

# Investigating the Sinking of the Titanic by Using Various Machine Learning Algorithms

Philip Hartout, Vincent Roest & Bas Straathof

Amsterdam University College

December 19, 2016

## Introduction

The transatlantic passenger liner RMS Titanic sank during her maiden voyage on April 15, 1912 due to a collision with an iceberg. The RMS Titanic disaster is one of the most infamous shipwrecks in history, since 1,502 out of the 2,224 people on board could not be saved. In this study, the chances of survival of a passenger on the RMS Titanic using a labeled data set of 891 passengers are predicted with the use of different classifiers.

The data set was obtained from the Kaggle competition *Titanic: Machine Learning From Disaster*, and contains 10 features: passenger class, name, sex, age, number of spouses and siblings on board, number of parents on board, ticket number, fare cost, cabin number, and location of embarkation. In this paper, predictions are made using three classifiers from the `scikit-learn` Python library: a Decision Tree classifier, a Random Forest classifier, and a Support Vector Machine classifier. The `GridSearchCV()` package in the `scikit-learn` library is used to find the optimal estimators for all three classifiers. The purpose of this research paper is to see how well these classifiers can predict if a person would survive the RMS Titanic disaster or not, based on the set of features in the Kaggle data set. The performance and results of the classifiers are assessed through an analysis of their learning curves, performance curves and confusion matrices.

## 1. Pre-processing the data

### a. Missing and irrelevant data

Of the 891 passengers in the data set, there are 177 for which the age has not been specified. The mean age of the 714 passengers for which the age is specified is computed and this value (29.7 y.) is inserted in all the places where the age of a passenger is missing. Nevertheless, the statistical variance of the age feature vector in the data set has decreased because of this, which might still result in skewing. A more appropriate approach would be to generate data from a Gaussian distribution with mean 29.7; however, this would result in more noise in the data which is why we opted for the former option.

Additionally, numerous cabin numbers are missing, so this feature is discarded. Moreover, the features 'passenger name' and 'ticket number' should also be discarded because they do not contain any classifiable information that would significantly change the results. The only useful information that a name could comprise is information about family relations, which is stored in two separate features: `SibSp`, which denotes the number of siblings and spouses of a given passenger aboard, and `Parch`, which contains the number of parents and children of a passenger aboard. The rest of the features in the data set do not contain irrelevant or missing data.

The ratio of people that survived the RMS Titanic disaster (staff included) is approximately 32%. It is calculated that the survival ratio of the passengers in this data set is approximately 38.4%. Hence, we can consider the data to be

a good representation of the actual survival ratio.

## b. Data-types

All data should be numeric for classification, and this is achieved as follows. There were three locations of embarkation for the RMS Titanic: Southampton, Great-Britain; Queenstown (now Cobh), Ireland; Cherbourg, France. In the original data set, the place of embarkation is denoted with a string 'S', 'Q', or 'C', in a single feature vector – Embarked. This feature vector is discarded after having split it into three separate features: 'Embarked\_S', 'Embarked\_Q' and 'Embarked\_C'. For any passenger, 1 is inserted in the feature vector corresponding to the place of embarkation of the passenger, and 0 is inserted in the two remaining feature vectors.

In the Sex feature vector, gender is indicated by a string 'female' or 'male', which is mapped to 0 and 1, respectively, in a new feature vector named Gender. The original feature vector – Sex – is discarded from the data set.

There are now 9 numeric features in the data set that do not contain any missing data: Pclass (the passenger class), Age, SibSp, Parch, Fare, Embarked\_C, Embarked\_Q, Embarked\_S and Gender.

## c. Data handling

Since a data set of 891 learning examples is not extensive, a 80-20 split of the data is chosen, which can be specified using the `train_test_split()` function of `scikit-learn`'s `cross_validation` package. Furthermore, 7-fold cross-validation split is used to let `GridSearchCV()` find the best estimator for all four classifiers, based on a comparison on scores on the training set and a mean over the scores on the cross-validation set. Please note that because of the way in which the  $k$ -fold cross-validation algorithm is implemented in `scikit-learn`, the results evaluated in this paper are not exactly reproducible. Rerunning the code will

yield different results because the 7 cross-validation regimes will be chosen differently. Even though the mean scores and standard deviation over the folds are used as accuracy measures on the cross-validation set for all four classifiers discussed in this paper, changes between scores can still be rather big. This is due to the small size of the data set. This hurdle could be overcome in future research by predefining the cross-validation splits, evading the built-in `scikit-learn` function. Additionally, the fact that the decision tree and random forest algorithms are greedy may change the results when rerunning the code.

# 2. Decision Trees

## a. Parameters

The function `GridSearchCV()` is used to find the optimal estimators of all classifiers discussed in this paper. The `scikit-learn` decision tree classifier that is used in this study to predict survival or loss during the RMS Titanic disaster is `DecisionTreeClassifier()`. Also, a parameter space consisting of a handful of parameters, in which `GridSearchCV()` is allowed to search for optimal parameters, is constructed. The first parameter is the maximum number of features allowed – `max_features` –, which can be either the real number of features, or a square root or logarithm with base two amount of the features. The square root and the logarithm with base two are used to prevent a high variance problem. In order to do pruning, the parameter space for the maximum depth of the tree<sup>1</sup> – `max_depth` – contains integers between 5 and 9. Moreover, `GridSearchCV()` is allowed to look for a minimum number of samples to split on per decision – `min_samples_split` – between 1 and 8, and a minimum number of leaf nodes – `min_samples_leaf` – between 1 and 7. An additional feature of the `DecisionTreeClassifier()` is that it is able to presort features, so `GridSearchCV()`

<sup>1</sup>A decision tree can never be deeper than the amount of features that is given as input.

should also check if this improves classification. `GridSearchCV()` then finds the best estimator by calculating all possible combinations of the aforementioned parameters specified in the parameter space.

## b. Results

The best decision tree estimator for the Titanic survival classification problem that was found has the following parameters: `max_depth = 6`, `max_features = 'None'` (the actual number of features), `min_samples_leaf = 5`, `min_samples_split = 3` and by presorting the data. This best found estimator classified the passengers in the training set with an accuracy of 85.53%, and the mean accuracy on the 7 cross-validation folds was 81.74%, with a standard deviation of 1.1%. The accuracy on the test set was higher, namely 86.59%. The optimal decision tree estimator can be visualized by reading an compiled .dot file with the Graphviz software. A snippet of the visualized decision tree can be seen in Figure 1.

The `DecisionTreeClassifier()` of the `scikit-learn` library has the attribute `feature_importances_`, which returns normalized importance scores of the features. For the best found estimator, these normalized scores, approximated to three decimals and from high to low, are: `Gender = 0.514%`; `Fare = 0.146`; `Pclass = 0.164`; `Age = 0.106`; `SibSp = 0.059`; `Embarked_C = 0.008`; `Parch = 0.001`; `Embarked_Q = 0.000`; `Embarked_S = 0.000`.

## c. Error analysis

The confusion matrix of the best found decision tree estimator can be seen in Table 1. In the confusion matrices, a 0 denotes that a passenger survived the shipwreck, and a 1 denotes that a passenger did not survive the disaster. True Positive is defined if both the predicted and actual class was 0; thus, if it was predicted that a passenger would survive the disaster, and he or she did survive in reality as well<sup>2</sup>.

<sup>2</sup>This deviation to not define true positive as 'predicted: 1, and actual: 1' is chosen because the 'survived' label is

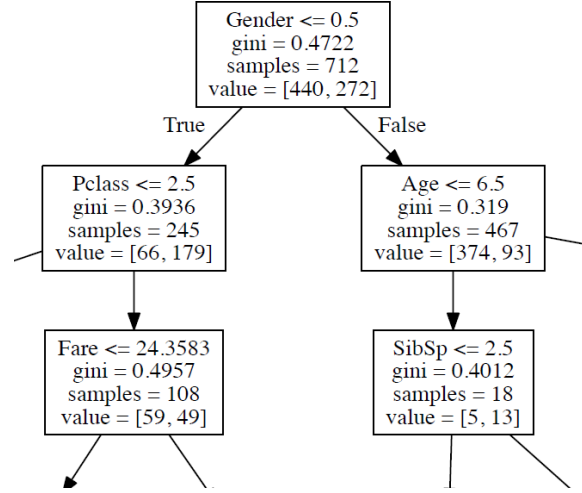


Figure 1: Decision Tree visualization

From the data in Table 1, the precision, recall and F1-score are computed and displayed in Table 2. The precision score is only slightly higher than the recall score, and the F1-score of this estimator is 0.80, which is quite high. This implies that both the the ratio of passengers that this estimator predicts to survive the disaster over the passengers that actually did survive the shipwreck is high, and the best found estimator is good at filtering out the people that survived.

The learning curve of the best found estimator on the training and (different!) 7-fold cross-validation sets is depicted in Figure 2. There is a gap between the final mean scores on the training and cross-validation set, but it does not seem too big or too small, which gives the impression that the variance-bias balance is about right.

		Actual	
		0	1
Predicted	0	TP: 48	FP: 11
	1	FN: 13	TN: 107

Table 1: Confusion matrix of the best found Decision Tree

denoted by a 0 in the data set.

	Precision	Recall	F1-score
Survived	0.81	0.79	0.80

**Table 2:** Precision Recall matrix of the best found Decision Tree



**Figure 2:** Learning curve of a Decision Tree

### 3. Random Forest

#### a. Parameters

Again, the function `GridSearchCV()` is used to find the optimal estimators; this time using `scikit-learn`'s `RandomForestClassifier()`. The parameter space of this classifier is specified as: the number of estimators – `n_estimators` – equal to 300, the maximum depth of the tree – `max_depth` – between integers 5 and 9, between 1 and 9 minimum training samples to split on – `min_samples_split`, and the minimum samples to constitute a leaf – `min_samples_leaf` – between 1 and 7. The `oob_score`, which can be set to either `True` or `False` to see if using out-of-bag samples to estimate the generalization accuracy was found to not yield a better estimator on checking with 10 estimators, so this is put to `False`. The `RandomForrestClassifier()` has no attribute to presort features. It should be noted that the accuracy increases as the number of trees increases, although the incremental gain becomes smaller and smaller. Ultimately, the computational cost outweighs the gain in accu-

racy, since an asymptote is reached. Training the algorithm on 300 estimators seems to be about the maximum capacity of our computers within a reasonable amount of time ( $\pm 30$  minutes).

#### b. Results

The parameters of the best found random forest estimator are: `max_depth = 8`, `max_features = 'auto'` (the actual number of features), `min_samples_leaf = 2` and `min_samples_split = 1`. The score on the training and test set are 89.75% and 86.59%, respectively, and the mean over the folds is 82.44%, with a standard deviation of approximately 1.2%.

The `RandomForestClassifier()` of the `scikit-learn` library has the attribute `feature_importances_` as well, and the normalized scores for the best found estimator are: `Gender = 0.380%`; `Fare = 0.215`; `Age = 0.162`; `Pclass = 0.116`; `SibSp = 0.051`; `Parch = 0.038`; `Embarked_C = 0.017`; `Embarked_S = 0.016`; `Embarked_Q = 0.006`. In the section 'Comparative analysis', the scores of the three different classifiers will be compared and evaluated.

#### c. Error analysis

The confusion matrix of the best found random forest estimator can be seen in Table 3, and in Table 4 the precision, recall and F1-score. The precision score is higher than the recall score, which implies that the the ratio of passengers that this estimator predicts to survive the disaster over the passengers that actually did survive the shipwreck is big, but the estimator is less equipped to filter out all people that survived. Nevertheless, the F1-score is 0.80, just as it is for the best found decision tree estimator. The learning curves of the best found estimator on the training and 7-fold cross-validation sets are depicted in Figure 3. It is evident from this figure that the gap between the final accuracy of the training set and the final mean accuracy of the cross-validation set is rather substantial, which implies that the

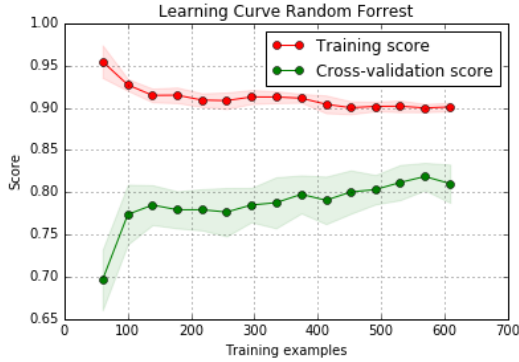
variance is high. As previously stated, the data set is quite small. If there would be a way to gather more data, the gap would probably decrease, implying that the estimator would perform better.

		Actual	
		0	1
Predicted	0	TP: 47	FP: 10
	1	FN: 14	TN: 108

**Table 3:** Confusion matrix of the best found Random Forrest

Survived	Precision	Recall	F1-score
	0.82	0.77	0.80

**Table 4:** Precision Recall matrix of the best found Random Forrest



**Figure 3:** Learning curve of the best found Random Forrest

#### 4. Random Forest without location of embarkation

We hypothesized that the location of embarkation would not have a great effect on the performance of the estimator. Hence, we ran `GridSearchCV()` on `RandomForestClassifier()` with the exact

same parameters, with the three features corresponding the location of embarkation dropped.

##### a. Results

The parameters of the best found random forest estimator with the location of embarkation dropped are: `max_depth = 7`, `max_features = 'auto'` (the actual number of features), `min_samples_leaf = 2` and `min_samples_split = 6`. The score on the training and test set are 90.03% and 84.36%, respectively, and the mean over the folds is 82.58%, with a standard deviation of approximately 4.1%.

Furthermore, the normalized scores for the best found estimator are: Gender = 0.348%; Fare = 0.251; Age = 0.206; Pclass = 0.111; SibSp = 0.050; Parch = 0.033. The best estimator with the location of embarkation dropped scores lower on all data sets, for this particular run, and also over a series of runs prior to the one described in this paper.

##### b. Error analysis

The recall, precision and F1-score are calculated from the confusion matrix of the best found random forest estimator (Table 5) and are depicted in Table 6. In our results, the recall and precision scores are equal if the locations of embarkation are dropped, but the F1-score is lower than with the place of embarkation feature maintained: 0.77.

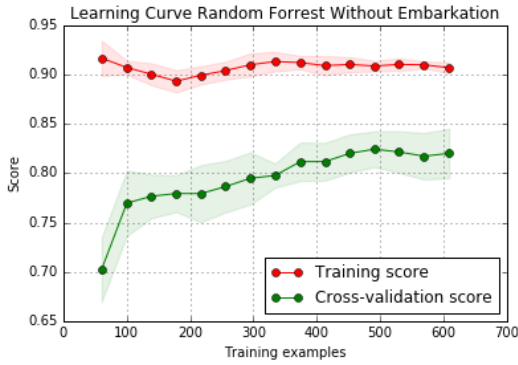
An interesting – and not intuitive – fact is that the gap between the final accuracy on the training set and the means over the folds is a little bigger, see Figure 4, implying that the variance is higher when the location of embarkation is dropped. However, through dropping features one would expect that the bias becomes bigger. It is due to the size of the data set that substantial changes can occur in the learning curves, which is most probably the case why we see this result. All in all, it does not seem favorable to drop the features corresponding to the location of embarkation.

		Actual	
		0	1
Predicted	0	TP: 47	FP: 14
	1	FN: 14	TN: 104

**Table 5:** Confusion matrix of the best found Random Forrest without place of embarkation

	Precision	Recall	F1-score
Survived	0.77	0.77	0.77

**Table 6:** Precision Recall matrix of the best found Random Forest without embarkation



**Figure 4:** Learning curve of the best found Random Forrest without embarkation

## 5. Support Vector Machine

### a. Parameters

Besides the function `GridSearchCV()`, scikit-learn's `Pipeline` package is used to find the optimal Support Vector Machine estimator. The pipeline is used to apply a list of transforms in sequence. In this case, the `SVC()` (Support Vector Classification) is used, as well as `StandardScaler()` - a function that standardizes features by removing the mean and scaling to unit variance.

The parameter space of the `SVC()` classifier is specified as: the penalty parameter of the error term  $C = 1, 5, 10$  or  $100$ ; a gamma of  $0.01, 0.05, 0.1, \frac{1}{891}$  or  $0.2$ ; four different ker-

nels: `rbf` (which stands for "Radial Basis Function"), `linear`, `sigmoid`, and `poly`; the degree of the polynomial kernel - `degree`, which is either  $2$  or  $3$ ; the shape of the decision function `decision_function_shape: ovo` (one-vs-all) or `ovr` (one-vs-rest).

### b. Results

The parameters of the best found support vector machine estimator are: `gamma = 0.06`, `degree = 3` (n.a.), `kernel = 'rbf'`, `C = 5` and `decision_function_shape = 'ovr'`.

The score on the training and test set are  $83.15\%$  and  $86.03\%$ , respectively, and the mean over the folds is  $82.44\%$ , with a standard deviation of approximately  $2.0\%$ . Unfortunately there is no attribute to determine the significance of the features for a support vector machine estimator in scikit-learn.

### c. Error analysis

The confusion matrix of the best found support vector machine estimator can be seen in Table 7, and the precision, recall and F1-score in Table 8. For this SVM estimator, the precision and recall are not significantly different and the F1-score is good:  $0.79$ .

From the learning curve of the best found SVM estimator (see Figure 5) can be seen that the gap between the final accuracy on the training set and the accuracy of the mean over the folds is rather small, so there is no immediate need for more training samples.

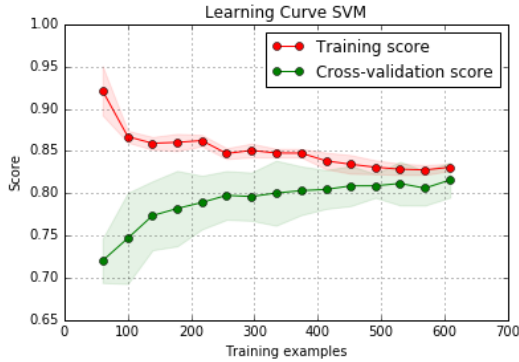
		Actual	
		0	1
Predicted	0	TP: 48	FP: 12
	1	FN: 13	TN: 106

**Table 7:** Confusion matrix of the best found Support Vector Machine



	Precision	Recall	F1-score
Survived	0.80	0.79	0.79

**Table 8:** Precision Recall matrix of the best found Support Vector Machine



**Figure 5:** Learning curve of the best found Support Vector Machine

## 6. Comparative analysis and discussion

A myriad of interesting observations have been done. The performance of the various classifiers can be compared using their ROC-curves (see Figure 6). The decision tree, random forest and support vector machine estimators perform comparably well, and dropping the place of embarkation feature seems to decrease the score, albeit not substantial nor significant.

Moreover, it can be deduced from Figures 3,4 that both random forest estimators have relatively high variance, looking at the gap between final scores on the cross-validation and training sets. Therefore, increasing the number of training example will likely enhance the performance of a random forest classifier significantly. In contrast, in Figures 2,5 can be seen that the decision tree and support vector machine estimator have a variance-bias balance that seem about right (mind that the y-axis scales are not similar, which could set about a false first impression).

In general, it seems best to maintain the place of embarkation feature and the decision tree,

random forest, and support vector machine yield similar result. The data set is sufficiently small to yield slightly varying results over multiple runs, but the scores of the distinct classifiers were always comparably close over a series of runs that were executed during this research (which are not all discussed in this paper). We anticipated that the random forest classifier would outperform the decision tree classifier, but this is not necessarily the case (both scored 86.59% on the test set). Please refer to table Table 9 for an overview of the accuracies of the classifiers. Regarding the greediness of both the random forest and the decision tree algorithms, variety in the scores can be found upon multiple runs, however, the differences seem insignificant.

	Training set	CV sets	Test set
DT	85.53	81.74	86.59
RF	89.75	82.44	86.59
RFE	90.03	82.58	84.36
SVM	83.15	82.44	86.03

**Table 9:** Accuracies in % of the Decision Tree (DT), Random Forest (RF), RF without embarkation (RFE), and Support Vector Machine (SVM) classifiers.

## 7. Conclusion

Overall, all classifiers (decision tree, random forest and support vector machine) would be good choices to predict the chances of survival of individual passengers on the titanic.

Regarding the smallness of the data set, it seems that there is most room for improvement for the random forest classifier if it would be possible to obtain more data, since the variance seems to be high. For further research it is recommendable to find a way to create the same k-fold cross-validation sets for all the classifiers such that scores on the cross-validation sets can be compared more rigorously. An other inter-

esting classifier to predict survival during the disaster could be the  $k$ -nearest-neighbor classifier, especially if more data were obtainable. It was anticipated that regarding the small size of this data set, it would not have been useful to apply this method, for it would have been prone to strong over-fitting.

Furthermore, the results of the decision tree and random forest estimators show that of the 9 features, the most important feature for predicting chances of survival on the RMS Titanic is gender - by far. The mean normalized importance of the Gender feature for the best decision tree and random forest estimators is 0.447, followed by Fare, Age, and Pclass - which are proxies for socio-economic status, femininity and children- with respective scores of 0.181, 0.140 and 0.134.

Lost passengers and survivors of the Titanic can be predicted way better than we hypothesized, and all three classifiers that we tried yielded similar results - which is remarkable because of the seemingly categorical nature of the data set.

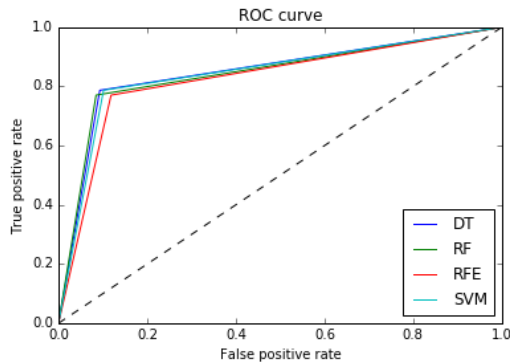


Figure 6: ROC-curves

### Link to the GitHub repository

You will find the experimental data as well as the implementations of the learning algorithm by accessing the following GitHub repository: <https://github.com/BasStraathof/ML-project-titanic>