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

Philip Hartout, Vincent Roest & Bas Straathof

Amsterdam University College

December 15, 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 Forrest 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 the this 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.

1. 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 [1] –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() 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. 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 based on a comparison on scores on the training set and a mean over the scores on the 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. Due to the small size of the data set, changes between scores will be rather big. This hurdle could be overcome in future research by predefining the cross-validation splits. Additionally, the fact that the decision tree algorithm is greedy may change the results when rerunning the code.

[1] A decision tree can never be deeper than the amount of features that is given as input.

c. Results

The best decision tree estimator for the Titanic survival classification problem that was found has the following parameters: `max_depth = 7`, `max_features = 'None'` (the actual number of features), `min_samples_leaf = 1`, `min_samples_split = 4` and by presorting the data. This best found estimator classified the passengers in the training set with an accuracy of 87.64%, and the mean accuracy on the 7 cross-validation folds was 83.42%, with a standard deviation of 2.5%. The accuracy on the test set was slightly higher, namely 85.47%. 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` = 0.434%; `Pclass` = 0.190; `Fare` = 0.187; `Age` = 0.115; `SibSp` = 0.058; `Parch` = 0.000; `Embarked_S` = 0.009; `Embarked_C` = 0.006; `Embarked_Q` = 0.000.

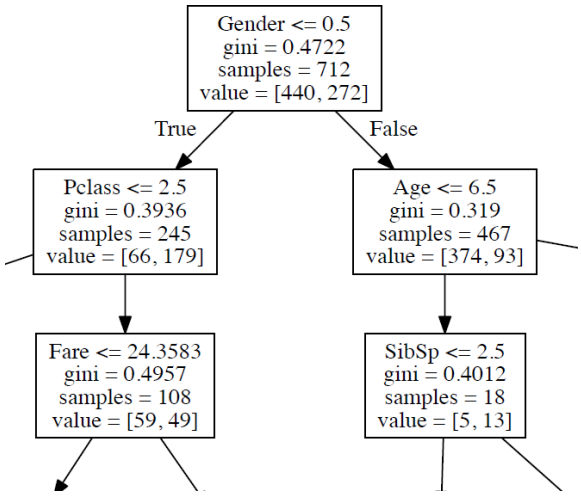

Figure 1: *Decision Tree visualization*

d. 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[2].

From the data in Table 1, the precision, recall and F1-score are computed and displayed in Table 2. 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, but the estimator is less equipped to filter out all people that survived. The F1-score of this estimator is 0.79, which is quite high.

The learning curve of the best found estimator on the training data 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: 50	FP: 6
	1	FN: 20	TN: 103

Table 1: *Confusion matrix of the best found Decision Tree*

3. Random Forest

a. Parameters

Again, the function `GridSearchCV()` is used to find the optimal estimators; this time using

[2]This deviation to not define true positive as 'predicted: 1, and actual: 1' is chosen because the 'survived' label is denoted by a 0 in the data set.

	Precision	Recall	F1-score
Survived	0.81	0.71	0.79

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

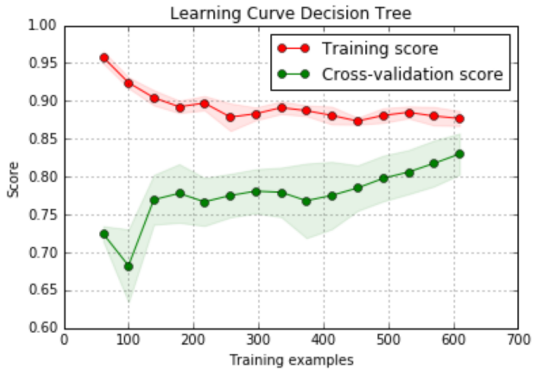

Figure 2: *Learning curve of a Decision Tree*

`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 accuracy, 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. Data handling

The data set is split up in an 80% training and 20% test set, again, and a 7-fold cross-validation split of the training set is used.

c. Results

The parameters of the best found random forest estimator are: `max_depth = 9`, `max_features = 'auto'` (the actual number of features), `min_samples_leaf = 2` and `min_samples_split = 7`. The score on the training and test set are 90.31% and 83.24%, respectively, and the mean over the folds is 82.16%, with a standard deviation of approximately 3.6%.

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` = 0.366%; `Fare` = 0.223; `Age` = 0.159; `Pclass` = 0.121; `SibSp` = 0.055; `Parch` = 0.035; `Embarked_C` = 0.019; `Embarked_S` = 0.015; `Embarked_Q` = 0.007. In the section 'Comparative analysis', the scores of the three different classifiers will be compared and evaluated.

d. 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 recall score is slightly lower than the precision score, but the difference seems insignificant if the size of the data set is taken into account. The F1-score is 0.79, just as it is for the best found decision tree estimator. The learning curve of the best found estimator on the training data and different 7-fold cross-validation sets is 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: 58	FP: 13
	1	FN: 17	TN: 91

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

	Precision	Recall	F1-score
Survived	0.82	0.77	0.79

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 88.62% and 82.68%, respectively, and the mean over the folds is 84.13%, with a standard deviation of approximately 4.1%.
Furthermore, the normalized scores for the best found estimator are: `Gender` = 0.398%; `Fare` = 0.223; `Age` = 0.148; `Pclass` = 0.141; `SibSp` = 0.050; `Parch` = 0.040. 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. The difference between recall and precision is larger if the locations of embarkation are dropped, which implies that this estimator is not well-equipped to filter out all people that survived the shipwreck. Due to this, the F1-score is significantly lower (0.76).

An interesting – and intuitive – fact is that the gap between the final accuracy on the training set and the means over the folds is significantly smaller, see Figure 4, implying that the bias is higher when the location of embarkation is dropped. However, due to the size of the data set, substantial changes can occur in the learning curves, which could potentially overshadow this effect. All in all, it does not seem favorable to drop the features corresponding to the location of embarkation.

		Actual	
		0	1
Predicted	0	TP: 50	FP: 8
	1	FN: 23	TN: 98

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

	Precision	Recall	F1-score
Survived	0.86	0.68	0.76

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

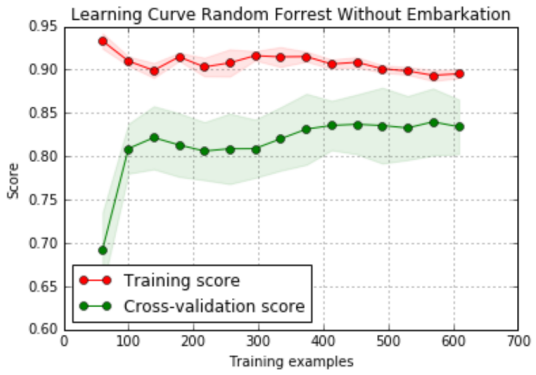

Figure 4: *Learning curve of the best found Random Forrest without embarkation*

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 `StandardScalar()` - 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 kernels: `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. Data handling

The split of the data set is again a 80-20 ratio and 7-fold cross-validation is used.

c. Results

The parameters of the best found support vector machine estimator are: `gamma = 'auto'` ($\frac{1}{891}$, `degree = 3` (n.a.), `kernel = 'rbf'`, `C = 5` and `decision_function_shape = 'ovr'`.

The score on the training and test set are 85.39% and 82.12%, respectively, and the mean over the folds is 83.3%, with a standard deviation of approximately 5.3%. Unfortunately there is no attribute to determine the significance of the features for a support vector machine estimator in scikit-learn.

d. 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 recall is significantly lower than the precision, but the overall F1-score is not too bad (0.76).

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.

6. Comparative analysis and discussion

It was late noticed that a thorough comparison of the best found estimators of the three different classifiers is difficult to do because the data has been split into an 80% training and 20%

		Actual	
		0	1
Predicted	0	TP: 51	FP: 9
	1	FN: 23	TN: 96

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

	Precision	Recall	F1-score
Survived	0.85	0.69	0.76

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

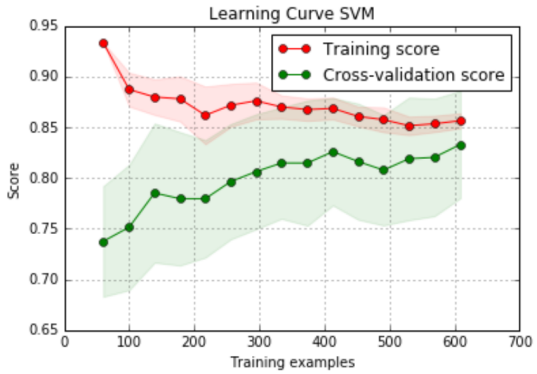

Figure 5: *Learning curve of the best found Support Vector Machine*

test set for each classifier separately. Since the `scikit-learn` function `train_teset_split()` splits the data set randomly, each classifier handles a different training and test set. Assuming homogeneous distribution of the data, this would not be a concern for a big data set. However, in this case, the comparative analysis will lose rigor due to this fact. Besides, it might have been a wiser idea to not divide the data set into a training and test set at all, but to just divide it into k-fold cross-validation sets and do a comparative analysis of the mean accuracies of the folds. Due to time constraints, this has not been implemented, but it is recommended for future implementations of classification algorithms.

Regardless, 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 and random forest estimators slightly outperform the support vector machine and the random forest without location of embarkation features, but the difference is not substantial nor significant.

Moreover, it can be deduced from Figures 2,4,5 that the decision tree, the random forest without location of embarkation, and the SVM estimators have relatively high bias. Therefore, increasing the number of training example will not necessarily enhance their performance significantly. In contrast, in Figure 4 can be seen that the random forest estimator with the location of embarkation features included seems to have a rather high variance. Consequently, its performance would likely increase if more training samples were available.

In general, the random forest classifier with the embarkation features included seems most suitable to the task to predict chances of survival during the RMS Titanic disaster. Over a series of runs (which are not all discussed in this paper) this classifier seems to give the most constant scores on both the training and cross-validation sets, outperforming all other classifiers. The random forest classifier seems more accurate than the decision tree classifier because of a number of reasons. Firstly, the standard deviation of the learning curve regarding the cross-validation set is not smaller than the decision tree algorithm (see Figures 2,3) and the random forest's precision and recall are more equal than those scores for the decision tree, with both best estimators yielding the same F1-scores. Secondly, the most plausible explanation for the best decision tree estimator scoring better on its test set than the best random forest estimator does is luck, due to the small size of the test sets and the greediness of both algorithms.

7. Conculsion

Overall, despite limitations of the data set and the way in which this data set was split up in training, cross-validation, and test sets, chances are likely that a random forest classifier would be best at predicting the chances of survival of individual passengers on the titanic. However, all classifiers performed reasonably well.

To improve classification, it is recommendable to find more passenger (and crew) data, as well as finding a way to make sure that the classifiers all use the same k-fold cross-validation sets and learn, and predict, on the same training and test set. If more data was included, it would also be possible to try out how well a k-nearest-neighbor classifier can predict survival during the disaster. 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 be prone to strong over-fitting.

Furthermore, the results show that of the 9 features, the most important feature for predicting chances of survival on the RMS Titanic is by far the gender. The mean normalized importance of the `Gender` feature for the best decision tree, random forest, and random forest without embarkation location features is 0.399, followed by `Fare` and `Pclass` – which are proxies for socio-economic status – with respective scores of 0.211 and 0.151.

The hypothesis remains that the random forest classifier is the best solution to this problem. Even though the results obtained from this study are not clear enough to persuasively claim this statement, they do give a strong indication that this is indeed the case.

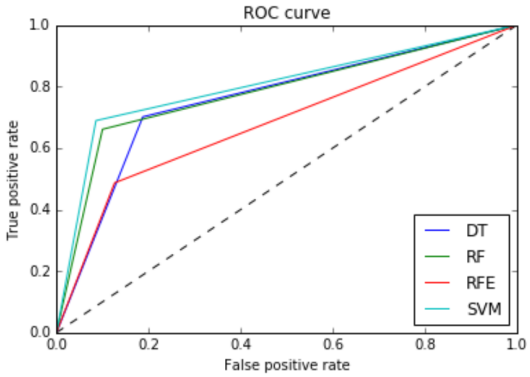

Figure 6: *ROC-curves*

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