Predicting Survival on Titanic

Sheng Hsin Hsu – Master of Science

1752559

Fairleigh Dickinson University

Predicting Survival on Titanic

In this article, we are going to purpose a practice to predict if a person survival or not in the sinking of Titanic by using machine learning algorithm from Kaggle.com: Logistic Regression and Support Vector Machine with Gaussian kernel and we will analyze the performance and compare the performance of both methods.

Introduction

The dataset contains different features about the passengers who were on the Titanic. In this Titanic dataset, it consists with 11 features and we obtain totally 891 data. First, we must analyze and clean up the features before we fit our data into machine learning algorithm. The goal of preprocessing features is to eliminate the redundant features, clean the missing value and convert the string features to the number, due to both algorithm cannot take a string and NaN value as input. While doing the step of analyzing, we extract the meaningful information from data and look at relationship between them such as the standard deviation, mean and number of NaN value…etc. After we preprocesses the data, we will fit our data to the algorithms.

	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Figure 1: Data information for the numerical data

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Figure 3: First 3 row of the Titanic dataset

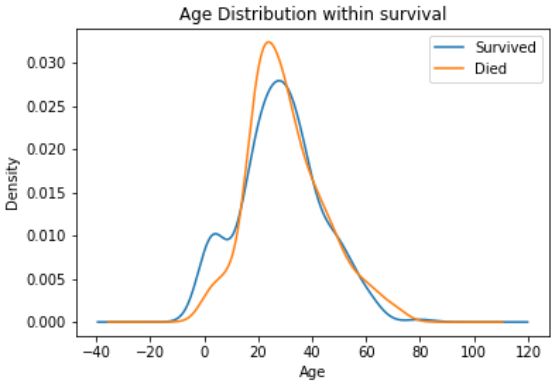

Figure 2: Age Distribution within survival: We can see must of people belong to the age between 20 to 40

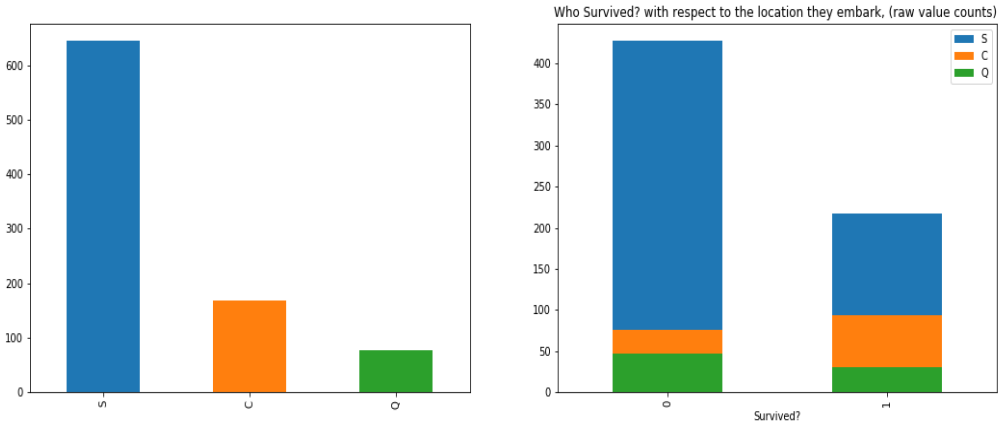

Figure 4: Number of people in different embark location. Right show the number of people, who survived or died with the different embarked location

Date wrangling: Data Preprocessing

As figure 1 and figure 2 shown, the data consist with 4 issues: redundant feature, missing data, unusable data and different feature scaler. We will discuss those issues in the following:

1. Redundant Features: Cabin

We can see the feature PassengerId is redundant because it duplicates from the index. Second, the feature Cabin contains 687 missing value, which is about 77.1%. Cabin has too much missing value, thus it's not good practice to fill the missing value. It is will cause significantly problems that we continuously to use this feature. At this point we will drop this feature. Next. we are going to extract gathering more features from our features.

2. Extract Features from Features: Embarked, Name. PClass

We have three features which is string or categorical features: embarked, PClass and Name. First, we convert the Embarked and PClass to dummy features, as shown as below:

$$\textbf{Embarked}: \begin{matrix} S \\ C \\ Q \end{matrix} \rightarrow \begin{matrix} S & C & Q \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \qquad \boldsymbol{Pclass}: \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \rightarrow \begin{matrix} Pclass:1 & Pclass:2 & Pclass:3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

Second, we need to see the relationship of the Name features. We observe that we can extract the 17 different kind of prefix from the Name features as following : 'Mr', 'Mrs', 'Miss', 'Master', 'Don', 'Rev', 'Dr', 'Mme', 'Ms', 'Major', 'Lady', 'Sir', 'Mlle', 'Col', 'Capt', 'the Countess', 'Jonkheer'. This can be useful for us to expand our dataset. We will categorize those into:

Title_Dictionary = { "Capt": "Officer", "Col": "Officer", "Major": "Officer", "Jonkheer": "Royalty", "Don": "Royalty", "Sir" : "Royalty"," Dr": "Officer", "Rev": "Officer", "the Countess": "Royalty", "Mme": "Married", "Mlle": "Unmarried", "Ms": "UnKnow_Married", "Mr" : "UnKnow_Married", "Mrs" : "Married", "Miss" : "Unmarried", "Master": "Young", "Lady": "Royalty"}

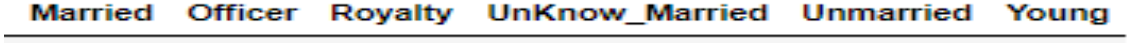

Figure 5: Extract meaningful feature from our name features.

We can use this as a new feature and convert it into dummy features as figure 5, but we need to further check if the marry status is correct, which if Married = 0 and Unmarry = 0, then UnKnow_Married should be 1.

3. Missing Value: Age and Embarked

The feature of Embarked has 2 missing and the feature of age has 177 missing. We have 3 options to conduct the missing value: simply drop them, fill with the most value/mean value or use a machine learning algorithm to generate the missing value. Due to it just has the few number of missing value in Embarked, we will drop the feature. However, the feature Age contains 177 missing value, we will use Linear Regression to produce the missing value. Below we introduce Linear Regression:

- Linear Regression

  The goal of the Linear Regression is to predict the numerical output. We put our data into the following cost function to compute the cost:

$$\mathrm{J} = \frac{1}{2 \times \mathrm{m}} \sum (H(\theta) - Y)^2, \; H(\theta) = \theta^T X, \quad R^{n \times m}$$

  Where θ is the weight for the features, m is the number of the examples we have, and X is our training dataset. H (X) is the predicted output and J will be our cost function. Y is our correct result. We want to $\min_{\theta} J$. We will use gradient descent to converge to find the θ. Here is the gradient decent:

$$\theta \coloneqq \theta - \frac{\delta J}{\delta \theta}$$

$$\frac{\delta J}{\delta \theta} = \sum (H(X) - Y) \cdot X$$

  We will repeatedly update the $\theta \; with \; \frac{\delta J}{\delta \theta}$ until finding the minimum optima.

We will split the Non-Null value into training dataset and cross-validation set (70% and 30%), because we need to use cross-validation set to test our performance. After we fit our data to the Linear Regression, we will see the result as below:

	Mean Square Error	R^2
Training Set	117.1080	0.44
Cross-Validation Set	131.152	0.39

Table 1:Predicting linear regression performance for training set and cross-validation

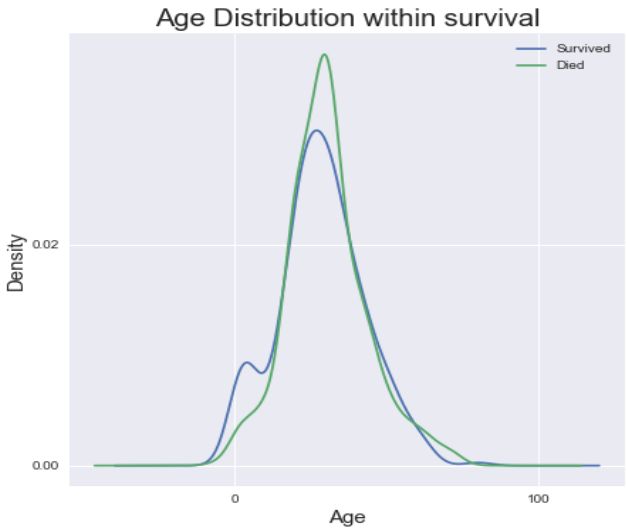

Figure 6: The Age distribution after we fill the missing data by using linear regression. We can see the distribution does not change a lot. We will consider this approach successfully

However, the cost is a little bit of high, but according to the figure 5, this approach seems successfully predict the missing value in age seems the distribution doesn't change significantly.

Features Scaler: Standardization

We can see most of our data land in range between 0 to 1, but Age and Fare don't have the same scale as others do. Therefore, we use Standardization to scale our features in training dataset and cross-validation dataset the dataset using the following formula:

$$X^{(i)} = \frac{X^{(i)} - X_{min}}{\sigma_x}$$

σ_x is the standard deviation of the x.

Fit the Model

We clean up the missing data and make all the features to numerical. The next step we are going to use Logistic Regression to learn from our data.

Logistic Regression

The goal of the Logistic Regression is to predict the categorical output. The Usage of the algorithm is similar, but there are few different. The cost function will be:

$$J = \frac{1}{\mathrm{m}} \sum \big(Y \cdot \log H(X) + (1 - Y) \cdot \log\big(1 - H(X)\big)\big)$$

Where θ is the weight for the features, m is the number of the examples we have, and X is our training dataset. If the predicted value is 1 and expected value is 1, the cost will be close to 0 whereas if the predicted value is 1 and the expected value is 0, the cost will be close to the finite. The H (X) will be:

$$H(X) = g(\theta^T \mathrm{X}), \qquad g(\mathrm{Z}) = \frac{1}{1 + e^{-Z}}$$

H (X) is the predicted result and J will be our cost function. G(Z) is the sigmoid function. Y is our target. The sigmoid function will generate the value between 0 to 1. We want to $\min_{\theta} J$ so we use gradient descent to converge to find the θ. Here is the gradient decent:

$$\theta := \theta - \frac{\delta J}{\delta \theta}$$

$$\frac{\delta J}{\delta \theta} = \sum (H(X) - Y) \cdot X$$

We will repeatedly update the θ $with$ $\frac{\delta J}{\delta \theta}$ until finding the minimum optima.

Performance

After we fit the mode with the Logistic Regression, first we compare the accuracy between cross-validation set and the training set:

	Accuracy
Training Set	84.1%
Cross-Validation Set	81.01%

Table 2: Performance for logistic regression. Overall, it performances well.

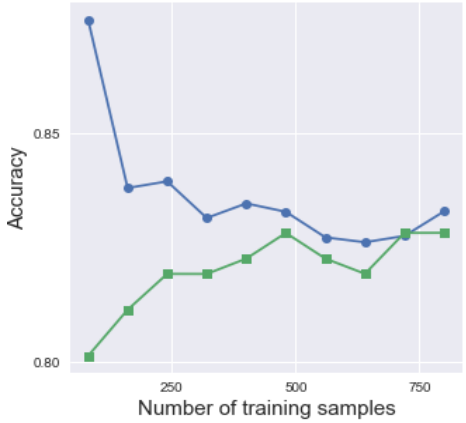

Figure 7: Learning Curve. We train the model with different size of the training dataset and predict the training and cross-validation dataset. As we can see, the model performance really well for both

The gap between two lines is small, which indicates that the model doesn't suffer from the overfitting and the accuracy is pretty good. Overfitting, also called high variance, is the situation when the performance is well in training set, but bad on cross-validation set.

Support Vector Machine with Gaussian Kernel (SVM)

This time we use SVM with Gaussian kernel to train our data, and SVM compare with logistic regression. Alternative to Logistic Regression, the SVM will try to make the decision boundary to have large margin as shown in figure 8.

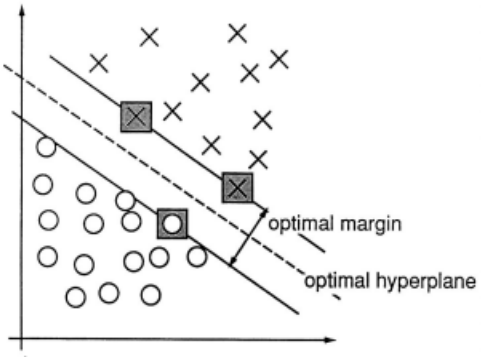

Figure 8: An example of a separable problem in a 2-dimensional space. The support vectors marked with grey squares, define the margin of largest separation between the two classes (Corinna, C., V;adomir, V., 1995)

Performance

The purpose of the kernel is make the decision boundary to be non-linear. Let see if we improve the accuracy using SVM kernel.

	Accuracy
Training Set	84.47%
Cross-Validation Set	82.01%

Table 3: Performance for the SVM.

The table 3 doesn't shows too much different from Logistic Regression. However, the model has a significant overfitting shown on figure 9:

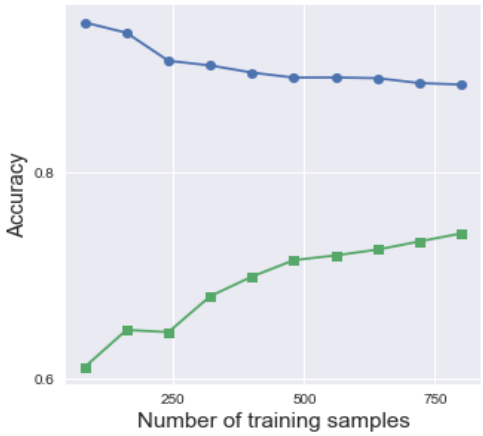

Figure 9: The learning curve for SVM.

The gap between two lines is too large, which indicate the SVM suffer from overfitting. Due to the significant overfitting in SVM, Logistic Regression is better to be used in this specific

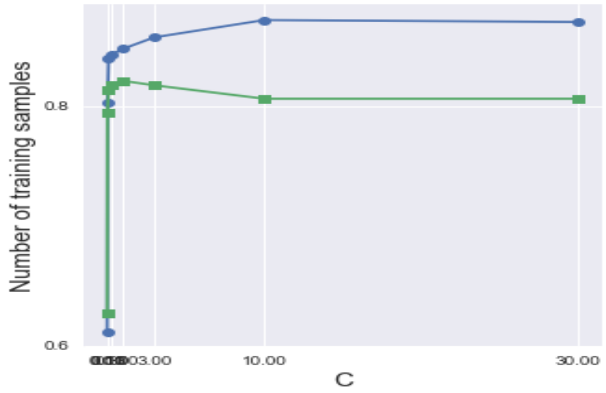

Figure 10: Tuning C. We can see even though we tune the C with different value, the model still suffers from overfitting.

problem. There is some technique to fix the problem such as acquiring more examples, or increase the parameter; however, we are not able to gain more examples, thus we will try to tune the regulation parameter C. The result shows as follow:

We can see the model doesn't improve by increasing C and we are impossible to obtain for more examples. Therefore, we can conclude that in this specific case, the Logistic Regression performance better than SVM.

Conclusion

While we obtain the data, we first glance over the all the dataset and visualize the data which can be more convince for us to analyze the information in the data. After understanding the examples, we clean up the dataset to make the dataset available to fit into the algorithm. We can see we three major problems in our dataset: redundant feature, unusable features, missing value and vary scale between feature. To solve situation of redundant feature, we simply drop the column. Next, the feature Name and Embarked are not usable due to it is String. We convert this feature to dummy features for the Embarked value and we extract the prefix from Name to generate new features. Last thing to clean up is Age, Age contains about 20% missing value. We are going to use linear regression to predict the missing value. As shown in figure 6, the distribution of the age does not change significantly.

Second, after cleaning up the data, we fit the data to Logistic Regression to predict if the passenger is survived or not. The accuracy of this model for training set is 83.39% where the cross-validation set is 81.17%. According to the result, we can see it performance well, though accuracy is a little bit low; however, it doesn't suffer from the overfitting whereas we use the SVM significantly suffer from the overfitting. We will use Logistic Regression to predict our new data.

All in all, we have an opportunity develop the machine learning project in the real data. The real task is more complex than the text book's example which the data is messy, and the model doesn't performance as we expected. We have to analysis the data and understand the data before we start to learn whereas in the text book, the data doesn't need to preprocess. When the data fit into the machine learning algorithm and the performance of the data doesn't usually like what we expected. We can use the learning curve help us to see if the model is overfitting or underfitting. However, with this opportunity, we can have a real experience to enjoy tuning the parameter and work on the training process.

References

Corinna, C., V;adomir, V. (1995). Support-Vector Network. *Kluwer Academic Publishers*,.

Kaggle.com (2012). Titanic: Machine Learning from Disaster. https://www.kaggle.com/c/titanic

Sebastin, R. (2015). *Python for Machine Learning.* PACKET.