Predict Survival on the Titanic

Li Qi(lq2156)   Chen Xu(cx2177)

Abstract—The sinking of Titanic is one of the most infamous shipwrecks in history and although there was some element of luck involved, some people were more likely to survive than others. In this report, we use logistic regression to build a model to predict what sorts of people are likely to survive and comparing result with the real situation.

Key words: Python, Data Analysis, Machine Learning, Logistic Regression

I.Introduction

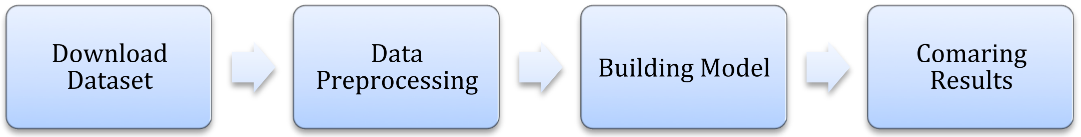
1.1 Project background

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 crews. 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 passengers.

1.2 Project's goal

Our project's goal is to build a predicting model that can predict what sorts of people can survive during this disaster. And then we will compare these results with the real situation and calculate the accuracy.

1.3 Project's Structure



First we download the dataset from the website and then we need to do some data preprocessing to make data useful for our model. The data

preprocessing includes importing data with Pandas, cleaning the data that are useless for our model and then exploring our data through visualizations with matplotlib.
After finishing our preprocessing we will use machine-learning techniques to build a model. We will mainly use logistic regression to build our model and Support Vector Machine to enhance our model.
The final step is to compare the results of our model with the actual results and calculate the accuracy.

II. Related Works

Information is given on a training set of passengers of the Titanic, for which the survival outcome is known. Given the training set information, the goal remains to predict each passenger's survival outcome from a test set of passengers.
We will apply machine-learning tools to build the model. In this case, we may consider approaches based on random forests. And will use Python package to plot the result. The python package used includes:

- NumPy
- Pandas
- SciKit-Learn
- SciPy
- StatsModels
- Patsy
- Matplotlib

2.1Analyzing the data

In the training dataset, there are 891 passengers. Each passenger has 12 attributes. Except the ”passangerId” and ”Survived” attribute, we need to con- sider 10 other attributes and predict the survival possibility.

2.1.1 Take care of missing values

The features Ticket and Cabin have many missing values and so can?t add much value to our analysis. To handle this we will drop them from the data frame to preserve the integrity of our dataset. What's more, we will remove NaN values from every remaining column. Using drop() and dropna() function in could easily achieve goal. Now we have a clean and tidy dataset that is ready for analysis, we cut the dataset from 891 to 712, and get 8 effective attributes to do prediction.

2.1.2 Graphical view of data

The point of this competition is to predict if an individual will survive based on the features in the data like:

• Traveling Class (called Pclass in the data)

• Sex

• Age

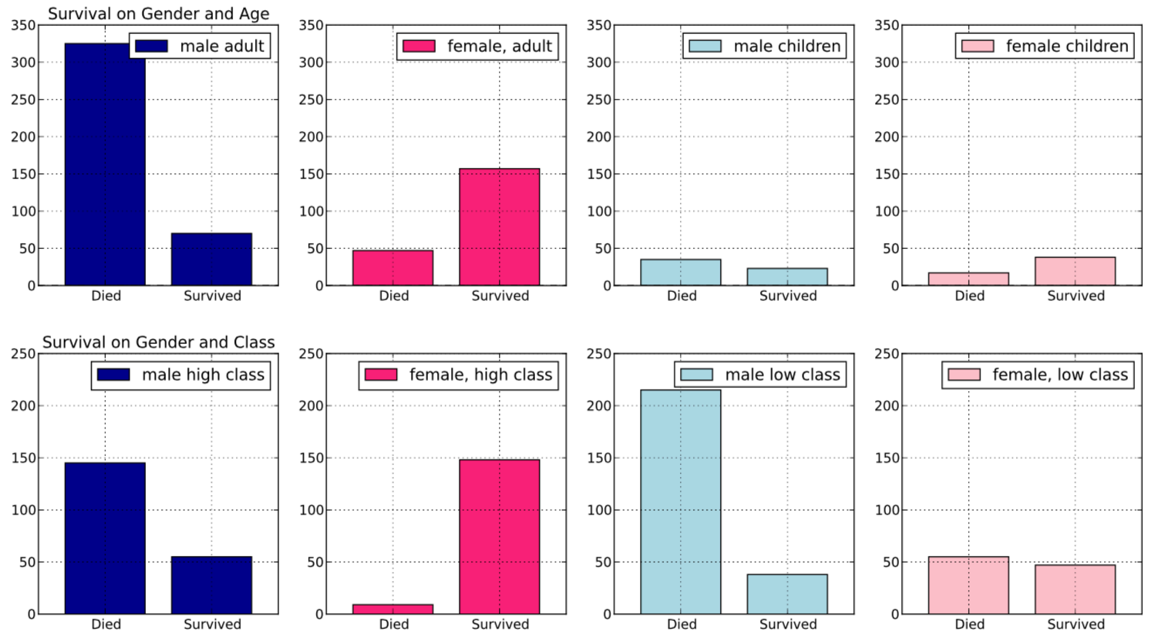


Figure 1: Survival on gender, age and class

III. Dataset Preprocessing

In the film, they tried to rescue women and children first, so a good guess would be on gender. From the graphical view of the data, its clear that although more men died and survived in raw value counts, females had a greater survival rate proportionally(25%), than men (20%).

After applying that all the female would survive, the survival rate become 36.36%, which comes close to the original rate 38.38%. However, we can refine our results by considering other variables.

We ruled out the age factor, as the survival rate doesn't change much with or without the involvement of age. Thus we split the fare class into four payments range and assumes that any group with more than half survivors will always be modeled to survive, while the rest will be mapped to death. The result is different from the previous one in that women in 3rd class who paid more than $20 will not survive, which bring the survival rate closer to training set.

IV. Algorithm,Tools&Model

4.1 Logistic Regression

In particular, we are going to try Logistic Regression. In statistics, logistic regression or logistic regression is a type of regression analysis used for predicting the outcome of a categorical dependent variable. We use python package stats models to build the model and use predictor variable including "Plass", "Sex", "Embarked", "Age", "SibSp".

First, we define our formula for our Logistic regression. In the next cell we create a regression friendly data frame that sets up Boolean values for the categorical variables in our formula and lets our regression model know the types of inputs we're giving it. The model is then instantiated and fitted before a summary of the model's performance is printed. In the last cell we graphically compare the predictions of our model to the actual values we are trying to predict, as well as the residual errors from our model to check for any structure we may have missed. The result as Table 1 shows.

Table 1: Logit Regression Results

	coef	std err	z	P >z	[95.0% Conf.	Int.]
C(Pclass)[T.2]	-1.2673	0.299	-4.245	0.000	-1.852	-0.682
C(Pclass)[T.3]	-2.4966	0.296	-8.422	0.000	-3.078	-1.916
C(Sex)[T.male]	-2.6239	0.218	-12.060	0.000	-3.050	-2.197
C(Embarked)[T.Q]	-0.8351	0.597	-1.398	0.162	-2.006	0.335
C(Embarked)[T.S]	-0.4254	0.271	-1.572	0.116	-0.956	0.105
Age	-0.0436	0.008	-5.264	0.000	-0.060	-0.027
SibSp	-0.3697	0.123	-3.004	0.003	-0.611	-0.129

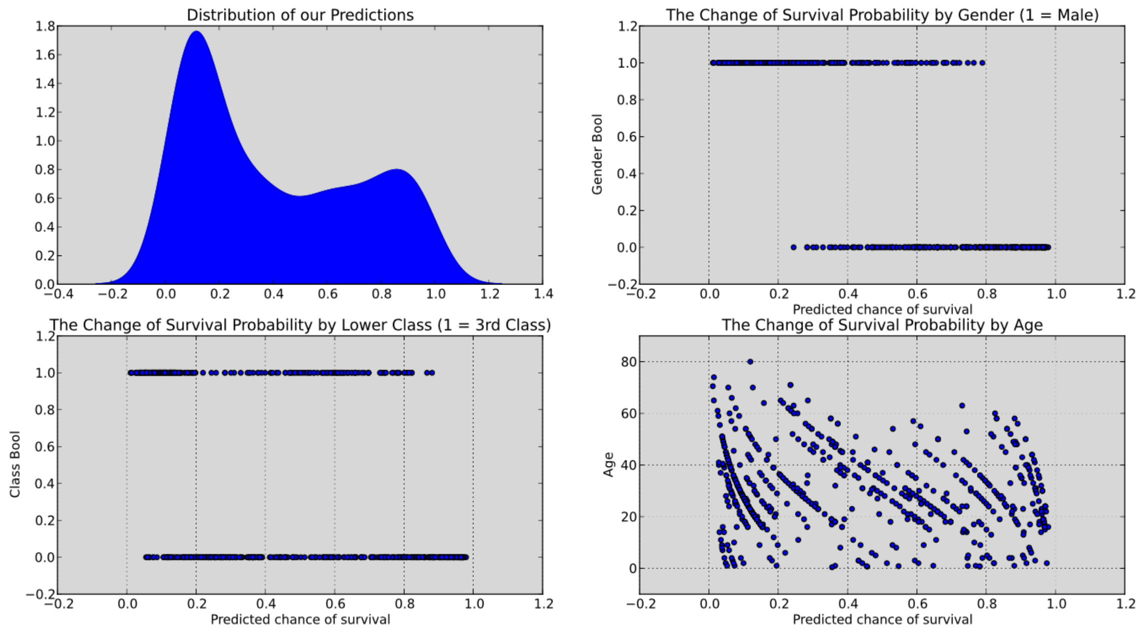


Figure 2: Prediction from Logit Regression

The Figure 2 shows the distribution of our predictions. The second graph also indicates that male has a higher possibility of died while female prefer to survive. This is consistent to our previous observations. Also, We can see from the third graph that ship class has slight effect on the prediction result.

After applying the model on the test data, we get an output file. Comparing the file to the actual result we can conclude that the accuracy is 77.03%.

4.2 Support Vector Machine

The logistic model we just implemented performed well and showed exactly where to draw our decision boundary or our survival cut off'. But when a straight line doesn't cut it, we have to apply a more complex decision boundary like a wave, circle, or maybe some sort of strange polygon, which would describe the variance observed in our sample better than a line. If the prediction of survival were based on age, that could be a linear decision boundary, meaning each additional time you've gone around the sun you were 1 unit more or less likely to survive. But it could be easy to use a curve, where a young healthy people would stand a better chance of survival than elderly or youth.

To get around the fact that our logistic model can only evaluate a linear decision boundary, we could transform our logistic equation from expressing a linear relationship like so:

Survived = β0 +β1P class+β2Sex+β3Age+β4SibSp+β5P arch+β6Embarked

We'll represent this for convenience as: y = x and transform that into a non- linear relationship: log(y) = log(x). An easy way to visualize this by looking at a graph an exponential relationship. Here if we transformed the non-linear by taking the log of our equation, log(y) = log(x^3) we would get a graph like this:

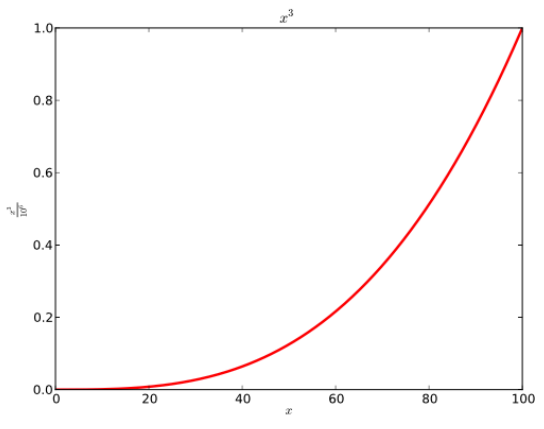


Figure 3: x^3

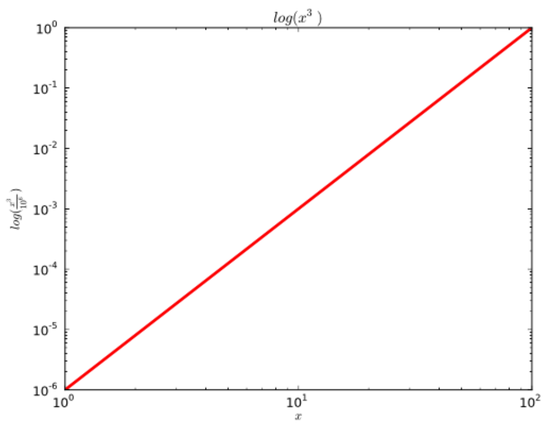


Figure 4: $log(x^3)$

This process of Support Vector Machine transforms models and expresses them in a different mathematical plane. We then implemented SVM model and examined the results after the SVM transforms the equation into three different mathematical planes. The first is linear, and is similar to our logic model. Next is a blank transformation and finally an exponential,polynomial transformation.

Any value in the blue survived while anyone in the red did not. The graph for the linear transformation creates its decision boundary right on 50%, which contribute to the guess from earlier. The remaining decision boundaries are much more complex than the original linear decision boundary. These more complex boundaries may be able to capture more structure in the dataset, if that structure exists, and so might create a more powerful predictive model.

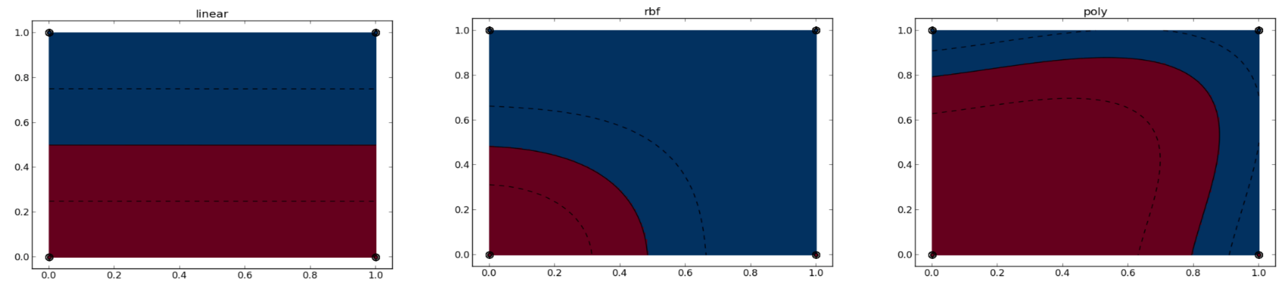


Figure 5: SVM results. Left: Linear kernel. Middle: RBF kernel. Right: Poly kernel

4.3 Random Forest

Random forests are an ensemble learning method for classification (and regression) that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes output by individual trees.

In this case, we will use random forest to predict the survival rate. Using Python package, we just need to initiate the model with training dataset and deal with small points like type matching and missing data filled.

A random forest algorithm randomly generates many extremely simple models to explain the variance observed in random subsections of our data. These models are all awful individually. Really awful. But once they are averaged, they can be powerful predictive tools. The averaging step is the secret sauce. While the vast majority of those models were extremely poor; they were all as bad as each other on average. So when their predictions are averaged together, the bad ones average their effect on our model out to zero. The thing that remains, if anything, is one or a handful of those models have stumbled upon the true structure of the data. The cell below shows the process of instantiating and fitting a random forest, generating predictions form the resulting model, and then scoring the results.

The procedure of using random forest could be concluded as these:

- Import the random forest package
- Create the random forest object which will include all the parameters for the fit
- Fit the training data to the Survived labels and create the decision trees
- Take the same decision trees and run it on the test data

Result shows that the accuracy is about 85.09%.

V. Software Description

We mainly use Python to code in our project. Dataset preprocessing and modeling are two parts of our code.

5.1 Dataset preprocessing

Importing data

```
df = pd.read_csv("data/train.csv")
```

Cleaning data

```
df = df.drop(['Ticket','Cabin'], axis=1)
# Remove NaN values
df = df.dropna()
```

Exploring data

```
# specifies the parameters of our graphs
fig = plt.figure(figsize=(18,6), dpi=1600)
alpha=alpha_scatterplot = 0.2
alpha_bar_chart = 0.55

# lets us plot many diffrent shaped graphs together
ax1 = plt.subplot2grid((2,3),(0,0))
# plots a bar graph of those who surived vs those who did not.
df.Survived.value_counts().plot(kind='bar', alpha=alpha_bar_chart)
# this nicely sets the margins in matplotlib to deal with a recent bug 1.3.1
ax1.set_xlim(-1, 2)
# puts a title on our graph
plt.title("Distribution of Survival, (1 = Survived)")

plt.subplot2grid((2,3),(0,1))
plt.scatter(df.Survived, df.Age, alpha=alpha_scatterplot)
# sets the y axis lable
plt.ylabel("Age")
# formats the grid line style of our graphs
plt.grid(b=True, which='major', axis='y')
plt.title("Survial by Age,  (1 = Survived)")

ax3 = plt.subplot2grid((2,3),(0,2))
df.Pclass.value_counts().plot(kind="barh", alpha=alpha_bar_chart)
ax3.set_ylim(-1, len(df.Pclass.value_counts()))
plt.title("Class Distribution")

plt.subplot2grid((2,3),(1,0), colspan=2)
# plots a kernel desnsity estimate of the subset of the 1st class passanges's age
df.Age[df.Pclass == 1].plot(kind='kde')
df.Age[df.Pclass == 2].plot(kind='kde')
df.Age[df.Pclass == 3].plot(kind='kde')
```

5.2 Modeling

Define formula for our model

```
# model formula
# here the ~ sign is an = sign, and the features of our dataset
# are written as a formula to predict survived. The C() lets our
# regression know that those variables are categorical.
formula = 'Survived ~ C(Pclass) + C(Sex) + Age + SibSp  + C(Embarked)'
# create a results dictionary to hold our regression results for easy analysis later
results = {}
```

```
# create a regression freindly dataframe using patsy's dmatrices function
y,x = dmatrices(formula, data=df, return_type='dataframe')

# instantiate our model
model = sm.Logit(y,x)

# fit our model to the training data
res = model.fit()

# save the result for outputing predictions later
results['Logit'] = [res, formula]
res.summary()
```

Use SVM in our model

```
# create a list of the types of kerneks we will use for your analysis
types_of_kernels = ['linear', 'rbf', 'poly']

# specify our color map for plotting the results
color_map = plt.cm.RdBu_r

# fit the model
for fig_num, kernel in enumerate(types_of_kernels):
    clf = svm.SVC(kernel=kernel, gamma=3)
    clf.fit(X_train, y_train)

    plt.figure(fig_num)
    plt.scatter(X[:, 0], X[:, 1], c=y, zorder=10, cmap=color_map)

    # circle out the test data
    plt.scatter(X_test[:, 0], X_test[:, 1], s=80, facecolors='none', zorder=10)

    plt.axis('tight')
    x_min = X[:, 0].min()
    x_max = X[:, 0].max()
    y_min = X[:, 1].min()
    y_max = X[:, 1].max()

    XX, YY = np.mgrid[x_min:x_max:200j, y_min:y_max:200j]
    Z = clf.decision_function(np.c_[XX.ravel(), YY.ravel()])

    # put the result into a color plot
    Z = Z.reshape(XX.shape)
    plt.pcolormesh(XX, YY, Z > 0, cmap=color_map)
    plt.contour(XX, YY, Z, colors=['k', 'k', 'k'], linestyles=['--', '-', '--'],
                levels=[-.5, 0, .5])

    plt.title(kernel)
    plt.show()
```

VI. Conclusion

In this project, we tried to use big data knowledge to predict the survival rate on Titanic based on the passenger's information including sex, class, age and so on. We build several models, from the most intuitive model about age and male, and following supervised machine learning model logistic regression, to the final unsupervised machine-learning model including SVM and random forest. We learned a lot about using Python package to data processing like Pandas, Numpy, Patsy, they are really helpful and powerful. We also learned how to use matploit to draw professional graphs.

Even though we keep trying more and more complicated model, we didn't get a significant improvement on the results. This is somehow frustrating, but it also inform us following tips:

• Sometimes, concise, simple views of data reveal their true patterns and nature, this means simple model could also get a good performance.

• Performance of model is closely related to the structure of dataset. A small dataset and simple binary result may limit the performance of complicate model.

VII Future work

Even though we have tried some models both in supervised and unsupervised machine learning, there are some more methods and models we would like to try for this dataset.

First is that we could use software Weka to do the analysis. We didn't do it for this project mainly because we can use it, but hard to learn the knowledge from the reason. So it would be better to implement it by us. However, playing the dataset in Weka will be effective and easy to tried different technology.

Second is that we can try dig deeper for each model and take more trial for different predictor. Or we can think about more ways to deal with empty data since currently we ignore incomplete data that make a negative effect on training dataset.

VIII Reference

[1] http://en.wikipedia.org/wiki/Logistic_regression
[2] http://en.wikipedia.org/wiki/Support_vector_machine