Bayesian Data Analysis - Course Project

December 9, 2018

RMS Titanic - Predicting survivors in the great maritime disaster

1 Introduction

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships.

One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.

This report aims to examine these differences in depth. We inted to create an accurate model for predicting survival probabilities of individuals. We chose survival status as our dependent variable and age, passenger class, sex, and city of embarktion to be our independent variables. We aim at obtaining new insights into the effect these features had on the survival of the participants in this great disaster.

2 Data

The dataset we worked on was ready-made and aquired from Vanderbilt Biostatistics dataset collection (http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/titanic.html). The dataset relies on multiple sources major one being the Encyclopedia Titanica webpage. It contains information about all 1313 passengers including their whole names, sex, age, passenger class, and survival status among others. The dataset does not include information about the RMS Titanic crew. Full description of dataset fields is included below.

Data features

	dtype	description
row.names	numeric	Name
pclass	string	Passenger class (1st, 2nd, 3rd)
survived	numeric	Survival (0 = No; 1 = Yes)
age	numeric	Age
embarked	string	Port of Embarkation
home.dest	string	Home/Destination
room	string	Cabin number
ticket	string	Ticket number
boat	string	Lifeboat (number of NaN)
sex	string	Sex (male, female)

We chose to investigate the effect of sex, age, passenger class, and port of embarktion on the survival probability. In addition, we tested whether being child or an elderly had effect on survival. The motivation behind this was to examine if the assumed policy of "women and children first" can be backed up with data. This was done by deriving two additional features, child and elderly, denoting passengers under the age of 15 and over the age of 65, respectively.

The dataset contained a designated variable for expressing whether individual had access to lifeboat. This variable was evidently correlated with survival (54%). However, we decided to exclude this variable, as we aimed to find the underlying reasons for some people ending in life boats and others not.

2.1 Preprocessing

The dataset required preprocesing before it could be used in modelling. First, categorical variables (sex, passenger class, city of embarktion) were transformed into binary representation. Second, all rows including missing values were dropped out. It is notable that this removes major part of the original data. Moreover, has different effect on different passenger classes as over 70% of 3rd class passenger data is lost. Third, age variable was scaled and normalized to avoid its larger magnitude to skew results. Finally, data set was divided into training (67 %) and test (33 %) set.

2.2 Key Statistics

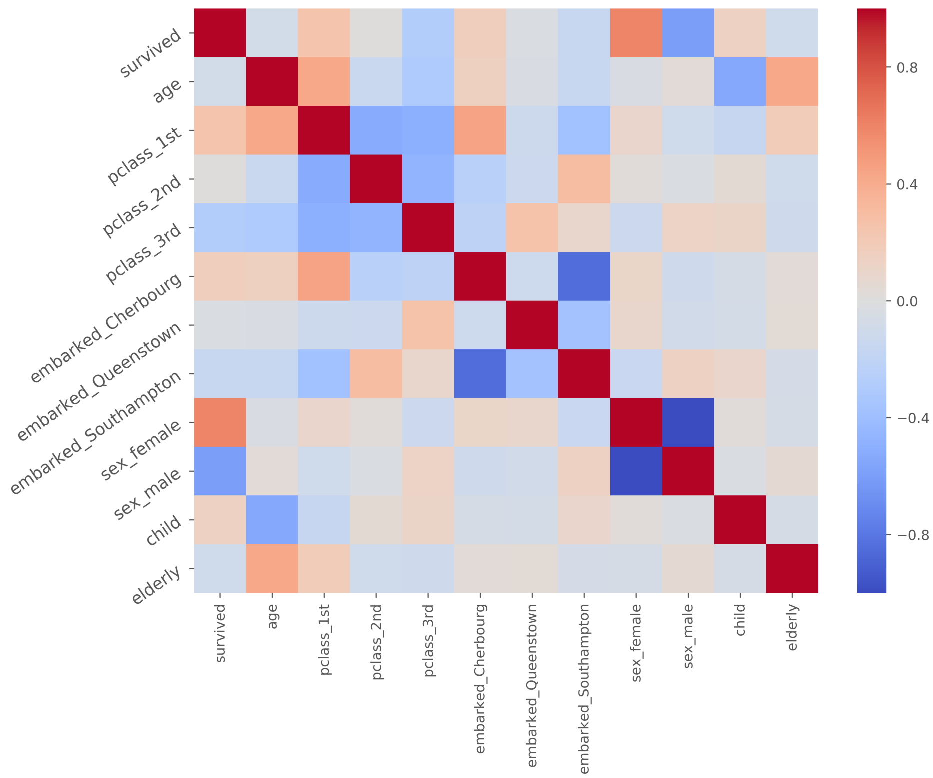
Passenger class specific key statistics were computed for the dataset and shown in the table below.

Key statistics by passenger class

pclass	Passengers	Men	Women	Age (avg)	Age unknown	Queenstown	Cherbourg	Southampton	Survived	Percentage
1st	322	179	143	39.67	96.0	3	142	167	193	0.60
2nd	280	173	107	28.30	68.0	7	28	237	119	0.42
3rd	711	498	213	24.52	516.0	35	33	169	137	0.19

The likelihoods of survival differ between different passenger classes. 60% of the first class passengers survived, as compared to 19% of the third class passengers. 711 of the passengers were travelling in the third class which is more than first and second class passengers combined. However, most of the third class information is missing which might introduce bias to the model. First class passengers were evidently older than second and third class passengers, which might increase the correlation between age and surivaval probability.

In addition, we analyzed the correlation between explanatory variables. The correlations are shown in the heatmap and table below. Besides the evident correlations (between passenger classes, between port of embarktion, one being either male or female, age and the derived quantities), there seems to be not many noteworthy correlations. One worth mentioning is the correlation beween age and first passenger class (0.43)



Heatmap of correlations between explanatory variables

Correlations between explanatory variables

	survived	age	pclass_1st	pclass_2nd	pclass_3rd	embarked_Cherbourg	embarked_Queenstown	embarked_Southampton	sex_female	sex_male	child	elderly
survived	1.00	-0.08	0.26	0.01	-0.28	0.17	-0.03	-0.15	0.60	-0.60	0.14	-0.09
age	-0.08	1.00	0.43	-0.14	-0.30	0.15	-0.03	-0.15	-0.03	0.03	-0.55	0.43
pclass_1st	0.26	0.43	1.00	-0.53	-0.50	0.45	-0.12	-0.38	0.10	-0.10	-0.16	0.19
pclass_2nd	0.01	-0.14	-0.53	1.00	-0.47	-0.24	-0.13	0.30	0.02	-0.02	0.05	-0.09
pclass_3rd	-0.28	-0.30	-0.50	-0.47	1.00	-0.22	0.26	0.09	-0.13	0.13	0.12	-0.10
embarked_Cherbourg	0.17	0.15	0.45	-0.24	-0.22	1.00	-0.12	-0.85	0.10	-0.10	-0.06	0.03
embarked_Queenstown	-0.03	-0.03	-0.12	-0.13	0.26	-0.12	1.00	-0.37	0.09	-0.09	-0.07	0.04
embarked_Southampton	-0.15	-0.15	-0.38	0.30	0.09	-0.85	-0.37	1.00	-0.14	0.14	0.10	-0.07
sex_female	0.60	-0.03	0.10	0.02	-0.13	0.10	0.09	-0.14	1.00	-1.00	0.03	-0.06
sex_male	-0.60	0.03	-0.10	-0.02	0.13	-0.10	-0.09	0.14	-1.00	1.00	-0.03	0.06
child	0.14	-0.55	-0.16	0.05	0.12	-0.06	-0.07	0.10	0.03	-0.03	1.00	-0.06
elderly	-0.09	0.43	0.19	-0.09	-0.10	0.03	0.04	-0.07	-0.06	0.06	-0.06	1.00

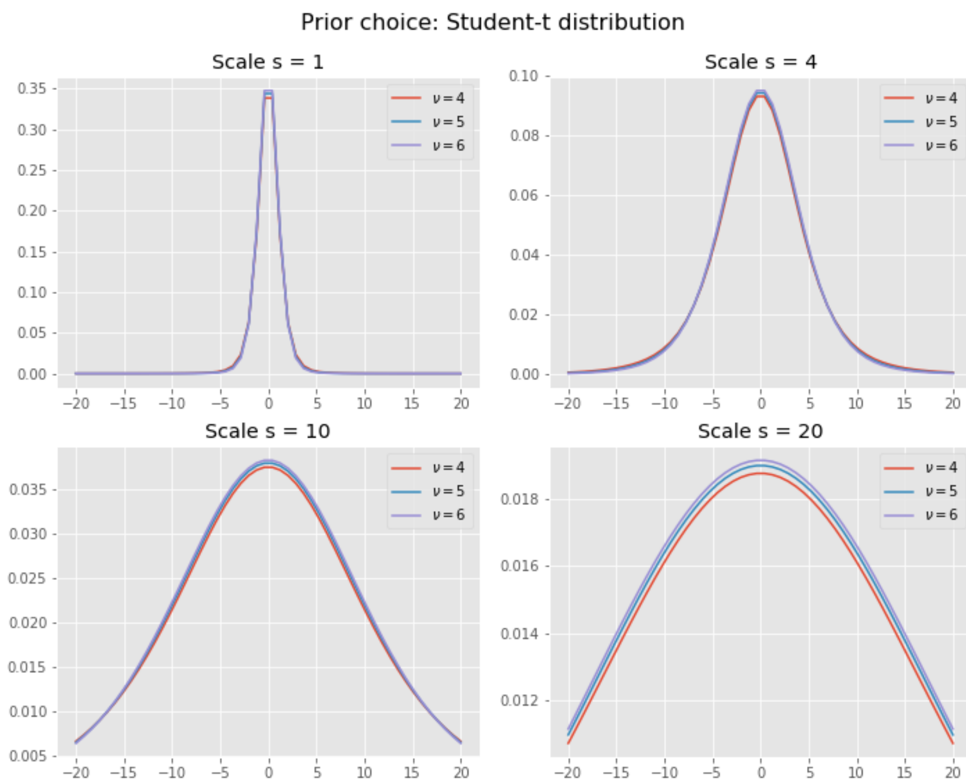
3 Model

Our choice is to examine the data with logistic regression. In logistic regression, linear regression is nonlinearized with a logistic function. The function limits regression values between 0 and 1 and thus provides a prozy for probability of survival.

3.1 Prior choice

As we don't have prior knowledge on how the data is distributed, a weakly informative prior is chosen according to stan development team guidelines. The rationale for using weakly informative prior is to let inferences be unaffected by information external to the current data - "let the data speak for itself", so to say. A Student-t distribution with zero mean is a suitable choice for logistic regression (see i.e. https://arxiv.org/pdf/0901.4011.pdf for more information).

Thus, for each regression coefficient, we assume a Student-t prior distribution with mean 0, degrees-of-freedom parameter ν within the range $3 < \nu < 7$, and scale s, where s is chosen to provide weak information on the expected scale. The figure below presents a few prior distribution candidates like this.



Student-t distributions with different scales and degrees of freedom

3.2 Model Selection

This prior is further adjusted in model selection. Several different combinations of degrees of freedom and scale values for prior were compared by applying PSIS-LOO cross validation to the logistic model. The table and the figure below present results of the cross validation. It can be seen that prior with 6 degrees of freedom and scale as 1 provides the best fit based on the PSIS-LOO value and corresponding k values.

PSIS-LOO cross validation results

prior df	prior scale	p_eff	PSIS-LOO
4	1	8.49	-180.58
	2	9.65	-181.07
	4	10.63	-181.94
	10	11.33	-182.64
	20	11.54	-182.87
5	1	8.56	-180.73
	2	9.61	-181.01
	4	10.55	-181.83
	10	11.38	-182.69
	20	11.37	-182.68
6	1	8.24	-180.50
	2	9.68	-181.09
	4	10.73	-182.05
	10	11.14	-182.45
	20	11.55	-182.90

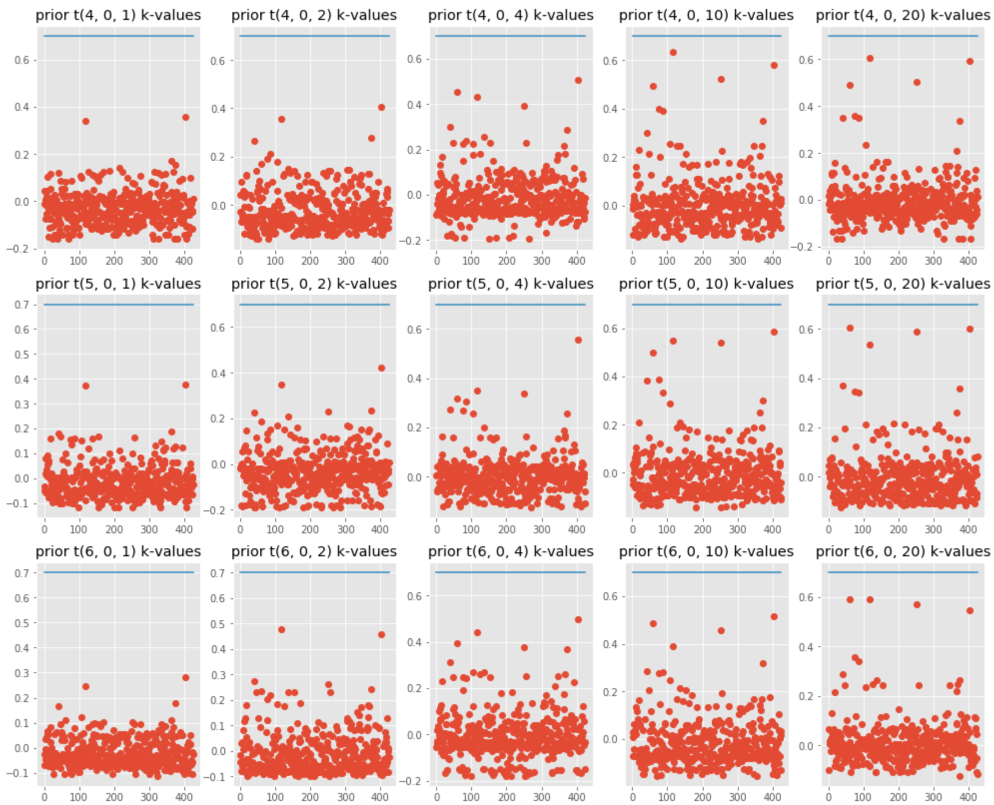
According to the model selection results we parametrize our prior as $\nu = 6$ and $s = 1$

$$p(\theta) \sim \text{Student-t}(\mu = 0, \nu = 6, s = 1) \tag{1}$$

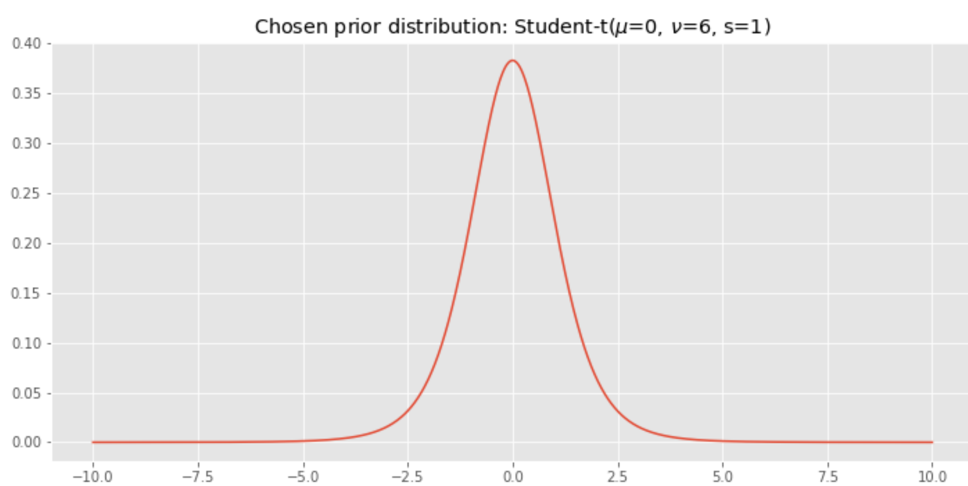
3.3 Sensitivity Analysis

As we don't have strong knowledge about the effect of features on survival probability, we choose a weakly informative prior. However, while choosing one prior distribution over others, we induce new modeling assumptions - this is true, regradless of whether we explicitely know what these assumptions are. Following the standard Bayesian Data Analysis convention, we investigate the sensitivity of posterior distribution towards the choice of the prior distribution by performing sensitivity analysis.

The tables below show the observed posterior means and posterior 95% condifence interval spreads for selected features. The 95% posterior spread was calculated by substracting the lower limit from the upper limit of the confidence interval. As can be seen from the tables, the obtained feature means are relatively robust against the prior choice. However, there is some variation in the feature means, arguably correlating with increasing prior scale. For example, different priors



PSIS-LOO k vales with different prior parameters



Chosen prior

assign age posterior means ranging from -1.74 to -2.58. Yet, the relative importance, or rank, the model assigns to different features, remains the same despite the choice of prior distribution.

The 95 % confidence interval spread is not as robust against the prior choice. This can be seen when, for example, comparing the port of embarktion spreads between different priors. One intepretation is that the robustness of posterior confidence interval spread is correlated with the quality of the featurewise model-evidence. The more the posterior-distribution is constructed from data (the more it can be backed-up) the stabler it is.

Sensitivity of the feature mean on prior choice

		age	child	elderly	pclass_1st	pclass_2nd	pclass_3rd	sex_female	sex_male
prior_df	prior_scale								
4	1	-1.85	1.07	-1.13	1.29	-0.14	-1.12	1.54	-1.62
	2	-2.20	1.15	-1.53	1.50	-0.10	-1.17	1.66	-1.63
	4	-2.42	1.14	-1.76	1.48	-0.21	-1.30	1.63	-1.73
	10	-2.58	1.12	-1.80	1.54	-0.17	-1.29	1.83	-1.55
	20	-2.58	1.14	-1.81	1.85	0.11	-1.00	1.61	-1.78
5	1	-1.78	1.08	-1.12	1.31	-0.11	-1.09	1.57	-1.59
	2	-2.18	1.15	-1.53	1.42	-0.17	-1.23	1.62	-1.67
	4	-2.46	1.12	-1.71	1.53	-0.17	-1.27	1.71	-1.65
	10	-2.56	1.13	-1.82	1.49	-0.23	-1.34	1.47	-1.90
	20	-2.60	1.11	-1.86	1.90	0.17	-0.95	1.45	-1.94
6	1	-1.74	1.09	-1.11	1.27	-0.14	-1.10	1.58	-1.56
	2	-2.18	1.15	-1.54	1.43	-0.16	-1.22	1.56	-1.73
	4	-2.48	1.12	-1.72	1.44	-0.26	-1.36	1.69	-1.67
	10	-2.56	1.12	-1.79	1.33	-0.38	-1.50	1.69	-1.70
	20	-2.57	1.12	-1.85	1.43	-0.31	-1.42	1.88	-1.51

Sensitivity of the feature 95% confidence interval on the prior choice

		age	child	elderly	pclass_1st	pclass_2nd	pclass_3rd	sex_female	sex_male
prior_df	prior_scale								
4	1	3.53	2.03	3.23	3.13	3.16	3.20	4.78	4.73
	2	3.72	2.10	3.78	5.23	5.18	5.15	6.82	6.80
	4	4.01	2.26	4.30	9.72	9.51	9.48	13.21	13.15
	10	4.14	2.23	4.39	22.79	22.61	22.66	30.65	30.74
	20	4.14	2.21	4.49	46.82	46.42	46.53	60.18	60.43
5	1	3.36	1.94	3.20	2.98	2.91	2.96	4.48	4.55
	2	3.83	2.13	3.87	4.84	4.75	4.78	6.91	6.72
	4	4.03	2.20	4.14	9.32	9.28	9.34	12.64	12.59
	10	4.09	2.24	4.47	23.45	23.10	23.20	30.23	30.14
	20	4.21	2.28	4.61	45.78	45.72	45.83	60.47	60.53
6	1	3.27	1.94	3.24	2.92	2.84	2.97	4.25	4.29
	2	3.59	2.07	3.76	4.88	4.86	4.93	6.83	6.79
	4	3.96	2.18	4.21	9.37	9.35	9.40	12.07	12.03
	10	4.21	2.30	4.36	23.06	23.06	22.95	30.24	30.33
	20	4.14	2.28	4.29	46.16	46.23	46.29	58.51	58.40

3.4 Convergence Diagnostic

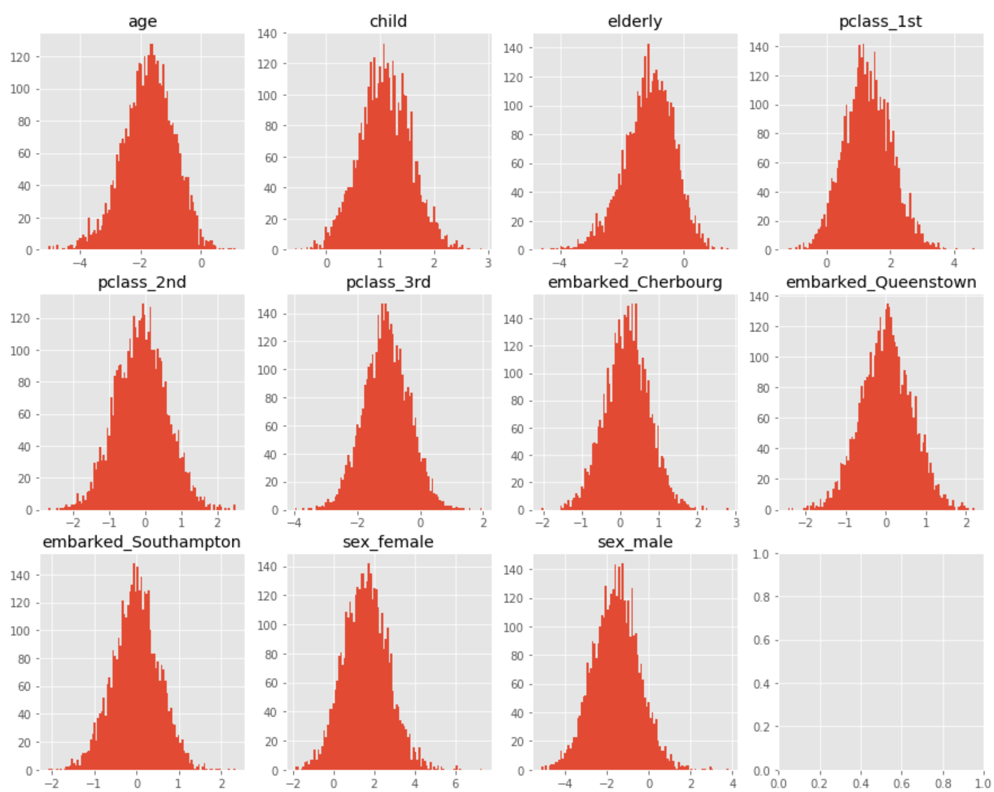
Based on the model selection the final predictive model was fitted with t(6,0,1) as prior, see table below for convergence diagnostics. Based on the convergence diagnostic we the model has convergenced well (R_hat < 1.01)

Logistic regression converge statistics

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
alpha	0.62	0.03	1.46	-2.36	-0.34	0.60	1.64	3.46	1934.55	1.0
beta[1]	-1.74	0.01	0.84	-3.46	-2.30	-1.71	-1.16	-0.19	3295.31	1.0
beta[2]	1.09	0.01	0.49	0.13	0.76	1.08	1.41	2.07	3481.09	1.0
beta[3]	-1.11	0.01	0.81	-2.88	-1.61	-1.06	-0.58	0.36	3434.66	1.0
beta[4]	1.27	0.02	0.75	-0.20	0.78	1.26	1.77	2.72	2127.68	1.0
beta[5]	-0.14	0.02	0.72	-1.60	-0.60	-0.14	0.35	1.24	1973.93	1.0
beta[6]	-1.10	0.02	0.74	-2.65	-1.57	-1.08	-0.62	0.32	2083.10	1.0
beta[7]	0.17	0.01	0.57	-0.93	-0.21	0.17	0.55	1.28	2771.44	1.0
beta[8]	0.02	0.01	0.65	-1.28	-0.43	0.02	0.47	1.29	2856.64	1.0
beta[9]	-0.02	0.01	0.55	-1.09	-0.39	-0.02	0.34	1.06	2587.75	1.0
beta[10]	1.58	0.02	1.10	-0.52	0.81	1.59	2.33	3.73	2218.26	1.0
beta[11]	-1.56	0.02	1.10	-3.73	-2.32	-1.54	-0.80	0.56	2288.58	1.0

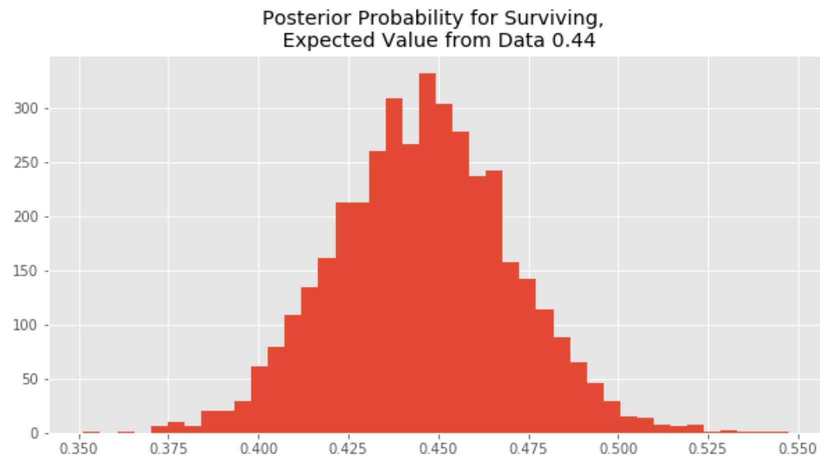
3.5 Model Parameters

Posterior distributions for the beta values of the regression model are presented in the figure below. The intepretation of the results is discussed in later sections.



Beta draws from posterior

Posterior predictice checking was conducted by drawing samples from posterior and comptuing the expected value for a singe draw. This value was then compared to the expected value from data. The posterior draws seem to model the data quite well, see figure below



Posterior predictive checking

4 Discussion

Based on the beta values, the probability of survival is increased by being young and female and having a higher class ticket. This supports also the assumed policy of "women and children first". The intuitive fact that port of embarktion has no effect on survival is backed up by data.

In addition to posterior predictive checking, we performed point-estimate based accuracy checking. Training and test error are presented in a table below. We can see that the logistic regression model achieves a significat improvement to a dummy model which predicts a person not surviving

F1-scores

	Dummy (All 1) training	training	Dummy (All 1) test	test
0	0.62	0.79	0.61	0.74

The most significant shortcomings of this model are related to the dataset as it had great deal of missing information. The missing information was not evenly distributed but concerned foremost third class passengers. The effect this bias had on the results is hard to estimate.

5 Conclusion

We were able train a classifier to predict survivols with 0.74 F1 score and found out the intuitive fact that being young, female and having higher class ticket improved individuals chances to survive in titanic. The model predictions are not perfectly accurate but this was expected, as the survival included an irremovable element of randomness.

6 Appendix

In [91]:
```
import os
import math
import sys
import pystan
import scipy.stats as st
import numpy as np
import pandas as pd
import itertools
import pickle
import seaborn as sn

from IPython.display import display
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from matplotlib import image, pyplot as plt
from tqdm import tqdm
from sklearn.metrics import f1_score
from utilities import psis, stan_utility
from utilities.my_utilities import *
```

In [92]:
```
plt.style.use("ggplot")

CONVERT_TO_PDF = True
RUN_MODEL_SELECTION = False
RUN_SENSITIVITY_ANALYSIS = False

figs = 'figs'
tex = 'tex'

if not os.path.exists(figs):
    os.makedirs(figs)

if not os.path.exists(tex):
    os.makedirs(tex)
```

In [93]:
```
# data
original_data = pd.read_csv("../data/titanic.txt", index_col="name")

numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']

dtypes = (original_data.dtypes.apply(lambda r: pd.Series({'dtype': 'numeric' if r in
numerics else 'string'})))

descriptions = [
    'Name',
    'Passenger class (1st, 2nd, 3rd)',
    'Survival (0 = No; 1 = Yes)',
    'Age',
    'Port of Embarkation',
    'Home/Destination',
    'Cabin number',
    'Ticket number',
    'Lifeboat (number of NaN)',
    'Sex (male, female)'
]

dtypes['description'] = descriptions

with open(os.path.join(tex,'dtypes.tex'), 'w') as f:
    f.write(dtypes.to_latex())

data = original_data.drop(["row.names",
                           "home.dest",
                           "room",
                           "ticket",
```

```
                                "boat"], axis=1)
```

6.0.1 Key statistics

```
In [94]: were_on_lifeboat = original_data.boat.notna().astype('int')

         print("correlation between access to lifeboat and survival is
         {:.2%}".format(were_on_lifeboat.corr(original_data.survived)))

         pclass_summary = data.groupby('pclass').agg({'survived': ['sum', 'mean'],
                                                      'age': [no_info, 'mean'],
                                                      'sex': [females, males, tot],
                                                      'embarked': [Queenstown, Cherbourg,
         Southampton, no_info, tot]
         })
         pclass_summary["tot"] = data.pclass.value_counts()

         final = pd.DataFrame()
         final["Passengers"] = pclass_summary["tot"]
         final["Men"] = pclass_summary["sex"]["males"]
         final["Women"] = pclass_summary["sex"]["females"]
         final["Age (avg)"] = pclass_summary["age"]["mean"]
         final["Age unknown"] = pclass_summary["age"]["no_info"]
         final["Queenstown"] = pclass_summary["embarked"]["Queenstown"]
         final["Cherbourg"] = pclass_summary["embarked"]["Cherbourg"]
         final["Southampton"] = pclass_summary["embarked"]["Southampton"]
         final["Survived"] = pclass_summary["survived"]["sum"]
         final["Percentage"] = pclass_summary["survived"]["mean"]

         create_tex_table(final, os.path.join(tex,'key_stats.tex'))

         if not CONVERT_TO_PDF:
             display(final)

correlation between access to lifeboat and survival is 54.38%


In [95]: # binarize categorical variables, drop NaNs and normalize and scale "age" between 0 and
         1
         data_binarized = pd.get_dummies(data).dropna(axis=0, how="any")
         data_binarized["child"] = (data_binarized["age"] < 15).astype(int)
         data_binarized["elderly"] = (data_binarized["age"] > 60).astype(int)
         data_binarized["age"] =
         preprocessing.minmax_scale(preprocessing.scale(np.array(data_binarized["age"])))

         corr_matrix = data_binarized.corr()
         create_tex_table(corr_matrix, os.path.join(tex, 'corr_matrix.tex'))

         if not CONVERT_TO_PDF:
             display(data_binarized.head(n=3))
             fig, ax = plt.subplots(figsize=(10, 8))
             heatmap = sn.heatmap(corr_matrix, cmap='coolwarm', ax=ax)
             heatmap.set_yticklabels(heatmap.get_yticklabels(), rotation = 35, fontsize = 12)

             plt.tight_layout()
             plt.savefig(os.path.join(figs, 'corr_matrix.png'), dpi=400)

In [96]: # create arrays for a stan model
         features = ["age",
                     "child",
                     "elderly",
                     "pclass_1st",
                     "pclass_2nd",
                     "pclass_3rd",
                     "embarked_Cherbourg",
                     "embarked_Queenstown",
```

```
"embarked_Southampton",
            "sex_female",
            "sex_male"]

y = np.array(data_binarized["survived"])
X = np.array(data_binarized[features], dtype=np.dtype(float))
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
random_state=42)
print("{0} samples in training set \n{1} samples in test set".format(y_train.size,
y_test.size))
```

```
424 samples in training set
209 samples in test set
```

6.0.2 Model Training and Selection

```
In [97]: def fit_model(prior_df, prior_s, debug=False):
             model_data = dict(n=X_train.shape[0],
                               d=X_train.shape[1],
                               X=X_train,
                               y=y_train,
                               p_beta_df=prior_df,
                               p_beta_scale=prior_s)
             n_iter = 50 if debug else 2000
             fit = model.sampling(data=model_data, seed=1, iter=n_iter,
         control=dict(max_treedepth=15))
             return fit
```

6.1 Prior choice

```
In [98]: if not CONVERT_TO_PDF:
             prior_dfs = [4, 5, 6]
             prior_scale = [1, 4, 10, 20]

             fig, axes = plt.subplots(2, 2, figsize=(10, 8))
             x = np.linspace(-20, 20)

             for df in prior_dfs:
                 for s, ax in zip(prior_scale, axes.flat):
                     ax.plot(x, st.t.pdf(x, df=df, scale=s), label=r'$\nu={}$'.format(df))

                     ax.legend(loc='best')
                     ax.set_title('Scale s = {}'.format(s))

             fig.suptitle('Prior choice: Student-t distribution', fontsize=16, )
             plt.tight_layout()
             plt.subplots_adjust(top=0.90)


             plt.savefig(os.path.join(figs,'prior_choice.png'))
             plt.savefig(os.path.join(figs,'prior_choice.eps'), format='eps', dpi=1000)

In [99]: prior_dfs = [4, 5, 6]
         prior_scale = [1, 2, 4, 10, 20]
         model = stan_utility.compile_model('logistic_regression.stan')

         if RUN_MODEL_SELECTION:

             datapoints = np.arange(1, X_train.shape[0] + 1)

             fig, axs = plt.subplots(len(prior_dfs), len(prior_scale), figsize=(17, 14))
             axs = axs.ravel()
```

```
    p_effs = []
    loo_sums = []
    Ks = []
    i = 0

    for df in tqdm(prior_dfs):
        for s in prior_scale:
            fit = fit_model(df, s)
            samples = fit.extract(permuted=True)

            # LOO CV
            loo, loos, ks = psis.psisloo(samples["log_lik"])
            loo_sums.append(loo)
            Ks.append(ks)

            lppd = np.sum(np.log(np.sum(np.exp(samples["log_lik"]), axis=0)/4000))
            p_effs.append(lppd-loo)

            axs[i].plot(datapoints, ks, 'o')
            axs[i].plot(datapoints, [0.7] * X_train.shape[0])
            axs[i].set_title("prior t({0}, {1}, {2}) k-values".format(df, 0, s))
            i += 1

    fig.savefig(os.path.join(figs,'k_values.png'), bbox_inches='tight')
    fig.savefig(os.path.join(figs,'k_values.eps'), bbox_inches='tight', format='eps',
dpi=1000)

    with open("loo_sums.pkl", 'wb') as f:
        pickle.dump(loo_sums ,f, protocol=2)
    with open("p_effs.pkl", 'wb') as f:
        pickle.dump(p_effs ,f, protocol=2)

if not CONVERT_TO_PDF:
    img=image.imread(os.path.join(figs,'k_values.png'))
    plt.figure(figsize = (50,50))
    plt.imshow(img)
    plt.show()
```

```
Using cached StanModel
```

```
In [100]: if not RUN_MODEL_SELECTION:
              with open("loo_sums.pkl", 'rb') as f:
                  loo_sums = pickle.load(f, encoding='latin1')
              with open("p_effs.pkl", 'rb') as f:
                  p_effs = pickle.load(f, encoding='latin1')

          psis_loo_results = pd.DataFrame()
          psis_loo_results["p_eff"] = p_effs
          psis_loo_results["PSIS-LOO"] = loo_sums
          psis_loo_results["prior df"] = [i for i, _ in itertools.product(prior_dfs, prior_scale)]
          psis_loo_results["prior scale"] = [j for _, j in itertools.product(prior_dfs,
          prior_scale)]
          psis_loo_results_table = psis_loo_results.set_index(["prior df", "prior scale"],
          drop=True)
          create_tex_table(psis_loo_results_table, os.path.join(tex,'psis_loo_results.tex'))

          if not CONVERT_TO_PDF:
              display(psis_loo_results)
```

6.1.1 Posterior Predictive Checking

```
In [101]: # final model
          largest_psis_loo_params = psis_loo_results[psis_loo_results["PSIS-LOO"] ==
                                                      np.max(psis_loo_results["PSIS-LOO"])]
          if not CONVERT_TO_PDF:
```

```
    fig, ax = plt.subplots(figsize=(10,5))
    x = np.linspace(-10, 10, 1000)
    df = int(largest_psis_loo_params["prior df"])
    scale = int(largest_psis_loo_params["prior scale"])
    ax.plot(x, st.t.pdf(x, df=df, scale=scale))
    ax.set_title(r'Chosen prior distribution: Student-t($\mu$=0, $\nu$={},
s={})'.format(df, scale));

    plt.tight_layout()
    plt.savefig(os.path.join(figs,'best_prior.png'))
    plt.savefig(os.path.join(figs,'best_prior.eps'), format='eps', dpi=1000)
```

```
In [102]: fit = fit_model(int(largest_psis_loo_params["prior df"]),
                          int(largest_psis_loo_params["prior scale"]))
          samples = fit.extract(permuted=True)
```

```
In [119]: # plot ppc
          if not CONVERT_TO_PDF:
              fig, axs = plt.subplots(1, 1, figsize=(10, 5))
              axs.hist(samples["p_hat_ppc"], bins=42)
              axs.set_title("Posterior Probability for Surviving, \n Expected Value from Data
          {:.2f}".format(np.mean(y_test)))
              plt.show()
              fig.savefig(os.path.join(figs,"ppc.png"))
```

```
In [104]: # plot betas
          if not CONVERT_TO_PDF:
              m = 4
              n = int(math.ceil(len(features)/float(m)))
              fig, axs = plt.subplots(n, m, figsize=(15, 12))
              for i, (ax, feature) in enumerate(zip(axs.flat, features)):
                  ax.hist(samples["beta"][:,i], bins=100)
                  ax.set_title(feature)
              fig.savefig(os.path.join(figs,'betas.png'), bbox_inches='tight')
              fig.savefig(os.path.join(figs,'betas.eps'), bbox_inches='tight', format='eps',
          dpi=1000)
```

6.1.2 Convergence Diagnostic

```
In [105]: filt = ['alpha'] + ['beta[{}]'.format(i+1) for i in range(len(features))]
          fit_logistic_regression = create_fit_table(fit,
          os.path.join(tex,'fit_logistic_regression.tex'), filter=filt)

          if not CONVERT_TO_PDF:
              display(fit_logistic_regression)
```

6.1.3 Sensitivity Analysis

```
In [106]: prior_dfs = [4, 5, 6] # redundant definition
          prior_scale = [1, 2, 4, 10, 20] # redundant definition
          model = stan_utility.compile_model('logistic_regression.stan') # redundant definition

          means = np.zeros((len(prior_dfs), len(prior_scale), len(features) + 1))
          intervals = np.zeros_like(means, dtype=(float, 2))

          if RUN_SENSITIVITY_ANALYSIS:

              for i, df in enumerate(tqdm(prior_dfs)):
                  for j, s in enumerate(prior_scale):
                      fit = fit_model(df, s, debug=False)

                      # filter alpha and betas
                      filt = ['alpha'] + ['beta[{}]'.format(i+1) for i in range(len(features))]
                      filtered = create_fit_table(fit, filter=filt)
```

```
                means[i, j, :] = filtered['mean']
                tuples = [t for t in zip(filtered['2.5%'], filtered['97.5%'])]
                intervals[i, j, :] = tuples

        multi_index = pd.MultiIndex.from_product([prior_dfs, prior_scale],
    names=['prior_df', 'prior_scale'])

        mean_sensitivity = pd.Panel(means).transpose(2, 0, 1).to_frame()
        mean_sensitivity.index = multi_index
        mean_sensitivity.columns = ['alpha'] + features

        spread = np.subtract(intervals[:,:,:,1], intervals[:,:,:,0])
        spread_sensitivity = pd.Panel(spread).transpose(2, 0, 1).to_frame()
        spread_sensitivity.index = multi_index
        spread_sensitivity.columns = ['alpha'] + features

        with open("sensitivity_mean.pkl", 'wb') as f:
            pickle.dump(mean_sensitivity ,f, protocol=2)
        with open("sensitivity_spread.pkl", 'wb') as f:
            pickle.dump(spread_sensitivity ,f, protocol=2)
```

```
Using cached StanModel
```

```
In [107]: if not RUN_SENSITIVITY_ANALYSIS:
              with open('sensitivity_mean.pkl', 'rb') as f:
                  mean_sensitivity = pickle.load(f, encoding='latin1')
              with open('sensitivity_spread.pkl', 'rb') as f:
                  spread_sensitivity = pickle.load(f, encoding='latin1')

          create_tex_table(spread_sensitivity[[f for f in features if 'embarked' not in f]],
          os.path.join(tex,'spread_sensitivity.tex'))
          create_tex_table(mean_sensitivity[[f for f in features if 'embarked' not in f]],
          os.path.join(tex,'mean_sensitivity.tex'))
```

6.1.4 Predictive Performance Assessment

```
In [108]: def logistic(x, beta, alpha):
              return (1+np.exp(-(alpha + np.dot(x, beta))))**(-1)

          def get_y_preds(data, beta, alpha):
              y_preds = []
              for x in data:
                  res = logistic(x, beta, alpha)
                  y_preds.append(1 if res > 0.5 else 0)
              return y_preds

          def check_accuracy(data, target, beta, alpha):
              ans_list = []
              for i in range(len(data)):
                  res = logistic(data[i], beta, alpha)
                  ans = 1 if res > 0.5 else 0
                  ans_list.append(ans == target[i])

              return np.mean(ans_list)

          def check_dummy_accuracy(target, res):
              ans_list = []
              for i in range(len(res)):
                  ans_list.append(res[i] == target[i])
              return np.mean(ans_list)

          mean_list = fit.summary()["summary"]
          beta = mean_list[1:len(features)+1, 0]
          alpha = mean_list[0, 0]
```

```python
predictive_performance = pd.DataFrame()
predictive_performance["Dummy (All 1) training"] = [f1_score(y_train, [1] *
len(y_train))]
predictive_performance["training"] = [f1_score(y_train, get_y_preds(X_train, beta,
alpha))]
predictive_performance["Dummy (All 1) test"] = [f1_score(y_test, [1] * len(y_test))]
predictive_performance["test"] = [f1_score(y_test, get_y_preds(X_test, beta, alpha))]

create_tex_table(predictive_performance, os.path.join(tex,'f1_scores.tex'))

if not CONVERT_TO_PDF:
    display(predictive_performance)
```

6.1.5 my_utilities.py

```python
"""
Provides utility functions for creating figures and tables.
"""
import pandas as pd
from functools import reduce

def create_tex_table(df, dest, decimals=2, index=True):
    """
    Create a .tex table from pandas.DataFrame
    """
    with open(dest, 'w') as f:
        f.write(df.round(decimals).to_latex(index=index))

def create_fit_table(fit, dest=None, filter=[], decimals=2):
    """
    Create a publication ready latex table of StanFit4model

    The table can be included in the notebook with

    \begin{table}
        \caption{Table caption}
        \input{dest}
    \end{table}

    Params:
        fit (StanFit4model)
        dest (str)                Name of the file to save the table. None for not to save
        filter (list[str]):       Names of the keys to include.

    Returns:
        pd.DataFrame
    """

    summary = fit.summary()
    df = pd.DataFrame(summary['summary'],
                      index=summary['summary_rownames'],
                      columns=summary['summary_colnames'])
    if filter:
        df = df.loc[filter]

    if dest is not None:
        create_tex_table(df, dest, decimals)

    return df.round(decimals)

# Filters

def count_matches(seq, match):
    return reduce(lambda agg, p: agg + 1 if p == match else agg, seq, 0)
```

```
def no_info(seq):
    return seq.isnull().sum()

def tot(seq):
    return len(seq)

def females(passengers):
    return count_matches(passengers, "female")

def males(passengers):
    return count_matches(passengers, "male")

def Queenstown(passengers):
    return count_matches(passengers, "Queenstown")

def Cherbourg(passengers):
    return count_matches(passengers, "Cherbourg")

def Southampton(passengers):
    return count_matches(passengers, "Southampton")
```

6.1.6 logistic_regression.stan

```
/**
 * Logistic regression with student's t prior
 */
data {
    int<lower=0> n;                 // number of data points
    int<lower=1> d;                 // explanatory variable dimension
    matrix[n, d] X;                 // explanatory variable
    int<lower=0,upper=1> y[n];      // response variable

    int<lower=1> p_beta_df;         // prior degrees of freedom for beta
    real<lower=0> p_beta_scale;     // prior scale for beta
}
parameters {
    real alpha;      // intercept
    vector[d] beta;  // explanatory variable weights
}
transformed parameters {
    // linear predictor
    vector[n] eta;
    eta = alpha + X * beta;
}
model {
    beta ~ student_t(p_beta_df, 0, p_beta_scale);
    y ~ bernoulli_logit(eta);
}
generated quantities {
    vector[n] log_lik;
    real p_hat_ppc = 0;

    for (j in 1:n) {
        int y_ppc = bernoulli_logit_rng(eta[j]);
        p_hat_ppc = p_hat_ppc + y_ppc;
    }
    p_hat_ppc = p_hat_ppc / n;

    for (i in 1:n)
        log_lik[i] = bernoulli_logit_lpmf(y[i] | eta[i]);
}
```