ECOLOGICAL DATA ANALYSIS USING DEEP LEARNING APPROACH

Chandini Ravindranathan Nair
University Of Windsor, School of Computer Science

Abstract
Machine learning is a method of data analysis that automates analytical model building. Using algorithms that iteratively learn from data, machine learning allows computers to find hidden insights without being explicitly programmed where to look. The iterative aspect of machine learning is important because as models are exposed to new data, they are able to independently adapt. They learn from previous computations to produce reliable, repeatable decisions and results.

Deep Learning is a set of Machine Learning algorithms which have one or more hidden layers in Neural Networks. Now Deep Learning systems are used for almost any tasks other Machine Learning algorithms are used for.

I. INTRODUCTION

In individual-based predator-prey model, each agent behaviour is being modelled by a Fuzzy Cognitive Map (FCM), allowing the evolution of the agent behaviour through the epochs of the simulation. The FCM enables the agent to evaluate its environment (e.g., distance to predator/prey, distance to potential breeding partner, distance to food, energy level), its internal state (e.g., fear, hunger, curiosity) with memory and choosing several possible actions such as evasion, eating or breeding. This is the only model that links between behaviour patterns and speciation. The simulation produces a lot of data including - the number of individuals, level of energy by individual, choice of action, age of the individuals, number of species etc. This study mainly investigates emergence of species in a simulated ecosystem and proposes a general framework for the study of invasive species and species diversity patterns [1].

Fig 1: Predator-Prey Example

Here there are two data sets- the training set and the test set, both corresponding to the data generated from several runs of the Individual based predator prey model simulation. There are three classes: Default, FR (Fluctuating Resources), LF (Low Food). Based on the different features that govern the behaviour of an individual, they can be classified into one of the three classes.

This paper provides a state-of-the-art report for ecological data analysis using deep learning approach. In addition to giving a comprehensive coverage of current deep learning methods used for data analysis, this report can also act as a tutorial that is easy for a novice to follow. This report also provides a vision for experienced readers (especially biologists) who are interested in pursuing further research on the data based on the results obtained in the project.

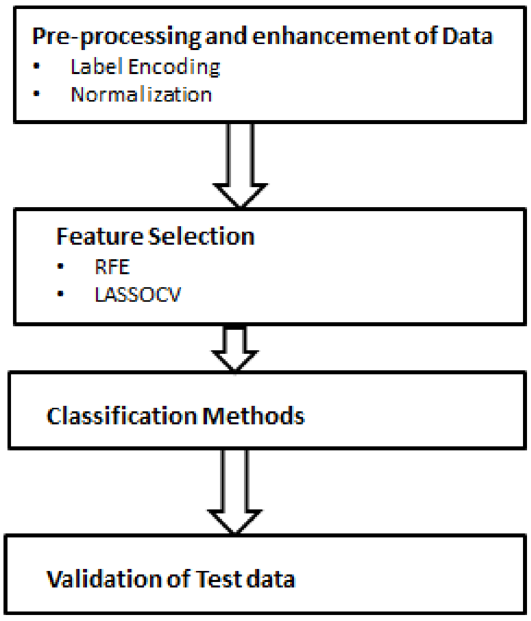
To achieve these goals, the paper is organized as follows. First the classification problem is well defined, followed by the different methodologies used to create the models to do the classification. Once the models do the classification of the test data the results are analysed to tell about the performance of the classifier and also to comment on the given data set. The report concludes by summarizing the results obtained by the different methodologies and what improvements can be done to get better results(if any).

II. PROBLEM DEFINTION AND DESCRIPTION OF DATA

The created model needs to be trained using any deep learning approach using the given training data set and then need to evaluate the performance of the classifier using the test data set. Here it is a classification problem which requires classifying the test data correctly into one of the three classes: Default, FR (Fluctuating Resource) and LF (Low Food). The data set consists of 171 different features which govern the behaviour of an individual, and based on these predictors the classification task needs to be done. Different deep learning approaches to classification are used to find the one that performs best on the given data set. Another aim is to identify which among the 171 features are the best contributing features for improved accuracy. On the whole the project aims to study the given data set and generate models that can best predict the classes for the given test set. The project also investigate the behaviour of the classifier and how well it predicts the classes for the test data.

III. METHEDOLOGY

A. Data Analysis Flow



B. Visualization of the data

There are two given data sets – the training set and the test set. The data comprises of 171 different features that govern the behavior of the individuals, and based on which classification is done to either of the three classes : **Default, LF(Low Food) and FR(Fluctuating Resources).**

For visualization purpose, we shall first try to make a scatter plot of the training data, by doing dimensionality reduction using **TSNE and PCA**.

We obtain the below results:

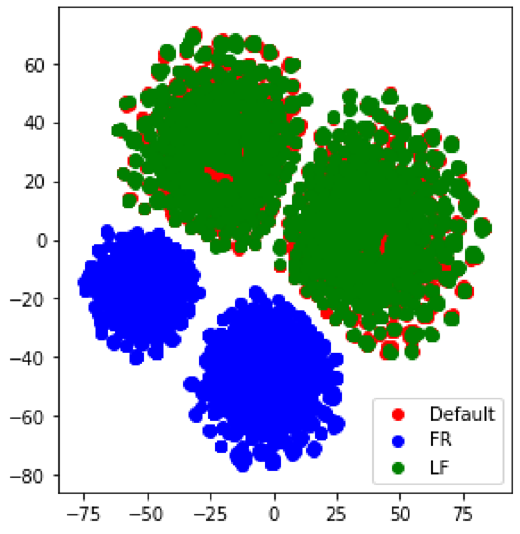


Fig 2: Scatter plot after dim reduction using TSNE

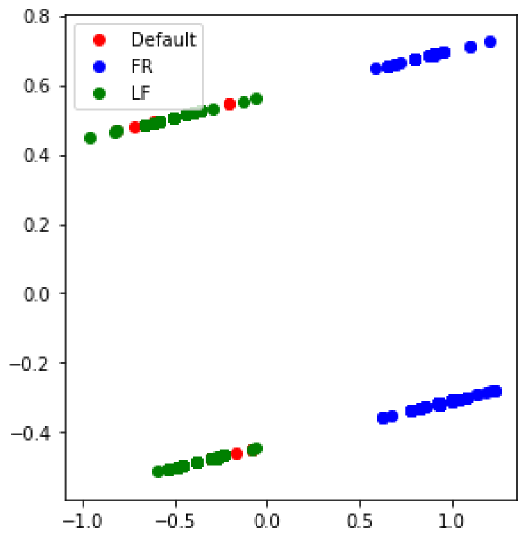


Fig 3: Scatter plot after dim reduction using PCA

From the above scatter plot it can be observed that the “Default” class and “LF” class data are difficult to separate by a linear separator. And also the data points of Default and LF classes seems to be so cluttered that, from a biological perspective we may say that the features of “Default” and “Low Food” are almost the same.

The data points corresponding to “FR” class are well separated from the other two classes.

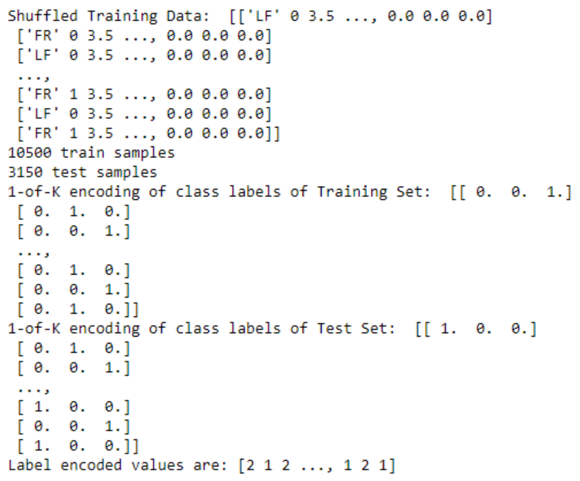
Hence it can be confirmed, that a non-linear classifier needs to be used for this classification problem.

C. Pre-processing and Enhancement of Data

Initially, the data set in CSV format was read using the pandas.read_csv function and stored to a data frame. Then the training data was shuffled to avoid any element of bias/patterns in the split datasets before training the ML model. This improves the ML model quality and also improves the predictive performance.

Normalization of data was also done to make the values of each feature in the data have zero- mean (when subtracting the mean in the numerator) and unit-variance.

Label encoding is an approach to encoding categorical values which is, simply converting each value in a column to a number. Our problem has 3 classes: Default, FR (fluctuating resources) and LF (Low Food), where Default is encoded as 0, FR as 1 and LF as 2.



```
Shuffled Training Data:  [['LF' 0 3.5 ..., 0.0 0.0 0.0]
 ['FR' 0 3.5 ..., 0.0 0.0 0.0]
 ['LF' 0 3.5 ..., 0.0 0.0 0.0]
 ...,
 ['FR' 1 3.5 ..., 0.0 0.0 0.0]
 ['LF' 0 3.5 ..., 0.0 0.0 0.0]
 ['FR' 1 3.5 ..., 0.0 0.0 0.0]]
10500 train samples
3150 test samples
1-of-K encoding of class labels of Training Set:  [[ 0.  0.  1.]
 [ 0.  1.  0.]
 [ 0.  0.  1.]
 ...,
 [ 0.  1.  0.]
 [ 0.  0.  1.]
 [ 0.  1.  0.]]
1-of-K encoding of class labels of Test Set:  [[ 1.  0.  0.]
 [ 0.  1.  0.]
 [ 0.  0.  1.]
 ...,
 [ 1.  0.  0.]
 [ 0.  0.  1.]
 [ 1.  0.  0.]]
Label encoded values are: [2 1 2 ..., 1 2 1]
```

Fig 4: Pre processing data results

D. Neural Networks

Neural networks were considered as the best approach to start off with the data analysis.

Neural networks require less formal statistical training, ability to implicitly detect complex nonlinear relationships between dependent and independent variables, ability to detect all possible interactions between predictor variables, and the availability of multiple training algorithms.

After learning from the initial inputs and their relationships, it can infer unseen relationships on unseen data as well, thus making the model generalize and predict on unseen data.

Unlike many other prediction techniques, Neural Networks does not impose any restrictions on the input variables (like how they should be distributed).

For the initial model setup, the hyper parameters where set to the below values:

NB_EPOCH = 20

BATCH_SIZE = 128

OPTIMIZER = Adam()

N_HIDDEN = 128

NO. OF HIDDEN LAYERS = 2

The model was created and it was trained using the training set. The model was then evaluated using the test set and it gave an accuracy value of just **66.66%.**

RESULTS	
Training Set Accuracy	67.19%
Validation Set Accuracy	65.48%
Test Set Accuracy	66.66%

Fig 5: Results of using NN model

Keeping EPOCH = 20 and BATCH_SIZE= 128, the model was modified for different Optimizers, Validation splits, Hidden nodes, No. of Hidden Layers and Adding and Removing Dropout layers, to study the differences in accuracy in each case, to find the model with the best parameters.

Few observations are below:

Hyper parameters	**Test Accuracy of Model**

OPTIMIZER = SGD()
HIDDEN NODES =64
VALIDATION_SPLIT=0.2
No of hidden layers=2
	33.33%

OPTIMIZER = SGD()
HIDDEN NODES =128
VALIDATION_SPLIT=0.2
No of hidden layers=3
	33.33%

OPTIMIZER = Adam() HIDDEN NODES =128 VALIDATION_SPLIT=0.2 No.of Hidden layers =3 Adding drop out layers	66.66%

OPTIMIZER = Adam()
HIDDEN NODES =128
VALIDATION_SPLIT=0.4
No.of Hidden layers=4
	68.82%

Fig 6: Results of using NN model with different values of hyper parameters

However, in order to confirm the results and find the best hyper parameters the Grid Search method was used.

Grid Search

In Grid Search there are a set of models (which differ from each other in their parameter values, which lie on a grid). Train each of the models and evaluate it using cross-validation. Then select the one that performed best.

Hyper parameter optimization is a big part of deep learning. The reason is that neural networks are difficult to configure and there are a lot of parameters that need to be set. On top of that, individual models can be very slow to train.

Grid search can be used to tune various hyper parameters like:

Batch size and training epochs

Optimization algorithms

Learning rate and momentum

Network weight initialization

Activation functions

Dropout regularization

The number of neurons in the hidden layer

Tuning Batch Size and Number of Epochs

Applying grid search using different batch sizes : [20,60,80,128] and different epochs : [20,50,100], below are the results obtained:

Batch Size	**Number of Epochs**	**Accuracy**
20	20	66.75%
20	50	67.14%
20	100	67.18%
60	20	66.30%
60	50	67.15%
60	100	67.13%
80	20	66.69%
80	50	66.97%
80	100	66.82%
128	20	66.66%
128	50	67.34%
128	100	67.11%

Fig 7: Results of applying grid search using different batch sizes and epochs

It can be seen that a batch_size of 128 and epochs of 50 gives the best results.

Tuning the Training Optimization Algorithm

Similarly using grid search on different Optimization Algorithms : [Adam, SGD, RMSprop, Adagrad, Adadelta, Adamax, Nadam], we obtain the below results:

Optimizer	**Accuracy**
SGD	33.21%
RMSprop	67.04%
Adagrad	66.40%
Adadelta	66.80%
Adam	67.62%
Adamax	67.25%
Nadam	33.33%

Fig 8: Results of applying grid search using different Optimization Algorithms

It can be seen that the 'Adam' Optimizer gives the best results.

Now, using these best hyper parameters obtained from Grid Search a new model was created with **4 hidden layers**. The below results were obtained:

RESULTS	
Training Set Accuracy	68.11%
Validation Set Accuracy	67.36%
Test Set Accuracy	68.15%

Fig 8: Results of using NN with best hyper-parameters from grid search

We obtain an increase in accuracy compared to the initial model. Now the **test accuracy is 68.15%.**

Graphs of Model Accuracy and Model Loss versus Epoch

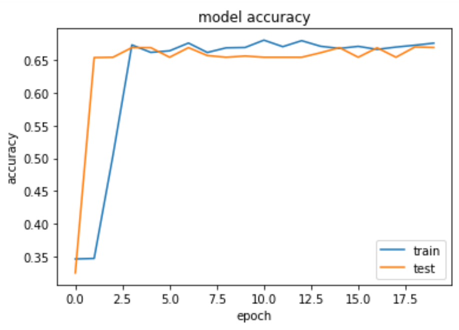


Fig 9: Accuracy vs Epoch

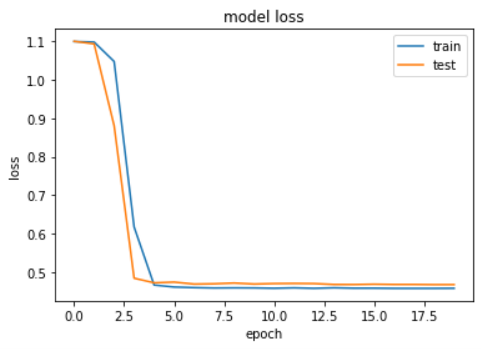


Fig 10: Loss vs Epoch

The results of the graph are quite as expected. The model accuracy increases as the epoch increases and the model loss decreases as the epoch decreases.

Analysis of the results obtained using neural network

As seen in Section A (Data visualization), this low test accuracy can be related to the given data set, where the data corresponding to Default and LF are almost overlapping each other.

Further analysis on this can be done, by printing the confusion matrix and plotting the ROC curves, and PR (Precision Recall) graph.

Confusion Matrix

		Predicted Class		
Actual Class		**Default**	**FR**	**LF**
	Default	147	0	903
	FR	0	1050	0
	LF	96	0	954

Fig 11: Confusion Matrix

From the above confusion matrix, it can be observed that the "Default" class has been many times wrongly predicted as "FR" class by the model. From the scatter plot for data visualization it was observed that the attribute values of class "Default" and class "LF" where almost same. Hence the low accuracy of the model can be related to this, i.e. the Default class being most of the time wrongly predicted as class "LF".

Also it can be seen that, the model is able to correctly predict class "FR" all the time in this particular run, and almost 100% correctly in several other runs.

ROC Curve

Receiver Operating Characteristic curve (or ROC curve.) It is a plot of the true positive rate against the false positive rate for the different possible cut points.

ROC curves are typically used in binary classification to study the output of a classifier. In order to extend ROC curve and ROC area to multi-class or multi-label classification, it is necessary to binarize the output.

Accuracy is measured by the area under the ROC curve.

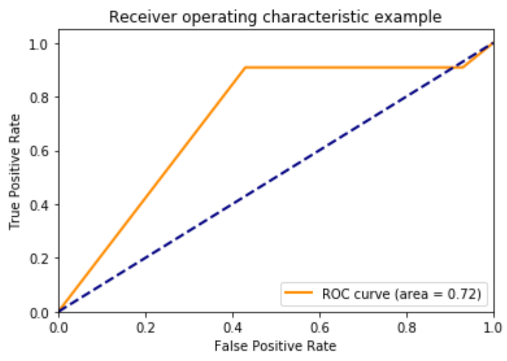


Fig 12: ROC curve

The area under is curve is 0.72 and is justifiable as from the confusion matrix it is clear that only about 2/3rd of data was correctly predicted.

Precision Recall Graph

Precision-Recall is a useful measure of success of prediction when the classes are very imbalanced.

Precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned.

The precision-recall curve shows the trade off between precision and recall for different threshold. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate.

High scores for both show that the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall).

A system with high recall but low precision returns many results, but most of its predicted labels are incorrect when compared to the training labels. A system with high precision but low recall is just the opposite, returning very few results, but most of its predicted labels are correct when compared to the training labels. An ideal system with high precision and high recall will return many results, with all results labeled correctly.

```
Average precision score, micro-averaged over all classes: 0.50
```

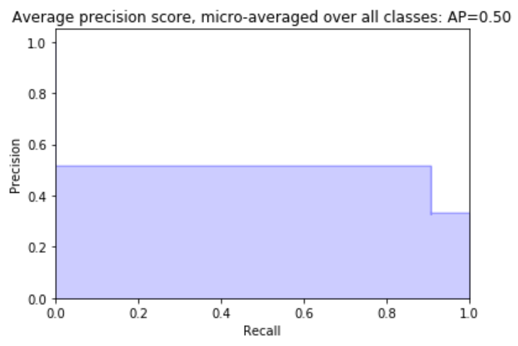


Fig 13: Precision Recall Graph

From the confusion matrix it is clear that many of the values where wrongly predicted. Hence a low average precision score is expected.

E. Feature Selection

Feature selection is also called variable selection/attribute selection.

It is the automatic selection of attributes in the data (such as columns in tabular data) that are most relevant to the predictive modeling problem under work.

Feature Selection Algorithms:

Filter Methods

Examples: Chi squared test, information gain and correlation coefficient scores.

Wrapper Methods

Examples: recursive feature elimination algorithm.

Embedded Methods

Examples: LASSO, Elastic Net and Ridge Regression

In this project, **LASSOCV** and **RFE** algorithms were used for feature selection.

LASSOCV selected 19 best features among the 171 features in the data set. Using RFE algorithm the rank of the features can be printed. The features with rank 1 are the best features.

LASSOCV

The LASSOCV algorithm selects 19 best features and when training the model using these best features , we are getting an test accuracy of 67.1%.

Number of features selected : 19

RESULTS	
Training Set Accuracy	67.27%
Validation Set Accuracy	67.02%
Test Set Accuracy	67.11%

Fig 14: Results after using LASSOCV

RFE

Using RFE algorithm, we are able to print the feature ranking and the selected features. RFE also returned test accuracy of about 66.66%

The features selected using RFE are the below:

RANK	**FEATURE**
1	Gender
1	Fear->Escape
1	Hunger->Escape
1	Curiosity->Socialize
1	Fear->Exploration
1	Search Partner->Exploration
1	Nuisance->Exploration
1	Satisfaction->Wait
1	Hunger->Eat
1	Sedentary->Eat
1	Satisfaction->Eat
1	Fear->Reproduce
1	SearchPartner->Reproduce
1	Curiosity->Reproduce
1	Sedentary->Reproduce
1	Move2StrongestPrey->Reproduce
1	SearchPartner>Move2StrongestPrey
1	Satisfaction->Move2WeakestPreyCell
1	Eat->Move2WeakestPreyCell

Fig 15: Results after using RFE

F. Convoluted Neural Networks

Convolutional Neural Networks are very similar to ordinary Neural Networks: they are made up of neurons that have learnable weights and biases.

There are four main operations in the Convolutional Neural Networks:

- Convolution
- Non Linearity (ReLU)
- Pooling or Sub Sampling
- Classification (Fully Connected Layer)

CNN's are mostly used in case of images which makes use of local patterns of the data. Here since the data has no local dependence, a low accuracy is expected.

It can be observed that the data set performs relatively poorly with the convolutional neural networks. The overall accuracy is as low as **33.33%.**

Let's take a look at the results.

RESULTS	
Training Set Accuracy	34.05%
Validation Set Accuracy	32.43%
Test Set Accuracy	33.33%

Fig 16: Results of using CNN model

The performance of the model was also very bad as the computations were very slow compared to normal NN.

As you may see, the accuracy is very low. This is because the features are independent of each other. Here the data has no local dependence, which would make no sense in case of the CNN's and give poor results. Therefore, it wouldn't perform well on the given data set.

Hence we can conclude CNN's are not a good option for training this data set.

G. Auto Encoder

Auto encoder is a simple 3-layer neural network where output units are directly connected back to input units. E.g. in a network like this:

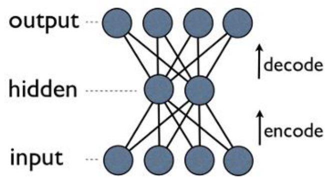


Fig 17: Network of Auto encoder

Output[i] has edge back to Input[i] for every i. Typically, number of hidden units is much less then number of visible (input/output) ones. As a result, when you pass data through such a network, it first compresses (encodes) input vector to "fit" in a smaller representation, and then tries to reconstruct (decode) it back. The task of training is to minimize an error or reconstruction, i.e. find the most efficient compact representation (encoding) for input data.

An accuracy of only **65.87%** is obtained after reconstructing input data.

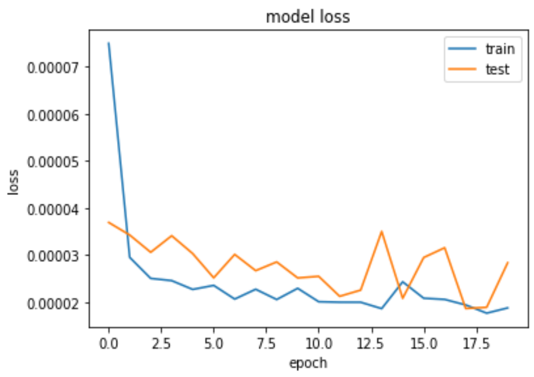


Fig 18: Loss vs Epoch

The reconstruction error on the training data seems to converge while that of test data has many peaks and valleys.

IV. DISCUSSION

From the data visualization it was observed that the feature values corresponding to data from class “Default” and class “LF” are mostly similar and hence the scatter plot shows that the data points of these two classes seems to be cluttered. This information can be of great interest to the biologists as they may find a strong similarity between these two classes. Within the scope of this project, the test data was classified by classifiers trained using the given training data set.

Convoluted neural networks were a bad idea for this data set as expected as the given data features were independent of each other and had no local dependence which are present in case of images.

It was seen that the given data set performs better with Neural Networks with more hidden layers (we have used 4), however the accuracy doesn’t go beyond about 68%. The classifier was not able to predict properly the “Default” class data. As we saw from printing the confusion matrix , most of the time the “Default” class data was being predicted as “FR” class.

The feature values are not good enough to make exact predictions.

However, from confusion matrix gives an idea that, more than half of the data set is being predicted correctly.

Hence including better features or adding on better data (if any) which can distinctly identify between the three classes, would produce improved results.

V. CONCLUSION

From the results obtained by using the deep learning approaches for classifying the given data set, it is clear that the maximum accuracy that be achieved on the data set is not more than 69%. For investigative purpose, the SVM classifier was used on the same data set and it also gave accuracy of 67.02% for linear kernel and an accuracy of 65.87% when kernel ='rbf'. Hence it can be concluded that new features might need to be added to the given data set that can actually distinctly classify between the "Default" class data and "Low Food" class data, as mentioned in the discussion.

From the results obtained above, it can be confirmed that neural networks with more hidden layers were at least able to classify 2/3rd of the given test data set correctly, even though the given data set was not having enough features to perfectly separate the data points of the three classes.

VI. FUTURE ENHANCEMENTS

Collect more unique data for each of the three classes corresponding to the runs of the Individual based Predator Prey simulations and train the models to see if accuracy is improved.

Apart from deep learning techniques and SVM, other classification techniques like Random Forest, Naive Bayes can be tried on the given data set to check if they would provide better results.

Based on the work of this project, biologists can try to find if really the two classes "Default" and "Low Food" have all similar characteristics or what actually differentiates between the two. This might contribute while studying the individual behavior in the Individual based predator prey model.

REFERENCES

[1]https://www.researchgate.net/publication/26234726_An_Individual-Based_Evolving_Predator-

Prey_Ecosystem_Simulation_Using_a_Fuzzy_Cognitive_Map_as_the_Behavior_Model

[2] https://gogul09.github.io/software/first-neural-network-keras

[3] https://machinelearningmastery.com/multi-class-classification-tutorial-keras-deep-learning-library/

[4] https://machinelearningmastery.com/feature-selection-machine-learning-python/

[5] https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/

[6] http://cs231n.github.io/convolutional-networks/

[7] https://medium.com/@curiously/credit-card-fraud-detection-using-autoencoders-in-keras-tensorflow-for- hackers-part-vii-20e0c85301bd

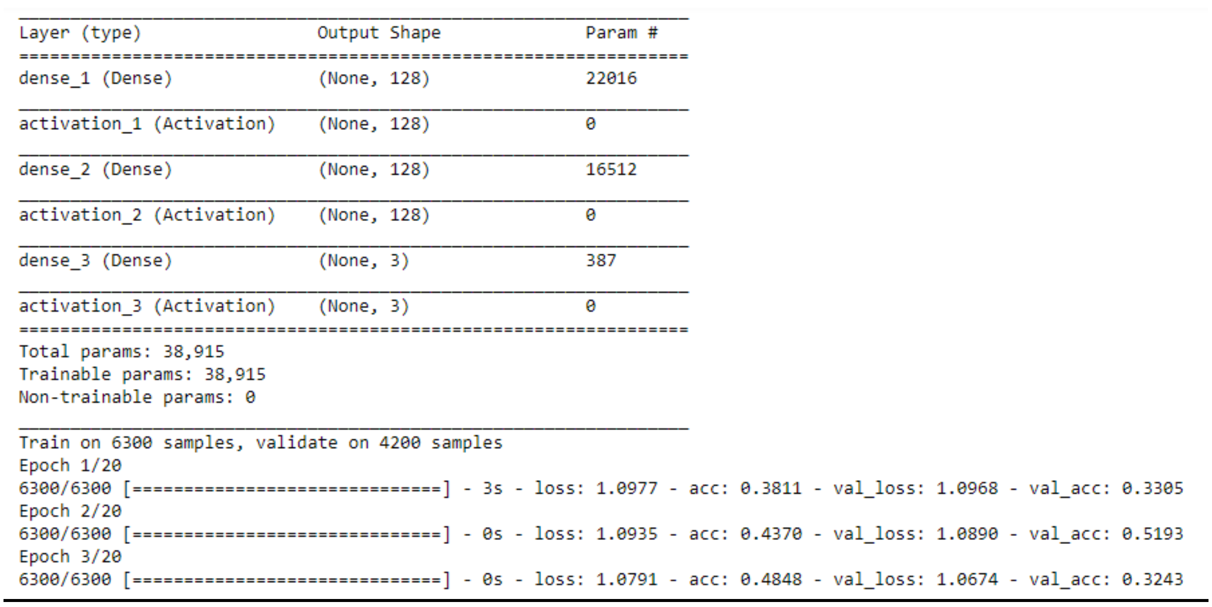
[8] https://machinelearningmastery.com/an-introduction-to-feature-selection/

[9] Google, Lecture Slides (Dr. R. Gras)

APPENDIX

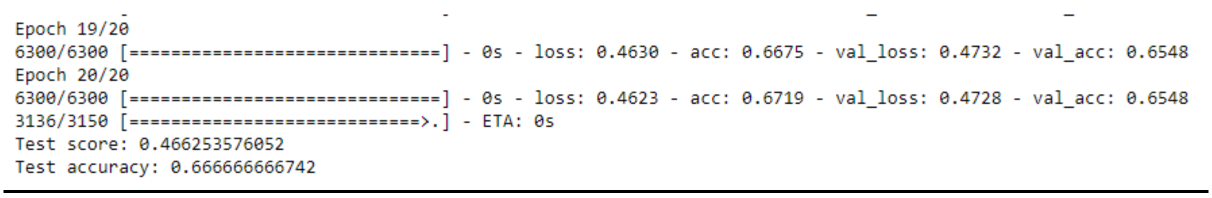
SCREENSHOTS

1. Results of using the First Neural Network model with hyper parameters chosen before grid search



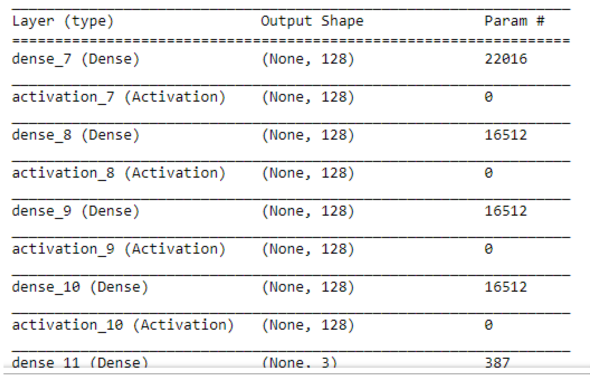
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_1 (Dense)              (None, 128)               22016     
_________________________________________________________________
activation_1 (Activation)    (None, 128)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 128)               16512     
_________________________________________________________________
activation_2 (Activation)    (None, 128)               0         
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 387       
_________________________________________________________________
activation_3 (Activation)    (None, 3)                 0         
=================================================================
Total params: 38,915
Trainable params: 38,915
Non-trainable params: 0
_________________________________________________________________
Train on 6300 samples, validate on 4200 samples
Epoch 1/20
6300/6300 [==============================] - 3s - loss: 1.0977 - acc: 0.3811 - val_loss: 1.0968 - val_acc: 0.3305
Epoch 2/20
6300/6300 [==============================] - 0s - loss: 1.0935 - acc: 0.4370 - val_loss: 1.0890 - val_acc: 0.5193
Epoch 3/20
6300/6300 [==============================] - 0s - loss: 1.0791 - acc: 0.4848 - val_loss: 1.0674 - val_acc: 0.3243
```

…….



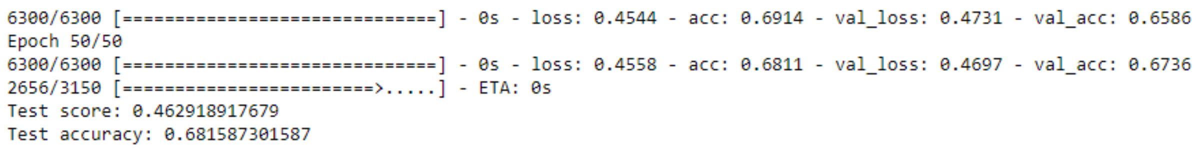
```
Epoch 19/20
6300/6300 [==============================] - 0s - loss: 0.4630 - acc: 0.6675 - val_loss: 0.4732 - val_acc: 0.6548
Epoch 20/20
6300/6300 [==============================] - 0s - loss: 0.4623 - acc: 0.6719 - val_loss: 0.4728 - val_acc: 0.6548
3136/3150 [============================>.] - ETA: 0s
Test score: 0.466253576052
Test accuracy: 0.666666666742
```

2. Results of using the Neural Network Model with hyper parameters chosen by grid search



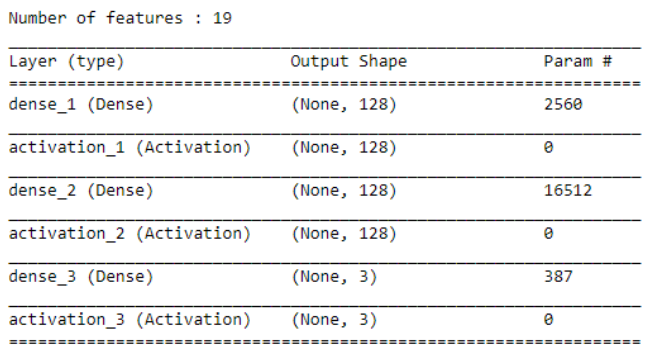
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_7 (Dense)              (None, 128)               22016     
_________________________________________________________________
activation_7 (Activation)    (None, 128)               0         
_________________________________________________________________
dense_8 (Dense)              (None, 128)               16512     
_________________________________________________________________
activation_8 (Activation)    (None, 128)               0         
_________________________________________________________________
dense_9 (Dense)              (None, 128)               16512     
_________________________________________________________________
activation_9 (Activation)    (None, 128)               0         
_________________________________________________________________
dense_10 (Dense)             (None, 128)               16512     
_________________________________________________________________
activation_10 (Activation)   (None, 128)               0         
_________________________________________________________________
dense_11 (Dense)             (None, 3)                 387       
```

…….

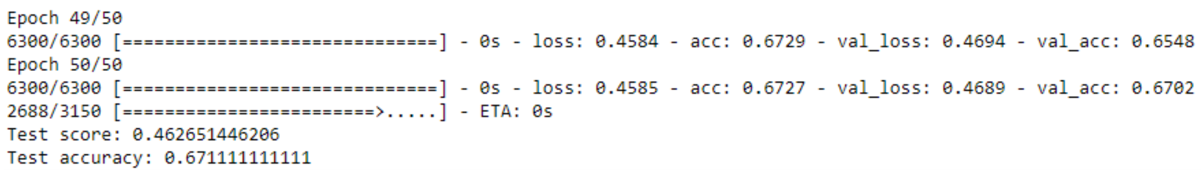


```
6300/6300 [==============================] - 0s - loss: 0.4544 - acc: 0.6914 - val_loss: 0.4731 - val_acc: 0.6586
Epoch 50/50
6300/6300 [==============================] - 0s - loss: 0.4558 - acc: 0.6811 - val_loss: 0.4697 - val_acc: 0.6736
2656/3150 [========================>.....] - ETA: 0s
Test score: 0.462918917679
Test accuracy: 0.681587301587
```

3. Results obtained after doing feature selection

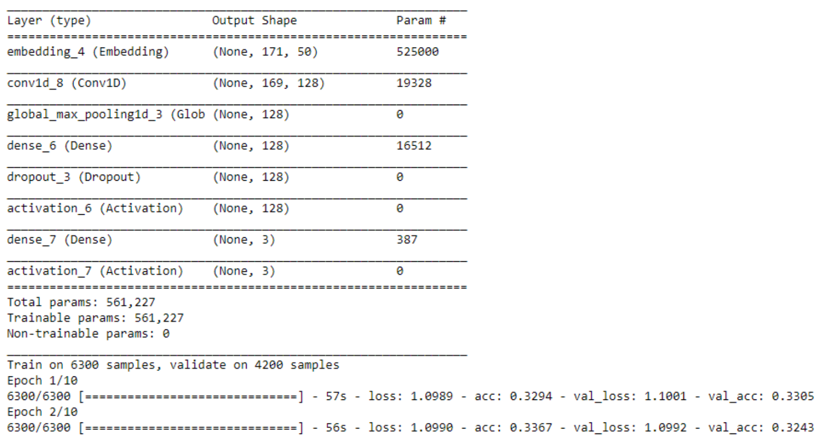


```
Number of features : 19
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               2560
_________________________________________________________________
activation_1 (Activation)    (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               16512
_________________________________________________________________
activation_2 (Activation)    (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 387
_________________________________________________________________
activation_3 (Activation)    (None, 3)                 0
=================================================================
```

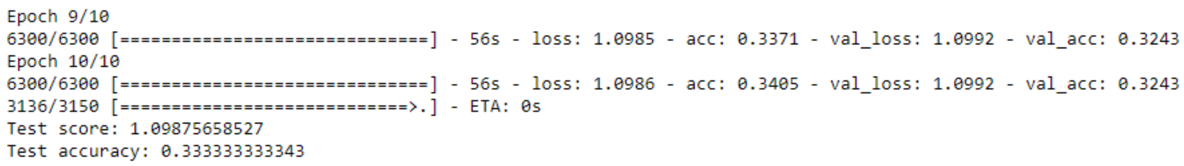


```
Epoch 49/50
6300/6300 [==============================] - 0s - loss: 0.4584 - acc: 0.6729 - val_loss: 0.4694 - val_acc: 0.6548
Epoch 50/50
6300/6300 [==============================] - 0s - loss: 0.4585 - acc: 0.6727 - val_loss: 0.4689 - val_acc: 0.6702
2688/3150 [========================>.....] - ETA: 0s
Test score: 0.462651446206
Test accuracy: 0.671111111111
```

4. Results obtained by using the CNN model



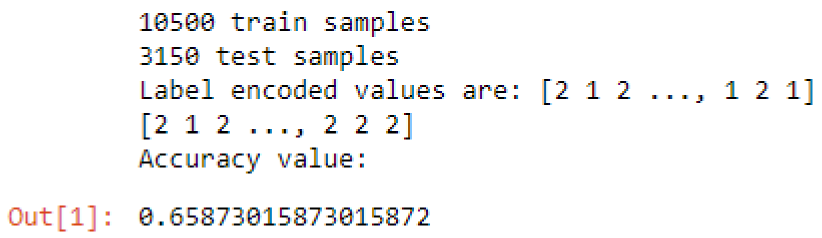
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_4 (Embedding)      (None, 171, 50)           525000
_________________________________________________________________
conv1d_8 (Conv1D)            (None, 169, 128)          19328
_________________________________________________________________
global_max_pooling1d_3 (Glob (None, 128)               0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               16512
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
activation_6 (Activation)    (None, 128)               0
_________________________________________________________________
dense_7 (Dense)              (None, 3)                 387
_________________________________________________________________
activation_7 (Activation)    (None, 3)                 0
=================================================================
Total params: 561,227
Trainable params: 561,227
Non-trainable params: 0
_________________________________________________________________
Train on 6300 samples, validate on 4200 samples
Epoch 1/10
6300/6300 [==============================] - 57s - loss: 1.0989 - acc: 0.3294 - val_loss: 1.1001 - val_acc: 0.3305
Epoch 2/10
6300/6300 [==============================] - 56s - loss: 1.0990 - acc: 0.3367 - val_loss: 1.0992 - val_acc: 0.3243
```



```
Epoch 9/10
6300/6300 [==============================] - 56s - loss: 1.0985 - acc: 0.3371 - val_loss: 1.0992 - val_acc: 0.3243
Epoch 10/10
6300/6300 [==============================] - 56s - loss: 1.0986 - acc: 0.3405 - val_loss: 1.0992 - val_acc: 0.3243
3136/3150 [============================>.] - ETA: 0s
Test score: 1.09875658527
Test accuracy: 0.333333333343
```

5. Results obtained by using SVM model

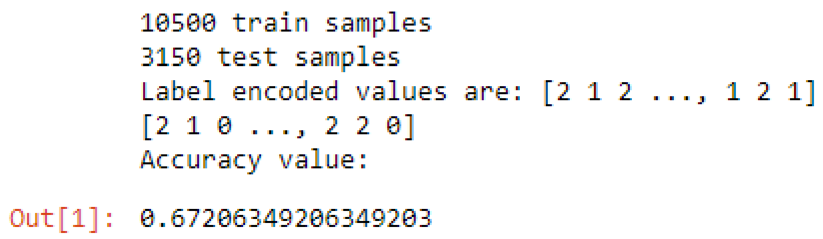
When kernel ='rbf'



```
         10500 train samples
         3150 test samples
         Label encoded values are: [2 1 2 ..., 1 2 1]
         [2 1 2 ..., 2 2 2]
         Accuracy value:

Out[1]:  0.65873015873015872
```

When kernel = ‘linear’



```
           10500 train samples
           3150 test samples
           Label encoded values are: [2 1 2 ..., 1 2 1]
           [2 1 0 ..., 2 2 0]
           Accuracy value:

  Out[1]:  0.67206349206349203
```