CS 589 Final Project Report

Trung Dang[1]

[1]tmdang@umass.edu - SPIRE ID 33858723

Contents

1 Introduction

This report and the code that complements it deal with the Kaggle's Titanic - Machine Learning from Disaster challenge. Students were presented with a dataset of people onboarding the RMS Titanic and were tasked with classifying survivors against victims of the disaster. The training and testing datasets were provided in separate files, with the "Survived" column omitted from the test set. More insights into the datasets will be provided in subsection 2.1. Specifically for this project, we will be training the model using three different approaches, using Fix-shape universal approximators, Neural-network-based universal approximators, and Tree-based approaches. We will then compare the accuracy of the models over the training period as well as on the test set provided by Kaggle.

2 Dataset and Data Preprocessing

2.1 Dataset

The dataset of the challenge consists of two parts. The train set contains 891 entries of 342 survivors and 549 victims, covering a wide range of attributes as noted in Table 1:

Label	Definition	Notes
Name	Name	
Survived	Survival	0 = No, 1 = Yes
pclass	Ticket Class	[1-3] : high to low
sex	Sex	
Age	Age in years	with decimals only to newborns
sibsp	# of siblings or spouse on board	
parch	# of parents or child on board	
ticket	Ticket number	
fare	Ticket fare	
cabin	Cabin number	
embarked	Port of Embarkation	[C, Q, S]

Table 1: Fields and description of data fields

2.2 Data Encoding

The test set contains 418 entries of unknown survivors - victims ratio. Most data fields were numerical, which can be readily modeled by an ML model (namely, age, survival, pclass, sibsp, patch, and fare). Other fields such as "sex" and "embarked" are not in numerical forms but can also be easily converted to one-hot vectors. Note that we avoid numerical conversions such as $\{\text{male} \rightarrow 1, \text{female} \rightarrow 0\}$, which frequently confuses machine learning models due to distance discrepancy.

This leaves us with a few troublesome fields, such as Name, Ticket, and Cabin. One can suggest dropping these three fields as they are spontaneous and offer little insight into whether a person survives. Nonetheless, I present several ways to exploit the three fields.

First, for *Names*, I noticed that we were provided with first/last names and titles. People with more "prestigious" titles are assumed to have a higher priority and access to rescue boats and thus may have a higher chance of survival. In total, I found 17 distinct titles used in the dataset, which are: 'Capt', 'Col', 'Don', 'Dona', 'Dr', 'Jonkheer', 'Lady', 'Major', 'Master', 'Miss', 'Mlle', 'Mme', 'Mr', 'Mrs', 'Ms', 'Rev', 'Sir', 'the Countess'.

Among those a few titles were translations from a different language. For instance, 'Jonkheer' is a Dutch translation of 'Sir', and 'Mme', an abbreviation of 'mademoiselle', is an equivalence of 'Miss'. Therefore, after regrouping of titles, I was left with 8 discernible groups: ['Lady.', 'Ms.', 'Rev.', 'Mr.', 'Officer.', 'Dr.', 'Mrs.', 'Sir.']

Next, for *Ticket*, a quick sorting of ticket numbers reveals duplicated entries, which suggest these people were traveling together. Inspection of the Encyclopedia Titanica confirmed this hypothesis and unveils certain interesting correlations: for example, Miss Roberta Elizabeth Mary Maioni was a servant to the Countess of Rothes, but would not otherwise have been listed in "parch" or "sibsp" column. However, due to the overwhelming number of single travelers as well as the presence of the "parch" and "sibsp" columns, such findings naturally become anomalies. Therefore I dropped the Ticket column

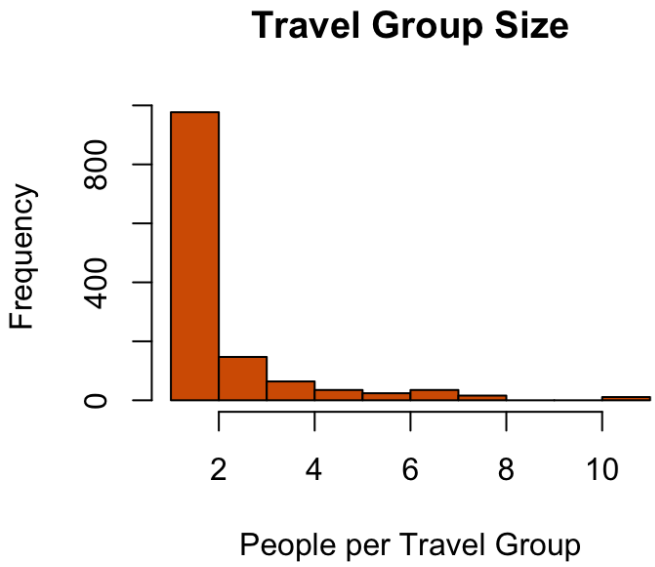

Figure 1: Histogram of number of people per travel group. Credit: Amber Thomas - Titanic Survival

Similarly, the *Cabin* column can also be dropped over the presence of the pclass column.

2.3 Data Normalization

The data obtained is not further normalized. This is because I found certain one-hot-encoded values (such as sex, or name titles) will not preserve their equal-distance property upon applicable normalization techniques, and perform very badly.

Within the training set, we use `sklearn.model_selection.train_test_split()` to split the training set into a training and cross-validation set with an 80-20 ratio. Reported model accuracies are evaluated on this cross-validation set, alongside the actual Kaggle score.

2.4 Handling of Missing Values

I noticed that there are missing entries in the "Age" and "Embarked" in the training set and the "Fare" of the test set. I also noticed that the data will be skewed if I fill in a lot of entries equal to the mean/median of the whole population onboard. Instead, I took the median of the population of the people with the same 'pclass' and 'sex' as the missing entry. Notice that since there are relatively few categories for 'pclass' and 'sex', there are generally many people with the same class and gender, so by the Law of Large Numbers, the estimation will likely be more accurate.

3 Implementation

The models are trained using `sklearn` API. We initialize each model using their respective wrapper class. To find the best hyperparameter, we use `sklearn.model_selection.GridSearchCV()` to perform an automatic grid search of hyperparameters. Models are then reported with Accuracy, ROC AUC score, and F1 score.

3.1 Kernel Method

For the Kernel Method, the model was initialized as such:

```
model_svm = SVC(max_iter=2000, tol=1e-5)
param_grid_svm = {
    "kernel": ["linear", "poly", "rbf"],
    "C": [10**i for i in range(-2, 3)],
    "gamma": [10**i for i in range(-2, 3)],
    "degree":[2,3,4]
}
clf_svm = GridSearchCV(model_svm, param_grid_svm, scoring="accuracy")
```

which covers several popular kernels. Recall that the kernel H can be defined without constructing the original feature matrix F, as:

$$H_{i,j} = \text{kernel}$$

where kernel falls into many categories, among those:

- linear: $\langle x, x' \rangle$
- poly: $(\gamma \langle x, x' \rangle + r)^d$, with d defined by the `degree` parameter
- rbf: $\exp(-\gamma |x - x'|^2)$, with γ defined by the `gamma` parameter

An exhaustive search of the parameters yields SVM Best params: {'C': 0.01, 'degree': 2, 'gamma': 0.01, 'kernel': 'linear'}. Note that since the degree parameters is only applicable to polynomial kernels, it is ignored in the final model yielded by grid search.

3.2 Neural Network Method

The Neural Network Based model is initialized as such:

```
hidden_layer_sizes = (30, 20, 10,)
model_nn = MLPClassifier(hidden_layer_sizes=hidden_layer_sizes,
                         max_iter=1000, random_state=42)
param_grid_nn = {
    "activation": ["logistic", "tanh", "relu"],
    "solver": ["sgd", "adam"],
    "alpha": [10**i for i in range(-3, 2)],
}

clf_nn = GridSearchCV(model_nn, param_grid_nn, scoring="f1")
clf_nn.fit(x_train, y_train)
```

Some important hyperparameters for the neural network model are the number of hidden layers, the size of each hidden layer, activation functions, the optimizer, and the learning rate. An exhaustive grid search of all these hyperparameters, however, would be unreasonable, provided the relatively slow training process and the runtime limit (15 minutes) of the problem. Therefore, through experiments, I have decided that the most suitable hidden layer number is 3, and the hidden layer size is (30, 20, 10,), as shown in Figure 2. Note that our input data has 18 features, so this choice is intuitive since it allows the model to explore combinations of features and slowly converge to the output layer. The `max_iter` hyperparameter is also capped at 1000, due to the slow convergence rate of the model that exceeds the runtime limit.

Other hyperparameters, however, must be tuned using grid search. This includes the activation function $\dot{a}$, which will map:

$$model(x, \theta) = \omega_{L+1}^T \dot{a}(\omega_L^T \dot{a}(\cdots \dot{a}(\omega_1^T \dot{x}) \cdots))$$

Some popular activation functions searched are logistic, tanh, and relu. Additionally, we perform a grid search for the optimizer, which regulates the gradient descent steps of the model: "sgd" and "adam". Finally, we also search for the most suitable learning rate. We noticed that a larger learning rate `alpha` will cause the model to diverge, thus limiting the search to 10^i for $i \in [-3, 1]$

The best NN Params are: {'activation': 'relu', 'alpha': 0.001, 'solver': 'adam'}

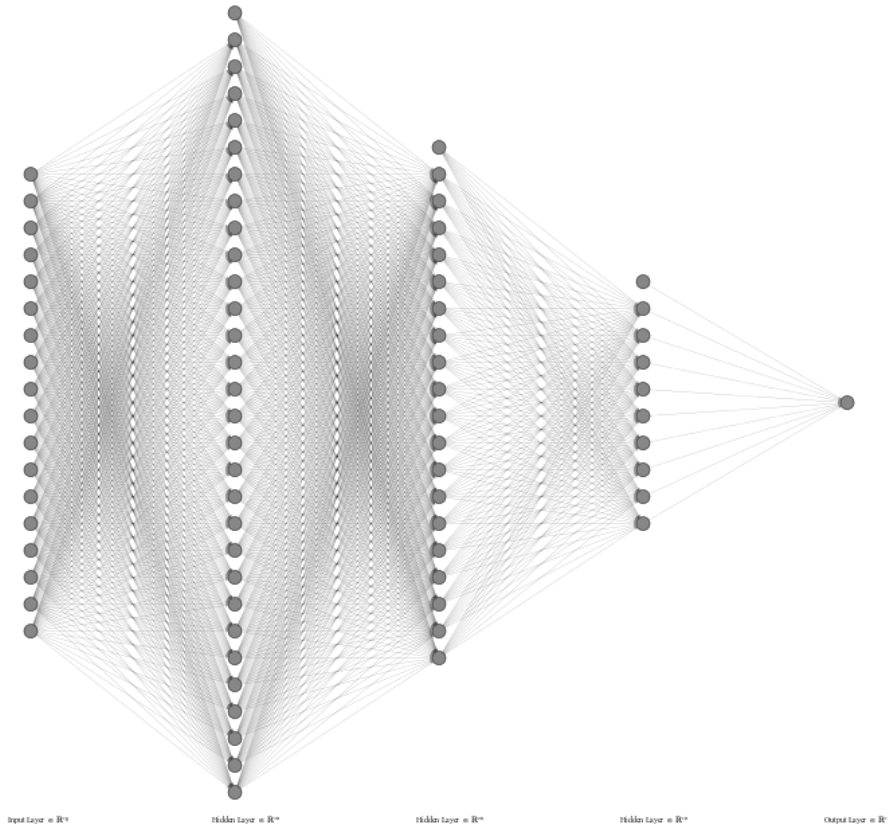

Figure 2: An illustration of the Neural Network used in the second approach

3.3 Tree-based Method

The Random Forest Model is initialized as such:

```
model_rf = RandomForestClassifier()
param_grid_rf = {
    "n_estimators": [100, 200, 300],
    "max_depth": [5, 10, 15],
    "max_features": ["sqrt", "log2"],
}
clf_rf = GridSearchCV(model_rf, param_grid_rf, scoring="accuracy")
clf_rf.fit(x_train, y_train)
```

Similar to the Neural Network model, an exhaustive grid search for all hyperparameters of the Random Forest model is not feasible due to computational limitations. Focusing on the most impactful parameters, we selected the following for grid search:

- **Number of estimators** (`n_estimators`): This parameter controls the number of decision trees in the forest. Higher values generally improve performance but also increase training time. We searched for values of 100, 200, and 300.
- **Maximum depth** (`max_depth`): This parameter limits the complexity of individual trees by restricting their maximum depth. Deeper trees can overfit the data, so we searched for values of 5, 10, and 15.
- **Maximum features** (`max_features`): This parameter controls the number of features considered at each split in the tree. We used two options: "sqrt" (square root of the total number of features) and "log2" (logarithm base 2 of the total number of features) for a balanced exploration without excessive computational cost.

Performing grid search using the above parameters revealed the following best parameters for the Random Forest model: {'`max_depth`': 5, '`max_features`': 'sqrt', '`n_estimators`': 100}

3.4 Validation Method and Cofigurations

As described above, each model was trained on a `GridSearchCV` with **default** 5-fold cross-validations and the reported model is the one with the highest average cross-validation accuracy score. An example of the code could be:

```
clf_rf = GridSearchCV(model_rf, param_grid_rf, scoring="accuracy")
```

4 Results

4.1 Model Saving and Inference

The model is saved using `pickle` to the format `{model}.pkl`. The inference is performed then by either loading a pre-trained model or re-training and inferring. Results are saved to `{model}.csv`.

4.2 Hyperparameter Analysis

Detailed analysis of the hyperparameters used in each model can be found in section 3. We list here the best hyperparameters for each model:

- Kernel Method: {'C': 0.01, 'degree': 2, 'gamma': 0.01, 'kernel': 'linear'}
- Neural Network Method: {`n_layers`:3, `layer_sizes`: (30, 20, 10,), 'activation': 'relu', 'alpha': 0.001, 'solver': 'adam', `max_iter`=1000}
- Tree-Based method: {`max_depth`: 5, '`max_features`': 'sqrt', '`n_estimators`': 100}

* Note: Even though we did not specify `max_iter` for the Kernel Method and Tree-Based Method, `sklearn` automatically supports early termination upon convergence, and our training process shows that the models successfully converges, eliminating the need for a `max_iter` hyperparameters

4.3 Evaluation Metrics

We use the accuracy, F1 score, ROC AUC, and the Kaggle score to evaluate the model. Among those, the accuracy measures the number of correct predictions (true positives and true negatives), and the Kaggle score is the accuracy of the model on the test set. The F1 scores and ROC AUC provide further insights into the recall and specificity of the models. However, we highlight that **accuracy is the main metric** of evaluation. The best models and hyperparameters were chosen out of higher accuracy.

4.4 Evaluation Results

Presented in Table 2 are the scores of the models trained on the best hyperparameters found using `GridSearchCV`

We also consider the learning curve of each model and the accuracy evolution as the models are trained on more training samples Figure 3, 4, and 5. It can be seen that the Kernel Method provides the most stable training curve, with the cross-validation score slowly converging to 0.78. Meanwhile, the neural network and random forest models fluctuate more during training, yet converge to a higher cross-validation accuracy. Evaluation by the Kaggle test set also indicates the superior performance of the Tree-based models, as the only model to cross the 0.8 threshold.

	Kernel	NN	Tree
F1 Score	0.73	0.75	0.76
ROC AUC	0.77	0.80	0.80
Accuracy	0.78	0.82	0.83
Kaggle	0.76	0.78	0.80

Table 2: Training results on different evaluation metrics of the selected models.

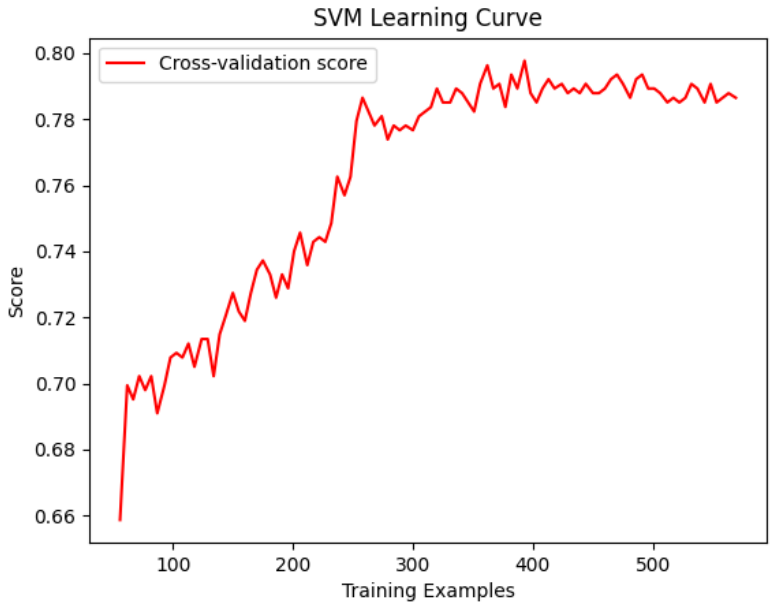

Figure 3: Kernel Method Learning Curve

Further insights into which features were important during the decision-making of the neural network and the random forest will be provided in Appendix A

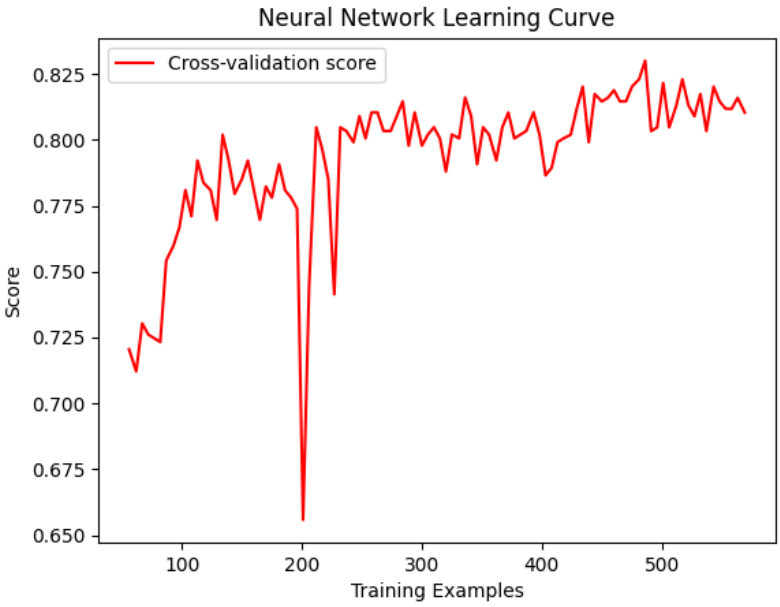

Figure 4: Neural Network-Based Method Learning Curve

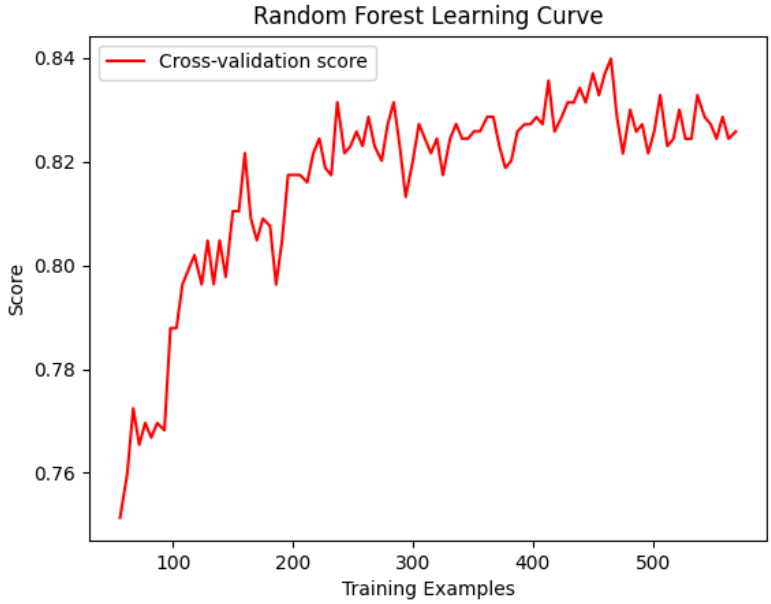

Figure 5: Tree-Based Method Learning Curve

5 Conclusions

The results show that within the scope of the Kaggle Titanic challenge, the Tree model yields the highest accuracy both during the training process and the final Kaggle score, and thus is the most suitable model to be used.

However, the result does not necessarily demonstrate the full potential of each model. For example, the Neural Network method was terminated early, and would likely have performed better given more training resources. Likewise, the random forest algorithm could have been trained on a more diverse set of decision trees at greater depth, which may shed light on some unexpected insights.

It is also noteworthy that a model with as few as 20-30 features is unlikely to capture all the dynamics of such a grave situation as the Titanic disaster. While certain factors such as gender, passenger class, and age were deemed critical (priority for women and children), in real life they are neither linearly correlated (a younger person does not necessarily have a higher survival chance) nor universal (many first-class passengers and crew did not survive). Therefore, any attempt to achieve an overwhelming accuracy (> 85 %) may simply create an overfitted model.

6 Acknowledgment

- Encyclopedia Titanica - https://www.encyclopedia-titanica.org/
- Amber Thomas - Titanic Survival - https://rstudio-pubs-static.s3.amazonaws.com
- `sklearn.svm.SVC` Documentations - sklearn.svm.SVC
- `sklearn.MLPClassifier` Documentations -sklearn.MLPClassifier
- `sklearn..RandomForestClassifier` Documentations - sklearn.ensemble.RandomForestClassifier
- Speeding up sklearn models on Intel CPU with `sklearnex` - Intel - StackOverflow

A Decision making of NN and RF models

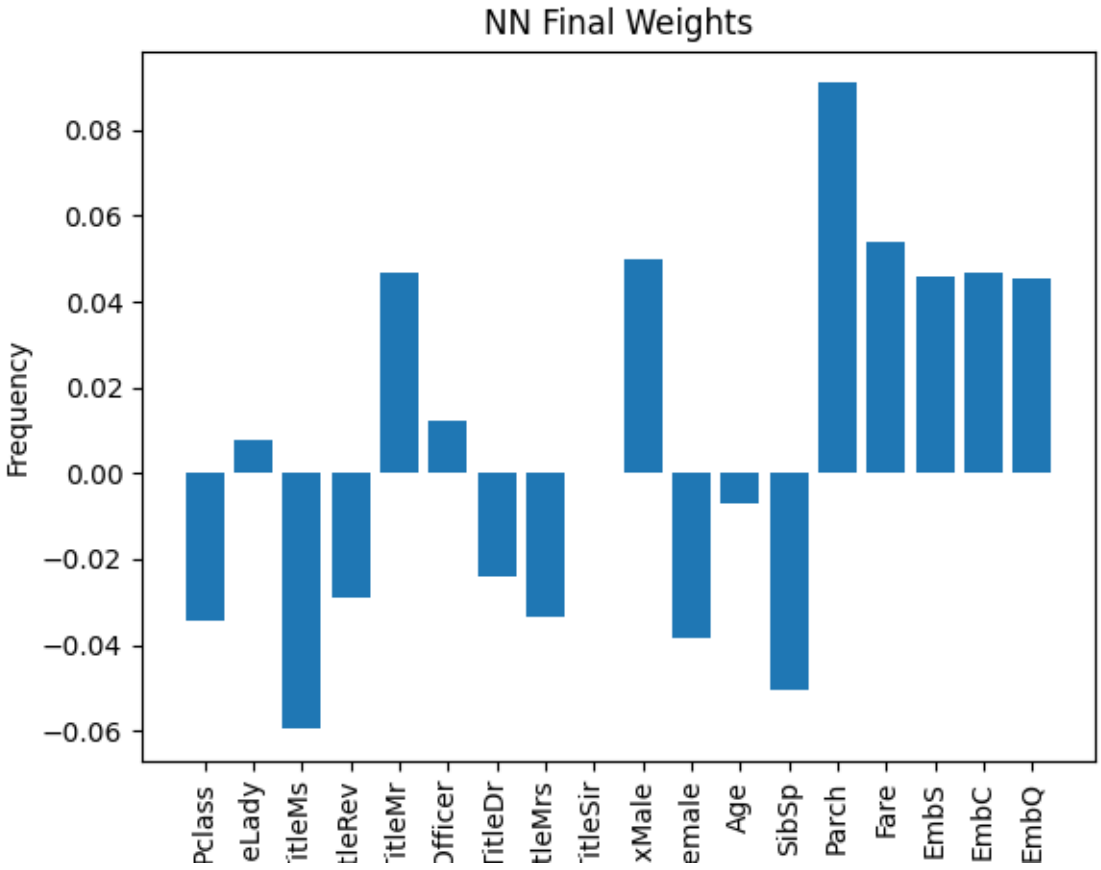

Figure 6: Final weights over parameters of the neural network models. Several important features include the sex of the travelers, their passenger class, and whether they traveled with parents or children.

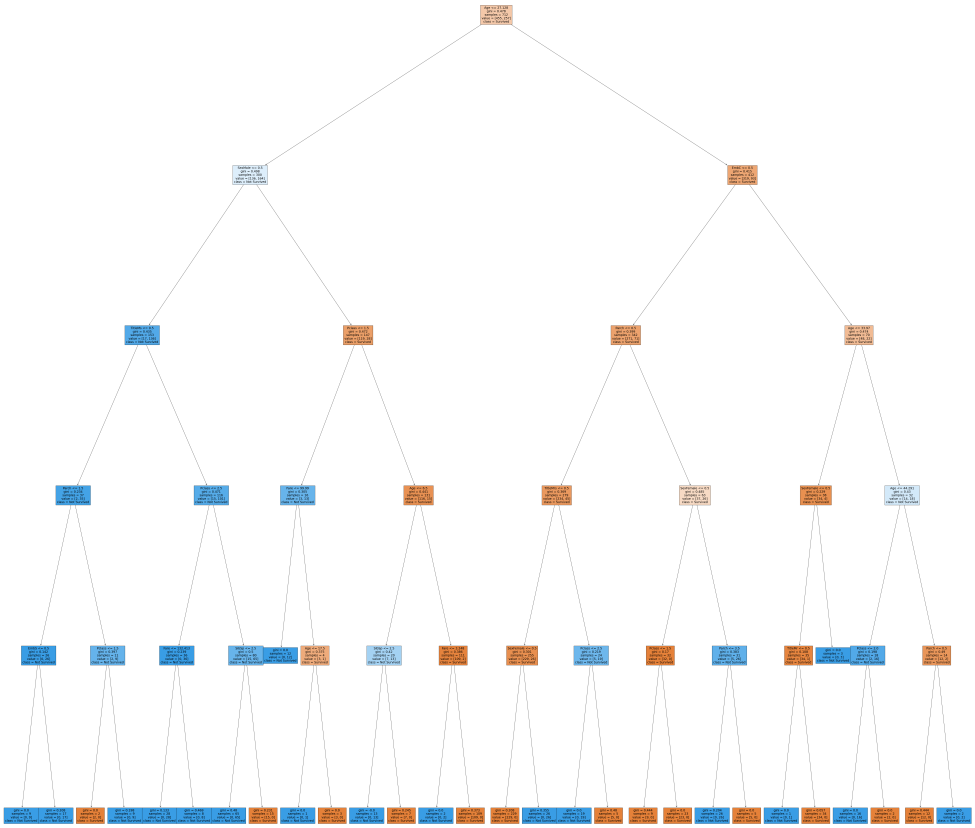

Figure 7: The decision tree of the Random Forest model. Yellow nodes indicate class = 'Survived', while blue nodes indicate class = 'Not Survived'. High resolution graph can be found here