Kaggle Titanic Data Analysis Report

P. Adames (284546)

4/3/2020

Transforming the Kaggle data into data frames

The Kaggle API (API v.1.5.6)for command line was used to get the data to start this analysis. The files downloaded from the site were decompressed and stored for backup as three different files, one for training, one ofr test, and another with a sample submission.

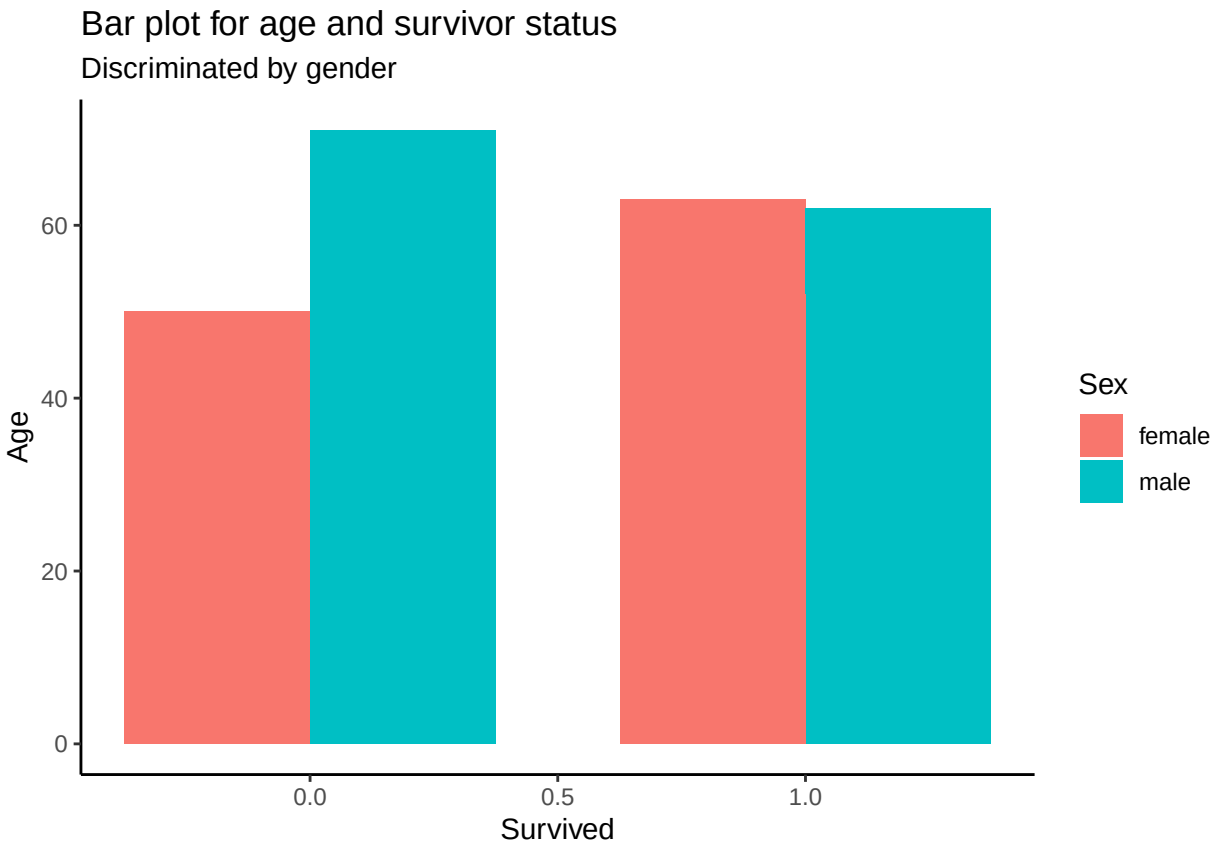
The way the data was given by Kaggle made it difficult to change the proportion of data in the train and test sets.

Data cleaning

All the rows with mising values were removed. As a result, there was a reduction of 125 records in the train set.

Data exploration

The actual proportion of passengers that died according to (Wikipedia contributors 2020) was 67.75%, the training data has a non-survival proportion of 59.57%. It is close but the sampling did not consider this proportion.

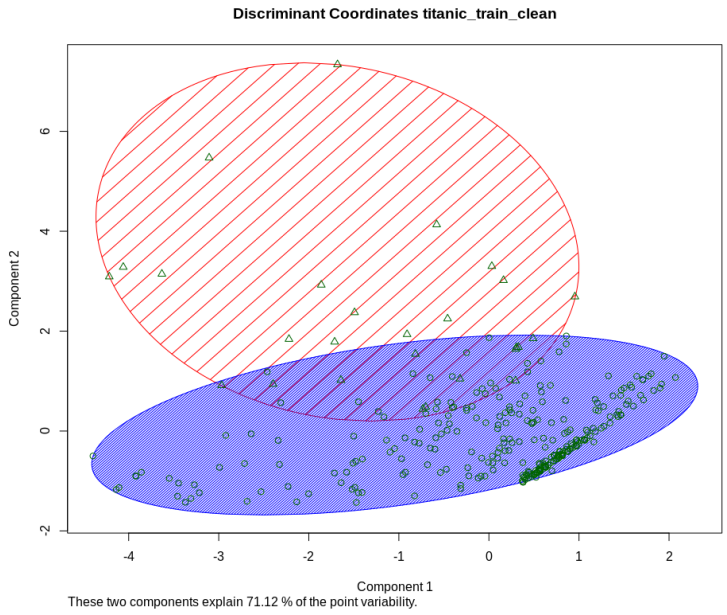


A clustering analysis with KMeans and two clusters, one with 302 passengers and a second one with 27. Between the two they explain 71.12% of the point variability of the data. The cluster centers are:

```
   Age      SibSp      Parch      Fare
```

1 30.11947 0.5430464 0.3874172 23.81876 2 33.10815 0.7037037 1.0740741 174.72669

Their plot appear below:



These clusters may indicate that there are data does naturally separate in two groups but the meaning of these clusters isn't very clear. Another analysis shows that the number of clusters stabilizes after approximately 5. This suggests that there is potential for dimensionality reduction, maybe using PCA. This was not pursued in the course of this analysis.

Exploration with a generalized linear model

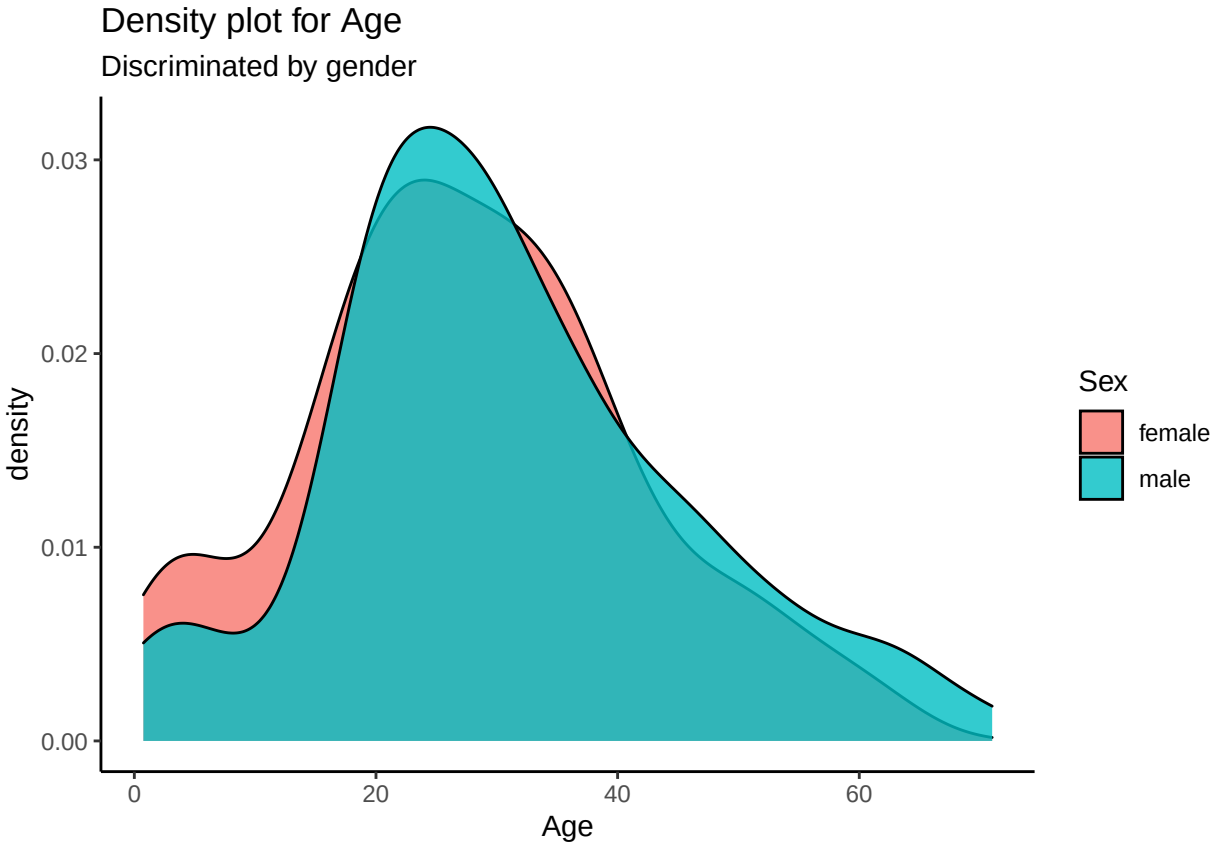
Using a generalize linear model with the `probit` function showed the coefficient of the following variables were statistically significant:

1. `` `SibSp`: number of siblings and/or spouse ``
2. `` `Age` ``
3. `` `Pclass`: the passengers class ``
4. `` `Sex`: the gender of the passenger ``

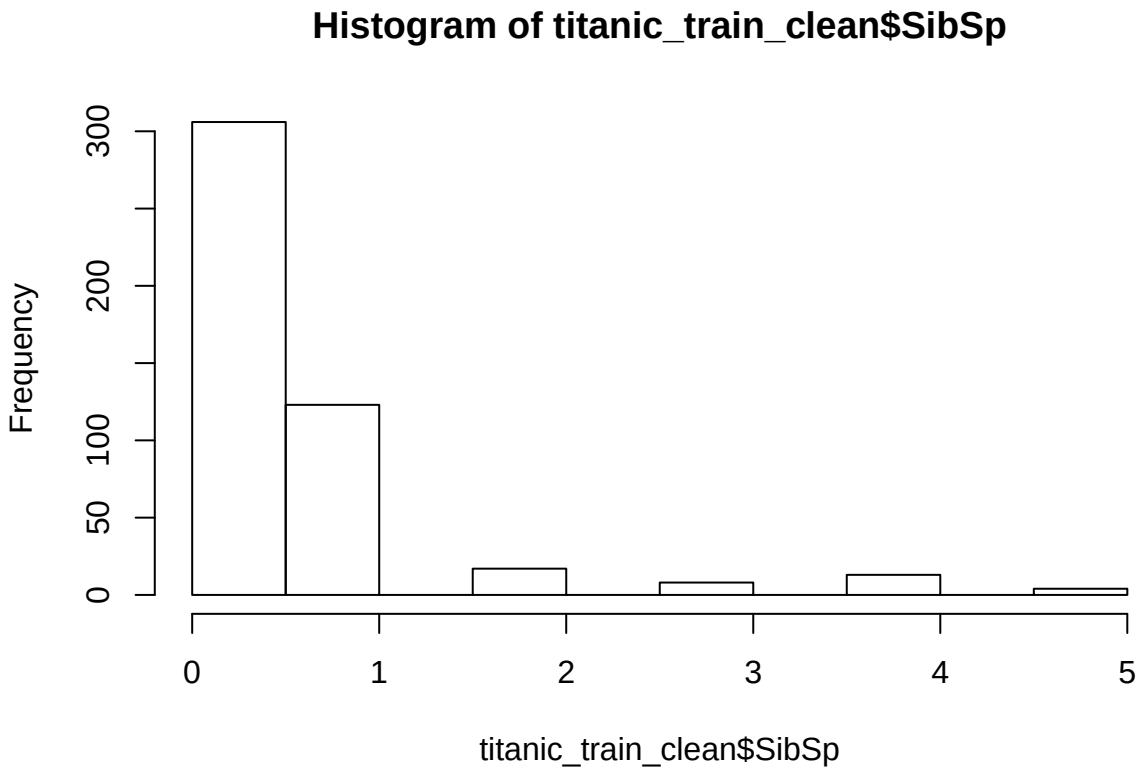
Numerical variables

The input numerical variables are `Age`, `SibSp`, `Parch`, and `Fare`.

`Age` has a relatively centered distribution.

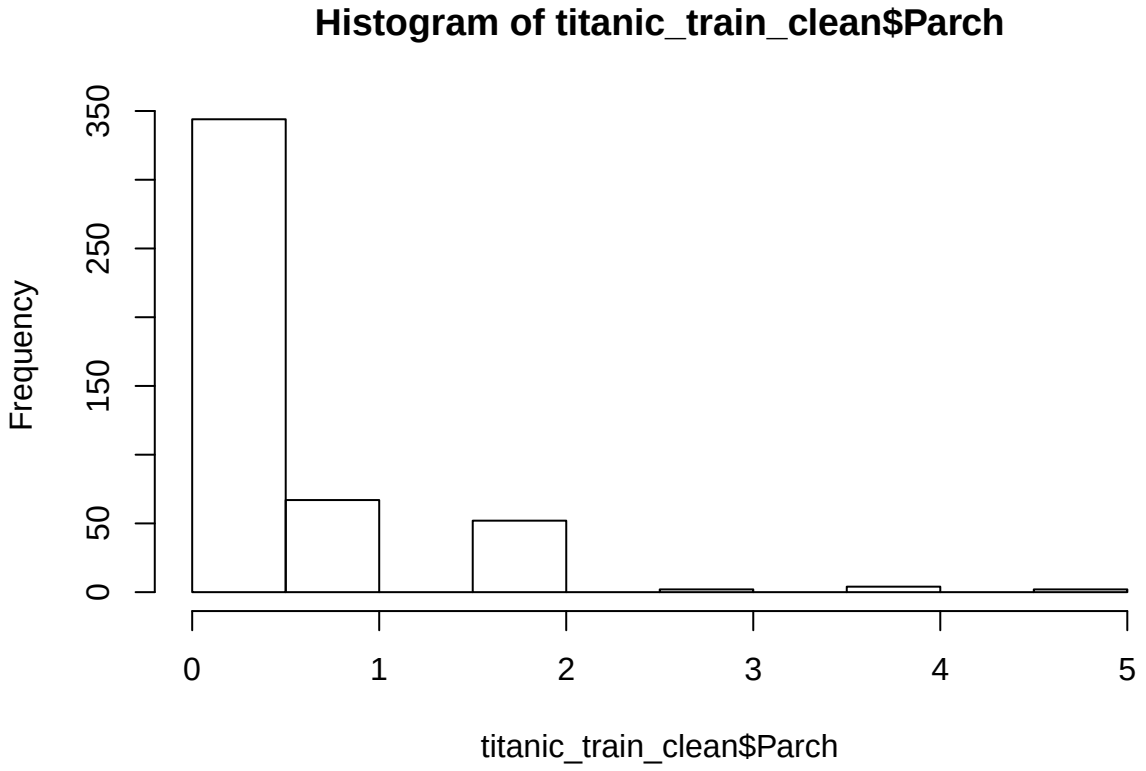


The variable `SibSp`, the number of siblings and spouse, can be treated as a continuous value between 0 and the maximum observed in the data, 5.



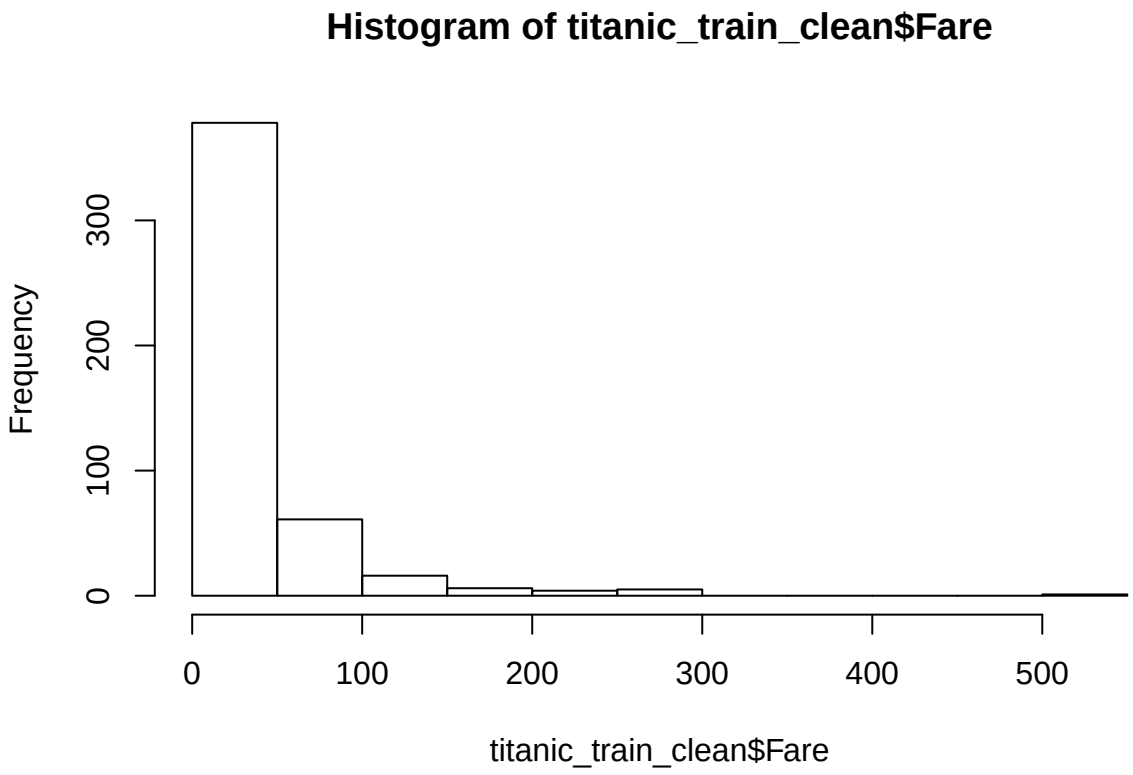
It also shows high values for skewness 2.265915 and kurtosis 5.217676. This indicates that this variable needs to be scaled with a standard scaler.

`Parch`, the number of parents and children accompanying the passenger, can also be treated as a continuous predictor.



As the histogram shows and the values of skewness 1.836532 kurtosis 3.152783 indicate it should also be normalized with a standard scaler.

Finally, `Fare` shows a very skewed distribution towards the low values. This is confirmed by the statistics of the variable: a mean of 36.20, a median of 16.70, a standard deviation of 52.52, skeness 4.050455, and kurtosis 24.262948. Therefore a standard scaler is th recommended preprocessing.



Categorical variables

The input categorical variables are `Sex` for gender Female (0) or Male (1), `Pclass` for the passenger category assigned by the shipping company, 1st, 2nd, or 3rd, and thus assigned three levels or factors.

The dependent or predicted variable is `Survived` that takes 196 (59.6%) negative outcomes, represented as 0, and 133 (40.4%) positive results, represented as 1.

Preprocessing

Only the numerical variables identified in the previous section were normalized. The same normalization used on the training set was used to predict with the test set for consistency. The predictor does not need any normalization.

To avoid missing predictions due to missing values in `Age` in the test set, I used K-nearest neighbours imputation as part of the preprocessing. This method with *k=10* finds the 10 nearest data vectors that look most like the data vector with the missing value and fills it in with the average from the cluster at that position. This gave slightly better results that not using this predictor at all to train the models. This restriction came from Kaggle requiring submissions for all records of the test set.

Preparing the data set to train the models

The variables `Name` and `Cabin` were dropped from this data frame because they hold very little meaning for prediction. The other variables mentioned in the previous sections were included in the data frame.

Preparing the data set for prediction

The test set was transformed in the same fashion as the training set, in this case using the same procedure with centering around the mean and scaling using the standard deviation.

Building and scoring the models

The emphasis will be on models that tradionally score well in classification competitions, that are insensitive to outliers and that are ensemble methods for structured data. In general repeated cross validation with 10 groups and 3 repetitions was used as often as practical. Parallel processing was enabled also to speed up the fitting.

Once each model is fitted to the training data a prediction is made with the test data and the result is saved to a `.csv` file in the folder for the results to be submitted to Kaggle. The score obtained from them is reported.

Logistic Model Trees

```
## Logistic Model Trees
##
## 471 samples
##   8 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 424, 423, 424, 425, 424, 424, ...
## Resampling results across tuning parameters:
##
##   iter  Accuracy   Kappa
##    1    0.7887431  0.5500104
##   21    0.7872045  0.5475188
##   41    0.7866000  0.5447399
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was iter = 1.
```

The accuracy on the training set was 0.7872045, 0.7866 with Kappa of 0.5475188, 0.5447399.

This result received a score of `0.23444` in Kaggle.

Bayesian generalized linear model

```
## Bayesian Generalized Linear Model
##
## 471 samples
##   8 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 424, 423, 424, 425, 424, 424, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7856807  0.5523862
```

The accuracy on the training set was `0.7868` with a Kappa value of `0.5538`.

This result received a score of `0.33971` in Kaggle.

Generalized linear model

```
## Generalized Linear Model
##
## 471 samples
##   8 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 424, 423, 424, 425, 424, 424, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.788533  0.5589737
```

This model shows and accuracy of 0.788533 on the training set.

The logistic regression model got a score of `0.24401` on Kaggle.

xGBoost

A default XGBoost model was built and some hyperparameter exploration was done only for the learning rate (the step size for the gradient estimation) and for the maximum depth of the trees. This took too long so several attempts at doing a grid optimization were abandoned. I think I need more experience with this method.

```
##   nrounds max_depth   eta gamma colsample_bytree min_child_weight subsample
## 1     100         4 0.025     0                1                1         1
```

These results obtained a score of `0.21531` on Kaggle.

Random forests

```
## Random Forest
##
## 471 samples
##   8 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 471, 471, 471, 471, 471, 471, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.7903065  0.5540096
##    7    0.7871034  0.5515029
##   12    0.7769229  0.5300991
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

The results form this model received a Kaggle score of `0.22966`.

Random forest with cross validation

```
## Random Forest
##
## 471 samples
##   8 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 470, 470, 470, 470, 470, 470, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.7855626  0.5404168
##    7    0.8025478  0.5826974
##   12    0.8067941  0.5904074
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 12.
```

The random forest grown with *leave one out cross validation* for the splitting on variables while growing trees, produces an accuracy of 0.8025478, 0.8067941 on the training set. However this translates into a Kaggle score of `0.22966`, identical to the one obtained for the regular random forest.

Support Vector Machine with Linear kernel

```
## Support Vector Machines with Linear Kernel
##
## 471 samples
##   8 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 424, 423, 424, 425, 424, 424, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7856659  0.5532718
##
## Tuning parameter 'C' was held constant at a value of 1
```

The training accuracy was 0.7856659 with the linear kernel.

The results from this model produced a score of `0.23444` in Kaggle.

Support Vector Machine with least squares Radial kernel.

This option has the Radial Basis Function kernel and sigma and cost as the hyperparameters to tune.

```
## Least Squares Support Vector Machine with Radial Basis Function Kernel
##
## 471 samples
```

```
##   8 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 424, 423, 424, 425, 424, 424, ...
## Resampling results across tuning parameters:
##
##   sigma       tau      Accuracy   Kappa
##   0.02546130   0.0625  0.7857700  0.5433099
##   0.02546130   0.1250  0.7864342  0.5444324
##   0.02546130   0.2500  0.7928198  0.5582053
##   0.02546130   0.5000  0.7942691  0.5625330
##   0.02546130   1.0000  0.7955963  0.5673397
##   0.02546130   2.0000  0.7914148  0.5594161
##   0.02546130   4.0000  0.7913262  0.5620204
##   0.02546130   8.0000  0.7856801  0.5503818
##   0.02546130  16.0000  0.7835524  0.5452143
##   0.02546130  32.0000  0.7758685  0.5208271
##   0.05040368   0.0625  0.7681847  0.5070227
##   0.05040368   0.1250  0.7730606  0.5169511
##   0.05040368   0.2500  0.7786611  0.5278481
##   0.05040368   0.5000  0.7822387  0.5349254
##   0.05040368   1.0000  0.7793562  0.5289183
##   0.05040368   2.0000  0.7842912  0.5422051
##   0.05040368   4.0000  0.7885940  0.5529882
##   0.05040368   8.0000  0.7913705  0.5606572
##   0.05040368  16.0000  0.7878540  0.5528661
##   0.05040368  32.0000  0.7794461  0.5294776
##   0.07534607   0.0625  0.7709496  0.5124035
##   0.07534607   0.1250  0.7751902  0.5210296
##   0.07534607   0.2500  0.7808055  0.5326934
##   0.07534607   0.5000  0.7836276  0.5376899
##   0.07534607   1.0000  0.7800513  0.5311087
##   0.07534607   2.0000  0.7807309  0.5336273
##   0.07534607   4.0000  0.7857706  0.5452772
##   0.07534607   8.0000  0.7865396  0.5477454
##   0.07534607  16.0000  0.7872045  0.5494747
##   0.07534607  32.0000  0.7737871  0.5131628
##   0.10028845   0.0625  0.7701633  0.5123139
##   0.10028845   0.1250  0.7736953  0.5191087
##   0.10028845   0.2500  0.7736959  0.5190992
##   0.10028845   0.5000  0.7744514  0.5197417
##   0.10028845   1.0000  0.7787234  0.5293517
##   0.10028845   2.0000  0.7801117  0.5333628
##   0.10028845   4.0000  0.7786322  0.5300567
##   0.10028845   8.0000  0.7829639  0.5401558
##   0.10028845  16.0000  0.7786919  0.5293983
##   0.10028845  32.0000  0.7701652  0.5010628
##   0.12523083   0.0625  0.7695305  0.5084512
##   0.12523083   0.1250  0.7674472  0.5042751
##   0.12523083   0.2500  0.7695755  0.5092610
##   0.12523083   0.5000  0.7702847  0.5108778
##   0.12523083   1.0000  0.7753103  0.5216775
```

```
##   0.12523083   2.0000  0.7787819  0.5280621
##   0.12523083   4.0000  0.7773782  0.5241311
##   0.12523083   8.0000  0.7702854  0.5083782
##   0.12523083  16.0000  0.7738906  0.5119877
##   0.12523083  32.0000  0.7688656  0.4918887
##   0.15017321   0.0625  0.7623760  0.4906819
##   0.15017321   0.1250  0.7623606  0.4906626
##   0.15017321   0.2500  0.7645191  0.4958934
##   0.15017321   0.5000  0.7680813  0.5043937
##   0.15017321   1.0000  0.7730619  0.5143034
##   0.15017321   2.0000  0.7710100  0.5100106
##   0.15017321   4.0000  0.7675236  0.5022157
##   0.15017321   8.0000  0.7667849  0.5002068
##   0.15017321  16.0000  0.7640656  0.4875845
##   0.15017321  32.0000  0.7596869  0.4644827
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.0254613 and tau = 1.
```

The training accuracy was with the linear kernel.

These predictions got `0.23923` in Kaggle.

Conclusions

The Kaggle titanic training set is quite challenging if one really wishes to score high on the leaderboard.

My perception is that it requires more thought into understanding the way the predictors, both categorical and continuous affect the prediction. Most of the implementations used automatically treat the categorical predictors like the tree based methods or convert them into numerica variables using hot-encoding or binary encoding as factors.

Some of the variables are not very balanced and this can cause problems.

I did not have the opportunity of doing more systematic hyperparameter tuning, especially due to the long run times.

The small data set and the small number of predictors discouraged my the use of neural networks.

The best entry in Kaggle scored `0.33971` for the Bayes generalized linear model. This is a Bayesian logistic regression with a reputation for being more stable numerically and computationally than the simple logistic regression. The coefficients are given a prior Student-t distributions to build a Naive Bayes estimator, using an Expectation-Maximization algorithm and a weighted least squares error reduction step.

References

Wikipedia contributors. 2020. “Passengers of the Rms Titanic.” April 1, 2020. https://en.wikipedia.org/wiki/Passengers_of_the_RMS_Titanic.