

**TITANIC**  
**MACHINE LEARNING FROM DISASTER**

A Project Report Presented to  
Project Advisor: Professor *Dr. Birsen Sirkeci* faculty of Department of Electrical  
Engineering San José State University  
By

Kishore Kumar Bheemappa Hanabar & 011103690 EE258, Fall 2017  
Susmitha Kolli & 011462594 EE258, Fall 2017  
Email address: kishorekumar.bheemappahanabar@sjsu.edu &  
phone number: +1-669-262-9502  
12th December 2017

Department of Electrical Engineering  
Charles W. Davidson College of Engineering  
San José State University San Jose, CA 95192-0084

## **TABLE OF CONTENTS**

1. OBJECTIVES.....	3
2. INTRODUCTION.....	3
2.1 DATASET.....	3
2.2 Jupyter Imports.....	5
3. PRE-PROCESSING WORK.....	5
3.1 Dataset Classification.....	5
4. BASLINE MODEL and MODEL IMPROVEMENTS.....	7
4.1 Multilayer Perceptron.....	7
4.2 Recurrent Neural Network.....	10
4.3 Deep Neural Network.....	12
5. DISCUSSION & CONCLUSION.....	14
6. REFERENCE.....	15

# 1 OBJECTIVES

- Downloading and importing the dataset to ipython file.
- Cleaning the data and exploring the data through Visualization and matplotlib.
- Processing the data with different methods of neural network to achieve good accuracy that the model is perfectly fitting the dataset.
- Show Sudo-Code of Important Code steps.

## 2 INTRODUCTION

RMS Titanic sinking is one of the most famous shipwrecks. The titanic sank after collided with an iceberg and killing 1522 out of 2224 people on the ship. This incident Shocked the entire world and made people to think more about improving the safety measures on the ship.

The main reason for this tragedy is because there were not enough lifeboats for the people who were there on that ship. Here we are trying to predict who all survived and who didn't, who had a great probability to survive. This process is currently an active competition in Kaggle which is a big platform for data enthusiasts. It enables people to learn more in the field of data science.

### 2.1 Dataset

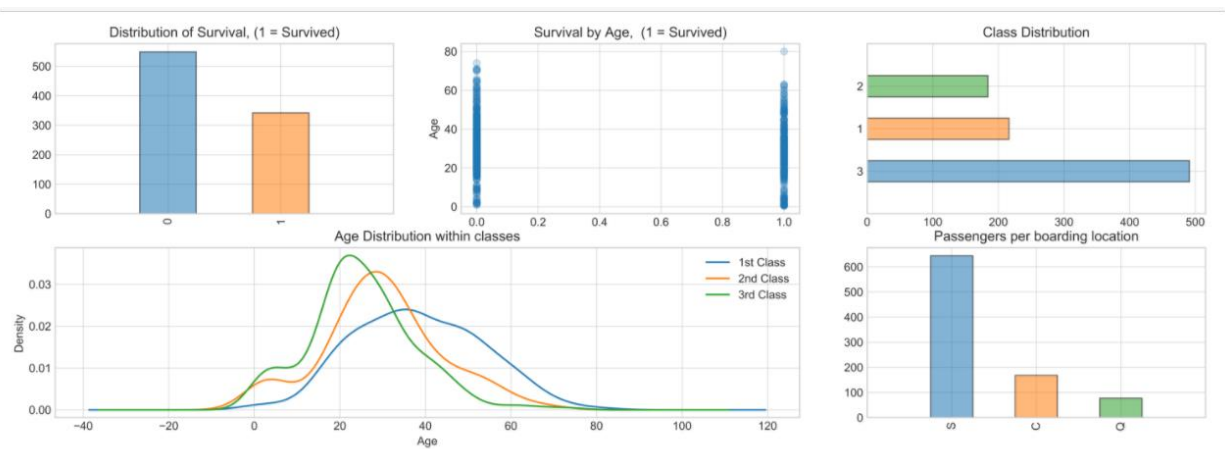
Titanic test and train dataset consists of unique features like passengerID, Passenger Class, name of each passenger, sex, age, number of siblings on the ship, parents, ticket number, fare of each ticket, Cabin number, and totall number of Embarked in different sections. Only train dataset consists a label of number of people Survived as this label is used to train the neural network.

```
train_df.head()
```

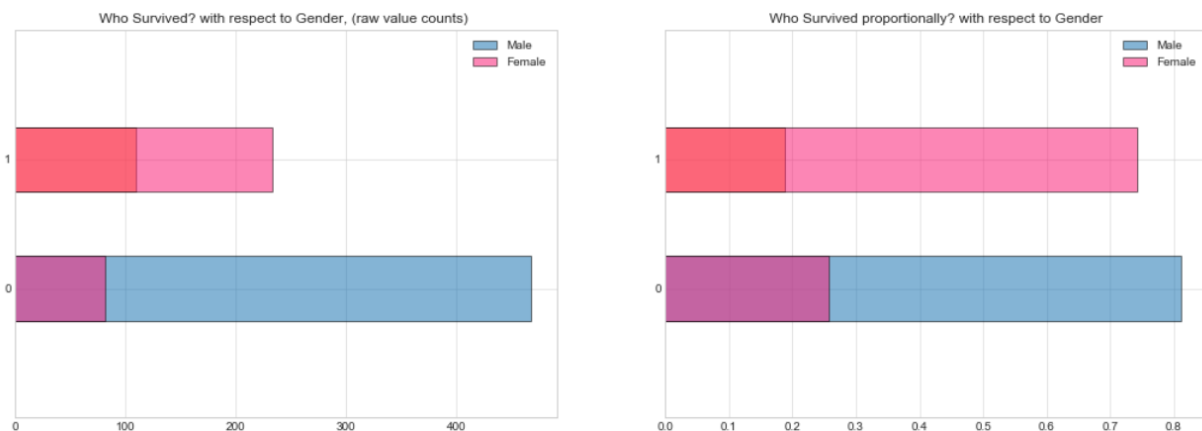
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 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

```
test_df.head()
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S



From above graph you can easily visualize the data clearly, first graph gives the distribution of number of people survived versus number of people died and as you can see (survived=1) survived people are very less compared to number of people died, in the second and third graph the same ratio is divided based on age and classes they were travelling in and you can clearly see the difference between them.



Two subplots showing the number of survived in category of male and female above in a bar graph and adjusted the bar graph so that it is showing the proportion of survival by gender.

Below I am showing the number of male and female based on class they were into in the Titanic ship and who survived and who didn't.



## 2.2 Jupyter Imports

```
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import Imputer
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from collections import namedtuple
from sklearn.preprocessing import Binarizer
plt.rcParams['patch.force_edgecolor'] = True
plt.rcParams['grid.color'] = 'k'
plt.rcParams['grid.linestyle'] = '-'
plt.rcParams['grid.linewidth'] = 0.5
sns.set_style('whitegrid')
```

```
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.optimizers import Adadelta
from keras.utils import np_utils
from keras.callbacks import Callback
```

## 3 PRE-PROCESSING WORK

After importing the dataset and visually explaining it we jump into data modification that we can use to process in any kind of neural networks.

In the dataset model replacing missing values in age, siblings, parents with suitable values using the padding function and then dropping the other unrelated features in the dataset which we don't use, or which give lot of noise if we consider it for feeding. They are PassengerId, Name, Ticket, Fare, Cabin, Embarked. Modified Training and Test data are show below.

```
train_data.head()
```

	Survived	Pclass	Sex	Age	SibSp	Parch
0	0	3	male	22.0	1.0	0.0
1	1	1	female	38.0	1.0	0.0
2	1	3	female	26.0	0.0	0.0
3	1	1	female	35.0	1.0	0.0
4	0	3	male	35.0	0.0	0.0

```
test_data.head()
```

	Pclass	Sex	Age	SibSp	Parch
0	3	male	34.5	0.0	0.0
1	3	female	47.0	1.0	0.0
2	2	male	62.0	0.0	0.0
3	3	male	27.0	0.0	0.0
4	3	female	22.0	1.0	1.0

3 dummy class are added to show the classes of the titanic and sex of male is changed 0 and sex of female is changed to 1

	Survived	Sex	Age	SibSp	Parch	Pclass_1	Pclass_2	Pclass_3
0	0	1	22.0	1.0	0.0	0	0	1
1	1	0	38.0	1.0	0.0	1	0	0
2	1	0	26.0	0.0	0.0	0	0	1
3	1	0	35.0	1.0	0.0	1	0	0
4	0	1	35.0	0.0	0.0	0	0	1

Still the Age features is in different scale as compared to all other features so normalized the age feature and brought the size back in the range 0 to 1

	Survived	Sex	Age	SibSp	Parch	Pclass_1	Pclass_2	Pclass_3
0	0	1	0.271174	1.0	0.0	0	0	1
1	1	0	0.472229	1.0	0.0	1	0	0
2	1	0	0.321438	0.0	0.0	0	0	1
3	1	0	0.434531	1.0	0.0	1	0	0
4	0	1	0.434531	0.0	0.0	0	0	1

This dataset is now ready to feed into the neural network with changing different hyperparameter while processing data and try to achieve best results out of it.

## 4 BASLINE MODEL and MODEL IMPROVEMENTS

### 4.1 MULTILAYER PERCEPTRON.

It is one the class of feed forward network and it uses supervised learning with non-linear activation function.

By using Sklearn learn library we are importing train\_test\_split function to split the train and test datasets and preparing the data for feeding the data into the neural network.

Built a single layer neural net by hardcoding and defining it as build\_neural network function as shown below. Data into network is passed as batches and the different results got by changing the hyperparameters is plotted and accuracy is calculated.

```
def build_neural_network(hidden_units=20):
    tf.reset_default_graph()
    inputs = tf.placeholder(tf.float32, shape=[None, train_x.shape[1]])
    labels = tf.placeholder(tf.float32, shape=[None, 1])
    learning_rate = tf.placeholder(tf.float32)
    is_training=tf.Variable(True,dtype=tf.bool)

    initializer = tf.contrib.layers.xavier_initializer()
    fc = tf.layers.dense(inputs, hidden_units, activation=None,kernel_initializer=initializer)
    fc=tf.layers.batch_normalization(fc, training=is_training)
    fc=tf.nn.relu(fc)

    logits = tf.layers.dense(fc, 1, activation=None)
    cross_entropy = tf.nn.sigmoid_cross_entropy_with_logits(labels=labels, logits=logits)
    cost = tf.reduce_mean(cross_entropy)

    with tf.control_dependencies(tf.get_collection(tf.GraphKeys.UPDATE_OPS)):
        optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)

    predicted = tf.nn.sigmoid(logits)
    correct_pred = tf.equal(tf.round(predicted), labels)
    accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

    # Export the nodes
    export_nodes = ['inputs', 'labels', 'learning_rate','is_training', 'logits',
                    'cost', 'optimizer', 'predicted', 'accuracy']
    Graph = namedtuple('Graph', export_nodes)
    local_dict = locals()
    graph = Graph(*[local_dict[each] for each in export_nodes])

    return graph

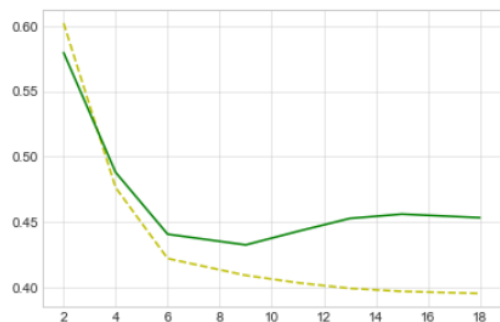
model = build_neural_network()
```

For 10 Hidden layers with 20 epochs and learning rate of 0.001 got accuracy of

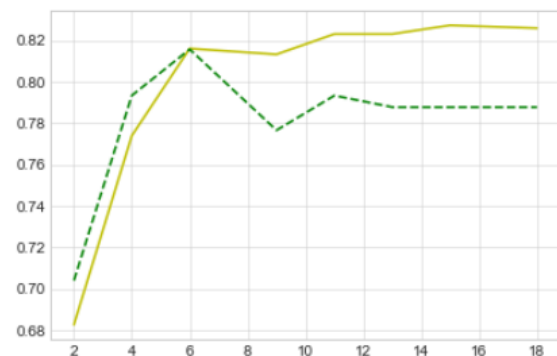
```
Train Loss: 0.4094 Train Acc: 0.8132
Validation Loss: 0.4327 Validation Acc: 0.7765
Train Loss: 0.3995 Train Acc: 0.8230
Validation Loss: 0.4529 Validation Acc: 0.7877
```

The Below graph shows the accuracy and loss of train and validation(test) of dataset on MLP using above mentioned parameters.

```
plt.plot(x_collect, train_loss_collect, "y--")
plt.plot(x_collect, valid_loss_collect, "g-")
plt.show()
```



```
plt.plot(x_collect, train_acc_collect, "y")
plt.plot(x_collect, valid_acc_collect, "g--")
plt.show()
```

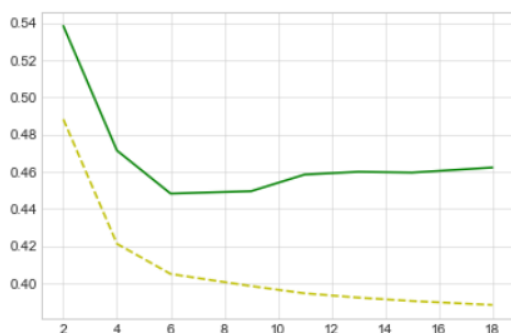


For 20 Hidden layers with 50 epochs and learning rate of 0.001 got accuracy of

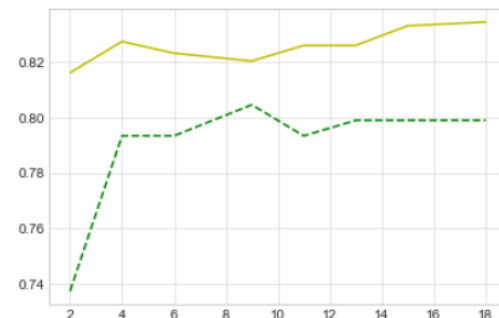
```
Validation Loss: 0.4599 Validation Acc: 0.7989
Train Loss: 0.3883 Train Acc: 0.8343
Validation Loss: 0.4622 Validation Acc: 0.7989
```

The Below graph shows the accuracy and loss of train and validation(test) of dataset on MLP using above mentioned parameters.

```
plt.plot(x_collect, train_loss_collect, "y--")
plt.plot(x_collect, valid_loss_collect, "g-")
plt.show()
```



```
plt.plot(x_collect, train_acc_collect, "y")
plt.plot(x_collect, valid_acc_collect, "g--")
plt.show()
```

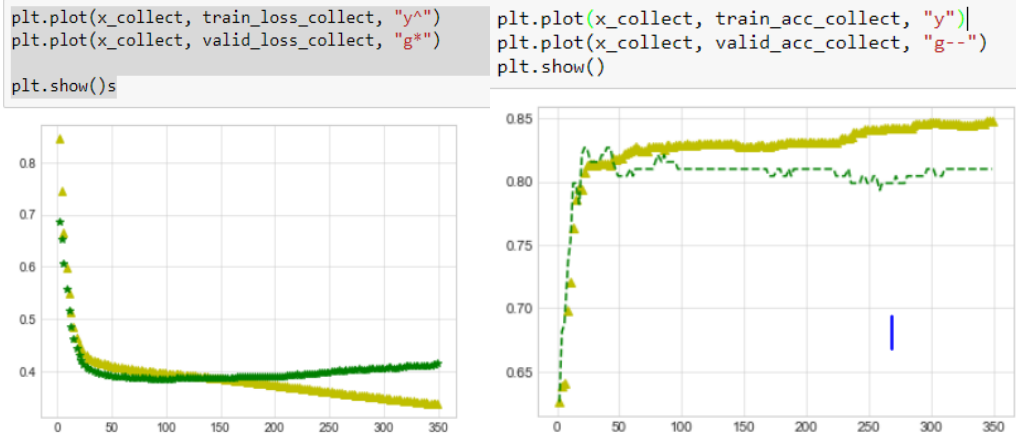




For 50 Hidden layers with 3500 epochs and learning rate of 0.0001 got accuracy of

```
Train Loss: 0.3327 Train Acc: 0.8539  
Validation Loss: 0.6341 Validation Acc: 0.7709  
Train Loss: 0.3313 Train Acc: 0.8553  
Validation Loss: 0.6404 Validation Acc: 0.7709  
Train Loss: 0.3300 Train Acc: 0.8539
```

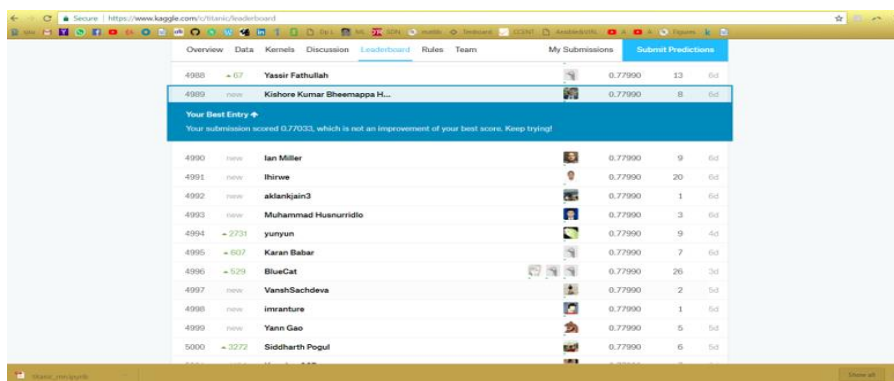
The Below graph shows the accuracy and loss of train and validation(test) of dataset on MLP using above mentioned parameters.



Achieved ranking of 4,989<sup>th</sup> out of 10,000 in Kaggle initial Submission which shown below.

Kaggle official name: Kishore Kumar Bheemappa Hanabar

Project: Titanic: machine learning from Disaster.



Rank	Score	Submission Name	Score	Rank
4988	0.77900	Yassir Fathallah	0.77900	13
4989	0.77900	Kishore Kumar Bheemappa H...	0.77900	8
4990	0.77900	Ian Miller	0.77900	9
4991	0.77900	Ithive	0.77900	20
4992	0.77900	akankjain3	0.77900	1
4993	0.77900	Muhammad Husnurridlo	0.77900	3
4994	0.77900	yunyun	0.77900	9
4995	0.77900	Karan Babar	0.77900	7
4996	0.77900	BlueCat	0.77900	26
4997	0.77900	VanshSachdeva	0.77900	2
4998	0.77900	imvantage	0.77900	1
4999	0.77900	Yann Gao	0.77900	5
5000	0.77900	Siddharth Pogul	0.77900	6

## **4.2 RECURRENT NEURAL NETWORK**

It is also another class of neural network. In this network connection between each neuron or unit form a directed cycle it uses internal memory to process the input as it takes sequence inputs.

In this model feature extraction is done entirely in a different manner as each feature in the dataset is converted to sequences of some length and then fed into the network.

Here we use LSTM which is one of the forms of neural network.

LSTM size is taken as 250, taken only one layer, with batch and learning rate 100, 0.001 respectively and taken maximum embed size as 300

```
embed_size = 300

with graph.as_default():
    embedding = tf.Variable(tf.random_uniform((n_words, embed_size), -1, 1))
    embed = tf.nn.embedding_lookup(embedding, inputs_)

with graph.as_default():
    lstm = tf.contrib.rnn.BasicLSTMCell(lstm_size)

    drop = tf.contrib.rnn.DropoutWrapper(lstm, output_keep_prob=keep_prob)

    cell = tf.contrib.rnn.MultiRNNCell([drop] * lstm_layers)
    |
    initial_state = cell.zero_state(batch_size, tf.float32)

with graph.as_default():
    outputs, final_state = tf.nn.dynamic_rnn(cell, embed,
                                             initial_state=initial_state)

with graph.as_default():
    predictions = tf.contrib.layers.fully_connected(outputs[:, -1], 1, activation_fn=tf.sigmoid)
    cost = tf.losses.mean_squared_error(labels_, predictions)

    optimizer = tf.train.AdamOptimizer(learning_rate).minimize(cost)
```

Obtained train loss and validation accuracy of 80% for training dataset

```
Epoch: 0/5 Iteration: 5 Train loss: 0.208
Epoch: 1/5 Iteration: 10 Train loss: 0.125
Epoch: 2/5 Iteration: 15 Train loss: 0.175
Epoch: 3/5 Iteration: 20 Train loss: 0.114
Epoch: 4/5 Iteration: 25 Train loss: 0.096
Val acc: 0.800
Epoch: 4/5 Iteration: 30 Train loss: 0.105
```

And on test dataset getting an accuracy of 85% which you can see below in the code.

```
test_acc = []
with tf.Session(graph=graph) as sess:
    saver.restore(sess, tf.train.latest_checkpoint('checkpoints'))
    test_state = sess.run(cell.zero_state(batch_len, tf.float32))
    for ii, (x, y) in enumerate(get_batches(test_x, test_y, batch_len), 1):
        feed = {inputs_: x,
                labels_: y[:, None],
                keep_prob: 1,
                initial_state: test_state}
        batch_acc, test_state = sess.run([accuracy, final_state], feed_dict=feed)
        test_acc.append(batch_acc)
    print("Test accuracy: {:.3f}".format(np.mean(test_acc)))
```

```
INFO:tensorflow:Restoring parameters from checkpoints\survival_preds.ckpt
Test accuracy: 0.850
```

### **4.3 DEEP NEURAL NETWORK**

An ANN with multiple hidden layer is known as Deep Neural Network know (DNN). Deep neural network means a neural network with many layers, a single layer neural network is show in our previous studies and Recurrent neural network is also show above and here now trying to analyze and get a better accuracy using Deep neural network (which contains more than one layers)

Using a different approach in feeding data to the network i.e. using keras library instead of Tensorflow what we used from the starting of our analysis.

In this model we have consider two hidden layers with activation function 'relu' for input and hidden layer and 'sigmoid' for the output layer

```
model = Sequential()

model.add(Dense(128, input_shape=(7, ), init="uniform"))
model.add(Activation('relu'))
model.add(Dropout(0.2))

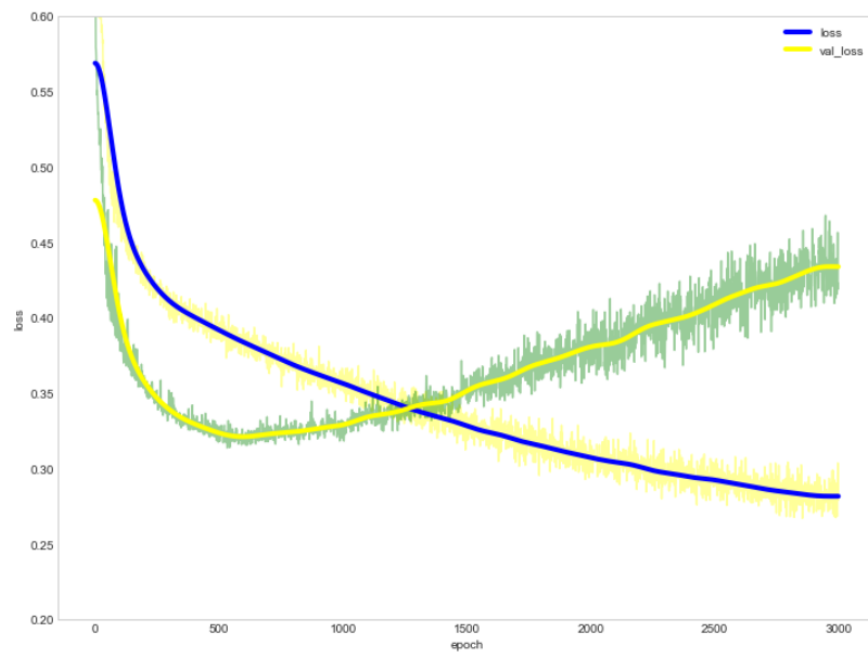
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.2))

model.add(Dense(2))
model.add(Activation('sigmoid'))

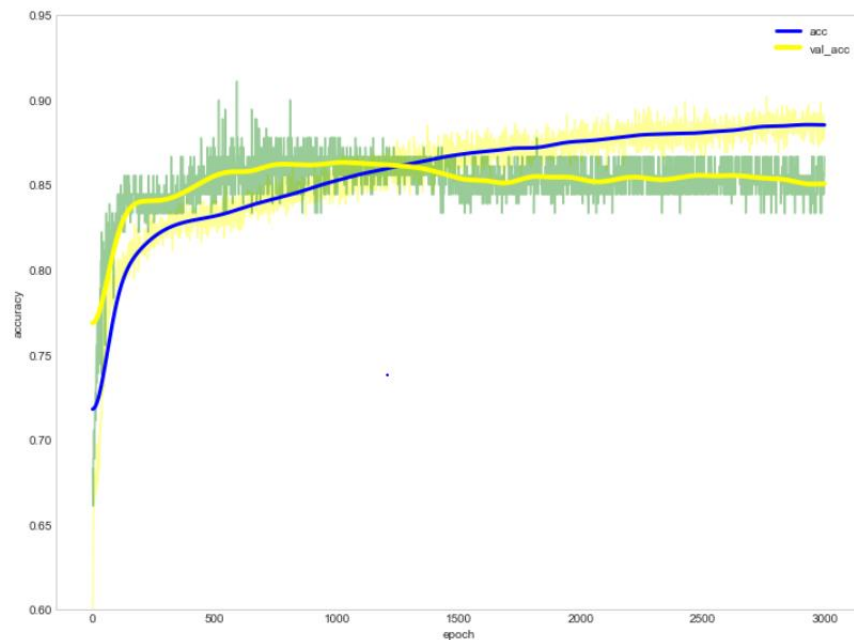
model.compile(loss='binary_crossentropy',
              optimizer=Adadelta(),
              metrics=['accuracy'])

call = Callback()
print("Training...")
hist = model.fit(X_train, y_train,
                batch_size=128,
                nb_epoch=3000,
                verbose=0,
                validation_split=0.1,
                callbacks=[call])
```

Below graph show the relation between loss and validation(test) loss parameters



Whereas over here plotted a graph of accuracy versus validation accuracy and shown the total Accuracy got from this model in a detailed manner.



fit\_acc

```
array([ 0.71808106,  0.71810412,  0.71815023, ...,  0.88534828,
        0.88534799,  0.88534782])
```

88.53% is the highest percentage of accuracy I have achieved in this model and from my previous leaderboard Kaggle rank of 4989 I have moved up to 1335<sup>th</sup> rank and there are total of 9,803 competitors are present in this competition.

The screenshot shows the Kaggle Titanic competition leaderboard. The user's submission, 'Kishore Kumar Bheemappa H...', is highlighted in blue, showing a score of 0.80382 and a rank of 1335. A blue banner below the submission states: 'Your Best Entry! You advanced 3,654 places on the leaderboard! Your submission scored 0.80382, which is an improvement of your previous score of 0.77990. Great job! Tweet this!'. The table lists other competitors with their scores and ranks.

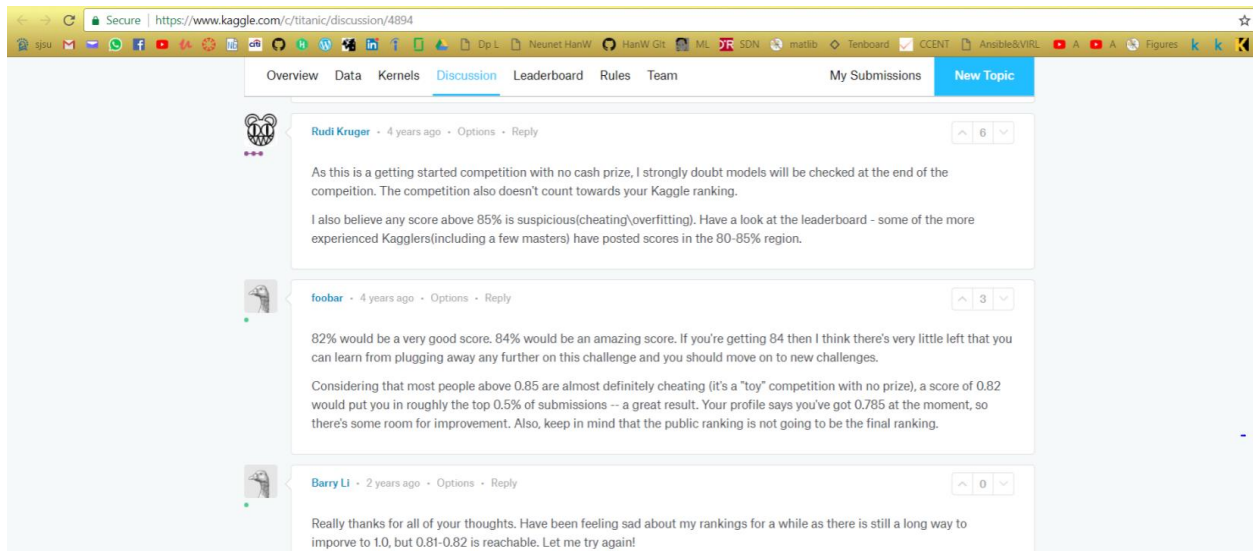
Rank	new	Username	Score	Placements	Time
1328	new	HriturajSingh	0.80382	1	1d
1329	new	antoinemrz 2	0.80382	3	1d
1330	new	Roan	0.80382	1	20h
1331	new	vishalrana16	0.80382	4	15h
1332	new	gravix	0.80382	15	11h
1333	new	ShankarGorkhe	0.80382	1	2h
1334	new	babusc112	0.80382	1	6m
1335	new	Kishore Kumar Bheemappa H...	0.80382	9	-10s
1336	▼ 129	vvenjuzina	0.79904	3	2mo
1337	▼ 129	unclemartin	0.79904	3	2mo
1338	▼ 129	hayyoshi	0.79904	2	2mo
1339	▼ 129	Michael Hutchinson	0.79904	6	2mo
1340	▼ 129	Bakuo	0.79904	6	2mo
1341	▼ 129	chalmersdan	0.79904	10	2mo
1342	▼ 129	JohnTravoltage	0.79904	16	1mo

## **5 DISCUSSION & CONCLUSION**

Submission format is a .CSV file which contains two main aspects of dataset i.e. PassengerId and Survived which give values of number of people survived. Results of this project is converted into a .csv format which is provided in the zip folder with the code and then it is compared with Kaggle results so the accuracy what we get over here might slightly vary from the accuracy we achieve in Kaggle submission. Totally 3 diverse types of neural network operations are tried on this dataset i.e. 1) Single Hidden MLP 2) RNN 3) DNN each Neural Net is processed in different ipynb files before combining it. DNN is my baseline model because I achieved my maximum accuracy of 88.53% while processing data with it and rank achieved in it is 1335 out of 9,803.

Main reason for this is explained as follows,

Accuracy in this competition you can achieve is of maximum about 87 to 88 percent to get better rank people change the values of their output csv file manually and try to match with the Kaggle answer, as the dataset is not huge possibility of doing it is high and after some trails you will get more than 90% accuracy. When I started searching of why I was not able increase my accuracy more than 88% then I got These Information from Kaggle Titanic project discussion board which you can see below from the screen shot.



## **6 REFERENCE**

- 1) <http://nbviewer.jupyter.org/github/agconti/kaggle-titanic/blob/master/Titanic.ipynb>  
Introductory data visualization inspiration
- 2) <https://github.com/pannous/tensorflow-speech-recognition>
- 3) <https://www.kaggle.com/c/titanic/kernels>
- 4) [http://proquest.safaribooksonline.com/book/programming/9781491971628/hello-tensorflow/titlepage01\\_html?query=\(\(tensorflow\)\)#snippet](http://proquest.safaribooksonline.com/book/programming/9781491971628/hello-tensorflow/titlepage01_html?query=((tensorflow))#snippet)
- 5) [http://proquest.safaribooksonline.com/book/programming/python/9781786464453/popular-open-source-libraries-an-introduction/ch03lv12sec25\\_html?query=\(\(keras\)\)#snippet](http://proquest.safaribooksonline.com/book/programming/python/9781786464453/popular-open-source-libraries-an-introduction/ch03lv12sec25_html?query=((keras))#snippet)