PROJECT REPORT

COMP 8610 - W2020

Prediction of Bankruptcy Using Artificial Neural Networks

Group Members:

Adarsh Sai Gupta: 105181433

Anshul Sharma: 110016388

Rishav Chatterjee: 110010348

School of Computer Science

Faculty of Science

University of Windsor

Abstract

Deep learning models use layers of artificial neural networks to extract features from data for prediction. One such prediction is the prediction of bankruptcy. Prediction of bankruptcy will help us to minimize its impact on financial stability if known in advance. Since, bankruptcy prediction is related to problem-solving knowledge and statistical calculations based on textual data, hence, we use a combined approach of three different deep learning models to further improve the prediction accuracy that is: Multi-layer Perceptron Model (MLP), Radial Basis Function (RBF) Network & Dense Neural Networks (DNN). A dataset based on bank customer attributes has been utilised for analysis and results from traditional models will considered as a benchmarking standard for comparison with the performance of the new prediction model. However, in this report we will consider some of their ideas as inspiration and construct our own bankruptcy prediction model in order to improve the previous results as much as possible.

Keywords: Bankruptcy Prediction, Multi-layer Perceptron, Radial Basis Function Network, Dense Neural Network, Deep Learning, Artificial Intelligence, Neural Networks

Table of Contents

List of Tables

List of Figures

Chapter 1.

1. Introduction

Bankruptcy is financial failure of a business and when an organization is not able to pay its debts. Impact of bankruptcy has shown disastrous outcomes and its affects whole community of people. Thus, an efficient and intelligent model is needed to clearly predict bankruptcy possibilities. These predictions refer to the business failure through financial and non-financial variables. Advancements in computation technique have ushered an era where Artificial Intelligence and Deep Learning Models form the backbone of bankruptcy prediction models. In the paper, (Wilson & Sharda, 1994), the authors showed a comparison of performance between Neural networks and multiple discriminant analysis for predicting bankruptcies, in which neural networks performed slightly better than the discriminant analysis. These results were useful for banking employees, neural networks researchers and for people in similar professional fields. However, use of more recent training models like artificial neural networks and multi-layer perceptron have proven to be more efficient over the conventional statistical algorithms for bankruptcy prediction. Considering the drastic growth in corporate firms, businesses & government organizations, accurately forecasting the financial risks involved would greatly provide useful information. Corporate bankruptcies can also be triggered by many additional factors such as incorrect investment decisions or a poor investment environment. Therefore, in this report an experimentation on a few different deep learning models is conducted to provide the best accuracy and performance for prediction of bankruptcy.

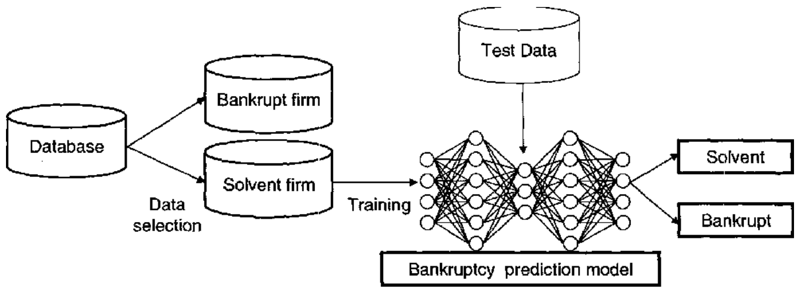


Figure 1: Example of the infrastructure of the model

In this report we demonstrate how we implemented Multi-layer Perceptron (MLP), Radial Basis Function Network (RBFN) & Dense Neural Network (DNN) to predict bankruptcy or more specifically all the bad investments made such as loans, bad credit score, that led to the situation of bankruptcy. In the paper, (Ravisankar et al., 2010), the authors used neural network genetic programming hybrids which comprises of multilayer feed forward neural network (MLFF) to predict the failure of dotcom companies. The authors used Wharton Research Data Services (WRDS) dataset and concluded that genetic programming yielded the most accurate results. The authors in (Lin et al., 2011) demonstrated on how to use hybrid manifold learning approach model to predict bankruptcy of firms. The model combined both isometric feature mapping and support vector machines (SVM). The hybrid approach produced the best result to predict financial crisis at an early stage. Thus, we conclude that compared to traditional statistical approaches, predicting bankruptcies or financial crisis using artificial neural networks leads to better accuracy although at the cost of performance. For smaller dataset, we have seen neural networks performs better compared to conventional statistical approaches, however, with larger samples of data, the complexity of the former approach gets increased and performance is hindered. The goal of this project is to predict an accurate estimation of bankruptcy while considering some technical and design limitations, while constructing the model which will be thoroughly discussed in the next chapter.

This part of the report is aimed towards a general outline of the project. The remaining part of the report is organized as follows,

(i) Chapter 2 provides a detailed discussion on the methodologies, implementation and results obtained from different tests regarding the prediction models used. All the results obtained are compared with each other for a better understanding of the performance and efficiency of the model.

(ii) Chapter 3 comprises conclusion, summary of the research & considerations on the report.

(iii) The final part of the report includes references.

Chapter 2.

2. Bankruptcy Prediction using Various Deep Learning Models

In this section we are going to discuss, the architecture of our model, its implementation and the results obtained. For our project we are using a dataset called “credit.csv” with 18 features and more than 90,000 samples. We are splitting the data set into two sets for training and testing in the ratio of 7:1 that is 70,000 samples for training and 10,000 for testing the accuracy our model. Some of the features of the dataset are ‘Loan ID’, ‘Loan Amount’, ‘Credit Score’ & ‘Annual Income’. Before we start with the implementation details, we briefly discuss the working principle of all the aforementioned models.

2.1. Prediction using Multi-layer Perceptron (MLP)

A perceptron is the simplest kind of neural network having only 1 hidden layer and produces single output based on several real-valued inputs by forming a linear combination using its input weights. A perceptron works in close resemblance to a linear classifier that is, it is an algorithm that classifies input by separating the two categories with a straight line. Inputs are a feature vector ‘x’ multiplied by weights ‘w’ and added to a bias ‘b’. Similarly, as the name suggests, a Multi-layer Perceptron (MLP) is a deep neural network that consists multiple perceptron (see Figure. 2). They are comprised of an input layer to receive the signal, an output layer that decides or predicts about the input. In between the input & output layer we place an arbitrary number of hidden layers that are the true computational engine of the MLP. MLPs with one hidden layer can approximate any continuous function and hence they are widely used in supervised learning problems where each input vector is associated with a label defining its class.

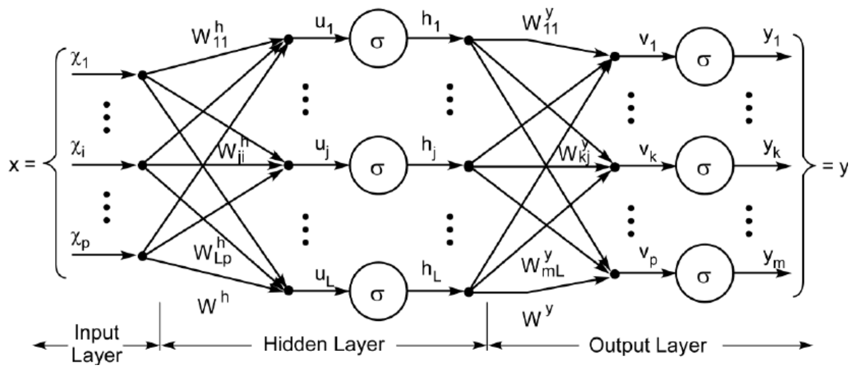


Figure 2: Example of a Multi-layer Perceptron

2.2. Prediction using Radial Basis Function Network (RBFN)

Usually, when addressing neural networks, we tend to refer to MLPs. As we discussed previously, each neuron in an MLP takes the weighted sum of its input values that is, every individual input value is multiplied by a coefficient, and the combined results are all added up together. A single MLP neuron is just a simple linear classifier, but complex non-linear classifiers can be built by combining these neurons into a network, which brings us to the idea of Radial Basis Function Networks.

Radial Basis Function Network (RBFN), has a much intuitive approach compared to MLPs. A RBFN performs the classification of data by measuring the input's resemblance to examples from the training set. Each RBFN neuron accumulates a "prototype", which is one of the examples from the training set. During classification of a new input, each neuron computes the Euclidean distance between the input and its prototype and produces a value between 0 and 1 which gives us a measure of similarity. If the matches the prototype, then the output of that RBFN neuron will be 1. Each and every RBFN neuron compares the input vector to its prototype. The neuron's response value is also known as its "activation" value. The input vector is the *d*-dimensional vector that we are trying to classify, where *d* is a non-zero integer. The entire input vector is made available to each of the RBFN neurons. The output of the network comprises a set of nodes, usually one per category that we are trying to classify. Then each output node computes a score for the associated category. In most cases, a classification decision is made by assigning the input to the category with the highest score.

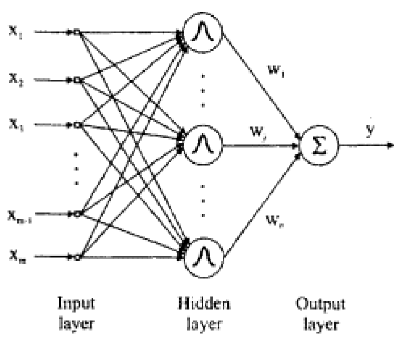


Figure 3: Example of Radial Basis Function Network

2.3 Prediction using Dense Neural Network (DNN)

Dense Neural Network, as the name suggests, are densely packed layers in a neural network model that are fully connected, to form a complete bipartite graph and hence termed as ‘dense’ (see Figure. 4). Each neuron in a layer receives an input from all the neurons present in the previous layer, therefore there are a lot of inputs received by each node.

As we know, deep learning models are built using neural networks that is a neural network takes inputs, which are then processed in hidden layers using weights that are adjusted during training phase. Then the model outputs a prediction. The weights are adjusted accordingly and autonomously to find patterns in the data set in order to make better predictions.

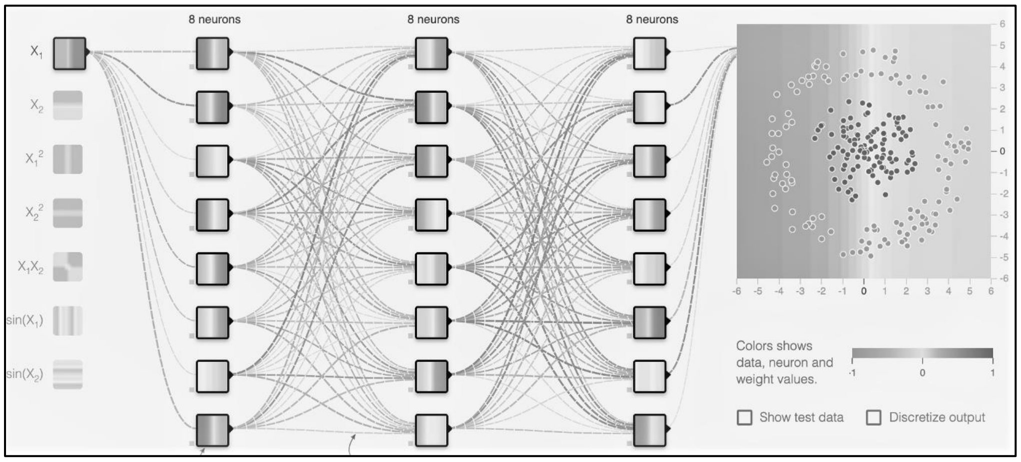


Figure 4: DNN model

To make the entire procedure work, we need to perform some preprocessing on data. Apart from data normalization, we also need to perform some feature column refactoring to make it compatible with our dense neural network classifier. To work with Dense Neural Networks, we use the Tensor Flow Estimator API. The optimized Tensor Flow Estimator API can also run on GPU providing faster learning.

2.4. Implementation & Results

A classifier uses backpropagation to enable a multi-layer perceptron to classify instances. The network is often built manually or be set up using a simple heuristic. The network parameters may also be monitored and modified during the training time. The nodes in this network are all sigmoid apart from for when the class is numeric, during which case the output nodes become linear units without any threshold.

In this report, we used WEKA Tool to generate the results of our model. WEKA is a collection of machine learning algorithms used for rendering tabloid outputs of a model. In WEKA, we select the option of MLP after performing data cleaning, i.e., eliminating some of the irrelevant attributes that does not affect the accuracy of the model and then we start training the model by tuning the parameters. We set learning rate to 0.3, the momentum value to 0.2, seed value to 0 and hidden layers as ‘a’, where a = (number of attributes + number of classes) / 2. It is better to keep a smaller learning rate value; hence we have chosen to keep it at 0.3. Thus, using the above parameter values, the model was constructed. We experimented with the training using both 10-Fold Cross-validation (CV) and the entire training set. The inference was that when we selected test option as, using the entire training data set, it gave us better results than 10Fold CV. So, first we train our input-output pairs which helps to find the dependencies or correlation between the inputs and outputs. This training involves adjusting the parameters or the weights and biases of the model in order to minimize the error. Then we use backpropagation technique to adjust the weights and biases relative to the error so as to minimize the error as much as possible. We achieved the following results form Multi-Layer Perceptron Algorithm:

Table 1: Summary of Results from Multi-layer Perceptron

Correctly Classified Instances	88.81%
Incorrectly Classified Instances	11.19 %
Kappa statistic	0.0522
Mean absolute error	0.0467
Root mean squared error	0.1862
Relative absolute error	70.5673 %
Root relative squared error	102.4389 %
Total Number of Instances	10000

Table 2: Summary of Detailed Accuracy by Class for MLP

	TP Rate	**FP Rate**	**Precision**	**Recall**	**F-Measure**	**ROC Area**	**Class**
	0.994	0.958	0.893	0.994	0.941	0.503	NO CHANCES
	0.035	0.007	0.375	0.035	0.064	0.499	CHANCES ARE QUARTER
	0	0	0	0	0	0.567	CHANCES ARE HALF
	0	0	0	0	0	0.736	CHANCES ARE 62.5%
	0	0	0	0	0	0.987	CHANCES ARE THREE QUARTERS
	0	0	0	0	0	0.999	SURE BANKRUPT
Weighted Avg.	0.888	0.853	0.833	0.888	0.844	0.504	

Table 3: Confusion Matrix Table of MLP

NO CHANCES	CHANCES ARE QUARTER	CHANCES ARE HALF	CHANCES ARE 62.5%	CHANCES ARE THREE QUARTERS	SURE BANKRUPT
8845	50	0	0	0	0
986	36	0	0	0	0
40	6	0	0	0	0
22	0	0	0	0	0
11	3	0	0	0	0
0	1	0	0	0	0

In order to understand the implementation of Multi-layer Perceptron model on our dataset, we take a look at Table 1 which summarizes the results obtained from Multi-layer Perceptron. The accuracy observed in the multi-layer perceptron is 88.81% for total 10,000 instances. The accuracy observed on each class is explained in Table 2. As, we can observe that ROC (Receiver operating characteristic) curve for sure bankrupt is the highest followed by 63% bankruptcy chances.

The training process of RBFN starts by selecting 3 sets of parameters: i) the prototypes & ii) beta co-efficient for each of the RBF neurons iii) the matrix of the output weights between RBF neurons and the output nodes. First, we need to select the prototype. We are using the entire training set to train our model. Logistic regression has been applied to K-means clusters as basis functions, that is, Logistic Regression with ridge parameter of 1.0E-8. The total time taken to build the model is 14.35 seconds. The results from Radial Basis Function Network Algorithm are as follows:

Table 4: Summary of Results of RBFN

Correctly Classified Instances	96.57 %
Incorrectly Classified Instances	3.43 %
Kappa statistic	0.8337
Mean absolute error	0.0178
Root mean squared error	0.0937
Relative absolute error	26.835 %
Root relative squared error	51.5412 %
Total Number of Instances	10000

Table 5: Summary of Detailed Accuracy by Class for RBFN

	TP Rate	**FP Rate**	**Precision**	**Recall**	**F-Measure**	**ROC Area**	**Class**
	0.0977	0.088	0.989	0.977	0.983	0.986	NO CHANCES
	0.932	0.026	0.802	0.932	0.862	0.99	CHANCES ARE QUARTER
	0.043	0	0.667	0.082	0.043	0.996	CHANCES ARE HALF
	0	0	0	0	0	0.974	CHANCES ARE 62.5%
	0.857	0.001	0.545	0.857	0.667	0.999	CHANCES ARE THREE QUARTERS
	0	0	0	0	0	0.999	SURE BANKRUPT
Weighted Avg.	0.966	0.081	0.965	0.966	0.964	0.987	

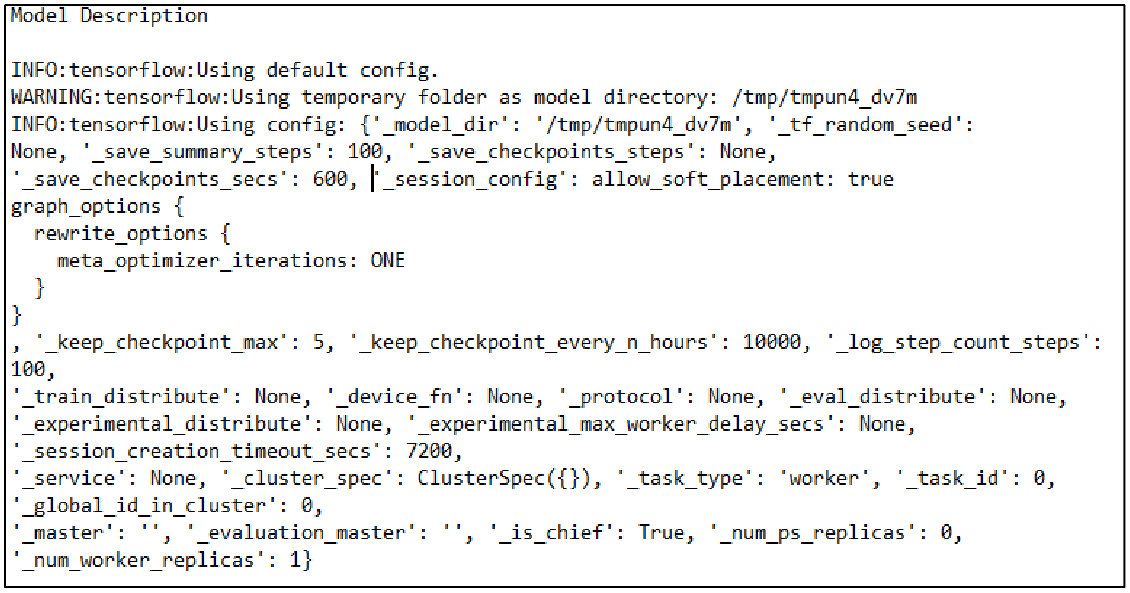
Table 6: Confusion Matrix Table RBFN

NO CHANCES	CHANCES ARE QUARTER	CHANCES ARE HALF	CHANCES ARE 62.5%	CHANCES ARE THREE QUARTERS	SURE BANKRUPT
8690	199	1	0	5	0
64	953	0	0	5	0
8	36	2	0	0	0
22	0	0	0	0	0
2	3	0	0	12	0
1	0	0	0	0	0

To summarize, the results obtained from or RBFN model, we got the highest accuracy of 96.57% for a total number of 10,000 instances. Table 2 illustrates the accuracy observed on each class. We can see that the FP (False-Positive) rate was quite low that is .081 whereas the ROC curve for sure bankrupt is the highest followed by 62.5% of bankruptcy chances.

For DNN, first, we import our data from 'csv' file that is our dataset using the Pandas library. Before we can pass the data to model, we need to structure and preprocess the data to remove the irrelevant data and to make the data consistent. Thus, we remove the column for customer id and loan id. Next, we create the required feature columns and the normalize our data so, that the values are logically related to each other. After creating the feature columns for some of the attributes of our bank dataset, we create two types of feature column, Numeric Column and Vocabulary List Categorical Column. The estimator API makes classification and structuring of data convenient for preprocessing. Finally, we set the input and result as x and y data set and then perform a train test split.

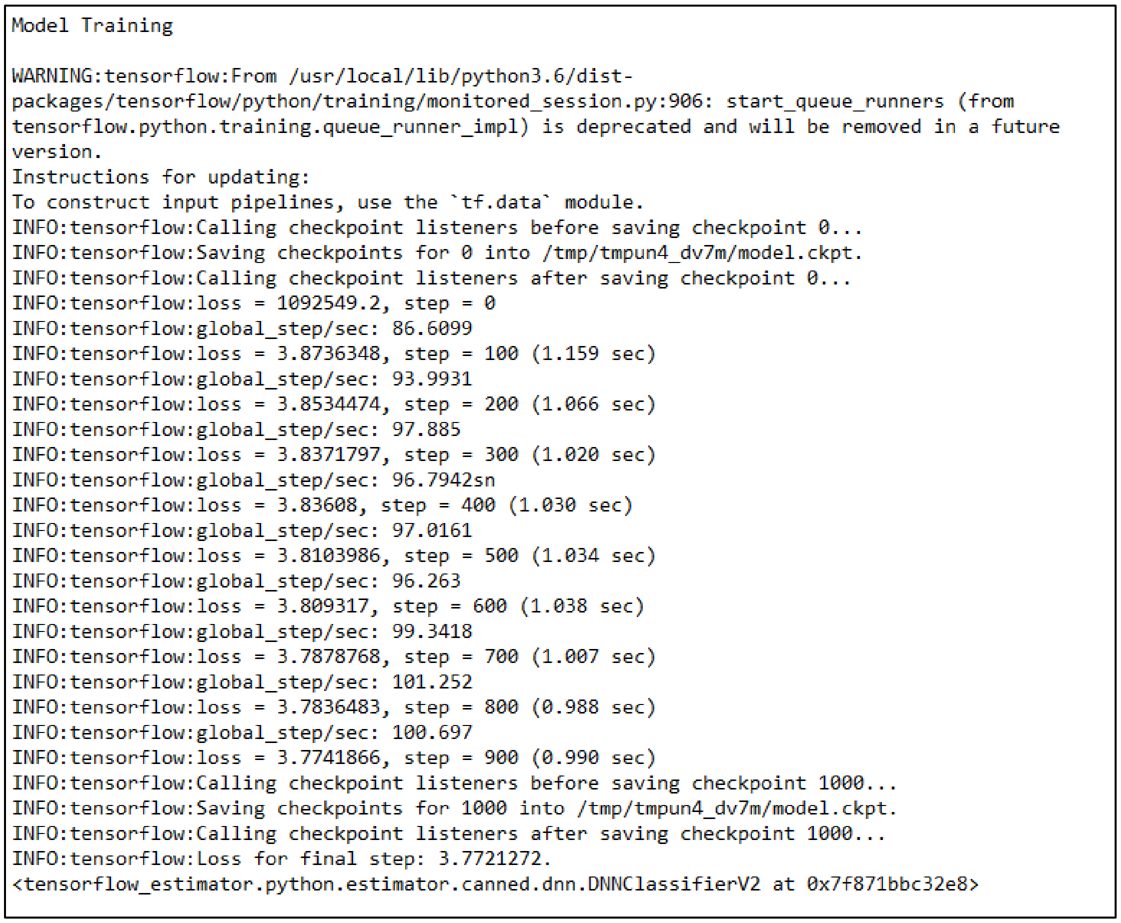
The data is split into training dataset and test dataset using the Scikit Learn train test split. The training data is given to the model that is generated using the Tensorflow estimator API. The properties are set to this model in the process of creating the model by passing them as parameters to the model description function. The model is optimized using the various possible combination for model properties. The properties such as batch size and activation function are configured to get optimized result. Trained model then is testing to check the prediction accuracy and model correctness. Prediction are made on data which is not previously shown to the model. This predicted data is then compared to actual results to check the relevancy of the model to a real scenario. The model using the Dense neural network has a prediction accuracy of 89%.



```
Model Description

INFO:tensorflow:Using default config.
WARNING:tensorflow:Using temporary folder as model directory: /tmp/tmpun4_dv7m
INFO:tensorflow:Using config: {'_model_dir': '/tmp/tmpun4_dv7m', '_tf_random_seed':
None, '_save_summary_steps': 100, '_save_checkpoints_steps': None,
'_save_checkpoints_secs': 600, '_session_config': allow_soft_placement: true
graph_options {
  rewrite_options {
    meta_optimizer_iterations: ONE
  }
}
, '_keep_checkpoint_max': 5, '_keep_checkpoint_every_n_hours': 10000, '_log_step_count_steps':
100,
'_train_distribute': None, '_device_fn': None, '_protocol': None, '_eval_distribute': None,
'_experimental_distribute': None, '_experimental_max_worker_delay_secs': None,
'_session_creation_timeout_secs': 7200,
'_service': None, '_cluster_spec': ClusterSpec({}), '_task_type': 'worker', '_task_id': 0,
'_global_id_in_cluster': 0,
'_master': '', '_evaluation_master': '', '_is_chief': True, '_num_ps_replicas': 0,
'_num_worker_replicas': 1}
```

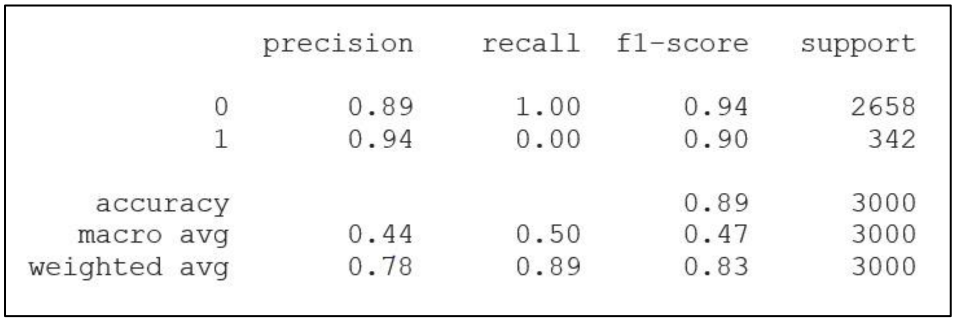
Figure 5: Model Description of DNN



```
Model Training

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow/python/training/monitored_session.py:906: start_queue_runners (from
tensorflow.python.training.queue_runner_impl) is deprecated and will be removed in a future
version.
Instructions for updating:
To construct input pipelines, use the `tf.data` module.
INFO:tensorflow:Calling checkpoint listeners before saving checkpoint 0...
INFO:tensorflow:Saving checkpoints for 0 into /tmp/tmpun4_dv7m/model.ckpt.
INFO:tensorflow:Calling checkpoint listeners after saving checkpoint 0...
INFO:tensorflow:loss = 1092549.2, step = 0
INFO:tensorflow:global_step/sec: 86.6099
INFO:tensorflow:loss = 3.8736348, step = 100 (1.159 sec)
INFO:tensorflow:global_step/sec: 93.9931
INFO:tensorflow:loss = 3.8534474, step = 200 (1.066 sec)
INFO:tensorflow:global_step/sec: 97.885
INFO:tensorflow:loss = 3.8371797, step = 300 (1.020 sec)
INFO:tensorflow:global_step/sec: 96.7942sn
INFO:tensorflow:loss = 3.83608, step = 400 (1.030 sec)
INFO:tensorflow:global_step/sec: 97.0161
INFO:tensorflow:loss = 3.8103986, step = 500 (1.034 sec)
INFO:tensorflow:global_step/sec: 96.263
INFO:tensorflow:loss = 3.809317, step = 600 (1.038 sec)
INFO:tensorflow:global_step/sec: 99.3418
INFO:tensorflow:loss = 3.7878768, step = 700 (1.007 sec)
INFO:tensorflow:global_step/sec: 101.252
INFO:tensorflow:loss = 3.7836483, step = 800 (0.988 sec)
INFO:tensorflow:global_step/sec: 100.697
INFO:tensorflow:loss = 3.7741866, step = 900 (0.990 sec)
INFO:tensorflow:Calling checkpoint listeners before saving checkpoint 1000...
INFO:tensorflow:Saving checkpoints for 1000 into /tmp/tmpun4_dv7m/model.ckpt.
INFO:tensorflow:Calling checkpoint listeners after saving checkpoint 1000...
INFO:tensorflow:Loss for final step: 3.7721272.
<tensorflow_estimator.python.estimator.canned.dnn.DNNClassifierV2 at 0x7f871bbc32e8>
```

Figure 6: Model Training of DNN


```
              precision    recall  f1-score   support

           0       0.89      1.00      0.94      2658
           1       0.94      0.00      0.90       342

    accuracy                           0.89      3000
   macro avg       0.44      0.50      0.47      3000
weighted avg       0.78      0.89      0.83      3000
```

Figure 7 : Summary of Results of DNN model

As we can see from Figure. 5, we have the model description which saves the checkpoint after every 600 seconds. In the model training (see Figure. 6), we are implementing 1000 steps, in which every hundred steps take 1 second approximately. We got the highest prediction accuracy for RBFN model that is a percentage of 96.57, followed by DNN model which had an accuracy of 89% and lastly, we got 88.81% accuracy for MLP model.

Chapter 3.

3. Conclusion & Future Research

3.1 Concluding Comments & Summary

To conclude our discussion, we observed that the results obtained form radial basis function network are the most accurate in comparison to the other two models i.e. MLP and DNN. The accuracy observed in this model was by far the most, that is 96.57 %, while the other two models DNN and MLP performed equally good with a similar result of 88.81% and 89% accuracy respectively. Overall, the models where able to predict the probability or chances of whether a person who is applying for loan, will or will not be able to payback the amount borrowed and would that adversely lead to a bankruptcy in the future. RBF networks are very popular for function approximation, curve fitting, time series prediction, control and classification problems because it uses a radial basis function as its activation function thus making it a special type of neural network. The features such as universal approximation makes it easier to grow new neurons during training. More compact topology, back-propagation for learning and hybrid approaches with unsupervised learning in the hidden layer are some of the special distinctive features that make RBFN algorithm more unique and providing faster learning speed. As our dataset had more than 15 attributes and huge amount of data, RBFN was successful in predicting bankruptcies with more accuracy. Optimized values for batch size, epoch, step size and various other setting where used to get optimal results for each model. Following is the summary of results of all the models implemented.

Table 7: Summary of All Results

Prediction Model	Results Obtained (Accuracy)
Multi-Layer Perceptron	88.81%
Dense Neural Network	89%
Radial Basis Function Network	96.57%

3.2 Future Work

We can further explore the domain of unsupervised learning algorithms to investigate the divergent results. Algorithms such as deep belief network & autoencoders may provide interesting results as they can perform non-linear dimensionality reduction and/or use pretrained layers from another model thus improving the accuracy of results. The prediction accuracy may be further improved by adjusting the parameters or by increasing the number of neurons. All these are speculations as of now and concrete or realistic observations are yet to be made to conclude factually.

References

[1]. Aghaie, A., Saeedi, A. (2009). Using Bayesian Networks for Bankruptcy Prediction: Empirical Evidence from Iranian Companies. International Conference on Information Management and Engineering, Pages. 450-455.

[2]. Hosaka, T. (2019). Bankruptcy prediction using imaged financial ratios and convolutional neural networks. Expert Systems with Applications, Vol. 117, Pages. 287-299.

[3]. Leshno, M., Spector, Y. (1996). Neural Network Prediction Analysis: The Bankruptcy Case. Neurocomputing. Vol 10, Pages. 125-147.

[4]. Lin, F., Yeh, C., Lee, M. Y. (2011). The Use of Hybrid Manifold Learning and Support Vector Machines in the Prediction of Business Failure. Journal Knowledge-Based Systems, Volume 24, Issue 1, Pages 95-101.

[5]. Naidu G. P., Govinda, K. (2018). Bankruptcy prediction using neural networks. 2nd International Conference on Inventive Systems and Control (ICISC), Pages. 248-251.

[6]. Mai, F., Tian, S., Lee, C., Ma, L. (2019). Deep learning models for bankruptcy prediction using textual disclosures. European Journal of Operational Research, Vol. 274, Issue 2, Pages. 743-758.

[7]. Ravisankar, P., Ravi, V., Bose, I. (2010). Failure Prediction of Dotcom Companies Using Neural Network–Genetic Programming Hybrids. Information Sciences, Volume 180, Issue 8, Pages 1257-1267.

[8]. Ribeiro B., Lopes N. (2011). Deep Belief Networks for Financial Prediction. Neural Information Processing. ICONIP 2011. Lecture Notes in Computer Science, Vol.7064.

[9]. Tsai, C., Wu, J. (2008). Using neural network ensembles for bankruptcy prediction and credit scoring, Expert Systems with Applications, Vol. 34, Issue 4, Pages. 2639-2649.

[10]. Wilson, R., Sharda, R. (1994). Bankruptcy Prediction Using Neural Networks. Decision Support Systems, Volume 11, Issue 5, Pages 545-557.