Titanic - Machine Learning from Disaster

Minting Chen
Xiaoyang Pan

March 23, 2021

Abstract

Titanic shipwreck is a famous historical tragedy, which took away around $1,500$ passengers' life. This project aims to build machine learning models to predict which passenger on the Titanic survived based on the following parameters: ticket class, sex, age, cabin number, fare, the port of embarkation, and the number of siblings and parents on board. In this project, we built five common Machine Learning models to output predictions, including Logistic Regression Model (LRM), Support Vector Machine (SVM), and Random Forest (RF), K-Nearest Neighbors (KNN), and Self-Organizing Map (SOM). Based on the accuracy of each model, Random Forest performs the best, having 77% accuracy. The accuracy is boosted to 78.7% after we remove the features with less than 0.3 correlation with the output. Last but not least, we also discuss the advantages and disadvantages of each technique in the report.

1 Research Problem

When the Titanic Disaster happened in 1912, 1502 out of 2224 passengers died. While there might be some factors of luck involved in surviving, some people were more likely to survive than others. So we investigate what the real factors that make a person more likely to survive are in this project. The dataset from Kaggle provides information about passengers, such as name, age, sex, the number of siblings and parents on board, etc. With that information, we aim to build models to predict which passengers survived the Titanic shipwreck.

2 Why is this problem important?

This project is a great topic to get us started on machine learning algorithms. Moreover, building ML models to predict which passengers survived the disaster and analyzing essential factors that influence survival have the potential to increase the survival rate in a disaster that is similar to the Titanic shipwreck. For example, if we find out that ticket class has a major impact on survival rate, ship companies should replenish equipment and resources for classes with low survival rates so that more passengers can survive a disaster.

3 Related Work

Singh et al. [1] utilized machine learning algorithms to predict the survival of passengers in the Titanic Disaster by using the same Titanic dataset from Kaggle. Before they built models to predict the survival of passengers, they found out there was a lot of missing data for age and cabin columns. The cabin column was dropped from the data set since most values were missing. Additional columns liked Name, Ticket, Cabin, and Embarked were removed from the data because the author believed that these variables were not relevant. Then, new features include mother, children, family size, and respectable were added to the data. At the same time, the missing values of Age were assigned by the average age of a group of people who had the same titles, such as Mrs, Mr, and Miss; the value of Sex column was replaced by 0 and 1(0 for male and 1 for female)

The rest of the features were taken as parameters for machine learning algorithms. The accuracy for the prediction models using Naive Gaussian Bayes, Logistic Regression, Decision Tree, and Random Forest were 91.39%, 94.26%, 93.06%, 91.8%, respectively. Singh et al. had also concluded that pclass, sex, age, children, and sibsp were the features are correlated with the prediction of survival, and more algorithm, such as Support Vector Machine and K-Nearest Neighbors, can be used to predict the survival of passengers.

Kakde and Agrawal [2] also analyzed the Titanic Passenger dataset from Kaggle to classify whether a passenger would survive or not. They first filled the missing value of Age, Cabin, and Embarked with random sample values from each category. Then, they compared individual features with the survival of passengers to have an initial understanding of the correlation. They decided to extract the title from the name column to be a new feature; they also combine the parch and sibsp column to be another new feature called family size. They selected features that include age, sex, cabin, title, Pclass, family size, fare, and embarked to implement machine learning models because they found little or no impact from other features on the survival of passengers. Three methods, Logistic Regression, Random Forest, and Decision Tree, and Support vector machine, were implemented. The evaluation of each model was done by examining the confusion matrix(sensitivity, specificity, positive predictive value, negative predicted value) and accuracy. Sensitivity is the percentage of actual positive which are correctly identified; specificity is the percentage of negative values which are correctly defined. Among all models, the logistic regression presented the highest accuracy of 83.7%, while the support vector machine, Random Forest, and Decision Tree yielded an accuracy of 83.2%, 82.6%, and 81.7%.

4 Proposed Solution

The implementation of this project is done in JupyterLab using Python. We also utilize the *sklearn* toolkit for building models and *matplotlib* for visualizing outputs. To have a better understanding of the problem and correlations between each feature and the survived status, we generate a correlation map shown in

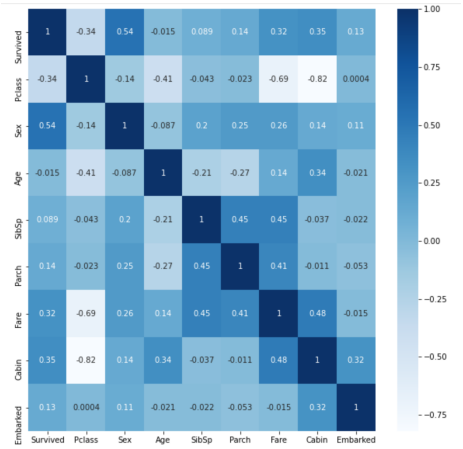

Figure 1: Correlation Matrix of features and survival status from Titanic

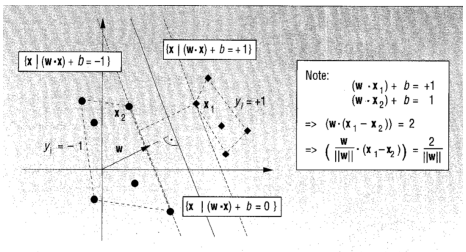

Figure 2: Illustration of SVM cited from [4]

Figure 1. Survived status does have a strong correlation with all features except age. That conclusion we draw from the correlation map is surprising because we commonly think that elderly are less likely to survive due to slow motion.

After understanding the dataset, data cleaning has to be done before building the actual models. Similar to [1], we decide to drop the name, ticket, and ID columns because those are unique to each passenger so they are not relevant to the model. Unlike [1] where cabin, age, and embarked features were removed due to missing data. We decide to keep those features to build our model to maximize useful information. We choose to deploy DataWig library [3] to run machine learning algorithms on those columns to predict missing entries. (more detail is discussed in the Method section) The final step of data cleaning is to convert the categorical features to numerical data for building models. Then we research the five ML technique introduced above (Logistic Regression Model, Support Vector Machine, and Random Forest, K-Nearest Neighbors, and Self-Organizing Map) and understand how to deploy them in this project. We determine the best performing model by comparing the accuracy and identify potential reasons for the variation of the accuracy of models. Finally, the performance of each model is discussed.

5 Methods

5.1 Data Preparation

Four common methods for handling missing data are (1) replacing missing entries by the mean of the feature, (2) deleting the entire row that contains missing data, (3) creating a new category for missing data, and (4) using machine learning algorithms to impute missing data. In the train set, the age feature misses 177

out of 891 entries and the cabin feature misses 687 out of 891 entries. Since a significant amount of data is missing for both features, the first three methods for recovering missing data are not suitable in this case. For example, the mean that is calculated based on incomplete data is not accurate; a lot of information is lost if we delete the rows; creating a new category for a large amount of missing data induces new confusion to the model. Therefore, we decide to use the DataWig library [3] that run machine learning algorithms to predict the values for missing data. We use the ticket class, number of parents and children on board, and number of siblings on board to determine the age of the passenger; to recover missing data from the cabin feature, we use ticket class, fare, and port of embarkation.

The process of choosing the feature is quite simple in this project. Out of the nine features given in the data-set, we decide to keep all the features except passenger name and ticket number because these are unique to every passenger; therefore, they are not very helpful in the training process.

To train any model, each value needs to be numeric. The factorize function from the pandas library is utilized and it converts categorical data into numerical data. The next step is to examine various types of machine learning algorithms.

5.2 Support-Vector Machine

Support Vector Machine (SVM) [5] is a supervised machine learning technique. We use SVM with a linear kernel. SVM maps the data to a higher dimensional feature space, where two classes can be better classified. The decision function is $f(x) = sign((w * x) + b)$ as defined in Figure 2 [4], where x is an input vector, w is a weight vector and b is a scalar in the feature space such that $y_i * ((w * x) + b) > 0$ where y_i is the class. The goal of SVM is to separate two classes (survived/ not survived) and maximizes the distance between them to avoid misclassification. For the implementation of the model, we utilize the sklearn.SVM toolkit. Then, we compare the prediction produced by our model with the real outputs given by Kaggle to calculate the accuracy.

The confusion matrix below shows that we predict all the samples correctly which results in an accuracy of 71%. The confusion matrix is shown in Figure 3. The accuracy of SVM is not high due to various reasons like the linear kernel might not be the best choice in this case. Also, SVM does not suitable for a large dataset because the computation complexity of the SVM algorithm can be intensive when the number of samples goes to infinity.

5.3 Self-Organizing Map

A self-Organizing map(SOM) is an unsupervised learning technique. It produces low dimension, 2D in most cases, representation of the input data. The row size of the 2D will be x, and the column size of the 2d map will be y. Each node in the map representation has i weights, where i is the number of input data features, and their weights are randomly assigned at the beginning. Then, a vector was at random from the data set to compute the Euclidean Distance

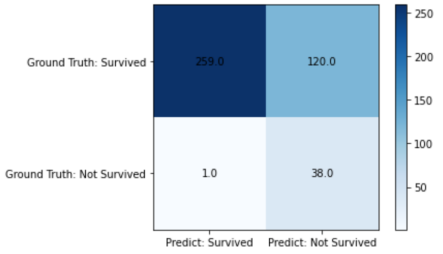

Figure 3: Confussion Matrix for SVM

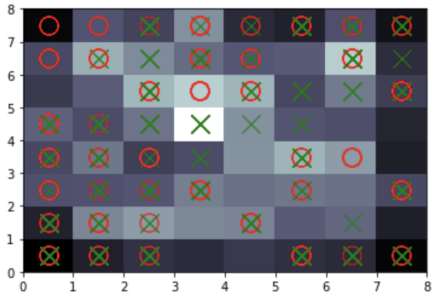

Figure 4: SOM 8*8 output map representation: X means not survived,O means survived.

between the vector and each node in the 2d map. The node with the smallest value of the euclidean distance is called the Best matching unit (BMU). Weights of nodes around the Best matching Unit are changed. The initial radius r_0 is determined as max(x,y)/2. The radius decays exponentially over each iteration as following $R(t) = r_0 * \exp(\frac{-t}{\lambda})$, where t is a number of iteration, r_0 was the initial radius, and λ is a time constant. Nodes inside the radius have their weight updated as following: $W(t+1) = W(t) + \Theta(t)L(t)(D(t) - W(t))$,where D(t) is the euclidean distance, W(t) was weight, $L(t) = l_0 * \exp(\frac{-t}{\lambda})$ and l_0 is the initial learning rate, and $\Theta(t) = \exp(\frac{-d^2}{2R^2(t)})$ and d is the Euclidean distance between the node and the best matching unit. Then, the radius is decayed according to R(t) and only nodes within the smaller radius have their weights changed. Weights are updated until the radius vanished. Then, another vector would be chosen to repeat until a desired number of iteration is reached.

SOM maintains the reveals correlation that is not easily identified; it also classifies data without supervision. However, it might be not suitable for this problem. The output map was shown in Figure 4. A green X means not survived, and a red O means survived. Multiple nodes have both X and O. In other words, the survival of a test input can not be approximate because multiple nodes in the input data can be predicted as either survived or not survived. The prediction of survival might be misleading. Also, if the size the 2d map increases, there are still some results overlapped and more space with no classification, which could lead to errors. Therefore, SOM will not be considered to predict the survival of passengers. Nonetheless, SOM will be better to predict the outliers in the input data, such as the white square in Figure 3, and classify numbers of input data.

5.4 Logistic Regression Model

The logistic regression model is a classification algorithm that approximate $P(Y|X)$ by learning. The Assumption made is $P(Y|X)$ can be estimated as a sigmoid function as following $P(Y = 1|X = x) = \sigma(\theta^T x)$,where $\theta^T x = \sum_{i=1}^{n} \theta_i x_i$,

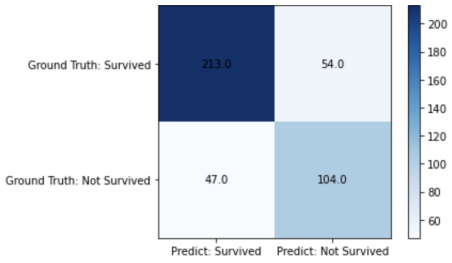

Figure 5: Confusion Matrix for Logistic Regression Model

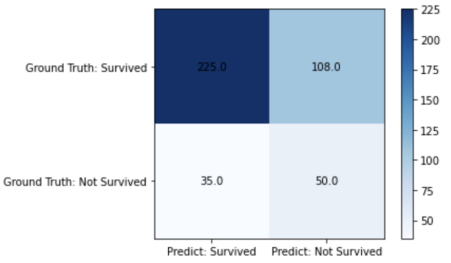

Figure 6: Confusion Matrix for K-Nearest Neighbors with p = 1

and n is the number of input features. And $P(Y = i|X) = (\sigma(\theta^T x)^y * (1 - \sigma(\theta^T x))^{1-y}$. The log-likelihood of P(Y|X) is used for updating the value of θ. By applying the gradient descent method, the derivative of the log likelihood is computed with the formula $\frac{\partial L(\theta)}{\partial \theta_j} = \sum_{i=1}^{n}[(y - \sigma(\theta^T x_i))]x_j$,where $L(\theta)$ is the log likelihood of P(Y=i|X). Then, the $\theta_j^{new} = \theta_j^{old} + \eta * \frac{\partial L(\theta^{old})}{\partial \theta_j^{old}}$, where η is the learning rate. The learning process will stop once the value of θ converges.

The missing data of the test data are filled in the same way the training data do. Then, the trained logistic model takes the test data as input. The predicted outputs are compared with the ground truth from Kaggle, and the accuracy is 75.83%, while the confusion matrix is shown in Figure 5.

5.5 K-Nearest Neighbors

The K-Nearest Neighbors(KNN) is a supervised learning algorithm that does not require training before making predictions. With the given training data with labeled information, the data can be a breakdown as survival or not survival, and all other information such as sex, age, ticket, etc, are viewed as their coordinate in space. Two main parameters for KNN are K, the number of nearest data points(neighbors), and the distance function, Euclidean Distance, Manhattan Distance, or Minkowski Distance. The distance metric for two-point x and y with dimension k is defined as below:

Euclidean Distance = $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$, Manhattan Distance = $\sum_{i=1}^{k}|x_i - y_i|$, Minkowski Distance = $(\sum_{i=1}^{k}(|x_i - y_i|)^p)^{\frac{1}{p}}$.

The parameter p in Minkowski value can be any real value, but it is typically chosen to between 1 and 2.If p = 1, Minkowski Distance is equivalent to Manhattan Distance; if p = 2, Minkowski Distance is equivalent to Euclidean Distance. As new data come in, the distance from the new data to each existing data can be computed to identify the near data points and their categories. Then, the new data is assigned to the class where a majority of the neighbors are. The value of K should be odd to avoid a tie for prediction and usually is set the $\sqrt{n}$, where n is the total number of data.

The KNN algorithm is implemented with $\sqrt{n}$ for the value of K and Minkowski Distance for the distance metric. The value of p ranged from 1 to 2 are evaluated, and p value of 1 presents the best accuracy, 65.79%, and p value of 2 presents the worst accuracy, 65.31%. The Manhattan Distance yields better results because the data is high dimension. The data might not be near the mean of the distribution, and the euclidean distance might exaggerate the discrepancy by adding all the square of distance difference. The data matches with this intuition. The confusion matrix for this approach are shown in Figure 6.

5.6 Random Forest

Random Forest is one of the most efficient and simplest ML techniques that we commonly use. RF is made of multiple decision trees, and each tree produces a prediction; then the prediction with the highest number of voting is the actual output of the model. The use of multiple trees reduces the risk of over-fitting. More importantly, RF maintains high accuracy even when a large portion of the data is missing because it can adapt to the data structure taking into consideration the high variance or the bias. The main goal of RF is to reduce entropy, which is the measurement of randomness in the dataset. First, we partition the dataset into small subsets which reduce the high entropy to a lower value, then we keep splitting the dataset based on the entropy value and gain. When a subset has entropy closed to 0, no further splitting is needed. We again use the sklearn.RandomForestClassifier to build the model. The confusion matrix is shown in 7, and the accuracy of Random Forest model is 77%.

5.7 Result Analysis

After training our models with eight features, we decide to drop four out of eight features that have low correlation to the survived status shown in Figure 1, which are age, number of siblings on board, number of parents on board, and port of embarked. Their correlation to survived status are less than 0.3. Figure 8 shows the accuracy of the four algorithms before and after we remove those unnecessary features. We see that accuracy of SVM increases the most due to removal of features. After we removed the unnecessary features, the data are mapped to a lower dimensional features space compare to before, where the noise is reduced; therefore, the SVM classifier performs better than before. A similar conditions could apply to the other three model, KNN, Random Forest,and Logistic Regression model, because some correlations in high-dimension data might be spurious; the decision boundary of classification were much complicated and error prone. Therefore, the performance accuracy for each model increased by a percentage ranging from 1 to 3 after dropping the four features.

We also add a new column called family size, which is the sum of the number of siblings and the number of parents on board plus one. However, the accuracy does not improve due to this additional feature; therefore, we decide to drop it.

The best performance of our model is approximately 80%, which is similar to the performance of models from Kaded and Agrawal [2]. We have tried something

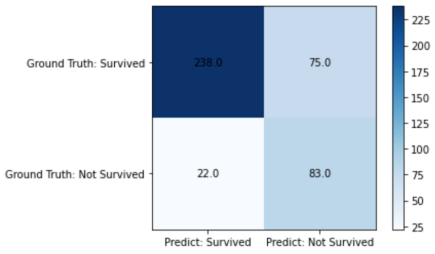

Figure 7: Confusion Matrix of RF

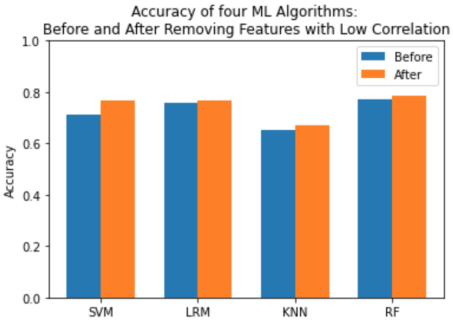

Figure 8: Accuracy of four ML algorithm before and after removing features with correlation < 0.3

similar to Singh et al. [1] by add new features, but the accuracy does not reach 90% or higher, the only time it happens is fitting the model to a sample data output from Kaggle called "gendersubmission.csv", which is not actual survival information for the testing data.

6 Conclusion

We investigate the performance of five machine learning models for predicting survived status of passengers in the Titanic shipwreck. By following the procedure of data analysis, data cleaning, building models, and accuracy comparison, we found that Random Forest performs the best in this case with a 78.7% accuracy. Future improvement could be done by adding new features into the dataset, like family size, and exploring more advanced machine learning algorithms. The link to Python implementation of the project: https://github.com/mintingchen/ECE269_Project

7 Tentative Timeline

We will follow the schedule listed below to complete our project:

Date	Weekly Goal
Jan 27 - Feb 2	ML Strategies research (SVM, SOM, KNN, RF, LRM)
Feb 3 - Feb 17	Implement SVM,SOM, and LRM models
Feb 18 - March 2	Implement KNN and RF models
March 3 - March 10	Improve current models
March 11 - March 23	Documentation

References

[1] A. Singh, S. Saraswat, and N. Faujdar, "Analyzing titanic disaster using machine learning algorithms," in *2017 International Conference on Computing, Communication and Automation (ICCCA)*, 2017, pp. 406–411.

[2] Y. Kakde and S. Agrawal, "Predicting survival on titanic by applying exploratory data analytics and machine learning techniques," *International Journal of Computer Applications*, vol. 179, pp. 32–38, 05 2018.

[3] F. Biessmann, T. Rukat, P. Schmidt, P. Naidu, S. Schelter, A. Taptunov, D. Lange, and D. Salinas, "Datawig: Missing value imputation for tables." *Journal of Machine Learning Research*, vol. 20, no. 175, pp. 1–6, 2019.

[4] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[5] W. S. Noble, "What is a support vector machine?" *Nature biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006.