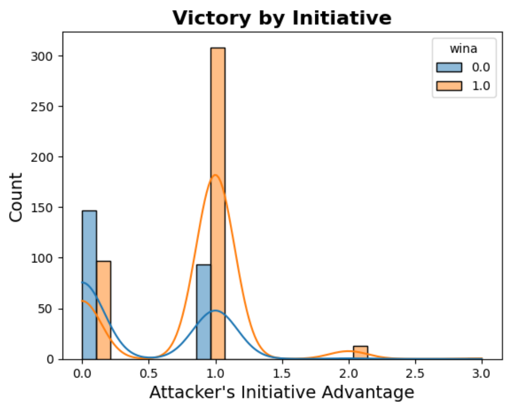

Figure 2. Initiative has a high impact on the outcome of a battle.

Another variable with a high score, 0.37, is *plana*, the attacker's relative planning advantage, something related to the commanders:

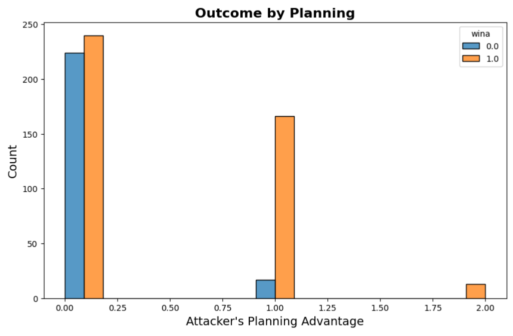

Figure 3. Commander's planning has a high degree of influence on a battle.

There are other variables that, before checking the correlations, we would think would be important for the outcome of a battle, like overall troop strength, have obtained low correlation values. Although that does not seem logical at first glance, it is true that large armies normally go to battle with other large armies, and therefore the direct correlation of the army's strength to winning is difficult to calculate. [2]

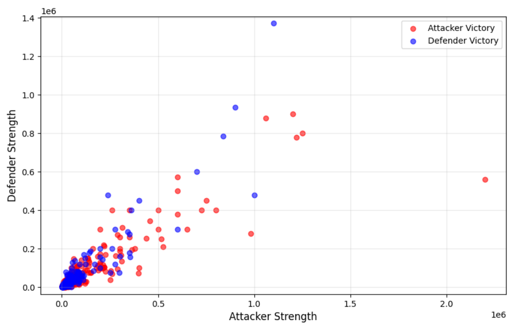

Figure 4. The battles are mostly on the diagonal, reinforcing the theory that large armies draw other large armies, reducing the effect of these variables on the outcome.

[2]It must be said that I've set a threshold of 0.1 for the correlation values. Any variable under it will be cut from the dataset.

4. Preprocessing

After a preliminary analysis of our dataset, we are finally able to process it. After separating our data into the train and test subsets, we must look into the different features and look what type they are, numerical or categorical with low (4 or less unique values) and high (more than 4 unique values) cardinality. After doing that, I have decided to use the following:

- **Numerical features**: **StandardScaler** to standardize the variables. At first I was using RobustScaler, but the latter does not work well with the SVM model we'll use later on. For imputing, I've used **SimpleImputer**, imputing the median. I've decided against KNNImputer due to the low amount of battles in the dataset. [3]
- **Categorical features with low cardinality**: **SimpleImputer**, imputing the variables with the most frequent value (for categorical variables we have only one with 2 NaNs and another with 1, so it's not a problem). After that, I've used **OneHotEncoder** to turn them into numerical (binary) variables.
- **Categorical features with high cardinality**: **TargetEncoder**, encoding the variables with its winning percentage.

To achieve all these, I've defined *Pipelines* as transformers to these variables, before combining them into one.

On top of that, I've applied a PCA to the train subset, intent on looking for a simple division between attacker and defender's victories, using only two components. However, this does not yield good results, so we can conclude that the dataset is not simple.

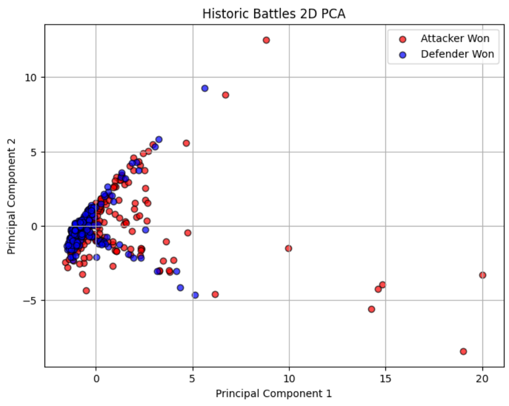

Figure 5. PCA shows the data is not distributed in a simple way.

5. Metric Selection

Before jumping to the models, we must decide which metric will we use to label their performance. We will start by fitting a Logistic Regression model to the train subset, in order to look at the values of the different metrics.

	Precision	**Recall**	**F1-Score**	**Support**
0	0.97	0.90	0.93	193
1	0.95	0.98	0.96	335
Accuracy			0.95	528
Macro Avg	0.96	0.94	0.95	528
Weighted Avg	0.95	0.95	0.95	528

With an accuracy score of 0.95.

As we can see, the results are positive, especially for class 1, since they are the majority class for the dataset. We must take into account, though, that these numbers are based on the train subset of the dataset, so for test they presumably not be nearly as good.

[3]As a side note, we must remember have also done a small imputing part above, when assigning 0's to some variables with NaNs

Since it is equally important to determine the False Positive (a predicted attacker's win when the defender actually won) as the False Negatives (the contrary), the primary metric for the model analysis will be the **F1-Score**. Because of this also, I have decided against using the PR-Curve, since it doesn't take into account the True Negatives. However, here's the **ROC Curve**:

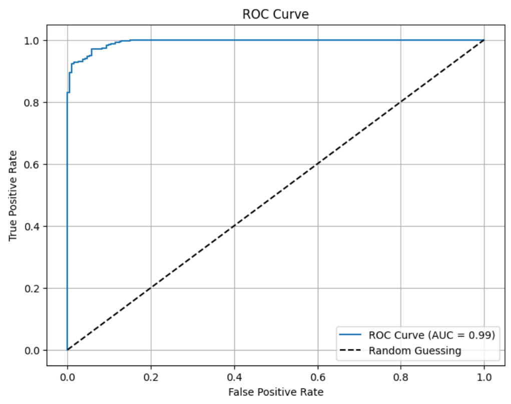

Figure 6. Logistic Regression captures clearly the distinction in battle outcomes, indicated by its high ROC-AUC score

However, we must take into account that in imbalanced datasets, the ROC Curve favours the majority class. For this reason, we can't say the AUC score is good, the only thing we can do is compare this value to other models.

6. Prediction Models

In this section, we delve into the predictive models selected for determining battle outcomes. The choice of models was driven by their ability to handle diverse data structures, interpretability, and robustness against imbalanced datasets. By leveraging both traditional and ensemble-based approaches, we aim to strike a balance between model simplicity and performance.

When deciding the models I will use, things like class imbalance, the dataset size, computational and time costs of the model must be taken into consideration. I've finally decided for the following:

- **Logistic Regression**: an easy model to both implement and interpret, Logistic Regression works well for binary datasets, without much computational or time load. However, due to its simplicity, it's a model that will likely not yield the best results, couple that with the fact that the extensive amount of features in the dataset might cause overfitting. However, I believe it's good to have it to serve as a baseline to other models.
- **Random Forest**: a robust model that works well with imbalanced datasets, it captures complex relations, avoids overfitting and deals well with high dimensionality.
- **SVM**: a model that works well with small and medium sized datasets like Historic Battles, that excels when dealing with high-dimensional data and can work well with imbalanced datasets.
- **XGBoost**: an advanced implementation of Gradient Boosting, it often provides high accuracy, excelling with imbalanced datasets. Its main downside is the computational cost and execution time, which is considerable.
- **LightGBM**: a lighter version of Gradient Boosting, it is significantly faster and less costly than XGBoost, all while retaining good performance, capturing non-linear relations.

After deciding the models I will use, I defined the range of the parameters of each model, tuning them after several executions, looking for the best scores.

To obtain the best parameters, we use **BayesSearch**. This technique improves its counterpart, GridSearch, in efficiency and computation time, since the latter tries every combination of the specified parameters, while the former uses a Bayesian optimization in order to arrive at the best solution faster. **StratifiedKFold** has also been used, both in the search for the best hyperparameters, and in the cross validation, in order to obtain better results while accounting for the class imbalance present on the dataset.

After knowing which are the best parameters for every model, we can compute the scores for the train and test subsets and compare the different models:

Model	Train F1-S.	Test F1-S.	CV F1-S.
Logistic Regression	0.943303	0.729401	0.926189
Random Forest	0.964213	0.765659	0.922248
SVM	0.947136	0.720303	0.924458
XGBoost	0.980948	0.696775	0.923351
LightGBM	0.944162	0.689297	0.915413

Out of all these, the model that fits the data best seems to be Random Forest, since it obtains the best test F1-Score out of all the models, while the cross validation and train scores are similar across all models.

However, I wanted to see if an Ensemble Model could predict the outcome better than the individual ones, mainly Random Forest, which is the model to beat. The models are:

- **Voting Classifier**: A method that adds the predictions of the individual methods to come up with a final prediction. Because not every method is as good as the other, I have assigned weights to the methods based on their individual scores, trying different ones until reaching an upper limit for the score.
- **Stacking Classifier**: A method that combines multiple machine learning models' predictions, defining each of the individual models as base models, and constructing meta-model with the predictions of the base models. As in the Voting Classifier, I've run several meta-models, before settling with the best one.

I have computed the F1-Scores for the test subset of the dataset in the following table:

Model	Test F1-Score
Voting Classifier	0.7797967
Stacking Classifier	0.7481607

So, we can see that the voting classifier obtains even better results than the Random Forest. [4]

After this, I've wanted to look at Random Forest's feature importance, to see if there were ways to improve further the models. This shows two clear dominators, the defending and attacking first commander, with an importance of 0.29 and 0.23 respectively (the next most important feature is *plana* with 0.04), showing the importance of having an able commander in a battlefield. After some tests, I decided to take out of the dataset the features with an importance value of less than 0.001. However, Random Forest with these new features only achieves a 0.774501 F1-Score, still below the Voting Classifier.

7. Final Analysis

Having selected the best model (Voting Classifier) with all the hyperparameters of the individual models chosen, we will look at several metrics to see how the model performed.

First of all, let's look at the ROC-Curve, to see if indeed it has improved in comparison to the Logistic Regression one shown earlier:

[4]I have also briefly tried with other Ensembling Methods, like Blending, but had worse results than the other two and were discarded.

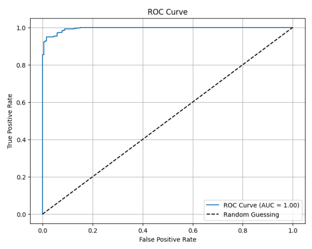

Figure 7. With an AUC of 0.995, the Voting Classifier improves the normal Logistic Regression.

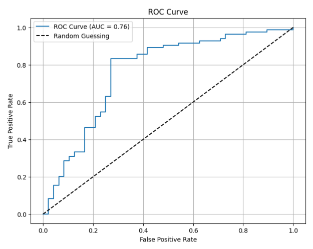

Figure 8. With an AUC of 0.763 for the test subset, the Voting Classifier improves a random classifier.

After searching the best threshold for the binary classification (0.51, which **improves the F1-Score up to 0.796**), we will look into the confusion matrix:

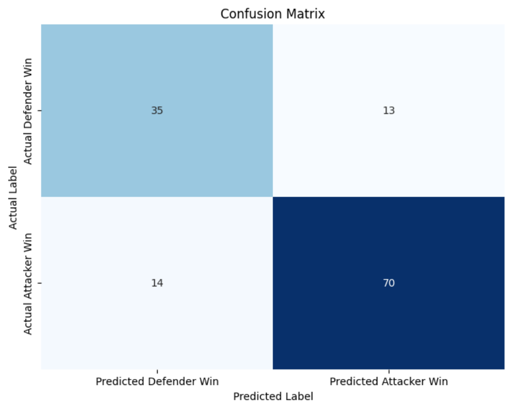

Figure 9. Most battles are predicted correctly, and the model does not favor the attacker's victory (same amount of FP as FN).

Now, let's look at the cases to see how the model behaves:

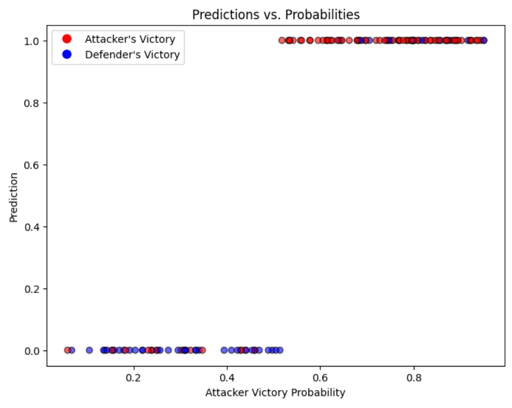

Figure 10. Most battles are predicted correctly, yet some outliers can be seen.

Checking the battles predicted as attacker victories when they were actually the defender's, one of them is the defense of Moscow in World War II. The model predicted the attacker (Germany) to win against the defender (Soviet Union). This is understandable, since, up until that point in WWII's eastern front, the Germans had not lost and seemed unstoppable. Reviewing the data from this battle, and looking at the top 10 features with most importance according to our Random Forest, the Germans had a substantial advantage in most of these features (initiative, mobility, planning) while not being at a disadvantage in any (both commanders, Von Bock and Stalin, have the same scores, while no army had any morale advantage). Other features that the model doesn't consider them important but are, like troop strength and tanks, were balanced or in favor of the Germans.

In conclusion, it is normal that the model predicted this battle as a victory for the Third Reich.

As for the battles predicted as defender's victories which were actually the attacker's, I have found defending commanders like Frederick the Great or Robert E. Lee, both great generals that have a very low score when transformed into numerical variables (as stated before, the first commanders are the features that contribute most towards the possibility). Added to that, most of these cases the features that show the advantage for the attacker (initiative, mobility ...) are 0, indicating that neither side had an edge in these fields.

Here we can see more in depth the impact some variables have in the outcome of a battle.:

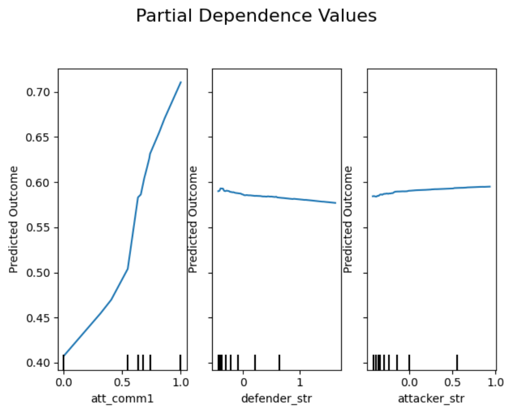

Figure 11. Variable's relation to the target.

The attacking commander, for instance, has a great impact on the outcome of a battle, as we had seen when viewing the feature importance of the Random Forest. Other variables that we would think would have more impact but do not are the total strength of each side, showing a slight trend, whether it be downwards or upwards.

8. Conclusions

All in all, I believe the Voting Classifier has achieved a respectable level of prediction, nailing 79.5% of the battles in the test subset, due to its combination of well-performing individual models. Although this might not seem much, some handicaps must be taken into consideration.

First of all, the dataset's dimensions. 660 battles constitutes a small dataset, giving the predicting models low capacity of learning. I have attempted to correct this as explained before, but it has not been possible.

Secondly, the amount of NaNs in the dataset. This forced me to delete entire columns of variables that could have been used by the models, along with imputing several others with values that in a considerable amount of battles would not be realistic. Even the target variable *wina* has been imputed by myself, personally searching every battle's outcome, and thus has a higher accuracy than the other variables that have been imputed by algorithms, probably has some mistakes I have made.

Anyways, as stated, I am satisfied with the result of more than 3 of every 4 battles correctly predicted.

In conclusion, the ability to predict battle outcomes with reasonable accuracy could offer insights into historical decision-making processes, shedding light on key factors that influenced victories and defeats. This study underlines the multifaceted nature of battles, where both quantitative (like troop strength) and qualitative (leadership) factors interact to determine outcomes. Future work could expand this research by incorporating additional battles, refining variable selection, and exploring more advanced modeling techniques such as neural networks. By doing so, we may move closer to a deeper understanding of the dynamics of warfare and the factors that determine its outcomes.