ML assignment 2: Titanic case

Team '**ML_222**': CRABIÉ Julien SARKARIA Tandejajit ZHANG Qiaolei

1 Feature Generation

1.1 Motivation behind the features

1.1.1 New features 'Title', 'FamilySize' and 'Companions'

We generate 'Title' from 'Name'. The column 'Name' is constructed by 3 parts: first name, title and family name. Title in the name may provide us information about the identity or social status of passengers. And considering that a passenger with higher social status may have higher priority to get on lifeboats, we extract this information from 'Name' and store it in new column 'Title', using it to train models and make prediction.

We generate 'FamilySize' using number of siblings, parents and kids. We consider that it's possible to improve the chances to survive if a passenger has family members on the Titanic. Those who get on the lifeboat may try to get their families on as well. And since kids have priority to get on boats, passengers who have kids are quite possible to get on lifeboats as well to take care of their children.

We generate 'Companions' using ticket information. Observing the ticket number, we find some duplicated numbers that may indicate people buy tickets together and thus they travel on Titanic as a group. And those who have unique ticket number may travel alone. We will use this feature to figure out whether having companions help increase the chance of survival.

1.1.3 Make 'Embarked' dummy variables

To apply machine learning models we have to convert the string variables into numerical variables. This can be done simply by the get_dummies function in python and outputs a dummy variable for every category of a string variable. For the string variable 'Embarked' which can have three values namely, C = Cherbourg, Q = Queenstown, S = Southampton,We will get three dummy variables for each port and the rows will have value 1 if a passenger embarked on a part and 0 otherwise. If a value has 0 for the first 2 ports it is understood that he embarked on the 3rd port which is why the last dummy variable can be dropped (the information is inherent in the first two dummy variables).

1.1.4 Categorize 'Age'

The 'Age' variable has 177 missing values that need to be accounted for. Age is an important variable as we see different survival rates for different age groups. It can also be predicted based on other related variables. We fill in the missing values for age by fitting a Random Forest Regressor on the known age and variables title, parch (number of parents or children aboard), sibsp (number of siblings or spouses aboard) and Pclass (Passenger Class). We use the model to predict the age for the row where it is missing.

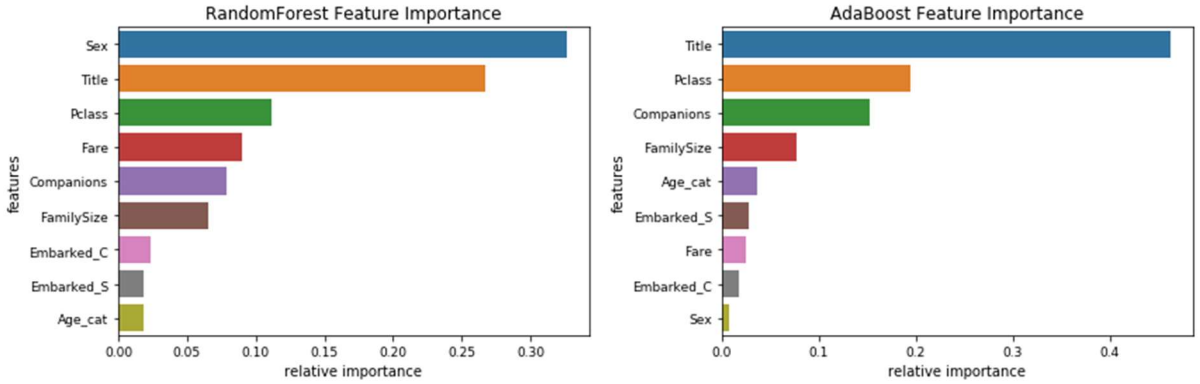
1.1.5 Dropping 'Cabin', 'Ticket', 'Name'

We drop the variable Cabin as it has 687 missing values in the data set. Even though there might be some relation between cabin location and survival, it is impossible predict so many missing values with the given information so we drop the variable. After we extract the title from the Name, we drop the variable as it doesn't provide us any useful information. From the ticket number, we extract the variable 'Companion' i.e some ticket numbers are identical for different observations implying that some people were travelling on the same ticket. After that we drop the column 'Ticket#' as any other information we might derive from it is also present in other variables such as Fare and Passenger Class.

1.2 Contribution of Features

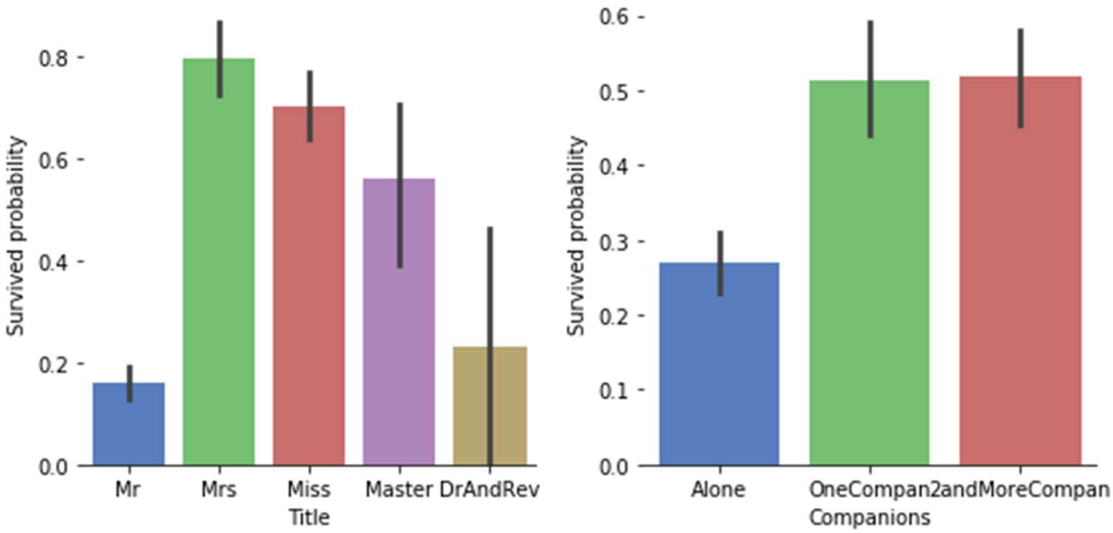
Among all models we trained, RandomForest got the highest training score and Adaboost got the highest Kaggle score.

In RandomForest model, Sex,Title, Pclass and Fare weight heavier than other features, gaining scores above 0.1. In AdaBoost model, Title, Pclass and Companions weight heavier than other features, gaining scores above 0.1.



Among the three new features we generated, Title and Companions are important for training models and predict labels.

Below shows the survival rate in different categories of Title and Companions. Mrs and Miss, which can be interpreted as female, had much higher survival rate then Mr. Also passengers had higher social status, Master or Doctor or Rev, have more chances to survive from the disaster than others. For Companions, passengers who traveled alone had lower survived probability than others, only gaining half of the survived rate of passengers with companions.



2 Model Tuning and Comparison

2.1 Accuracy for each model

Model	TrainAccuracy	ScoreKaggle
Adaboost	0.804790	0.79425

DecisionTree	0.804765	0.78947
RandomForest	0.830524	0.77990
LogisticRegression	0.786786	0.77033
SVC	0.786698	0.76555
KNN	0.814789	0.67942

2.2 Select and ensemble model

Since RandomForest got the highest training score and Adaboost got the highest Kaggle score, we used Voting classifier to ensemble these two models. The Kaggle score of VotingC is 0.81339, which is the best score we get on Kaggle.

2.3 Other models we tried

- KNN
  We also performed k-nearest neighbors (k-NN) method but the model didn't perform well. The parameter k was chosen by plotting the change in accuracy for the k-nn model (based on the training data) with the change in value of k. We used a K-fold cross validation and took the mean of the performance of 10 folds. The mean was then used to compute the accuracy. Using this method we found that the best value for k was 7. With k equal to 7 we tested our prediction on Kaggle and got a score of 0.67942.

- SVM
  We tried different types of kernels and C values to fit the model but got the best result with a linear one and C value at the default 1. While it performed relatively performed well on the training set, its accuracy dropped on the test set. We tried larger values of C to try and reduce misclassification instances but the resulting models show no improvement in training error or test error. Ultimately we decided against using an SVM model to do the predictions.

Using K-fold cross validation, the performance of 10 folds and the test score on Kaggle are shown below for each model:

- Decision tree
  >> [0.778 0.822 0.730 0.798 0.843 0.865 0.843 0.787 0.843 0.807]
  >>Mean Accuracy: 0.81 (+/- 0.08) | >>Score on Kaggle: 0.78468

- Random forest
  >> [0.822 0.899 0.775 0.865 0.854 0.854 0.798 0.798 0.888 0.82]
  >>Mean Accuracy: 0.837 | >>Score on Kaggle: 0.79425

- Logistic Regression
  >> [0.822 0.778 0.753 0.876 0.775 0.753 0.809 0.809 0.809 0.795]
  >>Mean Accuracy: 0.798 | >>Score on Kaggle: 0.77033

- Boosting using adaboost
  >> [0.767 0.811 0.719 0.831 0.843 0.854 0.854 0.798 0.820 0.864]
  >>Mean Accuracy: 0.816 | >>Score on Kaggle: 0.79425