Survey of Machine Learning Algorithms to Predict Survival of Passengers on the Titanic Dataset

Mahmood Rezaee Qotb Abadi 40217213
Soroush Merikhy 40216026
Pouria Pirian 40207625

Abstract

Titanic, the most powerful ship ever built at the time, had a miserable day on April $14^{th}, 1912$*. Many passengers were sinking, but the reason for survival remains unclear. This project aims to demonstrate how machine learning techniques can classify surviving people to predict their survival. Several data pre-processing and feature engineering methods are implemented on the Titanic dataset to improve the raw data and change them to a dataset that is more optimal and useful in the classification task. A survey is done on machine learning classification algorithms, namely logistic regression, K-nearest neighbors (KNN), support vector machine (SVM), Gaussian naive Bayes, random forest, and artificial neural network on the pre-processed dataset. For each algorithm, the performance of the algorithm in the classification task is improved by methods such as selecting the optimal subset of features, hyperparameter search and tune, and cross-validation. Lastly, the results from all the classifiers are compared to obtain the best classifier among them.*

1. Introduction

In the early hours of April 15^{th}, 1912, the British passenger liner Titanic sank in the North Atlantic Ocean after colliding with an iceberg on its inaugural journey from Southampton to New York City. The ship carried an estimated $2,224$ passengers and crew members, making it one of the biggest commercial peacetime maritime disasters in modern history with almost $1,500$ fatalities. The second of three Olympic-class ocean liners run by the White Star Line, the RMS Titanic was the biggest ship in existence when it entered service.
In this study, we will use machine-learning techniques to predict which passengers survived the Titanic using data from "Kaggle." Predictions will be based on factors such as ticket price, age, sex, and class. To compare the various machine-learning techniques, we take several approaches. By looking at the results of each technique, we can give some insights into the problem. In predicting the survival of each passenger, fare, sex/title, age, passenger class, and price were the most relevant predictors. A machine learning model helps tune a model by "learning" or "analyzing" each passenger to provide accurate predictions.

1.1. Dataset

The Titanic dataset used in this project is provided by Kaggle. In this dataset, there are 891 training samples of passengers with their associated survival labels. Each passenger was given his/her passenger class, name, sex, age, number of siblings/spouses aboard, number of parents/children aboard, ticket number, fare, cabin embarked, and port of embarkation. The test data consisted of 418 samples of the same format.
For several samples, one or more fields were missing (Not a Number, NaN) and marked empty (especially in the latter fields - age, fare, cabin, and port). However, all sample points contained information about the passenger class and gender at least.

1.2. Confusion matrix

In this report, one of the methods used in order to measure the effectiveness of the implemented algorithm on the dataset is the confusion matrix. A Confusion matrix is an NxN matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model.
The following 4 are the basic terminology to help us determine the confusion matrix.
True Positives (TP): when the actual value is Positive, and the predicted is also Positive.
True negatives (TN): when the actual value is Negative, and the prediction is also Negative.
False positives (FP): When the actual is negative, but the prediction is Positive.
False negatives (FN): When the actual is Positive, but the prediction is Negative.

The confusion matrix is shown with a heat map plot throughout this report.

1.3. Literature review

Ekinci et al. [3] applied fourteen machine-learning techniques to the Titanic dataset. A decision tree correctly classified 81% of instances, while a random forest correctly classified 84% of instances. With 87% accuracy, Gradient Boosting achieved the best results. Meyer et al. [5] compared SVM implementations with 16 classification algorithms, and for the Titanic dataset, they achieved 20.81% and 21.27% error rates with neural networks and SVM, respectively, as minimum errors. Shawn Cicoria et al. [2] performed decision tree classification and cluster analysis to suggest that "sex" is the most important feature compared to other features in determining the likelihood of the survival of passengers. Singh et al. [7] suggested that dimensionality reduction and playing more with the dataset could improve the accuracy of the algorithms. The most important conclusion reached by these authors is that more features utilized in the models do not necessarily lead to better results.
The rest of this report is organized as follows. Section 2 presents the problem statement and objectives through this project. In section 3, the data visualization and preprocessing are provided. In section 4, the machine learning classification algorithms are proposed and implemented for the stated problem. Moreover, the evaluations of each algorithm and the results are provided. The report ends with an overall evaluation, the conclusions of this work, and the references.

2. Problem statement and objectives

The aim of this project is mainly to apply different machine learning classifiers on the Titanic dataset to obtain the most accurate results and predictions about the survival of the passengers of this ship. The objective of this study is to use machine learning and feature engineering techniques to obtain the most accurate results from the raw and missing data. Besides, the visualization of the data as a significantly important step to understand and pre-process the dataset should be done. Pre-processing the dataset in order to obtain or make new features with the most correlation with the output is one of the initial goals. After doing a survey on machine learning classification algorithms for the task of binary classification (Survived / Dead) on the Titanic dataset, improving the performance of each classification algorithm is the next crucial process. Finally, the results from all the algorithms are aimed to compare.

3. Visualization and pre-processing of the data

The tables in Figure 1 show the description of all data, survived people, and dead people which give important information about the dataset and visualize it.

========== All of the Data ==========

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Emb1	Emb2
count	1309.000000	891.000000	1309.000000	1309.000000	1309.000000	1309.000000	1309.000000	1309.000000	1309.000000	1309.000000
mean	655.000000	0.383838	2.294882	0.355997	30.044098	0.498854	0.385027	33.280206	0.095493	0.207792
std	378.020061	0.486592	0.837836	0.478997	12.947325	1.041658	0.865560	51.741830	0.294007	0.405882
min	1.000000	0.000000	1.000000	0.000000	0.170000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	328.000000	0.000000	2.000000	0.000000	22.000000	0.000000	0.000000	7.895800	0.000000	0.000000
50%	655.000000	0.000000	3.000000	0.000000	29.000000	0.000000	0.000000	14.454200	0.000000	0.000000
75%	982.000000	1.000000	3.000000	1.000000	35.000000	1.000000	0.000000	31.275000	0.000000	0.000000
max	1309.000000	1.000000	3.000000	1.000000	80.000000	8.000000	9.000000	512.329200	1.000000	1.000000

========== Survived People ==========

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Emb1	Emb2
count	342.000000	342.0	342.000000	342.000000	342.000000	342.000000	342.000000	342.000000	342.000000	342.000000
mean	444.368421	1.0	1.950292	0.681287	28.477040	0.473684	0.464912	48.395408	0.093567	0.277778
std	252.358840	0.0	0.863321	0.466660	13.802919	0.708688	0.771712	66.596998	0.291652	0.448559
min	2.000000	1.0	1.000000	0.000000	0.420000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	250.750000	1.0	1.000000	0.000000	21.000000	0.000000	0.000000	12.475000	0.000000	0.000000
50%	439.500000	1.0	2.000000	1.000000	28.583132	0.000000	0.000000	26.000000	0.000000	0.000000
75%	651.500000	1.0	3.000000	1.000000	35.000000	1.000000	1.000000	57.000000	0.000000	1.000000
max	890.000000	1.0	3.000000	1.000000	80.000000	4.000000	5.000000	512.329200	1.000000	1.000000

========== Dead People ==========

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Emb1	Emb2
count	549.000000	549.0	549.000000	549.000000	549.000000	549.000000	549.000000	549.000000	549.000000	549.000000
mean	447.016393	0.0	2.531876	0.147541	30.812972	0.553734	0.329690	22.117887	0.085610	0.136612
std	260.640469	0.0	0.735805	0.354968	12.502648	1.288399	0.823166	31.388207	0.280043	0.343751
min	1.000000	0.0	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	211.000000	0.0	2.000000	0.000000	23.000000	0.000000	0.000000	7.854200	0.000000	0.000000
50%	455.000000	0.0	3.000000	0.000000	32.000000	0.000000	0.000000	10.500000	0.000000	0.000000
75%	675.000000	0.0	3.000000	0.000000	35.000000	1.000000	0.000000	26.000000	0.000000	0.000000
max	891.000000	0.0	3.000000	1.000000	74.000000	8.000000	6.000000	263.000000	1.000000	1.000000

Figure 1. The description of the data

The real-world data often has a lot of missing values. The cause of missing values can be data corruption or failure to record data. The handling of missing data is very important during the pre-processing of the dataset as many machine learning algorithms do not support missing values. Deleting Rows with missing values, imputing missing values, and predicting them are one of the popular methods that can be used for handling missing values. There is relatively a large number of missing values for the features "Age", "Fair", and "Cabin". For the first feature, the averaging method is utilized to deal with the missing values based on the title of the passengers. This title includes Mr., MS., and others that are obtained from the "name" of each passenger. For "Fair" the same method as "Age" has been done based on the "P class". It was necessary to devise some new features to handle the missing data in the "Cabin" feature.
For ML algorithms, it is always needed to convert categorized data to their corresponding digital binary numbers. As an example, we have four categories in the feature "Embarked" that have been converted as given in Table 1.
The same process has been done for "sex".

S	00
C	01
Q	10
NAN	11

Table 1. Digitalizing feature "Embarked"

3.1. New features

The heatmap of the correlation between each feature and the output yields that more features with a higher correlation with the output are needed. Combining "Age" and "Sex" features to make a new feature "Age&Sex" which is 0 for being a child, 1 for being a man, and 2 for being a woman. Moreover, the feature "Partner" is defined as the sum of "Parch" and "SibSp". The feature "Being_MoD" shows being a mother or daughter. In this feature, a binary value is assigned to each data sample that shows whether the person is a mother/daughter or not.
Knowing the value of fairs for each group of passengers, we tried to determine the fare for each individual by using the information of "Ticket" and "Common_Ticket" and making a new feature name "Single_Fare". The feature "Ticket" has also been used to make a new feature "Common_Ticket" for people with the same ticket number that helps in finding the values of "Single_Fare". Two features, "Single_Fare" and "Ticket" are used to make the feature "NN_cabin" for not being NaN "Cabin". After filling in the missing data in "Cabin", it has been labeled in a new feature named "Num_Cabin" from 0 to 7.
The heat map of the correlation between Survived and the "NN_Cabin" feature is satisfactory, therefore, a reliable feature has been extracted. Moreover, the heat map of the correlation between "Survived" and the "Cabin_Num" features is satisfactory too, so not only the NaN elements in the "Cabin" has been properly filled, but also "Cabin_Num" is a reliable feature.
Finally, we can say that we successfully extracted three new features, "Cabin_Num", "NN_Cabin", and "Being_Mother" which have a sufficient correlation with the "Survived" and can be used to predict our model.

3.2. Splitting train and test data

The dataset has been split into the training and test sets by the fraction of 80% for training and 20% for the test.

4. Machine learning classification algorithms

To do the binary classification of this model, machine learning algorithms such as logistic regression [10], K-nearest neighbors (KNN) [6], support vector machine (SVM) [4], naive Bayes [8], random forest [1], and the artificial neural network have been trained on data. For each method, the performance is evaluated and improved by tuning hyperparameters, if possible.

4.1. Logistic regression

Logistic regression is a machine learning algorithm used for classification problems; It is a predictive analysis algorithm based on probability. Logistic regression was used in the biological sciences in the early twentieth century. It was then used in many social science applications. Logistic regression is used when the dependent variable (target) is categorical.
Logistic regression can be considered a linear regression model, but logistic regression uses a more complex cost function; the cost function is known as the "Sigmoid" or "Logistic" function rather than a linear function. Logistic regression limits the cost function between 0 and 1.
The features that are used for logistic regression in order to obtain maximum test set accuracy are as follows
"Sex", "Age", "SibSp", "Common-Ticket", "NN-Cabin", "Being-MOD", "Multi-Ticket", and "P3SorQ".
At first test and training accuracies are reported. Features in the order of importance are gathered in Table 2. The importance of each feature is its coefficient in the logistic regression trained model.
As it is shown, the "Sex" feature has the most positive effect on the "Survived" feature, which means women ("Sex" = 1) have had more chances to survive compared to men. And "P3SorQ" has the most negative effect on the "Surviving" feature, which means those who have had a third Pclass ticket and embarked on S or Q harbors have lost their lives mostly. The confusion matrix is also provided in Figures 2 and 3 for both training and test datasets.

	Features	Coefficient
0	Sex	$+2.622335$
4	Partner	$+1.144275$
6	NN_Cabin	$+0.417665$
3	Emb1	-0.028983
1	Age	-0.030443
2	SibSp	-0.320433
5	Single_Fare	-0.614689
7	P3SorQ	-1.045628

Table 2. Feature importance in logistic regression

Accuracy	
Logistic Regression training accuracy	80.9256%
Logistic Regression test accuracy	86.5168%

Table 3. Logistic regression algorithm accuracy

4.2. K-nearest neighbors (KNN)

K-nearest neighbors (KNN) algorithms are non-parametric, supervised learning algorithms that use proximity to make predictions about the grouping of individual data points. Although it can be used for either regression or classification problems, it is typically used for classification, assuming that similar points can be

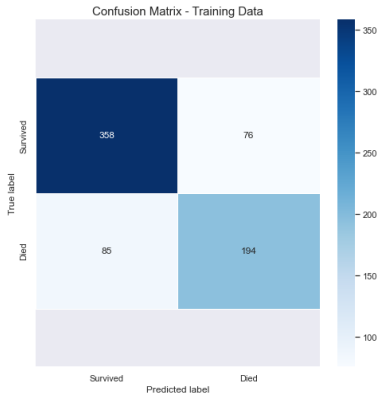

Figure 2. The confusion matrices of the training data in the logistic regression algorithm

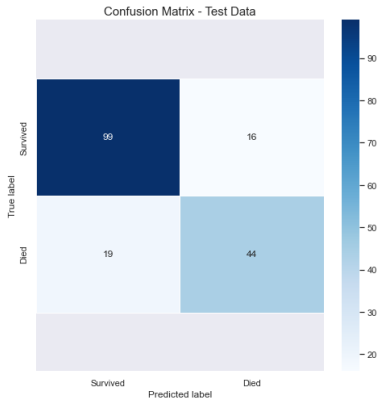

Figure 3. The confusion matrices of the test data in the logistic regression algorithm

found nearby. We can say that the KNN algorithm is, instance-based learning which means by comparing new data values to trained instances, the model generalizes to new data values using similarity measures (Euclidean distance), and lazy learning which means the model will do nothing in the training phase and just memorizes the training dataset. It is only during prediction that the model searches for the nearest neighbors from the entire training set, which is costly. The non-parametric term means it does not assume anything about the underlying data (e.g., linear regression assumes data is symmetrically distributed).

The features that are used for KNN in order to obtain maximum test set accuracy are as follows

"Pclass", "Sex", "Age", "SibSp", "Age&Sex", "Partner", "Single_Fare", and "NN_Cabin". Optimal features for the KNN method are selected and an instance of KNN method has been trained on the training dataset. Feature scaling has been done for both the training and test datasets.

Evaluating KNN:

KNN method with $n\_neighbors = K = 3$ and $p\ norm = 1, 2, 3$ has been implemented on the training dataset and accuracies are reported.

As it is shown, KNN with the second norm has the best

Accuracy	
KNN training accuracy for p=1	87.5175%
KNN test accuracy for p=1	89.3258%
KNN training accuracy for p=2	87.3773%
KNN test accuracy for p=2	91.0112%
KNN training accuracy for p=3	87.2370%
KNN test accuracy for p=3	90.4494%

Table 4. K-nearest neighbors algorithm accuracy for different p values

accuracy and better performance on the features.

After that, the KNN method with the second norm is implemented for different numbers of neighbors and its error on both the training and test dataset is plotted in Figure 4. The training error almost gets worse as the K value increases and the test error first decreases and then increases, which yields $K = 3$ as the best number of neighbors. This result seems reasonable because when you

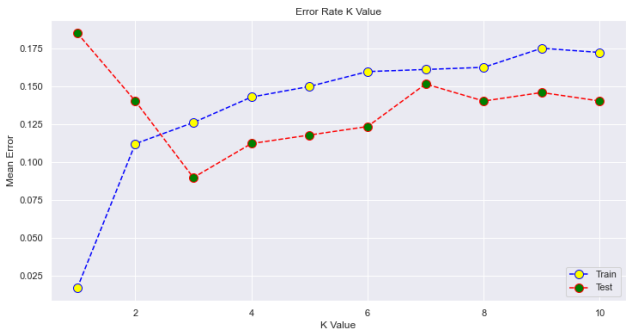

Figure 4. Mean error rate with respect to K values in the KNN algorithm

look at the first five nearest points to your test sample, you are considering more generalization and you classify your sample with a high level of confidence. The training error figure also shows that our samples are completely mixed together(they do not have separate clusters) as the error increases with higher K values.

The confusion matrix for the test data is also provided in Figure 5. Comparing this matrix to the same one in the previous part shows that we are performing better on the test dataset as the number of TP (True Positive) and TN (True Negative) samples increases by 9 and the number of FP (False Positive) and FN (False Negative) samples decreases by 9.

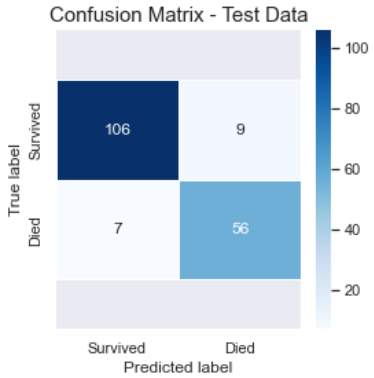

Figure 5. The confusion matrices of the training and test data in the K-nearest neighbors algorithm

4.3. Support vector machine (SVM)

Support vector machines (SVMs) are among supervised machine learning algorithms, which means they need to be trained on labeled data. SVM finds a hyperplane in N-dimensional space (N is the number of features) that distinguishes data points. There are several possible hyperplanes that can be used to separate the two types of data points, and he objective of SVM is to find a plane that has the maximum margin, i.e., the maximum distance between data points from both classes. By optimizing the margin distance, future data points can be classified with greater confidence.

The optimal features for SVM are reported as follows ”Pclass”, ”Sex”, ”Age”, ”SibSp”, ”Emb2”, ”Age&Sex”, ”Partner”, ”NN_Cabin”, ”Cabin_Num”, ”Being_MOD”, and ”Multi_Ticket”.

At first, SVM has been implemented for different types of kernels. For the ’linear’ and ’rbf’ kernels no feature scaling is used as the accuracies on the test and training datasets appeared to be more accurate in the absence of feature scaling. Values for ’C’ and ’gamma’ are manually tuned to have the best performance. However, with the ’poly’ kernel, feature scaling improves accuracies. For the ’poly’ kernel, the best degree is 4 based on the accuracy.

Evaluating SVM:

The accuracies for all three kernels are provided above. As we see, the ’rbf’ kernel is the best kernel to use for our dataset. F1 Scores are also calculated for all kernels. F1 score is defined as the harmonic mean between precision and recall.

$$\frac{TP}{TP+FP} = Precision$$

$$\frac{TP}{TP+FN} = Recall$$

Sometimes the positive class is more important for us compared to a negative class. So we want to have better precision value. On the other hand, sometimes negative class has higher priority for us and we want the Recall measure to be high enough. As we see, in different problems Precision and Recall have different importance, depending on the goal of classification. But sometimes both Recall and Precision measures are important to us, so one can use the F1 Score that considers both Precision and Recall measures. F1 Scores calculated above yield that

Accuracy	
kernel = Linear	
SVM training accuracy	78.2609%
SVM test accuracy	80.3371%
SVM training F1 score	0.7103
SVM test F1 score	0.7107
kernel = rbf	
SVM training accuracy	88.0785%
SVM test accuracy	83.1461%
SVM training F1 score	0.8429
SVM test F1 score	0.7541
kernel = poly	
SVM training accuracy	84.4320%
SVM test accuracy	87.0786%
SVM training F1 score	0.7845
SVM test F1 score	0.8067

Table 5. Support vector machine algorithm accuracy and F1 score for different kernels

methods with high accuracies also have better F1 Scores and are supposed to be better classifiers. The rbf kernel has the best F1 Scores too.

Dimensional reduction with PCA:

The dimensions of the dataset can be reduced using dimension reduction methods such as PCA [9]. The dimensions of the Titanic dataset have been reduced to 2 by implementing PCA. Then, an SVM classifier with rbf kernel and a high value of gamma (to see the overfitting phenomenon better) also with a low value of gamma (underfitting) has been implemented, and the plots of the training dataset with decision boundaries are shown in Figure 6 and 7. As the plots show and the accuracies confirm, our classifier in the first one is overfitting the data, and in the second one, it is underfitting it.

The best value for gamma is also found by hyperparameter search and the plot of the training dataset with decision boundary is shown in Figure 9. This gamma has been found based on the best suitable pair of training accuracy and test accuracy.

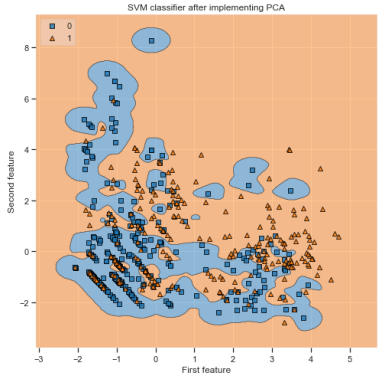

Figure 6. SVM classifier with gamma=20 and its decision boundary

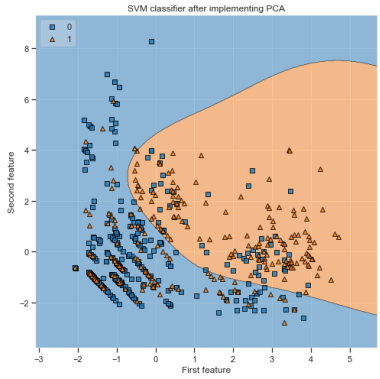

Figure 7. SVM classifier with gamma=0.1 and its decision boundary

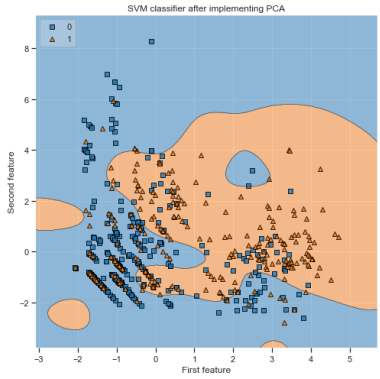

Figure 8. SVM classifier with best gamma=1.8 and its decision boundary

4.4. Gaussian Naive Bayes

In machine learning, a naive Bayes classifier uses probabilistic models to determine classifications. Bayes' theorem forms the heart of this classifier. The important assumption made here is that the predictors/features are independent. One feature does not affect the other, i.e., one does not affect the other. Hence it is called naive.

Evaluating Gaussian naive Bayes:
Optimum features for Gaussian Naive Bayes Classifier are selected and an instance of it is trained on the training dataset with the default value for smoothing. The accuracies on both training and test dataset are reported in Table 6.

Accuracy	
Gaussian naive Bayes training accuracy	80.5049%
Gaussian naive Bayes test accuracy	85.9550%

Table 6. Gaussian naive Bayes algorithm accuracy

Area Under Curve (AUC) for Gaussian naive Bayes:
AUC stands for Area Under Curve for ROC Curve. The ROC curve is created by plotting the true positive rate (TPR or Recall) against the false positive rate (FPR) at various threshold settings.

$$\frac{TP}{TP+FN} = TPR$$

$$\frac{FP}{FP+TN} = FPR$$

For a binary classification problem, AUC gives information about how the classifier is compared to random guesses. If the classifier performs well, AUC will be close to 1. AUC is a better measure of classifier performance than accuracy because it does not have a bias on the size of the test or evaluation data. However, accuracy is always biased by the size of the test data. Accuracy is calculated based on the class distribution of the test dataset or cross-validation. The TP rate and FP rate which are used to calculate AUC will not be affected by class distribution changes. This is one of the AUC measures advantages over the accuracy measure.

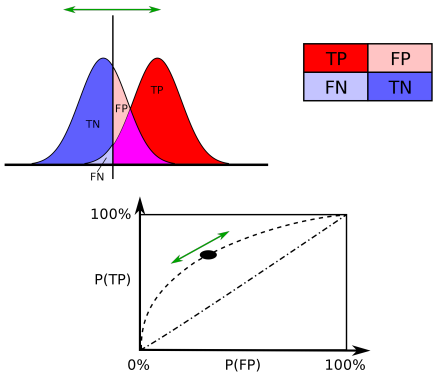

Figure 9. AUC for ROC curve

AUC for Gaussian naive Bayes	
AUC for Gaussian naive Bayes for training	0.7861
AUC for Gaussian naive Bayes for test	0.8411

Table 7. AUC for Gaussian naive Bayes

4.5. Random forest

Random forest is an ensemble of decision tree algorithms. The main advantage of using random forest is that it reduces the high variance problem of decision trees. It is an extension of bootstrap aggregation (bagging) of decision trees and can be used for classification and regression problems. Perhaps the most important hyperparameter to tune for the random forest is the number of random features to consider at each split point.

The optimal features found for the random forest algorithm based on accuracies are

"Sex", "Age", "SibSp", "Parch", "Single_Fare", "Cabin_Num", and "P3SorQ".

Five-fold cross-validation:

Five-fold cross-validation is implemented on the whole dataset for different n_trees (number of trees) and $depths$ (with ranges specified above).

Evaluating random forest based on cross-validation:

The best number of trees, Best depth of trees, Minimum error of cross-validation, best F1 Scores, and best accuracies are reported in Table 8. The random state is also specified to help compare errors and choose the optimum parameters.

Results	
Best n_trees (number of trees)	490
Best depth of trees	12
Min error of 5-fold cross-validation	0.1627
Random forest training accuracy	96.6339%
Random forest test accuracy	87.6404%
Random forest training F1 score	0.9557
Random forest test F1 score	0.83077

Table 8. Random forest algorithm evaluation based on 5-fold cross-validation

Evaluating random forest based on manually tuned hyperparameters (Best Result):

Although five-fold cross-validation proposed good parameters, better parameters have been found manually, and the best results of the random forest algorithm are reported in Table 9.

Results	
Best n_trees (number of trees)	220
Best depth of trees	19
Random forest training accuracy	98.5975%
Random forest test accuracy	89.32584%
Random forest total accuracy	96.7452%

Table 9. Random forest algorithm evaluation based on manually tuned hyperparameters

4.6. Artificial neural network

A subclass of machine learning called neural networks, commonly referred to as artificial neural networks (ANNs), are at the core of deep learning techniques. They mimic the way that organic neurons communicate with one another. They can be regarded as a non-linear function that transforms a set of inputs into an output variable. The Multilayer Perceptron (MLP) algorithm is the simplest type of artificial neural network. It is a model of a single neuron that can be used for binary classification problems.

To train an MLP classifier on our data, we used two different architects and tuned hyperparameters for them as follows.

Tuning hyperparameters for MLP with hidden layers (20,10):

To tune the learning rate and alpha (regularization penalty coefficient) for MLP with (20, 10) as hidden layers, the performance of the trained model on the test dataset for different learning rates and alpha values is evaluated.

Evaluating MLP with hidden layers with the size of (20,10):

MLP with hidden layers with the size of	(20,10)
Best learning rate	0.02
Best regularization penalty coefficient (alpha)	0.0001
Min error of tuning process	0.0955
MLP training accuracy	83.7307%
MLP test accuracy	90.4494%

Table 10. Multilayer perceptron (MLP) algorithm with hidden layers with the size of $(20, 10)$ evaluation

Tuning hyperparameters for MLP with hidden layers (100, 50):

To tune the learning rate and alpha (regularization penalty coefficient) for MLP with (100, 50) as hidden layers, the performance of the trained model on the test dataset for different learning rates and alpha values is evaluated.

Evaluating MLP with hidden layers with the size of (100, 50):

MLP with hidden layers with the size of	$(100, 50)$
Best learning rate	0.05
Best regularization penalty coefficient (alpha)	0.0001
Min error of tuning process	0.1011
MLP training accuracy	83.4502%
MLP test accuracy	89.8876%

Table 11. Multilayer perceptron (MLP) algorithm with hidden layers with the size of (100, 50) evaluation

Final evaluation for MLP:
As observed in previous sections, MLP with (20, 10) hidden layer performed better on both training and test datasets. In the (100, 50) case we have more weights in both hidden layers as compared to the (20, 10) case. we also use MLP with just 20 epochs which is a small number, and we faced converging problems during evaluating each MLP model most of the time. So in the (100, 50) case, the dimension of our objective function is far larger than the (20, 10) case, and this high dimension makes it difficult for the objective function to converge to its optimum value. So we expect that the (100, 50) model performs better for higher epoch values, but for only 20 epochs, the (20, 10) model has better accuracies. The best accuracies obtained by MLP are represented in Table 12.

Best Results for MLP	
MLP training accuracy	83.7307%
MLP test accuracy	90.4494%

Table 12. Multilayer perceptron (MLP) algorithm final evaluation

The confusion matrix of both training and test data are provided in Figures 10 and 11.

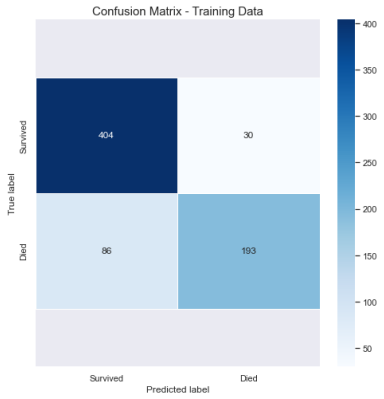

Figure 10. Confusion matrix for training data in the final MLP

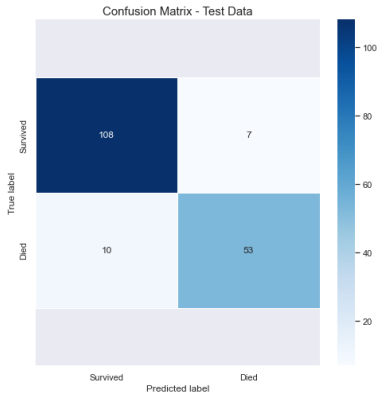

Figure 11. Confusion matrix for test data in the final MLP

Algorithm	Train accuracy	Test accuracy
Logistic regression	80.9257%	86.5169%
K-nearest neighbors	87.3773%	91.5730%
Support vector machine	93.6886%	87.6404%
Gaussian naive Bayes	80.5049%	85.9551%
Random forest	98.5975%	89.3258%
Neural network	83.7307%	90.4494%

Table 13. Training accuracies and test accuracies of all algorithms

5. Overal results and conclusion

As the last step, all training and test accuracies are gathered in Table 13. This yields that the random forest has the best accuracy on the training data and K-nearest neighbors has the best accuracy on the test data.

5.1. Conclusion

Machine Learning algorithms are a rich class of models that has become increasingly popular in many tasks such as classification. Despite the fact that ML algorithms have shown significant power and robustness in different types of classifications, some pre-processing methods are crucial for each and every method. In this project after pre-processing the data in order to obtain new features and select features with the highest correlation with the output, several machine learning classifiers have been implemented on the pre-processed data. Moreover, the performance of each method has improved using hyperparameter search to obtain the best ones based on performance. Results are relatively accurate in comparison to the state-of-the-art.

References

[1] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[2] Shawn Cicoria, John Sherlock, Manoj Muniswamaiah, and Lauren Clarke. Classification of titanic passenger data and chances of surviving the disaster. *Proceedings of Student-Faculty Research Day, CSIS, Pace University*, 2014.

[3] Ekin Ekinci, S İlhan Omurca, and Neytullah Acun. A comparative study on machine learning techniques using titanic dataset. In *7th international conference on advanced technologies*, pages 411–416, 2018.

[4] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[5] David Meyer, Friedrich Leisch, and Kurt Hornik. The support vector machine under test. *Neurocomputing*, 55(1-2):169–186, 2003.

[6] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[7] Aakriti Singh, Shipra Saraswat, and Neetu Faujdar. Analyzing titanic disaster using machine learning algorithms. In *2017 International Conference on Computing, Communication and Automation (ICCCA)*, pages 406–411. IEEE, 2017.

[8] Geoffrey I Webb, Eamonn Keogh, and Risto Miikkulainen. Naïve bayes. *Encyclopedia of machine learning*, 15:713–714, 2010.

[9] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[10] Raymond E Wright. Logistic regression. 1995.