Contents

List of Figures

1 ABSTRACT

This research is aimed at achieving a detailed data analysis and understanding the effect or parameters key to the survival of individuals on board a civilian transportation vessel. Considering the case of Titanic, a British cruise, the analysis will be conducted. Using highly optimised algorithms and search functions, we would like to predict the survivability rates. The bio-inspired algorithms will be implemented to optimise the performance of classifiers. Towards the end, accuracies of different algorithms based on features fed to them will be compared in a tabular form.

2 INTRODUCTION

In the recent industrial era, where science and technology have been growing with an exponential pattern, commercial civilian transportation has turned into a paramount industry. The modes of travel for civilians have never been this elaborate as they are to date. Transportation has become much more standardised and has resulted in greater movement of high density clusters of population. But with this magnanimous scale, there also comes a boon as a result of fiddling with the laws of nature

Disruptive effects on transportation systems are cause by anthropogenic disasters which impact infrastructures, terminals and modes. Every from of transport involves nature as a medium of propagation. Air travel involves movement across the vast blue skies and water transportation involves the abundant water masses on our planet. Whenever mankind has shown pride and pretension on its work, nature has always taught it a lesson.

One of the most critical infrastructure in the modern century is defined to be civilian transportation, since a disruption in one of its components can have significant impact on a plethora of lives

Disaster come in two variants : man made or artificial disasters and natural calamities. Natural disasters include floods, volcanic eruptions and earthquakes while man made disasters include terrorism bombings, nuclear leakage and gas poisoining Certain natural calamities are inherently cause because of man made drudgery which makes this categorization of disasters a conflict in itself . In this essay on disaster management, we will be talking about the importance of disaster management as well as how well countries are prepared for the upcoming disasters.

2.1 MOTIVATION

To mitigate the risk of disasters in commercial or private civilian transportation, the emphasis on disaster management has never been to this magnanimous scale. Disaster Management plays a paramount role to mitigate the risk during the time of a catastrophe.

Whenever mankind has shown pride and pretension on its work, nature has always taught it a lesson. In the month of April, on the 10th day of 1912, the Royal Mail Ship Titanic began its journey into the vast atlantic ocean. Titanic was termed as the "unsinkable" ship. The engineers and architects who designed the vessel were so swollen up on their pride that they skimped many safety measures and deployed less life boats on the titanic as it was, the unsinkable. On the night of 14th april, titanic hit an iceberg in the atlantic and sunk. What was the cost of making an unsinkable ship I ask. 1500 dead including men, women and children. The pride of making an unsinkable ship resulted in one of the most deadliest incidents in the history of civilian transportation and the most extravagant sinking of a superliner or cruise ship to date.

This is the major inspiration for our project. After this incident a majority of the work was focused on risk and disaster mitigation for civilian travel methods. We want to develop and deploy a model, that can help the disaster management authorities to mitigate the risk and increase the probability of surviving of the passengers on board a civilian vessel. Even if our model is able to increase the survival rate incase of a disaster, it will be an honour to have worked on it. Our model predicts the probability of survival of each passenger on board using certain attributes and metrics from a given dataset and this probability can be used by risk and disaster management authorities to facilitate the appropriate amounts of safety precautions on board.

Failures are stepping stones to success even if they are never welcomed. The reasons for failure are often highly unambiguous and highly unconditional.It is a trickster with a sense of cunning and irony which takes pride in tripping people when success is right at the end of the tunnel

3 LITERATURE REVIEW

Certain works have been proposed to analyse the data from the titanic, as a result of the incident's unprecedented scale. Some of the techniques which have been reviewed and executed on the titanic data include a data mining approach to extract cumulative features and highlight higher correlations, using spectral feature selections as a part of scavenging the remains of the ship underneath the atlantic and an approach to normalise the data by using gaussian kernel parameter filtering for implementing on support vector machines.

3.1 METHODS USED IN BASE PAPER

Some of the methods used in base paper include the following approaches

3.1.1 PEDAGOGICAL METHODS OF MINING DATA

Published by Jain, Nikita, and Vishal Srivastava, in a conference regarding science and technology, described the use of data mining approach on the titanic dataset. Analyzing various attributes associated with different types of data, falls under the category of data mining. It is a process of finding patterns and combinations, based upon the fact that the symmetry of the patterns is conserved throughout

3.1.2 SELECTING FEATRES BASED ON SPECTRAL CLASSIFICATION

Published by Farag, Nadine and Ghada Hassan, in a conference regarding science and technology, proposed selecting features based on spectral classification for guided as well as unguided machine learning algorithms. It is one of the techniques used to find relevant features on mixed datasets. For reducing dimensional hierarchy and for building comprehensible models with highly functional generalization performance, feature selection is used. There are many different feature selection algorithms, which are dependent upon the use case scenario and the heuristics of the problem

3.1.3 INTRICACIES IN THE SIGMOID PARAMETERS

Han along with Claudio and Jun Morag proposed the idea that intricacies in the parameters of sigmoid function lead to influence the speed of the convergence of the global cost on the datasets. Certain challenges are enforced to task based learning because of higher order dimensions which is termed as the curse cause due to dimensions of higher order. Because of certain higher dimensional data which serves no purpose in a model, algorithms become prone to overfitting and under-fitting. For dimensionality reduction and to identify highly coherent features, feature selection is used

3.1.4   DEPLOYMENT OF ITERATIVE DICHOTOMISER 3

Peng, Wei et al proposed the implementation if the id-3 algorithm. ID-3 stands for Iterative Dichotomiser 3, which is a classification approach to solving optimisation problems using the greedy approach. One of the major advantages of this approach is that it generates understandable prediction rules from the dataset

3.1.5   GAUSSIAN KERNEL PARAMETERS

Xiao, Yingchao, et al. proposed the use of gaussian kernel parameters for one support vector machine class and their applications to fault detection. A gaussian is also called as a bell shaped curve, more traditionally, which does an exemplary mode of normalising higher dimensional datasets

4   PROBLEM IDENTIFICATION

One of the major challenges that hinders in providing effective disaster management computation is the lack of advanced computers on board commercial vessels. Surely, the current generation of machines demonstrate abilities to compute large chunks of data, but to date, many of the vessels and aircrafts which are being used for transportations are equipped with aged technology because they have been in service since generations. For such vessels and carriers, computing second order partial derivatives of cost functions which are required for traditional gradient descent algorithms will levy a very high computational price.

In this case we would like to propose the use of bio inspired algorithms. This comes as an analogue to what to teach us survivability better than the nature itself. Bio Inspired algorithms rely on linear algebra based calculations rather than complex higher order partial derivates, like the stochastic gradient descent. This makes bio inspired algorithms much more computationally feasible than standard backpropogation techniques.

To mitigate the risk of disasters in commercial or private civilian transportation, the emphasis on disaster management has never been to this magnanimous scale. Disaster Management plays a paramount role to mitigate the risk during the time of a catastrophe.

After the incident of the titanic a majority of the work was focused on risk and disaster mitigation for civilian travel methods. We want to develop and deploy a model, that can help the disaster management authorities to mitigate the risk and increase the probability of surviving of the passengers on board a civilian vessel. Even if our model is able to increase the survival rate incase of a disaster, it will be an honour to have worked on it. Our model predicts the probability of survival of each passenger on board using certain attributes and metrics from a given dataset and this probability can be used by risk and disaster management authorities to facilitate the appropriate amounts of safety precautions on board.

4.1 OBJECTIVE

The preliminary objective of our proposed model is to predict the probability of survival of a passenger on board a civilian vessel based on various attributes. Our aim is to minimise the risk in the transportation industry and be impactful to help save lives of people. Based on the projections of our model, disaster management authorities can understand which group of people have a lower chance of survival during a catastrophe, and they can implement safety measures in advance to increase the probability of survival

We also propose using computational methods which rely on linear algebra based calculations which have been derived after observing how the nature survives in the wild. This will not only levy minute computational costs but also boost the run time, lowering it down to exponential values

We want to develop and deploy a model, that can help the disaster management authorities to mitigate the risk and increase the probability of surviving of the passengers on board a civilian vessel. Even if our model is able to increase the survival rate incase of a disaster, it will be an honour to have worked on it. Our model predicts the probability of survival of each passenger on board using certain attributes and metrics from a given dataset and this probability can be used by risk and disaster management authorities to facilitate the appropriate amounts of safety precautions on board.

One of the emerging algorithms which is based on the principles of biological evolutions and inspired from it, are bio-inspired algorithms, which can be used to develop robust techniques. To address the optimal solutions to complex problems, in the recent years, optimisation algorithms based around nature are being implemented.

5   SYSTEM METHODOLOGY

In our project we have incorporated industry standards of testing and developing the design of our model . We follow some of the universally accepted guidelines for deploying artificial neural networks and machine learning algorithms. These help in easing our flow of work while providing high quality of throughput.

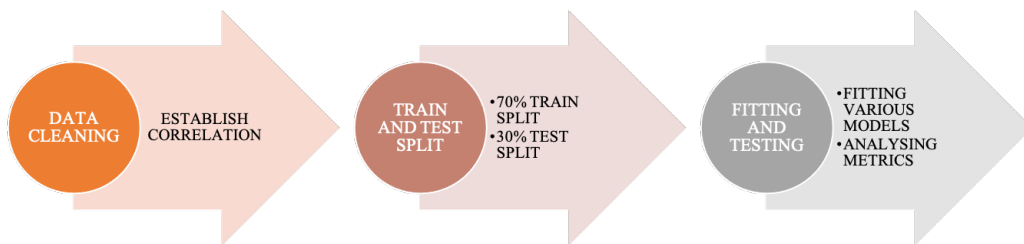The abstract level steps of the project development are illustrated in the figure below

Figure 1: Process flow chart

The broad steps include data manipulation, splitting of the data and testing and fitting of the models. Each of these steps ensure that the consistency in the dataset is maintained and any rogue values are eliminated giving us higher accuracies during the time of testing and training.

5.1 DATA CLEANING

This is one of the major steps which can make or break the entire model. We need to ensure that at every step in the training and testing phase, the model is provided with clean and consistent data. This includes that all the data fed in must be numerical and not categorical, there must be no missing and null values and that strongly correlated attributes are highlighted.
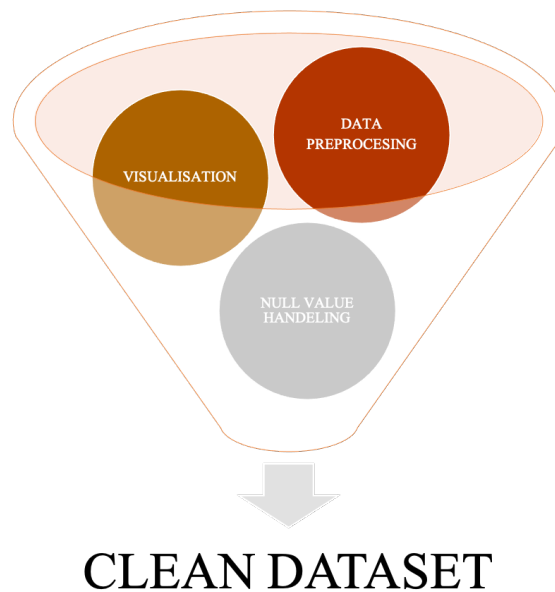

CLEAN DATASET

Figure 2: Eviscerating data

All the steps in the data preprocessing phase ensure that the data is highly consistent throughout. Emphasis is largely made on eliminating any occurring null values. The model does not work well under the influence of null values. Having null values increases the chances of bugs and errors in the data model which is highly non trivial. Also this might be highly non trivial, but null values take up more space than other placeholders

All the steps which ensure that there are no null values in our dataset are taken care in the data preprocessing phase. Not only are the null values truncated, but also categorical data must be replaced with equivalent numerical values. This is ensured by truncating categorical data into numeric counter parts. Machine

learning models do not recognise categorical data and hence they are truncated. One example of this is changing the label of male and female to 0 and 1 which stand respectively for their mentioned categorical counter parts.

The next part in this process is data visualisation. Knowing before hand, the attributes which are strongly co-related to each other ensures that the model can be emphasised to use them in the appropriate priority. We can setup a priority in training as to train highly co-related values with much emphasis cause these values have a higher chance of boosting accuracy figures
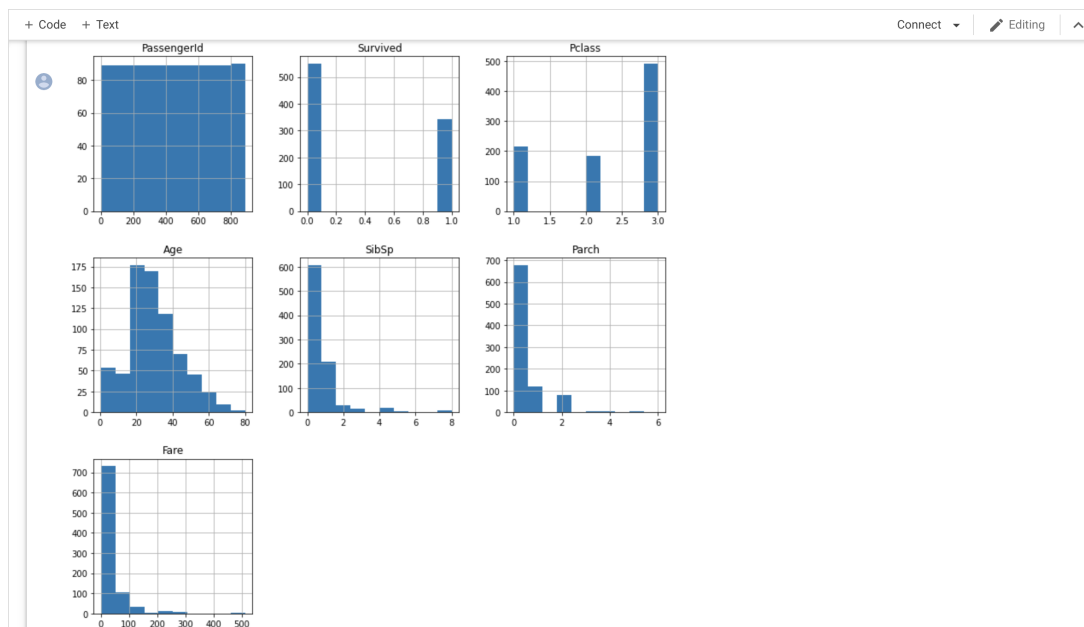

Figure 3: Data visualising

Emphasis on highly co-related data yields a higher chance in boosting accuracy metrics as co-related attributes directly impact on the survivability and probability rates.

5.2 TRAIN AND TEST SPLIT

We have split the data into 7:3 ratio in which the former part constitutes the training data and the latter the testing data. This ratio is appropriate the provide enough data in the testing phase such that the model neither overfits not does it under fit. This division is a sweet spot for general machine learning algorithms and works like a charm for our proposed as well as base paper implemented algorithms.
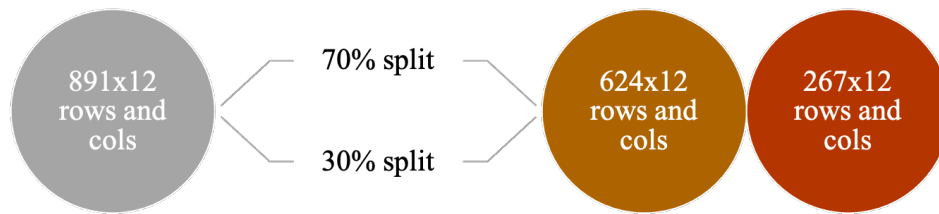

Figure 4: split scale

The main dataset contains 891 rows and 12 columns. After performing a 7:3 split the training data consists of the bulk of 624 rows and 12 columns whereas the testing data consists of 267 rows and 12 columns

5.3   MODEL FITTING AND TESTING

Now we arrive to the heart of our model, which is the model itself. In this phase we implement both the base paper algorithms which include the machine learning algorithms like support vector machines, decision tree classifiers and such, and also our proposed cutting edge flagship models which are Artificial Neural Networks and Bio-Inspired Particle Swarm Optimisation algorithms.
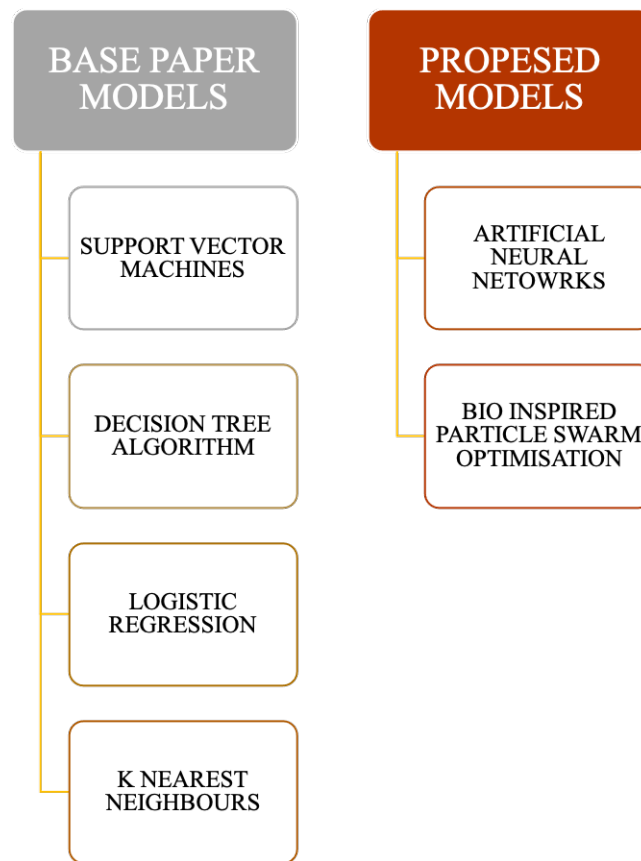

Figure 5: Implemented models

These models have been implemented to their highest working accuracy with respect to the dataset and have been observed to output desired results with tremendous accuracies. The accuracies of the base paper model also reconcile with the accuracies mentioned in the paper with corrections upto decimal point margin of errors

6 TECHNOLOGY DESIGN

We have used collaborative, open source and standardised software for building the model. The heart of this model is the Python programming language and the brain of the model is Tensorflow and Scikit learn libraries. We have ensured that all the technologies used are open source software, as it can be implemented by any disaster management authorities and scaled vertically or horizontally according to their needs without having to pay a penny.

We have also used data visualisation libraries like Matplotlib and data handling libraries like Pandas. Pandas help to create data frames which are easier to understand by the models and also easy to maintain and handle any future anomalies. Matplotlib functions help us to find highly co-related attributes by providing with methods which can visualise raw data.

We have also implemented the particle swarm optimisation algorithms from scratch and have built it on top of bio-isnspired code base. With the culmination of these elaborately defined libraries and tools, we were able to create a model could throughput probability survival rates with extremely precise accuracies.

Scikit learn is written extensively in python itself, and it uses numpy for higher order calculations likelinear algebra and matrix manipulations. It consists of many classes which can be used to implement both supervised and unsupervised machine learning algorithms.

Tensorflow is a library specifically designed for building complex neural networks. By using tensorflow one can build convolutional neural networks, Recurrent Neural Networks and can train them using different functions and optimisers. Tensorflow is completely open source and is used throughout the industry to build scalable artificial intelligence models. It is based on differential programming, and it treats objects as tensors. A tensor can be thought of as a high dimensional vector in pure mathematics.

6.1 ENVIRONEMNT SETUP

Python is the heart of the model. The entire source code as well as the driver code is written strictly in Python programming language. Python is an interpreter based language which is used to for general as well as scientific programming. It is an high level language which has been built on top of the C programming language

Figure 6: Python Programming Language

Python interpreters are supported for global operating systems and are available for mainstream programming language. It also has many open source reference libraries as well as additional plugins to boost run time like CPython.

Python has revolutionised how programming at an advanced scale can be implemented at such a collaborative level. It global presence and industry standard as well as open source architecture has enabled it to become on of the leading programming languages in the world.

We have run the python programming language over the google collaboratory notebooks. Using google colab, it becomes exponentially easier for one to work on collaborative project which specifically involve aspects of data science and artificial intelligence.

We write the code on the Jupyter notebooks and run it on a collaborative google colabs platform. Google colab provides the necessary tools so the the jupyter notebooks run 100 times faster than any other collborative environment. This is the main reason if have suited to use google colabs to run our Jupyter notebooks

Figure 7: Google Colab

It is highly effective to run arbitrary python code through a web server which specifically include methods of artificial intelligence and data science. It also provides a cloud based graphical processing unit capabilities, so that the system is never handicapped of lack of resources

6.2 LIBRARIES AND DEPENDENCIES

We have used google Tesorflow and Scikit Learn to implement our models. These libraries are rigorously worked out open source libraries which provide highly effective functions to deploy scalable machine learning and artificial intelligence models.

Scikit learn is written extensively in python itself, and it uses numpy for higher order calculations like linear algebra and matrix manipulations. It consists of many classes which can be used to implement both supervised and unsupervised machine learning algorithms.

Figure 8: Tensorflow

Figure 9: Scikit Learn

Tensorflow is a library specifically designed for building complex neural networks. By using tensorflow one can build convolutional neural networks, Recurrent Neural Networks and can train them using different functions and optimisers. Tensorflow is completely open source and is used throughout the industry to build scalable artificial intelligence models. It is based on differential programming, and it treats objects as tensors. A tensor can be thought of as a high dimensional vector in pure mathematics.

Using these two libraries as a framework we were able to implement a plethora of machine learning models with extremely high accuracy rates and completely open source in nature.

7   IMPLEMENTATION

The model has been implemented with python as the backbone programming language and also using libraries like Tensorflow, Scikit learn and Matplotlib. These libraries provide us with extensive tools and work environments so that the models can be deployed and run easily.

Python interpreters are supported for global operating systems and are available for mainstream programming language. It also has many open source reference libraries as well as additional plugins to boost run time like CPython.

Scikit learn is written extensively in python itself, and it uses numpy for higher order calculations like linear algebra and matrix manipulations. It consists of many classes which can be used to implement both supervised and unsupervised machine learning algorithms.

Tensorflow is a library specifically designed for building complex neural networks. By using tensorflow one can build convolutional neural networks, Recurrent Neural Networks and can train them using different functions and optimisers. Tensorflow is completely open source and is used throughout the industry to build scalable artificial intelligence models. It is based on differential programming, and it treats objects as tensors. A tensor can be thought of as a high dimensional vector in pure mathematics.

In our project we have incorporated industry standards of testing and developing the design of our model . We follow some of the universally accepted guidelines for deploying artificial neural networks and machine learning algorithms. These help in easing our flow of work while providing high quality of throughput.

In the following sections we will lay out the details of the code and how their design and testing in the said environments went by.

7.1 CODE

Primarily, in the colab notebook, we start by importing our trivial dependencies which have already been rigorously covered in the previous sections of this documentation.

Figure 10: Importing dependencies and peaking at the dataset

We import Pandas, for data handling, Matplotlib for visualisation and Numpy for matrix manipulations which will be used in data preprocessing phase. Seaborn is a style of matplotlib which gives certain visual appeal to our graphs and plots.

7.1.1 DATA PREPROCESSING

Next we move on to the data visualisation phase. In this phase our aim is to understand which attributes are highly co-related to each other and which attributes correspond the most to improving the accuracy of our model.

Knowing before hand, the attributes which are strongly co-related to each other ensures that the model can be emphasised to use them in the appropriate priority. We can setup a priority in training as to train highly co-related values with much emphasis cause these values have a higher chance of boosting accuracy figures
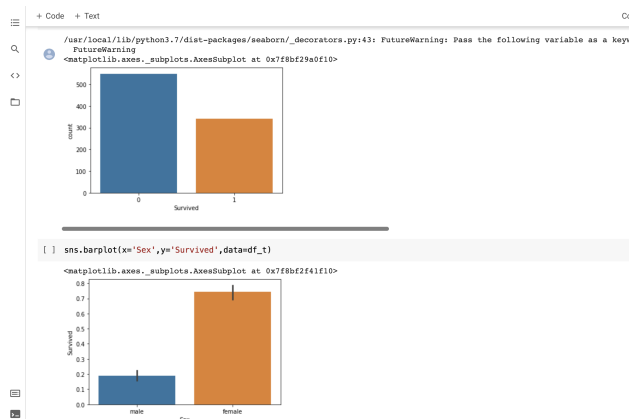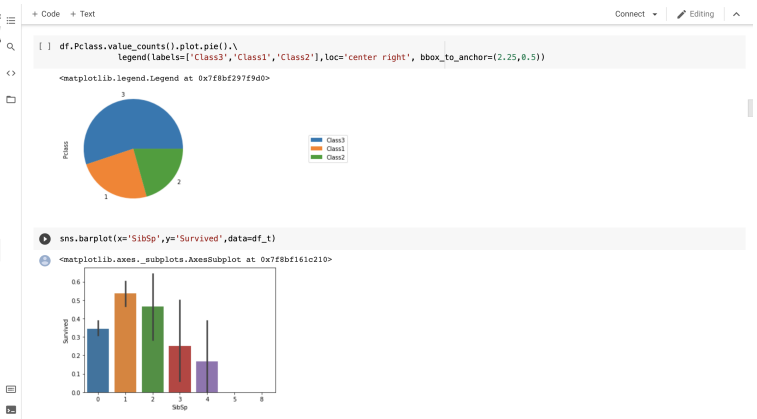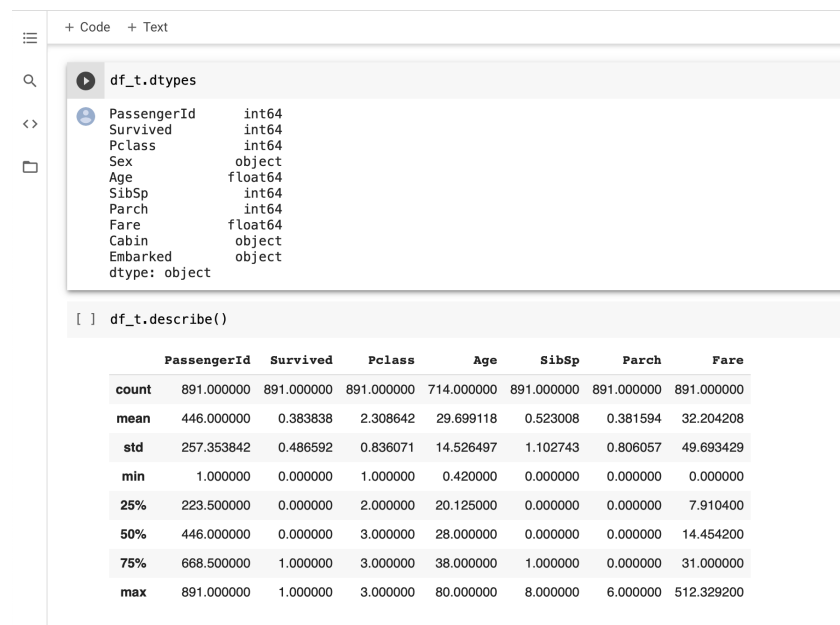

Figure 11: Data visualisation

Figure 12: Data visualisation

After getting around the intricacies of the data, we would like to eliminate any null values or categorical data. For this we look at the type of data value each column holds and check for any null values. If there exist any null values, We will truncate those columns

Emphasis on highly co-related data yields a higher chance in boosting accuracy metrics as co-related attributes directly impact on the survivability and probability rates.

All the steps in the data preprocessing phase ensure that the data is highly consistent throughout. Emphasis is largely made on eliminating any occurring null values. Having null values increases the chances of bugs and errors in the data model which is highly non trivial. Also this might be highly non trivial, but null values take up more space than other placeholders

This is one of the major steps which can make or break the entire model. We need to ensure that at every step in the training and testing phase, the model is provided with clean and consistent data. This includes that all the data fed in must be numerical and not categorical, there must be no missing and null values and that strongly correlated attributes are highlighted.

Figure 13: Data-type validation

All the steps which ensure that there are no null values in our dataset are taken care in the data preprocessing phase. Not only are the null values truncated, but also categorical data must be replaced with equivalent numerical values. This is ensured by truncating categorical data into numeric counter parts. Machine learning models do not recognise categorical data and hence they are truncated.

After this step we can see that put data has no null values and that it is highly consistent with no categorical attributes too

```
[ ] df_t.info()

    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 891 entries, 0 to 890
    Data columns (total 9 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   Sex          891 non-null    int64
     4   Age          714 non-null    float64
     5   SibSp        891 non-null    int64
     6   Parch        891 non-null    int64
     7   Fare         891 non-null    float64
     8   Embarked     889 non-null    object
    dtypes: float64(2), int64(6), object(1)
    memory usage: 62.8+ KB
```

Figure 14: Validating for null values

7.2 TESTING

We have now successfully truncated all the null and categorical data from our dataset and now we can ensure that our model will be fed with clean and consistent data throughout the entire phase of testing and training. This makes our data highly trivial and ensures consistent accuracies throughout the board

7.2.1 TESTING THE BASE PAPER MODELS

Now we come to the heart of our project that is to build the models. We start by exporting all our dependencies and also the library Functions. We implement all the algorithms mentioned in the base paper in a sequential order.

First we perform our train and test split in the ration of 70 and 30 percents respectively. For this we mention the test size in the code as 0.3 and the remaining 0.7 will automatically be reserved for training

```
[ ]  from sklearn.model_selection import train_test_split
     from sklearn import preprocessing
     X = df_t.drop(['Survived','PassengerId','Fare'],axis=1)
     y = df_t['Survived']

     X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=3296)
```

Figure 15: Train Test and Split formation

Now its time to move to building the base paper algorithms. With each model, the accuracy is mentioned with the code snippet. The algorithm accuracy reconciles with that mentioned in the base paper with minimal margin of error.

We start with the decision tree algorithm which yields an accuracy of 85.7 percent, which is in hair thin margin of error. The next algorithm which has been implemented is the k nearest neighbours algorithm which yields an accuracy of 75.6 percent, matching the value of the base paper figures

27

Next we move on to Logistic regression and Support Vector Machines, which yield accuracies of 82 percent and 82.3 percent respectively matching the numbers of the base paper.
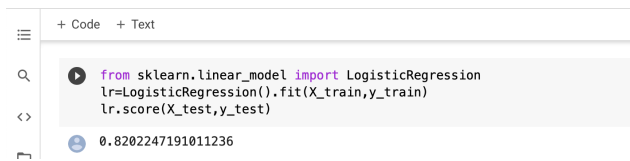
Figure 16: Decision Trees
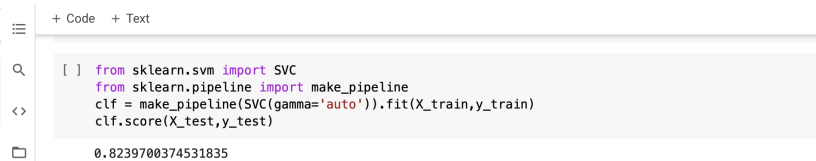
Figure 17: K Nearest Neighbours

Figure 18: Logistic Regression

Figure 19: Support Vector Machine

By this we have successfully implemented all the base paper algorithms with the accuracies aligning with the values mentioned in the paper. The algorithm with the highest accuracy in the base paper models was the decision tree classifier coming in at about 85.7 percent accuracy

7.2.2 DESIGNING AND TESTING PROPOSED MODELS

After the succesfull implementation of the base paper algorithms its time to implement proposed system designing and testing

ARTIFICIAL NEURAL NETWORKS Initially we will start by implementing our custom neural network. We have implemented a four layer sequential neural network, with the fourth layer being the output layer and the first layer being the input layer.

The first layern comprises of 39 Dense neurons. Dense, in neural networks means that all the nodes in this layer will be connected to each node in the next layer. There is a 1 to all relation between each node of first layer to every other node in the next layer and vice versa. The next layer is also a dense layer, but with only 27 neurons.

Hence, in total, all the 39 neurons of the first layer are connected to all the 27 neurons in the next layer. Hence there are a total of $39 * 27 = 1026$ connections between the first two layers

The third layer consists of 19 neurons which are again dense in nature. Which means that these 19 neurons will be connected to each of the previous 27 neurons. Hence in total there are $19 * 27 = 513$ connections between the 2nd and third layer.

Now the final layer consists of only one neuron as it is the output layer. Even this layer is a dense layer which means it is connected to all the previous 19 neurons of the third layer. Hence there are a total of $19 * 1 = 19$ connections between the last and third layer. In total our model has $1026 + 513 + 19 = 1558$ connections.

Moving on to the activation functions, the first layer uses an activation function called Rectified Linear Unit or simply called RELU. Relu, is preferred over the widely using sigmoid function because it avoids the problems of vanishing gradients. Relu takes an input and returns zero, if the value is less than zero, or return the input itself if the value is greater than zero.

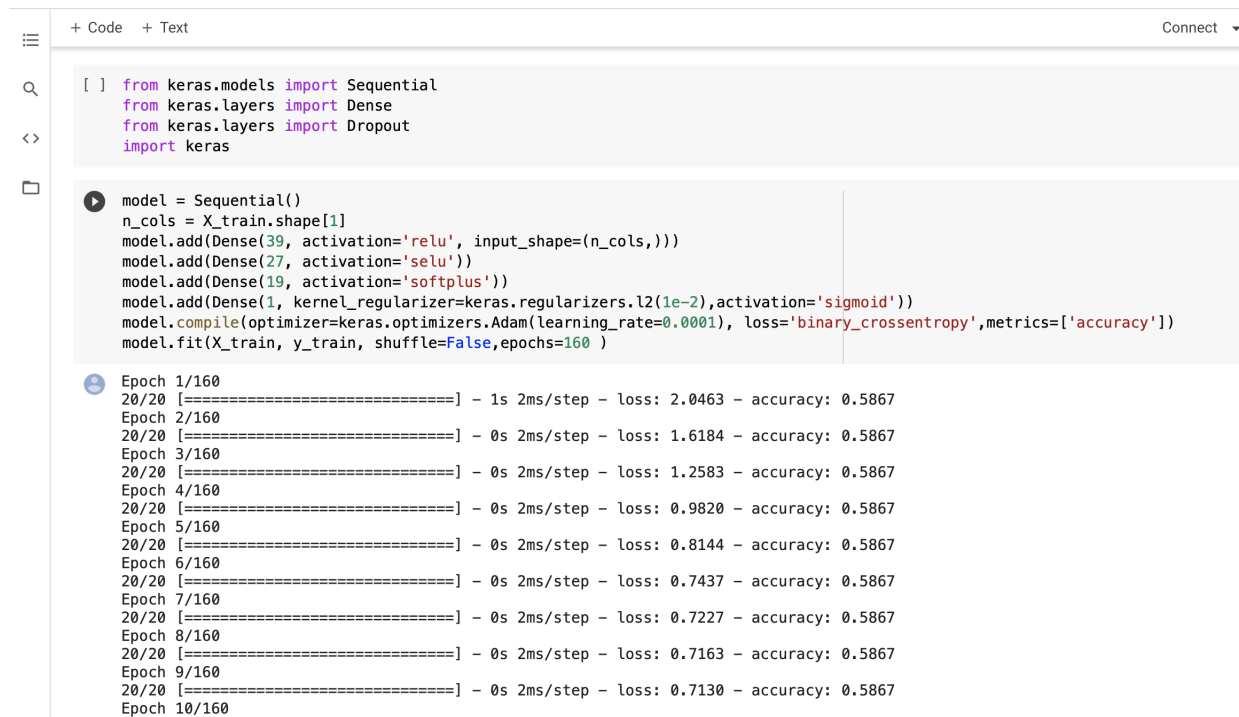$$Relu = max(0, x) \quad \text{Where x is the input} \tag{1}$$

The second layer uses what is called Exponential Linear Unit as a primary activa-

tion method. It is used to avoid the problem of static relu. Elu scales the input by a certain degree to avoid static relu problems

$$Elu = \begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases} \tag{2}$$

The third layer uses the softmax activation function, which turns the entire input into probability density. Softmax is used to parse an input into a probability density than can be later parsed into a sigmoid distribution.

Finally we compile the model using the Adam optimiser which is used in the place of traditional stochastic gradient descent. Adam has been derived from the phrase Adaptive Motion Estimation. Adam does not have any sort of constraints on hyper parameter tuning as well as they are quite intuitive in nature. It is also not computationally dense and can be used of large datasets without any hassle.

```
+ Code   + Text                                                                        Connect ▾

[ ]  from keras.models import Sequential
     from keras.layers import Dense
     from keras.layers import Dropout
     import keras

 ▶   model = Sequential()
     n_cols = X_train.shape[1]
     model.add(Dense(39, activation='relu', input_shape=(n_cols,)))
     model.add(Dense(27, activation='selu'))
     model.add(Dense(19, activation='softplus'))
     model.add(Dense(1, kernel_regularizer=keras.regularizers.l2(1e-2),activation='sigmoid'))
     model.compile(optimizer=keras.optimizers.Adam(learning_rate=0.0001), loss='binary_crossentropy',metrics=['accuracy'])
     model.fit(X_train, y_train, shuffle=False,epochs=160 )

 ●   Epoch 1/160
     20/20 [==============================] - 1s 2ms/step - loss: 2.0463 - accuracy: 0.5867
     Epoch 2/160
     20/20 [==============================] - 0s 2ms/step - loss: 1.6184 - accuracy: 0.5867
     Epoch 3/160
     20/20 [==============================] - 0s 2ms/step - loss: 1.2583 - accuracy: 0.5867
     Epoch 4/160
     20/20 [==============================] - 0s 2ms/step - loss: 0.9820 - accuracy: 0.5867
     Epoch 5/160
     20/20 [==============================] - 0s 2ms/step - loss: 0.8144 - accuracy: 0.5867
     Epoch 6/160
     20/20 [==============================] - 0s 2ms/step - loss: 0.7437 - accuracy: 0.5867
     Epoch 7/160
     20/20 [==============================] - 0s 2ms/step - loss: 0.7227 - accuracy: 0.5867
     Epoch 8/160
     20/20 [==============================] - 0s 2ms/step - loss: 0.7163 - accuracy: 0.5867
     Epoch 9/160
     20/20 [==============================] - 0s 2ms/step - loss: 0.7130 - accuracy: 0.5867
     Epoch 10/160
```

Figure 20: Compiling the Neural Network

As illustrated, We have successfully compiled our model and it has been designed

to run the test data over 160 epochs. 160 epochs, means that the model will be trained with the dataset over 160 iterations.

We have used the binary cross entropy as our loss function. The name of the loss function, is what an enthusiast will call a Log Loss. Binary cross entropy is a log loss function, and as we need discrete probabilities as our output, we have avoided using conventional mean squared errors as they are better suited for continuous valued Projections

PARTICLE SWARM OPTIMIZATION is a Bio-inspired heuristic search optimisation algorithm, that has been developed on the grounds of studying and observing the survival nuances of the wild. It is an iterative computational method which finds an optimal solution in a batch of candidate solutions based upon certain constraints which are enforced throughput the problem.

Bio- inspired Particle Swarm Optimisation, has been developed and modelled on, after observing groups of animals like a flock of birds or a school of fishes.

Nature has been a source of inspiration for many great endeavours and what better to teach us survival, than the nature itself. Survival instinct can be visualised among a group of animals by observing their behaviour and the way animals in the environment communicate with each other in groups and flocks

To find various optimal solutions upto a diverse degree of optimisation, and to eliminate the problem of traditional optimisation heuristic algorithms, Bio-inspired algorithms are being deployed. To tackle the concerning barriers to traditionally deployed algorithms, biologically synthesised algorithms, show a tremendous promise and a diverse future

One of the other main reasons to deploy bio inspired algorithms is that they are computationally less expensive. When compared to traditional algorithms like the Stochastic Gradient Descent algorithm, which uses first and second order partial differential equations for search optimisation, bio inspired algorithms use linear algebra which is not only less expensive on the syste, hardware but also faster to compute.

Particle Swarm Algorithm is a metaheuristic (Problem independent), bio inspired search optimsiation algorithm. Observations were made in the mid 19th century that laid grounds to the fact that animals is groups such as a flock of birds, communicate more effectively using natural patterns which increases the likelihood of survival and because of this they are able to survive the wild for substantial longer periods of time. This laid to the foundation of research in Bio-Inspired algorithms.

In a certain context, the movement of birds is choreographic in nature, as it is synchronised to the stroke of a constant in the view of a maximisation and minimisation problem.

To better get a visualisation of this algorithm consider the following scenario. When a flock of birds fly over a place they encounter problem which involve optimisations. Some of this include the fact that there is a maximisation problem, which would be to gather as much of food sources as possible and also a minimisation problem which would be to land in a geographical area which is less prone to predators.

Converging towards the technical details of implementation, The main crux of a Particle Swarm Algorithm include Position Vector, Momentum Vector, Fitness Function, Particles best position, Global Best position. The position vector is the goal of the optimisation problem, The fitness function is the objective function which determines the position, the velocity vector represents the speed at which the swarm moves. The updation occurs as follows

$$X_{ij}^{t+1} = X_{ij}^{t} + V_{ij}^{t+1} \tag{3}$$

$$V_{ij}^{t+1} = wV_{ij}^{t} + c_1 r_1^{t}(pbest_{ij} - X^{t}ij) + c_2 r_2^{t}(gbest_j - X_{ij}^{t}) \tag{4}$$

In an abstract view to optimise a problem tailored for particle swarm optimisation, we initialise a solution space with random population of points, which is analogous to our input data. the next step is updating the position and momentum vectors of each point, enforcing the above updation relations, and generating new solution space which is more optimal than the previous.

Each updated solution space is termed as a generation. We iteratively update the position and global and local solutions of every generations, while keeping every future generation more optimal than its ancestors. In each generation more and more points, which are termed as particles, get more and more optimal and the small local optimisations, turn into vast global optimisations over the course of iteration.

For each particle in a population, we randomly initialise the position and momentum vectors. Based on these values a fitness function is calculated which describes the optimality of our problem. Then the vectors are updated based on the above rules and then the local fitness function will be calculated. If this new fitness function is more optimal than the previous function, the new local function will be set as the initial function in the next generation. With this small local optimisations translate to large global optimisations.

The local best functions are updated with force according to the global best functions. Global updations trump Local updations in particle swarm optimisation as illustrated below
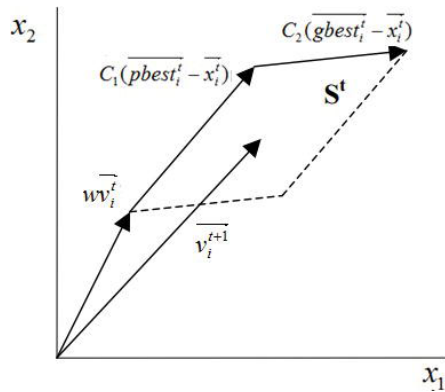
Figure 21: Before updation
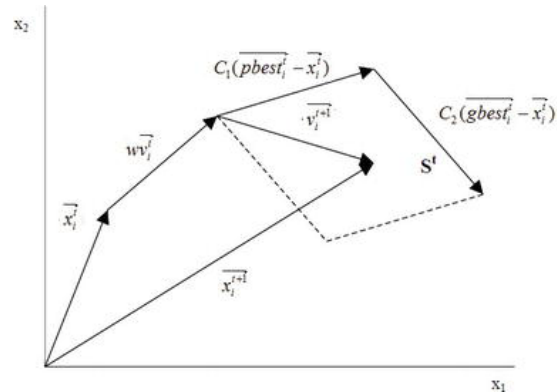
Figure 22: After updation

Now we move on to implementing this in out code. We import all the libraries in the first few cells

```
[ ]  import numpy as np
     import matplotlib.pyplot as plt
     import pyswarms as ps
     %load_ext autoreload
     %autoreload 2
```

Figure 23: Importing dependencies

Next we define two new classes which are particle class and the swarm class. Each class is initially randomly initialised as mentioned in the intricacies of the algorithm.

```python
import numpy as np

# This is a PSO(interia weight) variation...
class Particle:
    """
    Particle class represents a solution inside a pool(Swarm).
    """

    def __init__(self, no_dim, x_range, v_range):
        """
        Particle class constructor
        :param no_dim: int
            No of dimensions.
        :param x_range: tuple(double)
            Min and Max value(range) of dimension.
        :param v_range: tuple(double)
            Min and Max value(range) of velocity.
        """
        self.x = np.random.uniform(
            x_range[0], x_range[1], (no_dim,)
        )  # particle position in each dimension...
        self.v = np.random.uniform(
            v_range[0], v_range[1], (no_dim,)
        )  # particle velocity in each dimension...
        self.pbest = np.inf
        self.pbestpos = np.zeros((no_dim,))
```

Figure 24: Particle class

```python
class Swarm:
    """
    Swarm class represents a pool of solution(particle).
    """

    def __init__(self, no_particle, no_dim, x_range, v_range, iw_range, c):
        """
        Swarm class constructor.
        :param no_particle: int
            No of particles(solutions).
        :param no_dim: int
            No of dimensions.
        :param x_range: tuple(double)
            Min and Max value(range) of dimension.
        :param v_range: tuple(double)
            Min and Max value(range) of velocity.
        :param iw_range: tuple(double)
            Min and Max value(range) of interia weight.
        :param c: tuple(double)
            c[0] -> cognitive parameter, c[1] -> social parameter.
        """
        self.p = np.array(
            [Particle(no_dim, x_range, v_range) for i in range(no_particle)]
        )
        self.gbest = np.inf
        self.gbestpos = np.zeros((no_dim,))
```

Figure 25: Swarm class

The variables are initialised as shown in the snippet. we initialise all the variables as a part of a class of either the swarm or the particle. Local variables are initialised to the particle class and the global variables are initialised to the swarm

```python
        self.gbest = np.inf
        self.gbestpos = np.zeros((no_dim,))
        self.x_range = x_range
        self.v_range = v_range
        self.iw_range = iw_range
        self.c0 = c[0]
        self.c1 = c[1]
        self.no_dim = no_dim
```

Figure 26: Global and local variables

Each updated solution space is termed as a generation. We iteratively update the position and global and local solutions of every generations, while keeping every future generation more optimal than its ancestors. In each generation more and more points, which are termed as particles, get more and more optimal and the small local optimisations, turn into vast global optimisations over the course of iteration.

Next we call the main driver function and start optimising our solution spaces. In the end we print out the accuracy of our particle swarm optimisation model

```python
if __name__ == '__main__':
    no_solution = 100
    no_dim = (INPUT_NODES * HIDDEN_NODES) + HIDDEN_NODES + (HIDDEN_NODES * OUTPUT_NODES) + OUTPUT_NODES
    w_range = (0.0, 1.0)
    lr_range = (0.0, 1.0)
    iw_range = (0.9, 0.9)  # iw -> inertial weight...
    c = (0.5, 0.3)  # c[0] -> cognitive factor, c[1] -> social factor...

    s = Swarm(no_solution, no_dim, w_range, lr_range, iw_range, c)
    #Y = one_hot_encode(Y) #Encode here...
    s.optimize(forward_pass, X, y, 100, 1000)
    W = s.get_best_solution()
    Y_pred = predict(X, W)
    accuracy = get_accuracy(y, Y_pred)
    print("Accuracy: %.3f"% accuracy)
```

Figure 27: Driver code

With this we conclude the construction, testing and successful implementation of all our proposed models which display robust techniques and competitive accuracies.

8 RESULTS AND VALIDATION

Having implemented all the algorithms successfully, we have to now analyse all the accuracies which we have obtained by training and testing our model. The below tabular illustrates an overall accuracy picture which will give us detailed information for analysis purpose.

Table 1: Accuracy Validation (In Percentile)

Algorithm	Base Paper	Team Implementation
Decision Trees	85.5	85.7
K nearest neighbours	78.3	75.6
Logistic Regression	81.6	82.2
Support Vector Machines	82.1	82.0
Artificial Neural Networks	–	86
Particle Swarm Optimisation	–	83.4

From this we conclude that our implemented Custom Neural Network gives the highest accuracy of 86 percentile while giving the lowest amount of False values. The accuracies vary with a certain degree on margin of error because of system constraints and definition of functions. This margin of error is minimal as it was intended

Both our flagship models performed highly accurately while giving satisfying results. These results may vary after changing certain hyperparameters and depends upon the type of system used for testing as well

Our other main reasons to deploy bio inspired algorithms is that they are computationally less expensive. When compared to traditional algorithms like the Stochastic Gradient Descent algorithm, which uses first and second order partial differential equations for search optimisation, bio inspired algorithms use linear algebra which is not only less expensive on the system, hardware but also faster to compute.

We have successfully used the biggest strength of particle swarm optimisation which is parallelizability. The same algorithmic definition can be run with minute to null changes on various different classes of machines like high performing computers or single and multi-core computers. This provides a degree of freedom that no other algorithm does.

9 CONCLUSION

Nature has vividly captured the zeitgeist of survival and divulgence, throughout generations, and has bestowed mankind with the wealth of its knowledge. It has been a driver for sourcing inspiration into the development and deployment of highly optimised algorithms which happen to solve the problem present due to classical algorithms

We had picked the problem of optimising disaster management predictions in the case of natural or artificial catastrophes on board a civilian transportation service vessel.

We conclude that our models accurately predict the survivability of a passenger based on certain metrics provided to our custom learning algorithms, which can be used to provide a certain degree of refuge and increases the likelihood of survival of each individual

The highest accuracy was achieved by our custom developed neural network. All the hyper-parameters of our neural network were selected after a slew of trail and error runs, and in the end we were able to throughput an accuracy of 86 percent from our model. We were also saw satisfying performance from our bio-inspired algorithms, through which we were successfully able to exploit high parallelizability

Both of our flagship models performed highly accurately while giving satisfying results. These results may vary after changing certain hyper-parameters and depends upon the type of system used for testing as well

Based on the projections of our model, disaster management authorities can understand which group of people have a lower chance of survival during a catastrophe, and they can implement safety measures in advance to increase the probability of survival

All the algorithms implemented are highly optimised and yield desirable throughputs. Our neural networks outperform them all and high parallelizability is achieved from the particle swarm optimisation algorithm

9.1 FUTURE SCOPE

The formulation of this problem has been pertaining to a degree of uncertain variance and mean covariance with nominal risk exposure enforcing semi infinite constraints on partly infinite degree of the problem range

With all the perils that has been going on in the world, during the pandemic which has been set in, which has a death rate reaching unconscionable highs, protecting human lives and safeguarding interests has never been to this paramount degree. While we intricately formulate our models, we have to acknowledge that fact about what's going on outside the door of our rooms, in the outside world, were people are suffocating and are being denied a proper health care. Never has the world in recent times be in such a state of helplessness

During these times, if our model is able to successfully increase the survivability of even one individual, by giving throughputs of survival rates, it would have been an absolute honour to have worked on it.

We have also uploaded this documentation along side the source code to our github page, from where authorities, or any individual who wants to assess the survivability rates on their own dataset can do so with just a series of few clicks. They can pull our git repository directly to their local machines and import their dataset, and can readily get the survival rates without any hassle

The github repository link has been indexed in the appendix. In the end we would like to thank you for reading through this documentation and considering what we have achieved in the past months. It was with great enthusiasm that we have completed this project with and we hope that this helps in saving and safeguarding human lives.