MATH4995 Project 1: Predicting Survival on the Titanic

CHOW, Hau Cheung Jasper

24 September 2021

1 Introduction

In this project, I will conduct some preliminary analysis of the Titanic dataset (available on Kaggle) and use a variety of machine learning techniques to predict which passengers survive.

2 Data overview

The data comes in two datasets; namely "train.csv" and "test.csv". Each row in the dataset(s) correspond to a unique passenger with a "PassengerID" identifier column and 10 features/predictors. The values of these predictors are a mix of both numerical and categorical. In the "train.csv" dataset, there is an additional column called "Survived" which contains values in the set $\{0, 1\}$ describing whether that passenger survived the sinking of the Titanic (0 if they didn't, 1 if they did).

train.csv dimensions: 891 x 12
test.csv dimensions: 418 x 11

There are a variety of ways to access the data, from downloading it directly from Kaggle via a shell command in Google Colab to having an actual copy of the file. I chose the latter approach; namely, manually, downloading the dataset from Kaggle and then uploading it to Google Drive in a designated folder where it could be easily accessed. This was because I noticed that downloading the dataset from Kaggle was necessary whenever I wanted to run the notebook, and the process took a fair amount of time.

3 Understanding and transforming the data

Machine learning algorithms tend to accept numerical data. As such, I had to ensure that there were no non-numerical values anywhere in the training and test datasets. I will outline the approach I used to transform the training dataset into a "machine-learning-ready" state.

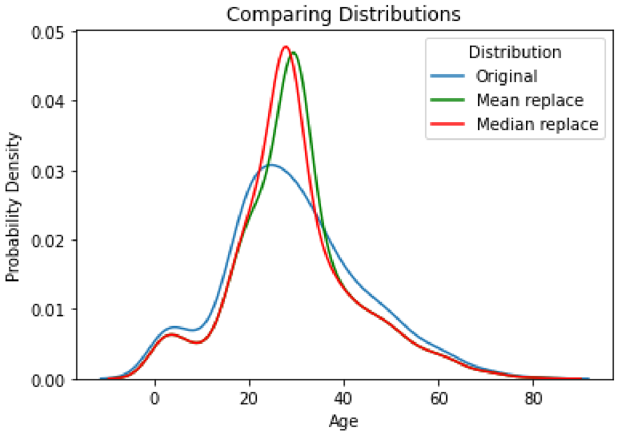
3.1 Missing values

The first thing I did was to check which columns of the data contained missing ("NA") values. In the training dataset "train.csv", I noticed 177 missing values in "Age", 687 values missing in "Cabin", and 2 values missing in "Embarked".

A brief look at the Cabin column revealed that the data tended to be in the form of "Deck-XXX" where Deck was a capital letter corresponding to which Deck that passenger was on, and XXX was some number. Since each passenger most likely occupied distinct cabins, I concluded there was no information to be gained from the specific cabin number. I did, however, extract the first letter from all data and saved it into the variable ”Deck”. For all missing values, I used the `fillna()` command to save their ”Deck” variable as ”X”.

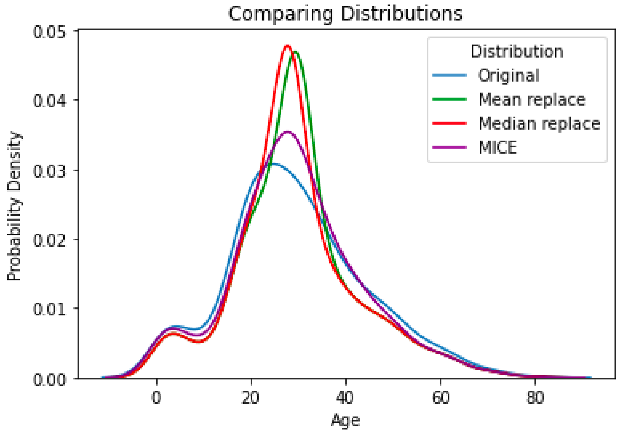
In the ”Embarked” section, values were one of $\{C, Q, S\}$. As such, I used `groupby()` to evaluate the median fare price for each `Pclass/Embarked` ordered pair, and filled in the missing values with the Embarked in the `Pclass/Embarked` pair whose median fare was **closest** to the fare of the missing values.

In the ”Age” section, there are 177 missing values out of 891 in the training set. As we can see from the distribution of the ages, if we use a more naive strategy like mean or median replacement, they are heavily centered around the middle, which is inconsistent with the original distribution.



Therefore, we must implement a way to fill in the missing values such that the resultant distribution is at least more similar to the original. The naive way would be to compute the original distribution function, then assign ’Age’ values to missing values randomly according to that distribution. However, that fails to take into account the idea that the age distribution is at least correlated with some other factors. For instance, an individual with ”Mrs” in their name you would not expect to be under 18, but the naive approach would not take that into consideration.

Therefore, we propose the approach of applying regression on each missing age value. Each missing Age value is defined to be the response variable Y in some regression involving predictors drawn from other columns of the training dataframe, and we repeat this regression procedure until all the missing values are filled. To do this, we chose the MICE (Multivariate Imputation by Chained Equation) and a procedure called predictive mean matching (PMM) to predict a distribution and randomly select values from it to use as the missing values. This method takes into account the relations between the OTHER predictors that may be correlated with different ages, and should therefore perform better than the naive approach.



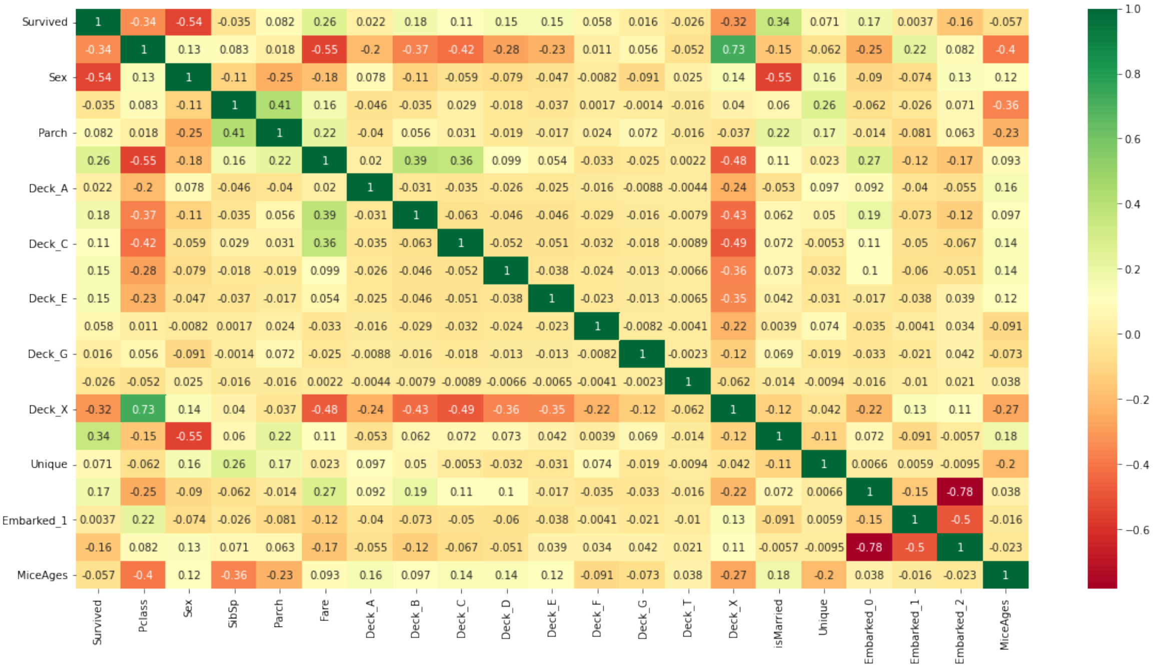
As seen, the distribution of the MICE-fitted ages is much more consistent with the original distribution. This, of course, assumes that the missing values follow the same distribution as the ages that we do have.

3.2 Indicator columns

We also prepared columns containing only indicator values (either 0 or 1) for the variables 'Deck' and 'Embarked'. The reason we did not leave them as a single value was because we could not safely assume that the variables were ordinal as opposed to categorical. For example, if we were to instead use a single column for a variable like 'Embarked' with values like 0 for C, 1 for Q, 2 for S might yield poor performance for any regression based model, since there is no evidence to suggest that passengers are most likely to survive if they embarked at C, then Q, then S.

3.3 Feature engineering

We also generated some new features from the dataset. On its own, the 'Name' column is quite useless, but after extracting the title of each individual, we were able to generate the 'Honorific' column, and then create the 'isMarried' feature (=1 for women who had the title Mrs, Ms, Mme, 0 otherwise) and the 'Unique' feature for those that had unique titles (eg Dr., Rev., Countess.)

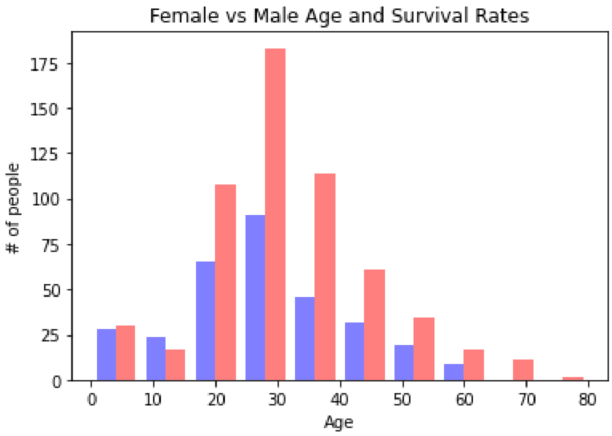


	Survived	Pclass	Sex	SibSp	Parch	Fare	Deck_A	Deck_B	Deck_C	Deck_D	Deck_E	Deck_F	Deck_G	Deck_T	Deck_X	isMarried	Unique	Embarked_0	Embarked_1	Embarked_2	MiceAges
Survived	1	-0.34	-0.54	-0.035	0.082	0.26	0.022	0.18	0.11	0.15	0.15	0.058	0.016	-0.026	-0.32	0.34	0.071	0.17	0.0037	-0.16	-0.057
	-0.34	1	0.13	0.083	0.018	-0.55	-0.2	-0.37	-0.42	-0.28	-0.23	0.011	0.056	-0.052	0.73	-0.15	-0.062	-0.25	0.22	0.082	-0.4
Sex	-0.54	0.13	1	-0.11	-0.25	-0.18	0.078	-0.11	-0.059	-0.079	-0.047	-0.0082	-0.091	0.025	0.14	-0.55	0.16	-0.09	-0.074	0.13	0.12
	-0.035	0.083	-0.11	1	0.41	0.16	-0.046	-0.035	0.029	-0.018	-0.037	0.0017	-0.0014	-0.016	0.04	0.06	0.26	-0.062	-0.026	0.071	-0.36
Parch	0.082	0.018	-0.25	0.41	1	0.22	-0.04	0.056	0.031	-0.019	-0.017	0.024	0.072	-0.016	-0.037	0.22	0.17	-0.014	-0.081	0.063	-0.23
	0.26	-0.55	-0.18	0.16	0.22	1	0.02	0.39	0.36	0.099	0.054	-0.033	-0.025	0.0022	-0.48	0.11	0.023	0.27	-0.12	-0.17	0.093
Deck_A	0.022	-0.2	0.078	-0.046	-0.04	0.02	1	-0.031	-0.035	-0.026	-0.025	-0.016	-0.0088	-0.0044	-0.24	-0.053	0.097	0.092	-0.04	-0.055	0.16
	0.18	-0.37	-0.11	-0.035	0.056	0.39	-0.031	1	-0.063	-0.046	-0.046	-0.029	-0.016	-0.0079	-0.43	0.062	0.05	0.19	-0.073	-0.12	0.097
Deck_C	0.11	-0.42	-0.059	0.029	0.031	0.36	-0.035	-0.063	1	-0.052	-0.051	-0.032	-0.018	-0.0089	-0.49	0.072	-0.0053	0.11	-0.05	-0.067	0.14
	0.15	-0.28	-0.079	-0.018	-0.019	0.099	-0.026	-0.046	-0.052	1	-0.038	-0.024	-0.013	-0.0066	-0.36	0.073	-0.032	0.1	-0.06	-0.051	0.14
Deck_E	0.15	-0.23	-0.047	-0.037	-0.017	0.054	-0.025	-0.046	-0.051	-0.038	1	-0.023	-0.013	-0.0065	-0.35	0.042	-0.031	-0.017	-0.038	0.039	0.12
	0.058	0.011	-0.0082	0.0017	0.024	-0.033	-0.016	-0.029	-0.032	-0.024	-0.023	1	-0.0082	-0.0041	-0.22	0.0039	0.074	-0.035	-0.0041	0.034	-0.091
Deck_G	0.016	0.056	-0.091	-0.0014	0.072	-0.025	-0.0088	-0.016	-0.018	-0.013	-0.013	-0.0082	1	-0.0023	-0.12	0.069	-0.019	-0.033	-0.021	0.042	-0.073
	-0.026	-0.052	0.025	-0.016	-0.016	0.0022	-0.0044	-0.0079	-0.0089	-0.0066	-0.0065	-0.0041	-0.0023	1	-0.062	-0.014	-0.0094	-0.016	-0.01	0.021	0.038
Deck_X	-0.32	0.73	0.14	0.04	-0.037	-0.48	-0.24	-0.43	-0.49	-0.36	-0.35	-0.22	-0.12	-0.062	1	-0.12	-0.042	-0.22	0.13	0.11	-0.27
	0.34	-0.15	-0.55	0.06	0.22	0.11	-0.053	0.062	0.072	0.073	0.042	0.0039	0.069	-0.014	-0.12	1	-0.11	0.072	-0.091	-0.0057	0.18
Unique	0.071	-0.062	0.16	0.26	0.17	0.023	0.097	0.05	-0.0053	-0.032	-0.031	0.074	-0.019	-0.0094	-0.042	-0.11	1	0.0066	0.0059	-0.0095	-0.2
	0.17	-0.25	-0.09	-0.062	-0.014	0.27	0.092	0.19	0.11	0.1	-0.017	-0.035	-0.033	-0.016	-0.22	0.072	0.0066	1	-0.15	-0.78	0.038
Embarked_1	0.0037	0.22	-0.074	-0.026	-0.081	-0.12	-0.04	-0.073	-0.05	-0.06	-0.038	-0.0041	-0.021	-0.01	0.13	-0.091	0.0059	-0.15	1	-0.5	-0.016
	-0.16	0.082	0.13	0.071	0.063	-0.17	-0.055	-0.12	-0.067	-0.051	0.039	0.034	0.042	0.021	0.11	-0.0057	-0.0095	-0.78	-0.5	1	-0.023
MiceAges	-0.057	-0.4	0.12	-0.36	-0.23	0.093	0.16	0.097	0.14	0.14	0.12	-0.091	-0.073	0.038	-0.27	0.18	-0.2	0.038	-0.016	-0.023	1

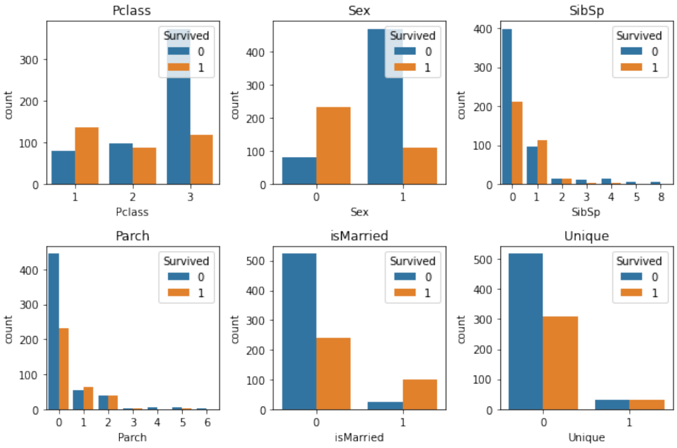
A correlation heatmap of the features show that the most important features in the model (highest absolute value of correlation) are Sex and Survived (-0.54), isMarried and Survived (0.34), Pclass and Survived (-0.34) and Fare and Survived (0.26). None of the correlations are particularly strong, but we observe a high negative correlation between values of the various indicator columns (which makes sense, since if a value is 1 in Embarked_0, it must be 0 for all the other Embarked indicator columns.) Surprisingly, we also observe that there is a strong positive correlation between Deck_X and Pclass (0.73) which is slightly reflected in the correlation between Deck_X and Survived (-0.32). However, we cannot be assured of the reliability of Deck_X as a predictor since Deck_X denoted the missing values which made up a large proportion of the dataset.

3.4 Visualisation

We plotted the number of individuals that survived by gender and noticed that females over the age of 20 are disproportionately more likely to survive than males of the same age range. However, males and females under the age of 20 are approximately equally likely to survive (likely on account of them being children.)



We also plotted the number of survivals by Sex, Age, Pclass, SibSp, Parch, isMarried and Unique.



Additionally, the ratio of individuals that survived to those that didn't is the highest for Pclass=1, for married women, and for individuals with unique honorifics. Additionally, those that had one or more sibling/spouse/parent/child were also relatively likely to survive compared to if they didn't. From these visualisations, we learn more about the inherent structure of the data and can use it to do sanity checks on our model (if our predictive model underweights the importance of 'Sex', for example, clearly something is wrong).

Since there is also a high correlation (0.73) between PClass and Deck_X, to avoid collinearity, we should remove one of these features. Since Deck_X has a high negative correlation with the other Deck indicator columns, if we removed it, we would still be able to identify a passenger from Deck_X because all the other Deck indicator columns would have value 0 (so we are not losing any information.) Hence we shall remove Deck_X from both the training and test dataframes. (We will also do the same for Embarked_2 to reduce collinearity.)

Therefore it makes sense to include the features 'Sex', 'isMarried', and either 'Pclass' or 'Fare' in our model.

4 Algorithms

We tried various approaches for determining an optimal classification. The models we tried were: k-Nearest Neighbours, Logistic Regression and Random Forest. We ignore single Decision Trees since they are susceptible to changes in training data and may tend to overfit; a random forest approach offers all the same advantages but also reduces variance.

4.1 KNN

Our initial training accuracy of the KNN model was 0.945, which initially seems extremely good. However, this heavily implies that the k-NN model overfits to the training data, because when K=1 the probability of one sample is estimated ONLY on ONE OTHER sample, which by definition is very susceptible to noise, outliers, etc. As such we cannot assume this model's training accuracy to be indicative of its performance on the unknown test set. In fact, one common estimate of an 'optimal' value of K in a dataset with n points (in our case $n = 891$ so $\sqrt{n} = 29$.)

When we observe the training accuracy score of the K-NN model at high values of K (23-29) we notice the accuracy score hovers around 75%. The validation accuracy score is around 70.9% for the chosen value of K=29, which is slightly lower than the training accuracy score (as expected, since all models will overfit to the training dataset to a small extent.)

4.2 Logistic regression

In logistic regression, we once again used 10-fold cross validation to estimate the test error, and the average validation training accuracy of the model (using the L2-norm and all features) was 0.8249, which was a significant improvement on KNN.

4.3 Random forest

The 10-fold cross validation accuracy score for our random forest model with 150 estimators and maximum depth of 7 was 0.835, which was a slight improvement on logistic regression. We chose maximum depth of 7 because trees that were too deep would be overfitted to the training data, and we want our trees to be 'not collinear' with other trees. However, we admit in the 'Future Analysis' section that more work could be done in tuning hyperparameters.

From our 3 models: K-NN, Logistic Regression and Random Forest, it appears Random Forest has the best training accuracy score, and even has a higher mean 10-fold cross-validation score than both the K-NN and Logistic Regression models. As such, we attempt to further tune the Random Forest model.

We also chose not to use the F1-score or precision/recall metrics because those tend to be used when there is significant class imbalance (there is only a 2:1 survived:not survived ratio) AND there is greater importance in predicting one particular class over another.

We further tuned our random forest model by selecting certain predictors to drop. We implement a form of backward subset selection by listing out and gradually dropping features in the random forest model starting with the least important ones. After iterating through all the features with < 0.05 importance, we find the highest mean 10-fold CV score and compute the range of values within 1SD of that value. We notice that all of the subsets here fall within this 1SD range of values, but there is a fairly significant drop off in mean cross validation score between keeping the last feature ('Parch') and removing it, so we decide to keep the 2nd last model generated, i.e. drop all features in the list except 'Parch'. This gives our model a 10-fold mean CV score of 0.835, which is very similar compared to the original model but has the advantage of not overfitting to the least relevant features.

4.4 Future analysis

When submitting our tuned random forest model to Kaggle, we find our test accuracy was only approximately 71%. As such, there is clearly much room for improvement.

For one, although we previously mentioned our reason for generating indicator columns, we could come up with different encodings for them (eg create a new feature called 'KnownCabin' =1 if the individual's cabin/deck number is known, and 0 if it is a missing value.) This is because may be bias that caused information about certain individuals' cabins to not be collected. This idea of using different 'encodings' of categorical variables can be extended. For instance, we can create separate features for Sibling and Spouse so that SibSp becomes two features, etc. Furthermore, we could also have attempted alternative data imputation methods besides MICE for the missing age values, since there is no guarantee that the missing age data follows the same distribution of ages that we already have.

Secondly, we could experience with different metrics. In logistic regression, we could attempt to use L1 norm (which penalizes more complex models compared to using L2 norm) to prevent overfitting. Additionally, we could have also experimented with the backwards subset selection in the logistic regression model to find the optimal set of features to apply the model to instead of applying it on all of the features like we did. For validation of these models, we could also use hypothesis testing (F-test) of various logistic regression models against each other.

Thirdly, we could have experimented with tuning the hyperparameters of the random forest. For instance, we could have implemented a grid search procedure for the number of estimators and the tree depth to exhaustively test various combinations, since the computational cost of the runtime is not high, this is a feasible approach.

As a starting point, we observe that our 10-fold CV mean error of 0.835 is significantly higher than the test accuracy of 0.719. As such, that suggests that our original choice of parameters is already significantly overfitting to the dataset and we should choose a smaller tree depth or change the number of estimators.

Finally, we could have experimented with an alternative bagging approach besides random forest; namely boosting (eg AdaBoost.) This would more heavily penalize misclassified datapoints since the loss function for boosting algorithms is exponential. However, it may have a tendency to overfit to the training data.