Titanic Survival Prediction

Lekai Song
Department of Electrical and Computer Engineering
Boston University
Allston, MA 02134
lksong@bu.edu

Jiaming Wu
Department of Electrical and Computer Engineering
Boston University
Allston, MA 02134
wu0523@bu.edu

Abstract

"Titanic: Machine Learning from Disaster" is a kernel from Kaggle which focuses on exploratory data analysis, feature engineering, and machine learning. There are many secrets to be revealed beneath the Titanic dataset. In this paper, we tried to find out some of those unknown factors that had affected the survival of passengers when the Titanic was sinking, and also to predict which passengers survived the tragedy with machine learning tools. As a result, we achieved a prediction with over 80% accuracy rate.

1 Introduction

Machine learning technology has experienced a tremendous growth since the last century. As a data analysis method that can realize the automatic analysis models building, machine learning is a branch of artificial intelligence based on the idea that the system can learn from some given data, use them to find the relationship between them, eventually identify patterns and make decisions with minimal human intervention. It is a classical problem to predict the outcome of a binary event. In layman's terms, this means, it either occurred or did not occur. For instance, one won or did not win; one passed the test or did not pass the test. Binary events create an exciting dynamic because we know, a random guess should achieve a 50% accuracy rate statistically. However, sometimes the result we predict can be too smart and even underperform a coin flip.

Let us walk through how to beat the odds. Define the problem. If data science, machine learning, big data, predictive analytics, or business intelligence is the solution, then what is the problem? In this case, the problem is to apply the tools of machine learning to predict which passengers survived the tragedy; Gather the data. We are drowning in data, yet starving for knowledge. Chances are, the datasets already exist somewhere, in some format. You have to know where to find it; Preprocess data. This step is a necessary process to turn "wild" data into "manageable" data, also "dirty data" to "clean data." It includes implementing data architectures for storage and processing, developing data governance standards for quality and control, data extraction, and data cleaning to identify aberrant, missing, or outlier data points; Perform exploratory analysis. It is vital to deploy descriptive and graphical statistics to look for potential problems, patterns, classifications, correlations, and comparisons in the dataset. Also, in order to understand and select the correct hypothesis test or data model, data categorization (i.e., qualitative vs. quantitative) is essential; Model data. Data modeling can summarize the pattern descriptively and predict future outcomes inferentially. The dataset and expected results will determine the algorithms available for use. Algorithms are just tools, so one

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
None
```

(a) Training dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
PassengerId    418 non-null int64
Pclass         418 non-null int64
Name           418 non-null object
Sex            418 non-null object
Age            332 non-null float64
SibSp          418 non-null int64
Parch          418 non-null int64
Ticket         418 non-null object
Fare           417 non-null float64
Cabin          91 non-null object
Embarked       418 non-null object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
None
```

(b) Testing dataset

Figure 1: Data overview

must still be clear how to select the right tool for the job. The same is true in data modeling. The wrong model will lead to poor performance at best and result in the wrong conclusion at worst; Validate and implement data model. After having trained a model based on a subset of data, it is time to test the model. Make sure that you have not overfitted (or under-fitted) the model, which means it only works well to the selected subset, but when it turns to another subset from the same dataset, it does not accurately fit it; Optimize and strategize. It is where to iterate back through the process to make it better, stronger, and faster than it was before.

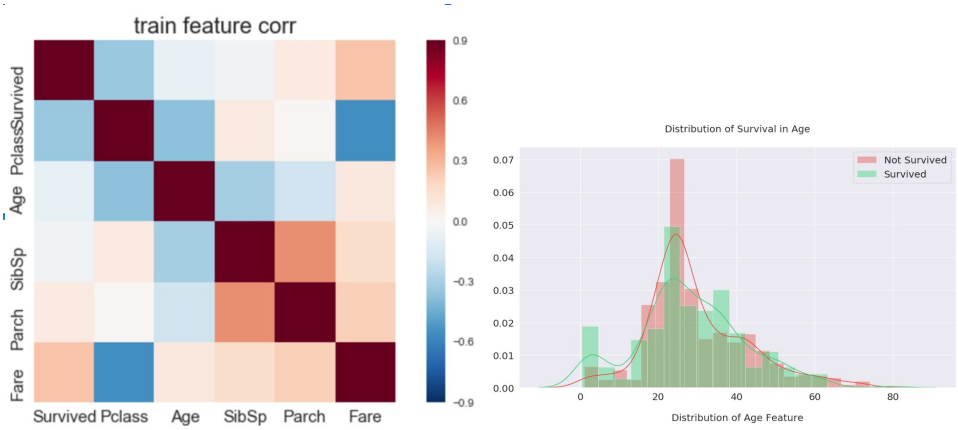
2 Workflow

2.1 Data Preprocessing

Before training models, we need to check what the dataset looks like and whether it is complete and valid, or not. We first import our data and use the info() and sample() function, to get a quick overview, shown as Figure 1. Then, we will clean dirty data by 1) correcting aberrant values and outliers, 2) completing missing information, and 3) converting fields to the correct format for calculations and presentation.

Correcting. Reviewing the data, it does not appear to be any aberrant or non-acceptable data inputs. Also, though we see there are potential outliers in age and fare, we will determine whether to include or exclude from the dataset after we complete our exploratory analysis since they look reasonable currently. Otherwise, if they are unreasonable values, for example, age is 200 instead of 20, then it is probably a safe decision to fix now. At the same time, we should understand modifying data from its original value maybe result in an inaccurate model.

Completing. There are 891 passengers on the training dataset and 418 on the testing dataset, with null or missing values in the age, cabin, and embarked features. As some algorithms do not know how to handle null values and missing values, the models will fail. Thus, it is crucial to fix before we start modeling because we will try several models and compare and contrast their performance. There are two common methods to deal with, either delete the record or populate the missing value with a reasonable input. As mentioned above, it is not recommended to delete the record, especially a large percentage of records. Instead, filling in missing values is better. A basic method for qualitative data is to fill in with the mode. A basic method for quantitative data is to fill in with the mean, median, or mean + randomized standard deviation, which depends on what feature it is. For this dataset, age will be filled in with the median, the cabin attribute will be binary classified and then dropped, and embark will be filled in with the mode.



(a) Correspondence (b) Distribution of survival in age

Figure 2: Feature engineering

Converting. Last, we will deal with formatting. Our categorical data imported as objects, which makes it difficult for mathematical calculations. For this dataset, we will convert object datatypes to categorical dummy variables.

2.2 Feature Engineering

Feature engineering is when we use existing features to create new features to determine if they provide new signals to predict our outcome. The graph shows the correspondence between every two original features. It seems that the age and number of siblings/spouses have no or weak relation to the survival rate. However, in fact, for example, from the distribution of survival in age, it is found that the feature age is not a bad factor as expected. Suppose training models with these two original features, and their contribution would be ignored, which is what we do not want to see. Thus what we can do is to create some new features from them and drop old ones. In this case, we combined the number of siblings/spouses and parents/children into a new feature named FamilySize, and grouped those whose last names are same together as FamilySurvival. For age, we split it into five parts, like 0 to 12 years old, 12 to 24, and go on. Besides, there are many unusual titles used today in names like jonkheer, Dona and Don, so we transformed them into much more common titles like Mr, Mrs, and Miss. Additionally, according to the references, the right end of Titanic sunk earlier than the left, so the cabins on the left are probably more likely to survival than that on another end. However, all of these features above is based on a guess, and nobody knows which is the most important one. We need a helper to determine if each of them played a role in survival.

2.3 Feature Selection

Random Forest Classifier is that helper to figure out what features matter. Random forest is a supervised learning algorithm. It can be used both for classification and regression. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. The random forest creates decision trees on randomly selected data samples, gets a prediction from each tree, and selects the best solution using voting. It also provides a pretty good indicator of the feature importance. Random forest uses Gini importance or mean decrease in impurity (MDI) to calculate the importance of each feature. Gini importance is also known as the total decrease in node impurity. This is how much the model fit or accuracy decreases when dropping a variable. The larger the decrease, the more significant the variable is. Here, the mean decrease is a significant parameter for variable selection. The Gini index can describe the overall explanatory power of the variables.

It works in four steps: 1) Select random samples from a given dataset. 2) Construct a decision tree for each sample and get a prediction result from each decision tree. 3) Perform a vote for each predicted result. 4) Select the prediction result with the most votes as the final prediction.

The result shows the top one is Mr, and the second is the length of the name, then FamilySurvival, gender, age, fare, and so forth. We just select the top 8 features for future model training.

(a) Random forest classifier (b) Features importance

Figure 3: Feature selection

2.4 Model training

Machine learning can be divided into supervised learning, unsupervised learning, and reinforcement learning. Supervised learning refers to the place where the model is trained by providing a training model with the correct answer to the training data set. Unsupervised learning is where the model is trained using a training data set that does not contain the correct answer. Reinforcement learning is a mixture of the first two, where the model does not immediately give the correct answer, but then strengthens the study after a series of events. We are conducting supervised machine learning because we train our algorithms by providing them with a set of features and their corresponding goals. Then we want to provide a new subset from the same data set and have similar results in terms of prediction accuracy.

There are many machine learning algorithms, but they can be classified into four categories: classification, regression, clustering, or dimensionality reduction, depending on the target variable and the data modeling goal. From previous experience, we can get those continuous target variables to require regression algorithms, while discrete target variables require classification algorithms. On the one hand, note that logistic regression, although there is a regression in the name, is a classification algorithm. Since our problem is to predict whether a passenger has survived or not survived, this is a discrete target variable. We will start our analysis using the classification algorithm in the sklearn library.

2.5 Model selection and classifier implementation

Logistic Regression Classifier:

Logistic regression is also called the generalized linear regression model, which is the same as the linear regression model. Both have ax+b, where a and b are parameters to be sought, the difference is that their dependent variables are different, and multiple linear regression is directly Ax+b is used as the dependent variable, i.e., y = ax+b, while logistic regression is used to map ax+b to a hidden state p, p = S(ax+b), and then according to p and 1-p The size determines the value of the dependent variable. The function S here is the Sigmoid function.

Support Vector Classifier:

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space, this hyperplane is a line dividing a plane into two parts where in each class lay in either side. The learning of hyperplanes in linear SVM is done by using some linear algebraic conversion problems. This is where the kernel plays the role.

For linear kernels, the equation for predicting new inputs using the dot product between input (x) and each support vector (xi) is calculated as follows:

$$f(x)= B(0)+ sum(ai *(x,xi))$$

This is an equation involving the calculation of the inner product of the new input vector (x) using all the support vectors in the training data. The coefficients B0 and ai (for each input) must be estimated from the training data by a learning algorithm. The polynomial kernel and the exponent can be written as

$$K(x,xi)= 1 + sum(x * xi)^{d}$$

$$K(x,xi)= exp(-gamma * sum((x-xi2)).$$

KneighborsClassifier:

The core idea of the kNN algorithm is that if the majority of the k most neighboring samples in a feature space belong to a certain category, the sample also belongs to this category and has the characteristics of the samples on this category. The method determines the category to which the sample to be classified belongs based on only the category of the nearest one or several samples in determining the classification decision. The kNN method is only relevant to a very small number of adjacent samples in class decision making. Since the kNN method mainly relies on the surrounding limited samples, rather than relying on the discriminant domain method to determine the category, the kNN method is more than the other methods for the crossover or overlapping sample set of the domain. To be suitable.

The main disadvantage of this algorithm in classification is that when the sample is unbalanced, such as the sample size of one class is large, and the sample size of other classes is very small; it may cause K of the sample when a new sample is an input. Samples of large-capacity classes in neighbors account for the majority. The algorithm only computes the "nearest" neighbor samples. The number of samples in a class is large, or the samples are not close to the target sample, or such samples are very close to the target sample. In any case, the quantity does not affect the results of the operation.

Another shortcoming of this method is that the amount of calculation is large, because the distance to all known samples is calculated for each text to be classified, and its K nearest neighbors can be obtained. Incomprehensible, unable to give rules like decision trees.

Decision Tree Classifier:

Decision tree algorithm is a method of approximating the value of discrete functions. It is a typical classification method, which first processes the data, uses the inductive algorithm to generate readable rules and decision trees, and then uses the decision to analyze the new data. Essentially, a decision tree is the process of classifying data through a series of rules.

The decision tree approach was first developed in the 1960s and the late 1970s. The ID3 algorithm was proposed by J Ross Quinlan, which aims to reduce the depth of the tree. However, the study of the number of leaves was ignored. The algorithm has been improved based on the ID3 algorithm. It has made great improvements to the predictive variable's missing value processing, pruning technology, and derivative rules. It is suitable for both classification and regression problems.

The decision tree algorithm constructs a decision tree to discover the classification rules implied in the data. How to construct a decision tree with high precision and small scale is the core content of the decision tree algorithm. Decision tree construction can be done in two steps. The first step is the generation of a decision tree: the process of generating a decision tree from a training sample set. In general, the training sample data set is a data set for data analysis and processing based on actual needs and a certain degree of comprehensiveness. The second step is the pruning of the decision tree: the pruning of the decision tree is the process of testing, correcting and repairing the decision tree generated in the previous stage, mainly using a new sample data set . The data validation check tree generates preliminary rules that cut off the branches that affect the accuracy of the pre-balance.

Random Forest Tree Classifier:

In machine learning, a random forest is a classifier that contains multiple decision trees, and the category of its output is determined by the mode of the category of the individual tree output. Leo Breiman and Adele Cutler developed an algorithm for deducing random forests. "Random Forests" is their trademark. The term came from a random decision forest proposed by Bin Kam Ho of Bell Labs in 1995. This method combines Breiman's' "Bootstrap aggregating" idea with Ho's "random subspace method" to build a collection of decision trees.

Gradient Boosting Classifier:

GBDT, also called MART (Multiple Additive Regression Tree), is an iterative decision tree algorithm composed of multiple decision trees. The conclusions of all trees are added up to make the final answer. It was considered together with the SVM to be a generalized algorithm when it was first proposed.

The tree in GBDT is a regression tree (not a classification tree). GBDT is used for regression prediction and can also be used for classification after adjustment.

The idea of GBDT gives it a natural advantage to discover a variety of distinguishing features and feature combinations.

2.6 Tune model with hyperparameters

In the sklearn python package, an integrated function GridSearchCV is implemented and can import directly. The GridSearchCV is used specifically to tune a model with hyperparameters. The basic idea is to set the hyperparameters as a grid and test them one by one like searching. The function will return the best parameters suitable for the model. After tuning the model, we achieved the best hyperparameters for the seven classifiers:

LR = LogisticRegression(penalty='11')

svc = SVC(C=3.0, kernel='poly')

knn = KNeighborsClassifier(nneighbors = 17)

DT = DecisionTreeClassifier(criterion='gini', maxdepth=11, maxfeatures=4, minsamplesleaf=2, minsamplessplit=14, randomstate=10)

RF = RandomForestClassifier(nestimators=410,maxdepth=6,minsamplessplit=4, minsamplesleaf=1,maxfeatures=4,oobscore=True,randomstate=10)

gbdt = GradientBoostingClassifier(learningrate=0.1, nestimators=160, maxdepth=5, minsamplesleaf=3, minsamplessplit=130, maxfeatures=2, subsample=0.8, randomstate=10)

xgb = XGBClassifier(maxdepth=3, nestimators=70, learningrate=0.1)

2.7 Analysing models using cross validation

Once we have completed the model training, we cannot assume that it handles data that we have never seen before. In other words, we are unable to determine if the model has the required accuracy and differences in the production environment. We need some assurance about the accuracy of the predictions that our model launches. To do this, we need to validate our model. The process of determining whether a numerical result of a hypothetical relationship between quantized variables is acceptable as a data description is called verification.

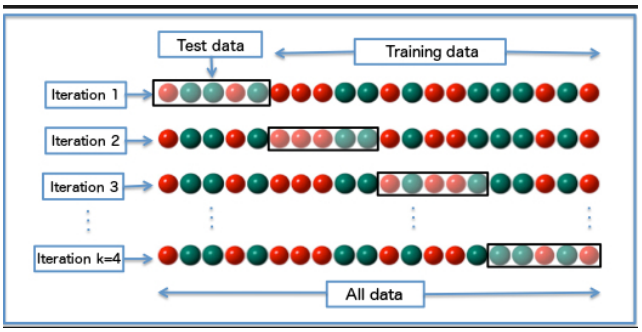
Cross-validation (CV) is one of the techniques used to test the effectiveness of machine learning models; if our data is limited, it is also used to evaluate model programs.

In order to evaluate the performance of any machine learning model, we need to test on some invisible data. Based on the performance of the model on invisible data, we can say whether our model is under-fitting/over-fitting/generalized. Cross-validation (CV) is one of the techniques used to test the effectiveness of machine learning models; if our data is limited, it is also a re-sampling procedure for evaluating models. In order to perform the traversal, we need to keep the samples/portions of the data that are not used to train the model. We will use this sample for testing and verification later.

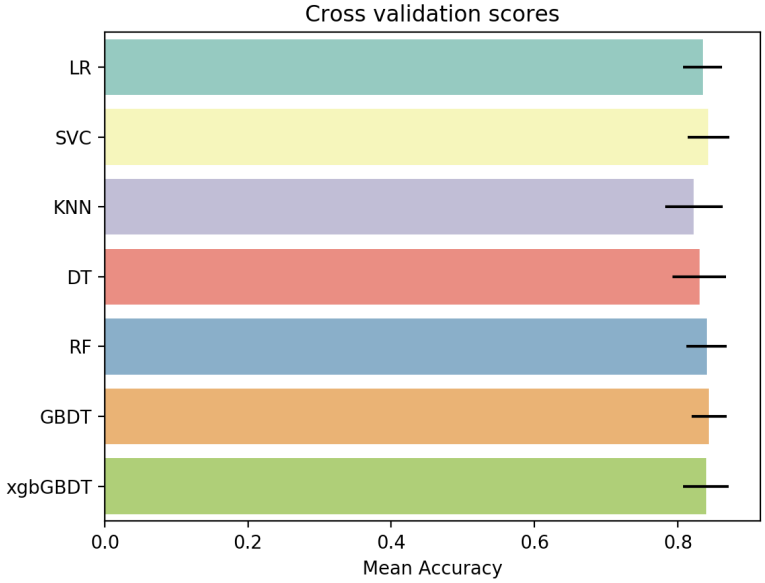
2.7.1 Train Test split approach

In this approach, we randomly split the complete data into training and test sets. Then Perform the model training on the training set and use the test set for validation purpose, ideally split the data into 70:30 or 80:20. With this approach, there is a possibility of high bias if we have limited data because we would miss some information about the data which we have not used for training. If our data is huge and our test sample and train sample has the same distribution, then this approach is acceptable.

(a) Train test split



(b) K-fold



(c) Models score

Figure 4: Cross Validation

2.7.2 K-Folds Cross Validation

K-Fold is popular and easy to understand; it generally results in a less biased model compare to other methods. Because it ensures that every observation from the original dataset has the chance of appearing in training and test set. This is one of the best approaches if we have limited input data.

Split the entire data randomly into k folds (value of k should not be too small or too high. Ideally, we choose 5 to 10 depending on the data size). The higher value of K leads to less biased model (but large variance might lead to overfitting), whereas the lower value of K is similar to the train-test split approach we saw before. Then fit the model using the K—1 (K minus 1) folds and validate the model using the remaining Kth fold. Note down the scores/errors. Repeat this process until every K-fold serve as the test set. Then take the average of your recorded scores. That will be the performance metric for the model.

2.8 Ensemble learning

Ensemble learning is a machine learning paradigm where multiple learners are trained to solve the same problem. In contrast to ordinary machine learning approaches which try to learn one hypothesis from training data, ensemble methods try to construct a set of hypotheses and combine them to use.

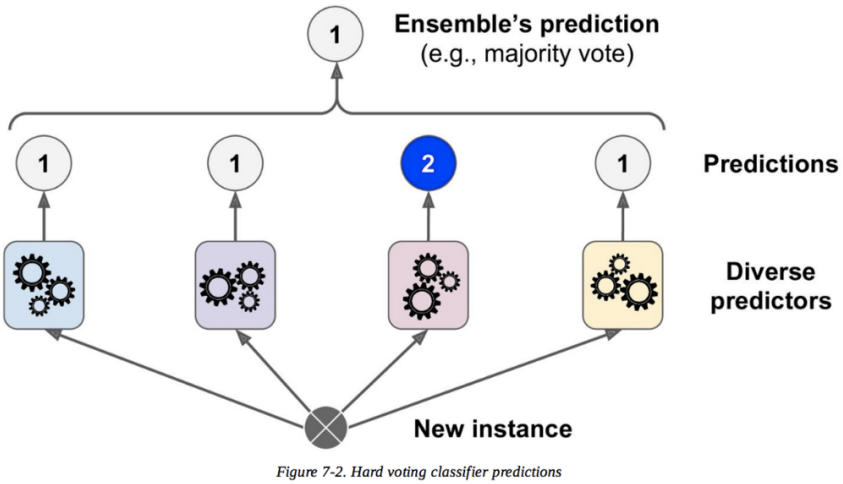


Figure 7-2. Hard voting classifier predictions

Figure 5: Ensemble learning

To implementing this, we import the sklearn bagging package to approach a bagging classifier. A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregates their predictions to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.

This algorithm encompasses several works from the literature. When random subsets of the dataset are drawn as random subsets of the samples, then this algorithm is known as Pasting. If samples are drawn with replacement, then the method is known as Bagging. When random subsets of the dataset are drawn as random subsets of the features, then the method is known as Random Subspaces. Finally, when base estimators are built on subsets of both samples and features, then the method is known as Random Patches.

3 Result

With the ensemble learning model classifier, we combine seven models to a final classifier with more than 80% accuracy and run the prediction list on Kaggle to achieve the above results.

References

[1] Chen, Tianqi; Guestrin, Carlos (2016). "XGBoost: A Scalable Tree Boosting System". In Krishnapuram, Balaji; Shah, Mohak; Smola, Alexander J.; Aggarwal, Charu C.; Shen, Dou; Rastogi, Rajeev (eds.). Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016. ACM. pp. 785–794. arXiv:1603.02754. doi:10.1145/2939672.2939785

[2] Kecman, Vojislav; Learning and Soft Computing — Support Vector Machines, Neural Networks, Fuzzy Logic Systems, The MIT Press, Cambridge, MA, 2001

[3] Zhi-Hua Zhou Ensemble Learning National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China