### 

### 

MachineLearning   
Project Report

*Titanic Dataset*

Team DataaCracker

17082871d LI Ruowei​

15101119d Bang Junyoung​

15100845d Hong Jiseok​

# 

# 

# 

# 

# Workload distribution

|  |  |
| --- | --- |
| Name | Contribution |
| LI Ruowei | Data preprocessing, feature engineering, modelling, discussion 1 on data leakage, final report drafting |
| Bang Junyoung | feature engineering, modelling, discussion 2 on Discrepancy between Leaderboard score and CV score, final report drafting |
| Hong Jiseok | Data visualization, PPT drafting, final report drafting |

# Introduction

On April 15, 1912, Titanic sank after colliding with an iceberg. Among the 2224 passengers onboard, only 1502 of them survived. It ranks one of the most infamous shipwrecks in history. After titanic, several safety regulations and issues and improvements concerning ship designs were brought back to surface to prevent future tragedies.

While a few causes such as bad choice of materials, shortage of lifeboats and poor designs of the water compartments have been discussed, hidden survival logic and patterns evidenced in data could lead to better understanding of the disaster and learn from it.

Kaggle, an open platform for data competitions and idea exchange among data scientists, has set up a **Titanic Challenge** for machine learning beginners to walk through the project and compete for best predictions. Following the rules and requirements of the competitions, our team would strive to get the highest test accuracy based on a set of variables describing the status of passengers abroad.

# Model results and Leaderboard

Admittedly, the leaderboard score of around 0.814, ranked 417 out of 18000.



|  |  |  |
| --- | --- | --- |
| **Classifiers** | **Accuracy with CV=10** | **Parameters after fine tuning** |
| Linear Regression | 0.8619 (+/- 0.0409) | 'C': 0.18019765146951916, 'penalty': 'l2' |
| Support vector machine | 0.8597 (+/- 0.0406) | 'C’: 1, 'decision\_function\_shape': 'ovo' |
| K nearest neighbor | 0.8508 (+/- 0.0181) | n\_neighbors: 11 |
| Random Forest | 0.8597 (+/- 0.0406) | 'criterion': 'gini', 'max\_depth': 4, 'n\_estimators': 300 |
| XGBoost | 0.8586 (+/- 0.0397) | 'learning\_rate': 0.01, 'max\_depth': 2, 'seed': 0 |
| Decision tree | 0.8676 (+/- 0.0385)\* | 'criterion': 'gini', 'max\_depth': 8, 'max\_features': 'auto', 'min\_samples\_leaf': 5, 'min\_samples\_split': 0.03 |
| Bagging random forest | **0.8608 (+/- 0.0291)\*\*** | criterion': 'gini’, ‘max\_depth': 5 |
| Neural network | 0.8530 (+/- 0.0465) | Input layer: 9, hidden 1: 8 units, hidden 2: 3 units |
| Voting (Hard) | 0.8463 (+/- 0.0970) | Hard voting |
| Voting (Soft) | 0.8429 (+/- 0.0887) | Soft voting |

\* Best validation results; \*\* Best model on leaderboard

# Problem definition

Given a series of N passenger details (x\_1, y\_1), . . . ,(x\_N, y\_N), where y\_1 ∈ {0, 1}, learn a set of **binary** classifiers that assign an effective survival label L ̂(x\_i) ∈ {0, 1} that minimizes a certain loss function ***Loss***(y\_N, L̂(x\_i) )and yields high prediction accuracy.

1. **Our objectives in this challenge are twofold:**

* Learn from disaster: achieve the highest prediction accuracy on passenger survival on test data with both classical and advanced machine learning classifiers;
* Learn from classifiers: Leverage multiple techniques to choose fine-tuned parameters, visualize and interpret classification results

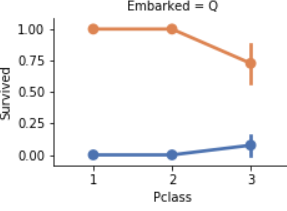
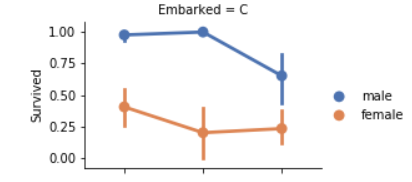
# Supplementary notes on data understanding and wrangling

There are two data sets retrieved through Kaggle Api to get train and test data sets. From the observation, data types in the data set can simply be divided into two big groups, categorical and numerical values. We need to deal with preprocessing data or feature engineering. Simply said that, we need to look into data and consider converting categorical variables into numerical values. It is because all models only use numbers to perform better when numerical input variables have a standard probability distribution. Binning is also important for developing a better model to group a number of more or less continuous values into smaller values in “bins”.

We focus on handling a large portion of missing values in Age and Cabin in common for both train and test sets. However, there was 1 missing value in the test data set where the train data does not have. Since it is a really small number of missing values, we just leave it to be bidding within “FareBin\_Code”.

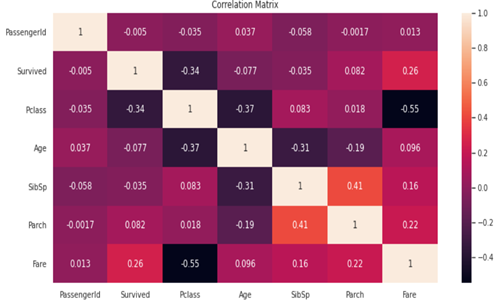
We can get essential information from alphabetical variables, but it is easy to miss it and hard to extract data. Such as the “Name” feature, it contains many valuable informations. “Name” combines information of “Sex” in detail with marital status and age. In addition, if they have the same last name, it will indicate that they are in the same family. The new feature “Title” is extracted from “Name” that each Mr and Master represents grown-up male and children.

It is said that there is no correlation between “Embarked” and “Survived” features. However, we also find other interesting points from correlating other categorical features. Females are more likely to survive rather than men, but there is an exception in Embarked C where men have a higher survival rate as shown graph below. Moreover, men also have better survival rate in Pclass3 compared to Pclass 2 for “Embarked” Q. Therefore, this unpredictable result may affect rule “female with higher survival rate” and corrupt the model caused by undesirable scores. In the end, we dropped the feature “Embarked”.



It is mentioned during the presentation that women and children are more likely to survive. We already distinguish groups of passengers traveling together whose members are either females or children. It was identified with the newly created feature “GroupIdentifier” and “TicketId”. In addition, we further do more feature engineerings with “InWcg” into “WcgAllDied” and “WcgAllSurvived”. These two features are put into values by computing the arithmetic mean of “Survived” ignoring NaNs values. Therefore, it is noted that these two features are only based on the training set only since test data does not have the feature “Survived”.

Another point that we only touched briefly is the before-,after- correlation table as shown below. It can be observed that the highest Pearson correlation coefficient changed from (-)0.55 to 0.73 and there are much more coefficients that exceed (±)0.5. This means that the feature engineering processing is successfully **amplifying** the existent data correlation.





# 

# Supplementary notes on modelling & performance

## 7.1 Ensemble methods (skipped in presentation)

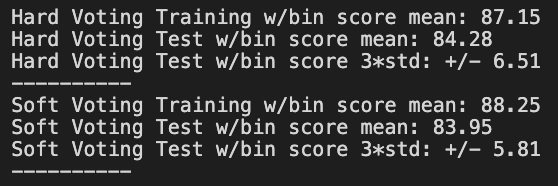
Ensembles method is a category of machine learning method that combines many base models at the same time. It can be broken up into three different sub-categories: Bagging, Stacking, Voting, and Boosting.

The bagging method uses one shallow learning model in multiple instances. It generates several subsets of given training data to train the model. Then, each subset data is trained with each instance of the model, typically a decision tree or a random forest algorithm. The method then uses the average of all models to produce output.

Boosting is another type of ensemble method. The basic idea is to sequentially train the same models but adjusting the weights on each classifier along the learning process. This process, in turn, converts weak learners into strong learners, improving accuracy and reducing bias.

In our implementation, we used Voting, Bagging and XGBoost. XGboost, which is one of the boosting algorithms, is highly scalable and has a well-written library that is easy to use and gives generally higher performance than other shallow learning algorithms. It is one of the most popular ML algorithms these days. In its learning process the weak learner, a decision tree, will gradually attempt to reduce the misclassification rate with adjusted weights.

Voting is another ensemble method, but instead of multiple instances of the same algorithm, it uses a diverse algorithm and they predict the final output by ensemble them. There are two ways to vote: Hard voting and Soft voting. Hard voting relies on a simple majority vote for accuracy (prediction rate) of each algorithm. If there are 3 different models being used in the voting method and the first two algorithms choose 'A' and the last one chooses 'B', the voting classifier returns 'A'. Soft voting calculates the probability of the outcomes and makes the best result by averaging the probabilities of each individual algorithm.



## 7.2 Scaling & Cross-Validation

After preprocessed and feature-engineered data, we scaled the dataset by StandardScaler.

Cross-validation is a necessary method to tune the machine learning model and, most importantly, to estimate the performance of the model on unseen data. Although the basic way of testing the model is to use the train\_test\_split method, CV gives the more reliable measurement of the model's performance with respect to the overfitting problem.

##### 7.3 What we are aware of: Data Leak

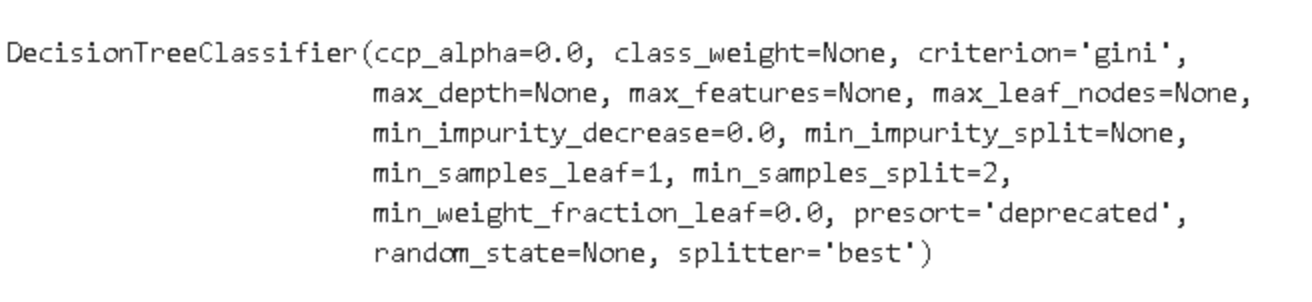
To neutralize the data leakage problem, see in Discussion 1, we made a pipeline of StandardScaler as the first operation and then a classifier. Every time the model is used in the cross-validation, the StandardScaler will be performed on each train and test data of the fold. For example, if the KFold is 3, the Scaler performs individually whenever the fold changes.

More details will be discussed in Discussion 1.

## 

## 7.4 Hyperparameter Tuning

Hyperparameters are parameters whose values are set as passed arguments before the learning process begins. For example, when we perform a decision tree with Scikit Learn, we can tune the model by specifying parameters such as max\_depth, max\_leaf-nodes, min\_samples\_split, etc. Likewise, C, Kernel, and gamma are the most important hyperparameters for the Support Vector Classifier of Scikit Learn library.



**7.5 Example of the Hyperparameters of Decision Tree**

Fine-tuning parameters of the algorithms is crucial to get the best performance out of whichever algorithms we choose. Those values are what control the learning process and ultimately derive the other parameters of the fitted model. Because it is often time-consuming to manually test the various configurations of hyperparameters, there are several methods to automatically test and find the optimal set of hyperparameters. The most typically used fine-tuning methods provided by Scikit Learn are GridSearchCV and RandomizedSearchCV.

The GridSearchCV searches for the best hyperparameters from combinations made of given lists of parameters. However, the candidate values of each parameter must be set by users. It could be inefficient if there are a lot of candidates for each hyperparameter because every possible combination would be performed. Meanwhile, the RandomizedSearchCV randomly samples the candidate values of hyperparameters from a given range and distribution specified by the user.

Both methods use the combinations of hyperparameters to train the model and score the results based on a cross-validation technique. However, the performance time of these methods can be relatively long.

Our approach extensively used both methods to find the optimal set of hyperparameters of the estimators.

## 7.6 Pipeline

Pipeline is one good way to automate the ML workflow. For example, we can streamline the preprocessing, scaling, Feature selection, Dimensionality reduction at the same time. Regarding the data leakage, this could be one solution to avoid the case of it, because on cross-validation or Grid Search we can use this pipeline to each fold of the data.

Our future implementation could use the pipeline more extensively. For example, applying the feature engineering process on each fold of the data, or using PCA on it.

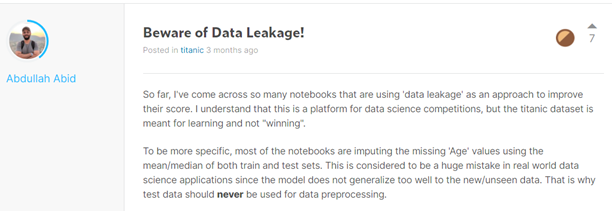
## 

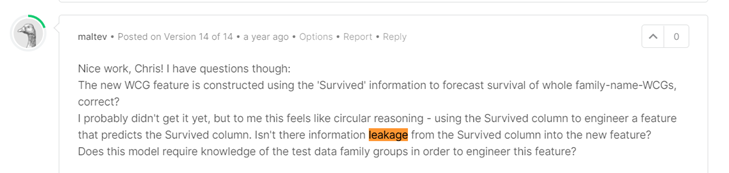
*How the pipeline are performed on steps*

# Supplementary notes on Discussion

## Discussion 1: In addition to data leakage and in response to Dr. Korris’s question

Data leakage is defined as the introduction of target information in the process of data mining which ought not to be legitimately available to mine from (Kaufman et al., 2011). The author also found significant signs of data leakage in data competition events and platforms. As we have undergone such debate in our group and seen a heated discussion on the problem of data leakage, we critically looked into the issue and gave a brief discussion during the presentation.





## In the question of “what have you observed that can lead to better performance when you submit to the platform”. This is exactly the data leakage we tried to convey in this section. In machine learning, leaking some information about the test dataset can provide some information that shouldn’t be available to the classifiers. It is believed that more data always leads to better performance (Google's Research Director [Peter Norvig](https://www.quora.com/profile/Peter-Norvig) said that “We don’t have better algorithms. We just have more data.”). This claim does not always hold in light of bias and variance, data quality and other issues but it does provide an intuition when trying to get the best result. Therefore, even if aware of this problem, we still learn from the notebooks and kernels of other peers to (partially) engineer at two features using the entire dataset. Experiment found that a marginal 2-3% accuracy can be achieved this way. In the future, we plan a statistical method (e.g. paired t-test) to better test the importance of data leakage.

## Discussion 2: Discrepancy between Leaderboard score and CV score

We've seen some gaps between cv scores of the model we trained and the Leaderboard score from the prediction of test data. There are many discussions on Kaggle about the improvement of cv score not leading to test scores.

The potential cause may be:

* A discrepancy of data in the training and test set with respect to the feature correlations. (Unseen data)​
* Cross-validation on the training dataset is still optimistic about unseen data. (Overfitting)​
* Models made additional predictions that were incorrect.​
* Or simply, not enough training Data

However, in our implementation we've seen less gap of scores when we use test data on the data preparation phase, as it is discussed on the Discussion 1.

For instance, the **best\_score\_** of our XgBoost model from the Randomized Grid Search is 0.854, while it's leaderboard score is down to around 0.75. However, the test score goes up to about 0.81 when we use the test data.

It may help us to find some more connections when we use the test data. This could lead to data leakage problems but better performance on the competition, hence supporting the idea that there are more correlations of features we may find out when both train and test data is used for the model training.



*Test score without test data*



*Test score with test data*

# Conclusion

Despite the tragedy of the sinking of the Titanic, the data we can infer from such an event can help us to get more insights to understand disasters and to prevent such occurrences in the future. Ship design standards and safety measurements have received significant attention after Titanic, indirectly contributing to the modernization of the industry.

Throughout this project, we have learned the important role of domain knowledge, EDA (data exploratory analysis), baselines used in peer’s work and the ensemble method that leverages collective wisdom (base classifiers). While intensive data preprocessing and feature engineering are done, the accuracy of machine learning algorithms, the performance is far from great and the hand-crafted feature engineering is rather labor-intensive and unscalable.

For this project, we could practice several new methods that improved the performance of our models, such as Pipeline, RandomizedGridSearch, or Voting methods. This experience led us to understand more about the workflow of the machine learning process in addition to a common mistake of data leakage that must be concerned about the case where both train and test data are given.

Furthermore, even though the change of the implementation didn't lead to notable performance improvement, the process of debating over methods and practices brought us more knowledge on handling data and applying the libraries such as Scikit Learn and Keras.

# References

•https://stackoverflow.com/questions/40569542/giving-multiple-pipelines-as-input-to-voting-classifier-sklearn

•<https://www.kaggle.com/dr1t10/surviving-the-titanic-step-by-step-with-groups/notebook>

•<https://www.kaggle.com/cdeotte/titantic-mega-model-0-84210/notebook>

•<https://www.kaggle.com/ylt0609/recursive-forward-elimination-workflow-to-0-82296/comments>

•<https://www.kaggle.com/cdeotte/titanic-wcg-xgboost-0-84688/comments>

•https://www.kaggle.com/amypeniston/titanic-simple-gender-family-group-model

•<https://www.kaggle.com/pliptor/optimal-titanic-for-gender-only-0-7655>

•https://www.kaggle.com/pliptor/how-am-i-doing-with-my-score

•Kaufman, S., Rosset, S., Perlich, C., & Stitelman, O. (2012). Leakage in data mining: Formulation, detection, and avoidance. ACM Transactions on Knowledge Discovery from Data (TKDD), 6(4), 1-21.

•<https://en.wikipedia.org/wiki/Titanic>

•https://www.britannica.com/topic/Titanic

•https://medium.com/@moritzkoerber/tune-your-preprocessing-steps-and-algorithm-selection-like-hyperparameters-c817e6572335