

# Titanic Machine Learning from Disaster

Xiaoxin Zhou  
Computer Science  
Ryerson University  
Toronto, Canada  
xiaoxin.zhou@ryerson.ca

Seit Sorra  
Computer Science  
Ryerson University  
Toronto, Canada  
ssorra@ryerson.c

## I. ABSTRACT

The sinking of the Titanic is a regrettable event in which 2224 were affected and more than 1500 died. From this event we have movies and books that attempt to narrate what happened. In our project, we will be tackling the Kaggle Titanic Machine Learning Project. Within this project we will be analyzing and observing a series of passenger data to predict the possibility of them surviving. Our results will be theoretical. Examples of data points that will be considered include socio economic class, gender, age, and family relations. In this report, we used four different models: Logistic Regression, Support Vector, Random Forest, and Neural Net. Our experiments show that Neural Net and without cross-validation provide the best result in predict survival rate.

Source code is available at:

[https://github.com/seitsorra/CPS803\\_Project](https://github.com/seitsorra/CPS803_Project)

## II. INTRODUCTION AND MOTIVATION

Our group found this Titanic Machine learning competition from Kaggle. It interested us in finding a model to predict the survival rate for passengers on the Titanic shipwreck. Also, we are interested in how the training data and hyperparameters affect the models, which can provide the best result. Learning from our past is a valuable advent and can help us gain critical skills that we may be able to apply to projects and data sets in our future. More importantly, creating a model for this disaster holds the potential for understanding other sinkings. Hence, we decided to choose this topic as our final project.

Before choosing the model, we decided to find the most valuable data and remove some useless information. Data Visualization and Analysis will explain how we choose the data and deal with it to judge our results better. Feature Engineering explained our understanding of the dataset and why we removed a few data cols from the dataset. In Models Design and Evaluation, we implemented Logistic Regression, Support Vector, Random Forest and Neural Net.

Brief Description:

Logistic Regression - is used to model the probability of a certain class such as True or False, in our project we are going to use logistic regression to predict the person who can survive or not. The returning data should be 0 or 1. [1]

Support Vector – This is a supervised learning model that works with classification and regression analysis. This model can help us classify if the database passenger is in the survival area or not. [2]

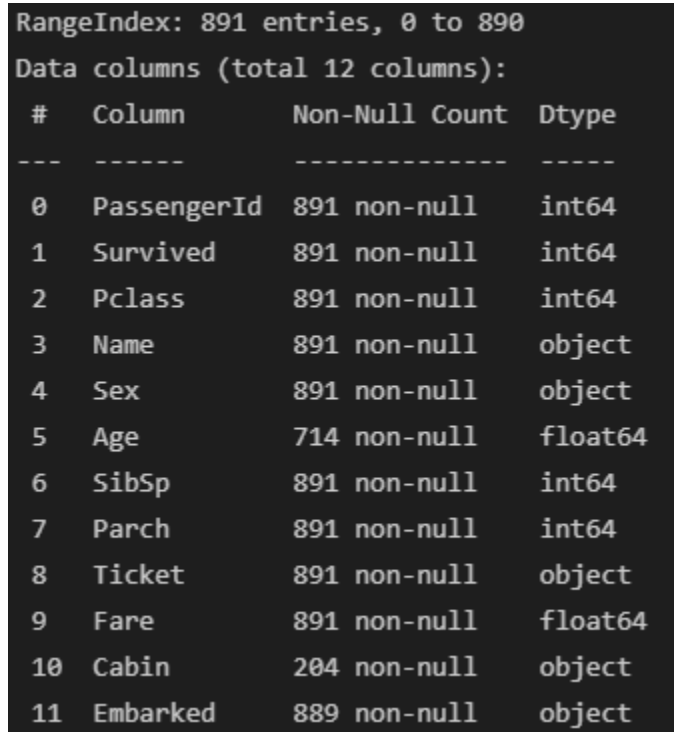
Random Forest – This is an unsupervised learning model, in which its performance is often comparable to logistic regression. The Random Forest model is strongly influenced by data characteristics, and hence requires a good data set.

Our team predicts this may be a strong model choice. [3]  
Neural Network – The neural network can “learn” the way a human does, and the strongest benefit is the hidden layers. The hidden layers can predict more accurately than another model.

## III. METHODS & EXPERIMENTS

### A. Data Visualizaiton and Analysis

Data for this project was provided on the Kaggle website. Here is an overview of the data:



RangeIndex: 891 entries, 0 to 890			
Data columns (total 12 columns):			
#	Column	Non-Null Count	Dtype
0	PassengerId	891 non-null	int64
1	Survived	891 non-null	int64
2	Pclass	891 non-null	int64
3	Name	891 non-null	object
4	Sex	891 non-null	object
5	Age	714 non-null	float64
6	SibSp	891 non-null	int64
7	Parch	891 non-null	int64
8	Ticket	891 non-null	object
9	Fare	891 non-null	float64
10	Cabin	204 non-null	object
11	Embarked	889 non-null	object

Fig 1: An overview of our data samples

Our first step was understanding the data we were dealing with to better judge our results.

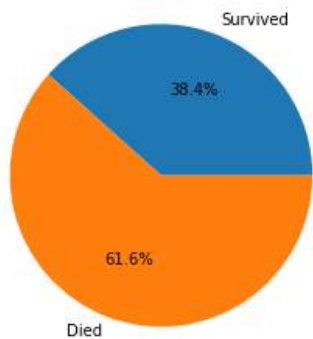


Fig 2: Percentage of passengers who survived versus those who died

To get a better idea of passengers who survived we looked at the passenger's gender and we noticed that majority of women survived the sink of the ship when compared to men as shown in Figure 3.

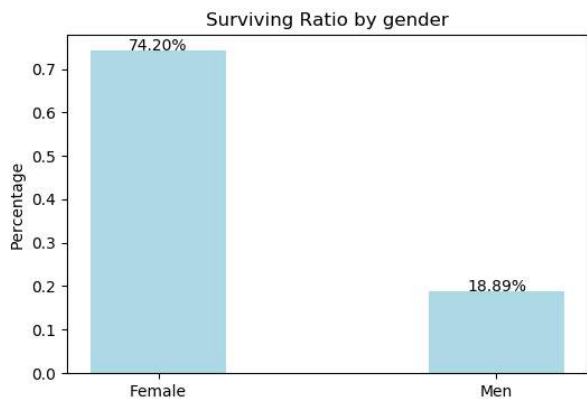


Fig 3: Surviving ratio of passengers by gender

Noticing this interesting trend about the gender of passengers we also investigated the passenger's age distribution and how the respective surviving ratio fared within a group age. We grouped ages as follows:

- 0 – 2 → Infant
- 2 – 4 → Toddler
- 4 – 13 → Kid
- 13 – 19 → Teen
- 20+ → Adults
- NaN → Unknown

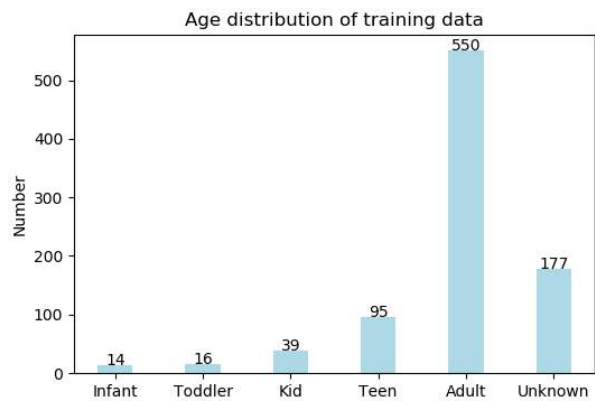


Fig 4: Passenger's age distribution by group age

Then looking at the survival ratio of those passengers we noticed that both adults and passenger without a known age were the ones who had died more as shown in Figure 5.

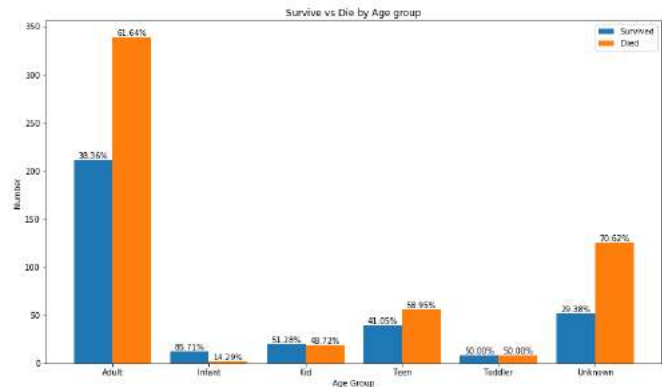


Fig 5: Survival/Die ratio within age groups

In addition, we knew that there were different ticket classes for passenger onboard, so we looked whether there was any relation between the ticket class and surviving ratio. Results, shown on Figure 6, proved that indeed passengers on the first class were more likely to survive than those in second or third class.

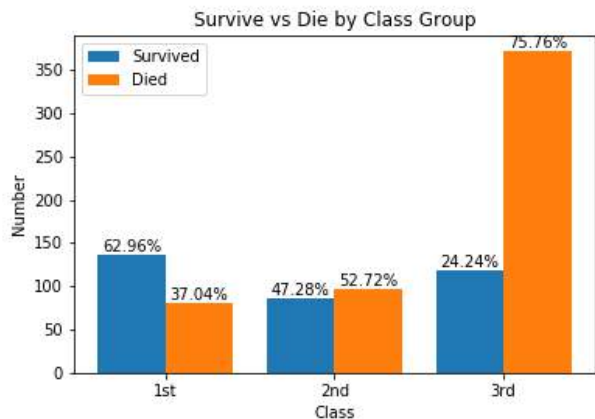


Fig 6: Surviving/Die ratio by ticket class

Another interesting observation was the embarking station. When analyzing the data, we were able to notice that those passengers who embarked the ship on Southampton, had a higher die ratio when compared to passengers who embarked on the other two stations as shown on Figure 7.

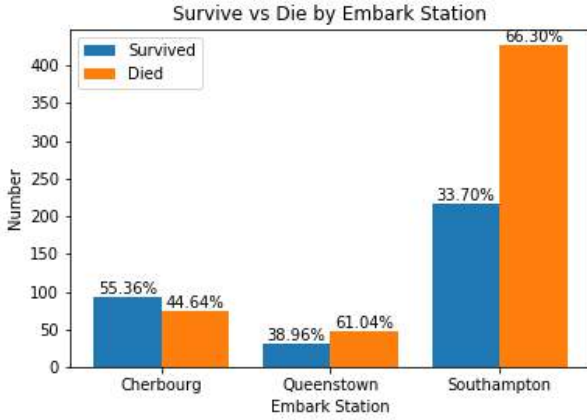


Fig 7: Survive/Die ratio of passengers by embarking station

### B. Feature Engineering

As shown on the overview of our data samples in Figure 1, data consists of 891 entries including 11 features for each sample, those being: Passenger Id, Ticket class (Pclass), Name, Sex, Age, Number of siblings/spouses on board (SipSp), Number of parent/children on board (Parch), ticket number (ticket), passenger fare (fare), cabin number (cabin), port of embarkation (Embarked) and lastly the label Survived with value 1 if passenger survived and 0 otherwise.

Considering the large amount of missing data for cabin numbers we decided to drop that. Similarly, we thought that the name of passenger and ticket number wouldn't have any impact on our model results, so we decided to drop them as well.

Furthermore, we did not have all the data about the passenger's age, with 19.8% of it missing. As a result, we decided to interpolate any of missing ages in our data set.

Lastly, we modified the dataset by converting any categorical data column to numerical values, to use for our model training.

## IV. MODELS DESIGN AND EVALUATION

### 1) Logistic Regression Model:

Considering our problem relates to determining whether a passenger survived, we thought that a logistic model would be a good choice to solve our problem.

Using an out of the box logistic regression model from the sklearn package we were able to get the following results:

Training accuracy: 0.8154657293497364				
Validation accuracy: 0.7887323943661971				
	precision	recall	f1-score	support
0	0.83	0.89	0.86	355
1	0.79	0.69	0.74	214
accuracy			0.82	569
macro avg	0.81	0.79	0.80	569
weighted avg	0.81	0.82	0.81	569

Fig 8: Logistic regression model using saga solver and 10000 iterations

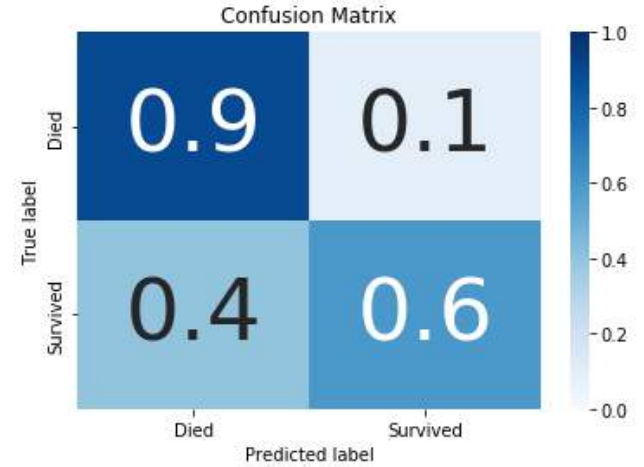


Fig 9: Confusion matrix for saga logistic regression model on validation data

As shown in figure 9, we are already getting relatively good results from an out of box logistic model.

Trying to improve the results of the logistic regression model, we ran some more experiments by tuning some of the parameters for the logistic model.

We decided to experiment with different values of Inverse regularization strength using a subsample of:

```
C_params = np.geomspace(1e-5, 1e5, 20)
```

The reason why we wanted to experiment with inverse regularization strength was to see how much the model could afford fitting the data without impacting our validation results.

While running the above experiments, we tracked the log loss and accuracy of the model. Results were as following (sorted by lowest log loss on validation data):

	index	C Param	Accuracy Train	Accuracy Valid	Log Loss Train	Log Loss Valid
0	17	8858.667904	0.811951	0.788732	0.447863	0.524981
1	16	2636.650899	0.813708	0.788732	0.447865	0.524982
2	15	784.759970	0.813708	0.788732	0.447869	0.524983
3	18	29763.514416	0.813708	0.788732	0.447869	0.524984
4	19	100000.000000	0.813708	0.788732	0.447869	0.524985
5	14	233.572147	0.813708	0.788732	0.447873	0.524985
6	13	69.519280	0.813708	0.788732	0.447884	0.524988
7	12	20.691381	0.813708	0.788732	0.447922	0.525002
8	11	6.158482	0.813708	0.788732	0.448054	0.525047
9	10	1.832981	0.815466	0.788732	0.448491	0.525202
10	9	0.545559	0.815466	0.788732	0.449973	0.525781
11	8	0.162378	0.810193	0.774648	0.454913	0.528110
12	7	0.048329	0.810193	0.753521	0.470260	0.537573
13	6	0.014384	0.755712	0.725352	0.506128	0.564691
14	5	0.004281	0.724077	0.690141	0.554204	0.603802
15	4	0.001274	0.694200	0.690141	0.591371	0.633168
16	0	0.000010	0.671353	0.647887	0.626535	0.647733
17	3	0.000379	0.676626	0.683099	0.611954	0.647869
18	1	0.000034	0.669596	0.654930	0.620221	0.648247
19	2	0.000113	0.666081	0.669014	0.619303	0.650592

Fig 10: results of logistic model by applying different C parameter

Using the best C parameter, we got the following results:

```
testModel = LogisticRegression(C=8858.667904, max_iter=10000, solver="saga")
testModel.fit(x_train, y_train)
print("Accuracy ", testModel.score(x_valid, y_valid))
✓ 0.5s
Accuracy 0.7887323943661971
```

Fig 11: Logistic model regression results using the best-found C parameter

Then using our final model, we trained it with all of our training data and tested it on test data that was never seen by the model. The model produced a training score of **79.6%** and test score of **79.2%**. Below is the confusion matrix on the test data:

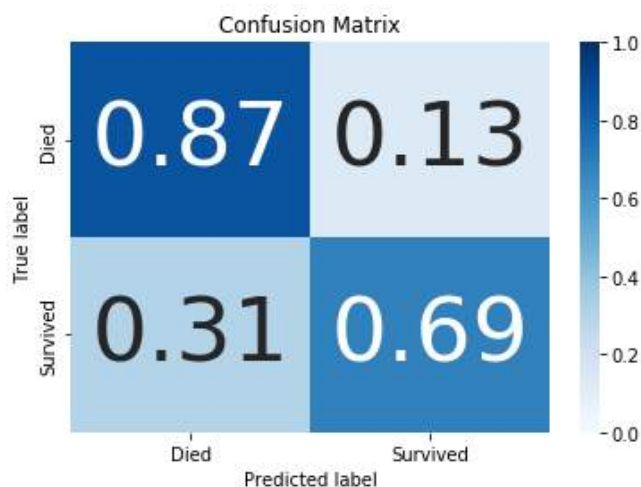


Fig 12: Confusion matrix on test data using Logistic Regression Model

## 2) Support Vector Classification Model:

Our next model was an SVC model from the sklearn package. Using the out of the box model from sklearn, we got the following results:

Training accuracy: 0.6748681898066784				
Validation accuracy: 0.647887323943662				
	precision	recall	f1-score	support
0	0.67	0.94	0.78	355
1	0.69	0.24	0.36	214
accuracy			0.67	569
macro avg	0.68	0.59	0.57	569
weighted avg	0.68	0.67	0.62	569

Fig 13: results from an out of box SVC model

We then experimented further by performing some cross-validation on the SVC model.

```
model = SVC()
param_grid = [
    {
        'kernel': ['linear', 'poly', 'rbf'],
        'C': np.logspace(1e-5, 1e5, 20),
        'max_iter': [10000]
    }
]
clf = GridSearchCV(model, param_grid=param_grid, cv=10, verbose=True, n_jobs=-1)
best_clf = clf.fit(x_train, y_train)
```

Fig 14: Parameter grid used for cross validation SVC model

Then using the best estimator from the cross validation experiment we achieved the following results:

```
print(f'Accuracy Train : {best_clf.score(x_train, y_train):.3f}')
print(f'Accuracy Valid : {best_clf.score(x_valid, y_valid):.3f}')
✓ 0.1s
Accuracy Train : 0.675
Accuracy Valid : 0.648
```

Fig 15: Results on the best estimator from a cross validation of SVC model

As shown on Figure 15, we were not getting better results even after trying some cross validation techniques, as a result we did not experiment any further with this model and moved on using other models instead.

## 3) Random Forest Classifier Model

Our next model experiment was Random Forest Classifier model from sklearn package.

Using an out of the box model, we got a training accuracy of **99.29%** and validation accuracy of **78.16%**. Below is the confusion matrix on validation data:

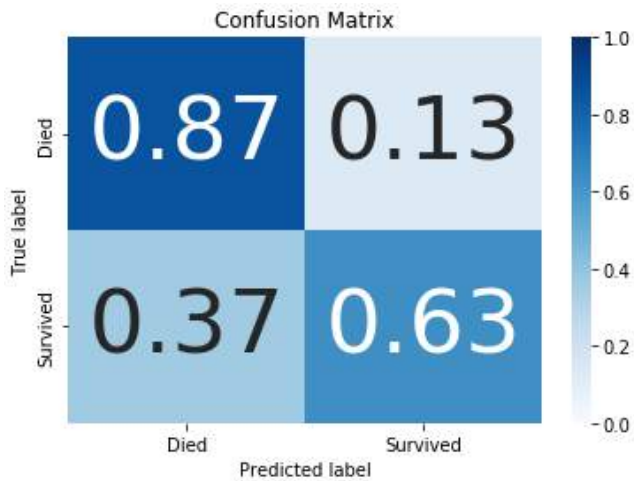


Fig 16: Confusion matrix of validation data using out of box Random Forest Classifier model

As shown in Figure 16, we were getting high accuracy on the training data, but not nearly as good results on the validation data set. This points out to overfitting where the model has low bias but high variance.

To improve the results, we experimented with Randomized Search Cross validation from sklearn using the following grid parameters:

```
# Create the random grid
random_grid = {
    'n_estimators': n_estimators,      # np.linspace(start = 200, stop = 2000, num = 10)
    'max_features': max_features,      # ['auto', 'sqrt']
    'max_depth': max_depth,            # np.linspace(10, 110, num = 11)
    'min_samples_split': min_samples_split, # [2, 5, 10]
    'min_samples_leaf': min_samples_leaf,  # [1, 2, 4]
    'bootstrap': bootstrap             # [True, False]
}
```

Fig 17: Parameters grid for randomized searched CV on Random Forest Classifier Model

Then using the best parameters generated from our cross validation experimenting, we build the final model and trained using both training and validation data and then tested it using the tested data which the model had never seen. Our results produced a training accuracy of **95.22%** and testing accuracy of **80.28%**. Below is the confusion matrix:

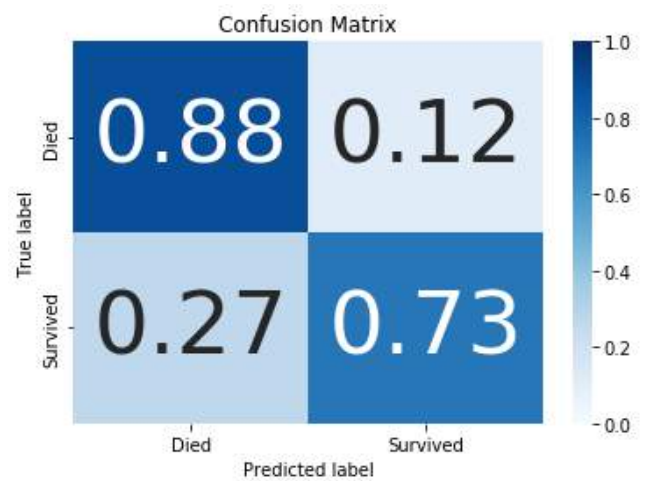


Fig 18: Confusion matrix of test results from our final Random Forest Classifier model

#### 4) Neural Network Model

Our next model experiment was a neural model from sklearn package

Our first attempt was a simple neural network with the following parameters:

```
model = Sequential()
model.add(Dense(24, input_dim=x_train.shape[1], activation='tanh'))
model.add(Dense(10, activation='tanh', kernel_regularizer=keras.regularizers.l2(0.2), bias_regularizer=keras.regularizers.l2(0.01)))
model.add(Dense(8, activation='tanh', kernel_regularizer=keras.regularizers.l2(0.01), bias_regularizer=keras.regularizers.l2(0.001)))
model.add(Dense(1, activation='sigmoid'))

# compile the keras model
opt = Adam(learning_rate=0.001)
model.compile(loss='binary_crossentropy', optimizer=opt, metrics=['accuracy'])

# fit the keras model on the dataset
lr_model_history = model.fit(x_train, y_train, epochs=1000, batch_size=1, validation_data=(x_valid, y_valid))
```

Fig 19: Neural Network model experiment

During the training of the above model, we kept track of model accuracy and loss for each epoch. Results were as follows:

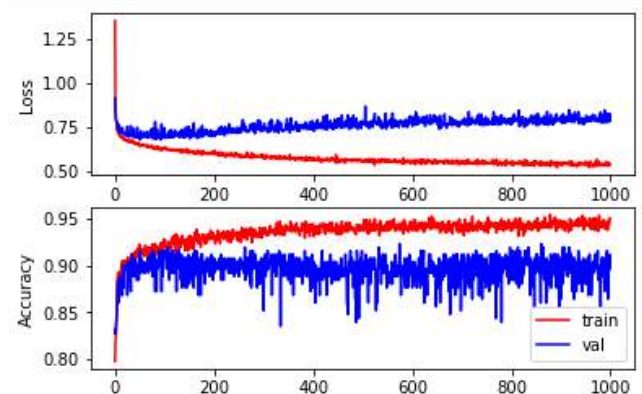


Fig 20: Accuracy and Loss of Neural Network model

The model reached an accuracy of **80.28%** on validation data and its confusion matrix was as depicted below:



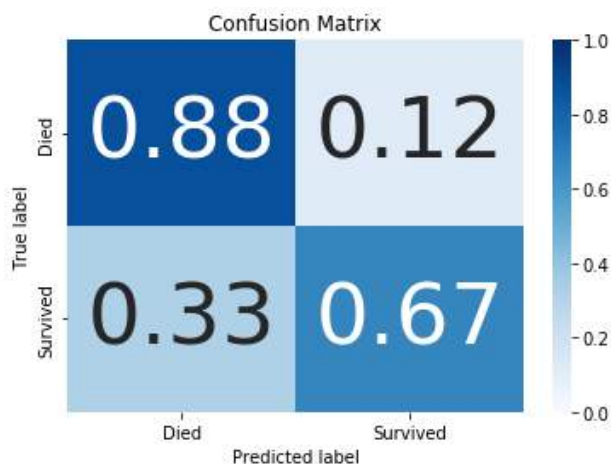


Fig 21: Confusion matrix of Neural Network Model

To improve the results, we decided to run some cross validation experiments with the following grid parameters:

```
param_grid = {
    "n_hidden": [1, 2, 3, 4],
    "n_neurons": [4, 8, 16, 24],
    "learning_rate": [0.05, 0.01, 0.005, 0.001]
}
```

Fig 22: Grid parameters used to run cross validation on Neural Network

However, using the best estimator from the above experiment produced an accuracy of only **70.13%** which was lower than our initial results.

## V. CONCLUSION

In this paper, we have talked about the motivation and introduction for this project. The visualization gives us an intuitive view of our dataset. Understanding the dataset is more important to our model. Our dataset is missing some vital information, such as whether the passenger is close to a lifeboat, if the passenger can swim or not, if floating objects were near the passenger, etc.

Findings from our experiments:

- **Logistic regression model** – in general logistic model was giving consistent results and without overfitting to our data. However, the lack of data did seem to penalize our experiments to further improve the results
- **Support Vector Machine** – all of experiments using the support vector classifier produced bad results when compared with other models.
- **Random Forest Classifier** – this model was quickest to overfit to our training data and although we were able to improve the overfitting by running some cross validation results, it remained an issue with our final model.

- **Neural Network** – attempts to building a simple enough model given the lack of data did produce decent results. However, further attempts to improve the results through some cross validation did not yield any higher performance.

In general, the lack of data has impacted our ability to experiment with the training of our models. Also, we noted that across all the models the performance is impacted mainly from the False Negative prediction being quite high than what we would normally want. Specifically, the models would be wrongly predicting that a passenger would die when in fact it survived. Further analysis could help understand more the reasoning behind these failures.

## VI. ACKNOWLEDGEMENTS

We would like to acknowledge TA{Name} for the supporting help throughout the development of this paper.

Xiaoxin zhou: Analyze the data, building models, training, maintain project repository on Github, 50% of the report writing.

Seit Sorra: Analyze the data, building models and running experiments (testing, validation), maintain project repository on Github, 50% of the report writing.

The workload for Xiaoxin Zhou and Seit Sorra is same in this project.

## VII. References

- [1] M. D. Juliana Tolles, "Review of Logistic Regression," JAMA, 02-Aug2016. [Online]. Available: <https://jamanetwork.com/journals/jama/article-abstract/2540383>.
- [2] T. Hastie, R. Tibshirani, and J. Friedman, "The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Second Edition," Elements of Statistical Learning: data mining, inference, and prediction. 2nd Edition. [Online]. Available: <https://web.archive.org/web/20091110212529/http://www.stat.stanford.edu/~tibsh/ElemStatLearn/>.
- [3] C. Cortez and V. Vapnik, "Support Vector Networks." [Online]. Available: [http://image.diku.dk/imagecanon/material/cortes\\_vapnik95.pdf](http://image.diku.dk/imagecanon/material/cortes_vapnik95.pdf).