CS6890 - Fraud Analytics - Assignment 4

Tanmay Garg CS20BTECH11063 | Aayush Patel CS20BTECH11001 | Shashank Shanbhag CS20BTECH11061 | Ganesh Bombatkar CS20BTECH11016 | Sushma CS20BTECH11051

***Abstract*—This document is a report on the implementation of the Cost-Sensitive Logistic Regression Model on Bank Transaction Data that has been provided in the assignment. We will compare the results of the data with different models.**

The libraries used are `PyTorch`, `Sklearn`, `NumPy`, and `Pandas`.

***Index Terms*—Logistic Regression, Cost Sensitive Logistic Regression, Artificial Neural Network**

I. Introduction

The cost associated with binary classifications for different misclassification errors is unequal in real-world scenarios. The impact of making an incorrect prediction varies depending on the problem statement.

For example, in the case of medical disease diagnosis, if a patient who is suffering from a disease is classified as healthy can lead to severe repercussions, but if a healthy person is classified as unhealthy will lead to some additional tests but will not be as harmful as the other case.

This report will address a binary classification problem where costs associated with the false negative are different for each sample. We will implement a cost-sensitive model that will take a custom cost value for each type of classification and give an optimal solution.

II. Problem Statement

We have been given Bank Transaction Data in CSV format containing multiple columns which are named as follows:

NotCount, YesCount, ATPM, PFD, PFG, SFD, SFG, WP, WS, AH, AN, Status, FNC

The below columns are independent variables:

- *NotCount*
- *YesCount*
- *ATPM*
- *PFD*
- *PFG*
- *SFD*
- *SFG*
- *WP*
- *WS*
- *AH*
- *AN*

The below columns are dependent variables:

- Status

Table I shows the cost associated with each type of classification:

	Predicted Positive	Predicted Negative
Actual Positive	$C_{TP} = 4$	C_{FN} = Data Defined
Actual Negative	$C_{FP} = 4$	$C_{TN} = 0$

TABLE I: Cost Matrix

The task is to implement a cost-sensitive logistic regression model that can consider these various costs and train a model that can minimize the overall cost incurred when predicting a transaction as fraudulent or not. The algorithms implemented will be discussed in Section IV and their results will be discussed in Section V.

III. Data Description

The CSV format data provided in the assignment contains 147636 rows and 13 columns, and the dependent and independent variables have been described in Section II.

We used `Pandas` library to analyze the data given and Table II shows the description.

	Mean	Std	Min	Median	Max
NotCount	7.72	7.57	0.00	4.00	23.00
YesCount	15.22	7.60	0.00	19.00	22.00
ATPM	0.25	0.37	0.00	0.03	1.00
PFD	0.03	0.34	0.00	0.00	79.84
PFG	0.05	0.37	0.00	0.00	51.94
SFD	0.02	0.30	0.00	0.00	61.56
SFG	0.07	1.14	0.00	0.00	209.02
WP	0.27	2.43	0.00	0.00	399.61
WS	0.49	0.76	0.00	0.10	5.00
AH	0.05	0.18	0.00	0.00	1.00
AN	0.01	0.10	0.00	0.00	1.00

TABLE II: Data Description

As we can see in Table III, the columns represent the statistic such as mean, median, standard deviation, minimum value, and maximum value.

NotCount seems more skewed towards the lower values, while *YesCount* seems more skewed towards the higher values.

PFD, PFG, SFD, SFG, WP, WS, WH, AH, AN seem to be more skewed towards the lower values as indicated by the mean and median of the dataset.

Status is the label of each sample, and its frequency distribution is given in Table III.

Label	Frequency
0	70.14%
1	29.86%

TABLE III: Frequency of Labels

FNC, *a.k.a* False Negative Cost is the false negative classification cost associated with each sample. Table IV shows the description of *FNC* column.

	mean	std	min	max
FNC	533.40	8774.01	0.00	1703185.65
	25%	**50%**	**75%**	
	0.28	11.84	106.98	

TABLE IV: FNC Data Description (3^{rd} row with 25%, 50%, 75% denote the percentile)

The mean of the costs is 533.40, and if we look at the standard deviation, it shows a wide disparity in the costs as the standard deviation is much higher than the mean. 50% percentile indicates that the median of the data is 11.84 and this signifies that 50% of the data is significantly less than the mean and the rest of the samples have a very high cost, and their frequency is also considerably less. Using the command from `Pandas` to print the number of null values for each of the columns:

```
>>> df.isnull().sum()
>>>
NotCount    0
YesCount    0
ATPM        0
PFD         0
PFG         0
SFD         0
SFG         0
WP          0
WS          0
AH          0
AN          0
Status      0
FNC         0
Total       0
dtype: int64
```

We will now try to understand the correlation between individual columns. Figure 1 shows the correlation values.

In the figure, we can see that *NotCount* and *YesCount* are completely negatively correlated which means there is a strong linear relationship among these 2 columns.

Similarly, *NotCount* is strongly correlated with *Status*, whereas *YesCount* is strongly negatively correlated with *Status*, this signifies there is a strong dependence of *Status* in the other 2 columns, which is true as *Status* is a dependent variable.

ATPM has some correlation with *WS* with a correlation value of 0.44.

PFD, PFG, SFG, WP are correlated with each other as they have high correlation values.

Using the command from `Pandas` and `Seaborn` to generate correlation matrix:

```
>>> sns.heatmap(df.corr("pearson"), vmax=1,
annot=True, cbar=False, fmt='.2f')
```

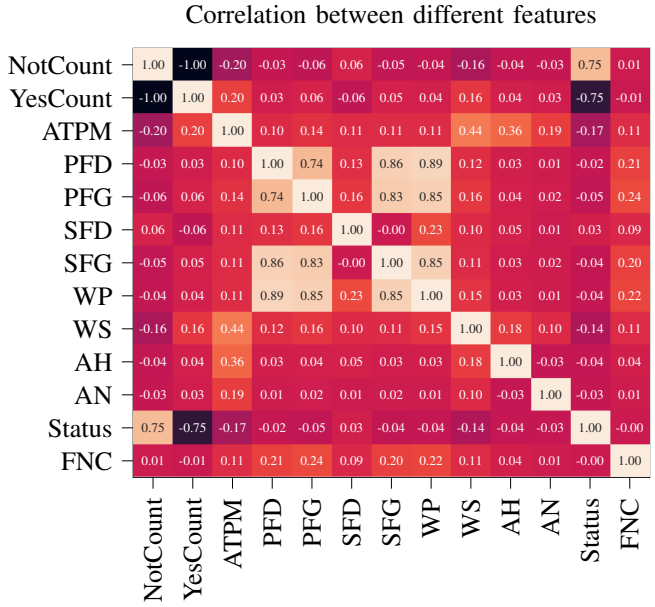


Fig. 1: Pearson Correlation of Data

IV. Algorithm

The task is to classify this data considering the different False Negative costs. Before discussing a cost-sensitive model, we will first discuss the algorithm of the usual Logistic Regression model.

A. Logistic Regression

Logistic Regression is a machine-learning algorithm for binary classification. It uses a logistic function to estimate the probability of an input belonging to a particular label.

$$h_\theta(\mathbf{x}_i) = P(y = 1|\mathbf{x} = \mathbf{x}_i) = g(\boldsymbol{\theta}^T \mathbf{x}_i) \tag{1}$$

Here the function $g(\cdot)$ is the sigmoid or logistic function and is defined below:

$$g(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

For Logistic Regression, the cost function of defined as an entropy function also known as Binary Cross Entropy Loss:

$$J(\mathbf{x}, y, \boldsymbol{\theta}) = -y \log(h_\theta(\mathbf{x})) - (1 - y) \log(1 - h_\theta(\mathbf{x})) \tag{3}$$

Here $\mathbf{x}$ denotes the samples and $\boldsymbol{\theta}$ denotes the parameter of the logistic regression model. The objective is to minimize this cost over all the samples that have been given. So we will write our total cost function as:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} J(\mathbf{x}_i, y_i, \boldsymbol{\theta}) \tag{4}$$

The objective function can be written as:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \tag{5}$$

B. Underlying Assumption for Logistic Regression

The underlying assumption with Logistic Regression model is that the cost for False Positive and False Negative is taken to be same and also the cost for True Positive and True Negative is taken to be same.

This does not take into account the real world scenarios in the case of fraud detection. The proof of the underlying assumption is as follows:

Proof 1:

Assuming that our parameters are kept constant as $\boldsymbol{\theta}$

Case 1: y = 0 and $h_\theta(\mathbf{x}) \approx 0$

$$\begin{aligned} J(\mathbf{x}, y, \boldsymbol{\theta}) &= -y \log(h_\theta(\mathbf{x})) - (1-y)\log(1-h_\theta(\mathbf{x})) \\ &\approx \lim_{\epsilon \to 0} -0 \cdot \log(\epsilon) - (1-0)\cdot \log(1-\epsilon) \\ &\approx 0 \end{aligned}$$

Case 2: y = 0 and $h_\theta(\mathbf{x}) \approx 1$

$$\begin{aligned} J(\mathbf{x}, y, \boldsymbol{\theta}) &= -y \log(h_\theta(\mathbf{x})) - (1-y)\log(1-h_\theta(\mathbf{x})) \\ &\approx \lim_{\epsilon \to 1} -0 \cdot \log(\epsilon) - (1-0)\cdot \log(1-\epsilon) \\ &\approx \infty \end{aligned}$$

Case 3: y = 1 and $h_\theta(\mathbf{x}) \approx 0$

$$\begin{aligned} J(\mathbf{x}, y, \boldsymbol{\theta}) &= -y \log(h_\theta(\mathbf{x})) - (1-y)\log(1-h_\theta(\mathbf{x})) \\ &\approx \lim_{\epsilon \to 0} -1 \cdot \log(\epsilon) - (1-1)\cdot \log(1-\epsilon) \\ &\approx \infty \end{aligned}$$

Case 4: y = 1 and $h_\theta(\mathbf{x}) \approx 1$

$$\begin{aligned} J(\mathbf{x}, y, \boldsymbol{\theta}) &= -y \log(h_\theta(\mathbf{x})) - (1-y)\log(1-h_\theta(\mathbf{x})) \\ &\approx \lim_{\epsilon \to 1} -1 \cdot \log(\epsilon) - (1-1)\cdot \log(1-\epsilon) \\ &\approx 0 \end{aligned}$$

C. Cost Sensitive Logistic Regression

As seen in Proof 1, we can see what the costs for 4 different classification scenarios look like and how it does not consider the real-world costs.

We will now modify the cost function given as per Table I into a new cost function as below:

$$\begin{aligned} J_c(\mathbf{x}, y, \boldsymbol{\theta}) = y\Big(h_\theta(\mathbf{x})C_{TP} + (1-h_\theta(\mathbf{x}))C_{FN}\Big) \\ + (1-y)\Big(h_\theta(\mathbf{x})C_{FP} + (1-h_\theta(\mathbf{x}))C_{TN}\Big) \end{aligned} \quad (6)$$

We can now use this modified cost function to write our total cost function by summing up all samples.

$$\mathcal{L}_c(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} J_c(\mathbf{x}_i, y_i, \boldsymbol{\theta}) \quad (7)$$

The modified objective function can be written as:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_c(\boldsymbol{\theta}) \quad (8)$$

D. Neural Network

Artificial Neural Network is a machine-learning model capable of learning complex patterns and structures of data and can be used for classification and regression.

They consist of *neurons*, that receive input, apply some transformation and then give output.

Let us take our network has n hidden layers. The network first performs a *forward pass* on the input.

Input Layer:

$$\mathbf{a}^{(0)} = \mathbf{x} \quad (9)$$

where $\mathbf{x}$ is the input data and $\mathbf{a}^{(0)}$ is the activation of the input layer which is same as $\mathbf{x}$.

Hidden Layers: For $l = 1$ to n, the activation of the l-th hidden layer can be calculated as:

$$\mathbf{a}^{(l)} = g(\mathbf{z}^{(l)}) = g(\mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}) \quad (10)$$

where $\mathbf{a}^{(l)}$ is the activation of the l-th hidden layer, $\mathbf{z}^{(l)}$ is the weighted sum of inputs for the l-th hidden layer, $\mathbf{W}^{(l)}$ is the weight matrix, $\mathbf{b}^{(l)}$ is the bias vector, and $g(\cdot)$ is the activation function such as *ReLU*, *Sigmoid*, *Tanh*, etc.

Output Layer: The activation of the output layer can be calculated as:

$$\mathbf{a}^{(n+1)} = g(\mathbf{z}^{(n+1)}) = g(\mathbf{W}^{(n+1)}\mathbf{a}^{(n)} + \mathbf{b}^{(n+1)}) \quad (11)$$

where $\mathbf{a}^{(n+1)}$ is the activation of the output layer, $\mathbf{z}^{(n+1)}$ is the weighted sum of inputs for the output layer, $\mathbf{W}^{(n+1)}$ is the weight matrix, $\mathbf{b}^{(n+1)}$ is the bias vector, and $g(\cdot)$ is the activation function *a.k.a Sigmoid*.

Here we can see a combination of linear as well as some non-linear operators to give us an output. The mathematical expression for probability of an input belonging to a particular label.

$$h_{\mathbf{w}}(\mathbf{x}_i) = P(y = 1 | \mathbf{x} = \mathbf{x}_i) = f(\mathbf{w}, \mathbf{x}_i) \quad (12)$$

Here w is the set of parameters that define the neural network function $f(\cdot)$ to transform input to the label. The cost can be defined as:

$$J(\mathbf{x}, y, \mathbf{w}) = -y \log(h_{\mathbf{w}}(\mathbf{x})) - (1-y)\log(1-h_{\mathbf{w}}(\mathbf{x})) \quad (13)$$

The total cost can also be written similarly to Equation 7:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} J(\mathbf{x}_i, y_i, \mathbf{w}) \quad (14)$$

The objective function can be written as:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \quad (15)$$

Similarly, for the cost-sensitive neural network model, we can modify our cost function in Equation 13 similar to Equation 6

$$\begin{aligned} J_c(\mathbf{x}, y, \mathbf{w}) = y\Big(h_{\mathbf{w}}(\mathbf{x})C_{TP} + (1-h_{\mathbf{w}}(\mathbf{x}))C_{FN}\Big) \\ + (1-y)\Big(h_{\mathbf{w}}(\mathbf{x})C_{FP} + (1-h_{\mathbf{w}}(\mathbf{x}))C_{TN}\Big) \end{aligned} \quad (16)$$

The new equation will be:

$$\mathcal{L}_c(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} J_c(\mathbf{x}_i, y_i, \mathbf{w}) \tag{17}$$

The modified objective will be:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathcal{L}_c(\mathbf{w}) \tag{18}$$

V. Results

We have conducted experiments on all four models discussed above, which are

- Logistic Regression
- Cost-Sensitive Logistic Regression
- Neural Network
- Cost-Sensitive Neural Network

The experiments have been conducted on multiple optimizers and their parameters. We will also discuss the results of these models in terms of accuracy and the expected cost of the test data.

The data has 147636 samples, and we have randomly split the data into $70 : 18 : 12$ for Train, Validation, and Test, respectively.

Experiments have been run over multiple hyperparameters and metrics for evaluating the models.

The different values of hyperparameters experimented on are:

- $lr = [0.001, 0.01]$
- momentum $= [0, 0.9]$
- weight decay = $[0, 0.0001]$

The optimizer used here is SGD optimizer from `PyTorch`.

For the Neural Network model, we have taken a single hidden layer network with *ReLU* activation layer and a final *Sigmoid* layer for classification.

The metrics on which the models are evaluated during validation are Accuracy and the Expected Cost of the validation dataset [1].

Testing different models for different hyperparameters are as follows:

Fig 2 shows the test data result for expected cost for logistic regression model.

Fig 3 shows the test data result for accuracy for logistic regression model.

Fig 4 shows the test data result for expected cost for cost sensitive logistic regression model.

Fig 5 shows the test data result for accuracy for cost sensitive logistic regression model.

Fig 6 shows the test data result for expected cost for neural network model.

Fig 7 shows the test data result for accuracy for neural network model.

Fig 8 shows the test data result for expected cost for cost sensitive neural network model.

Fig 9 shows the test data result for accuracy for cost sensitive neural network model.

[1]Results of Validation Dataset of different models and hyperparameters is given in Appendix A

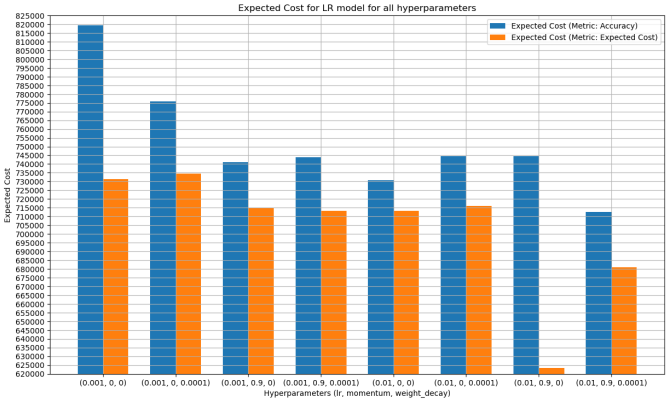


Fig. 2: Expected Cost of Logistic Regression

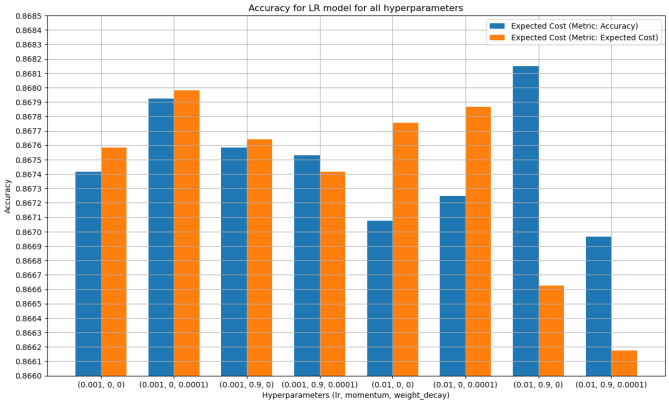


Fig. 3: Accuracy of Logistic Regression

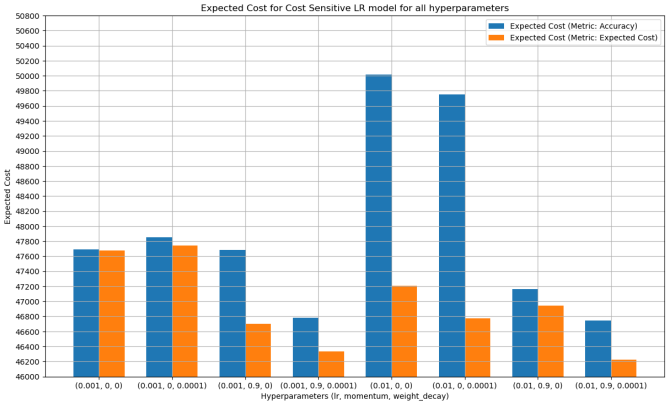


Fig. 4: Expected Cost of Logistic Regression

As seen from the plots discussed above, we have concluded that considering the metric as *Expected Cost* has given us models with the least expected cost, even performing almost similarly in terms of accuracy. However, in the case of Cost-Sensitive Logistic Regression, considering the *Accuracy* metric is better with the given architecture.

Opting not to take momentum and keeping a weight decay as low as possible has proven to give optimal results. For learning rate, both $0.01, 0.001$ have given optimal results.

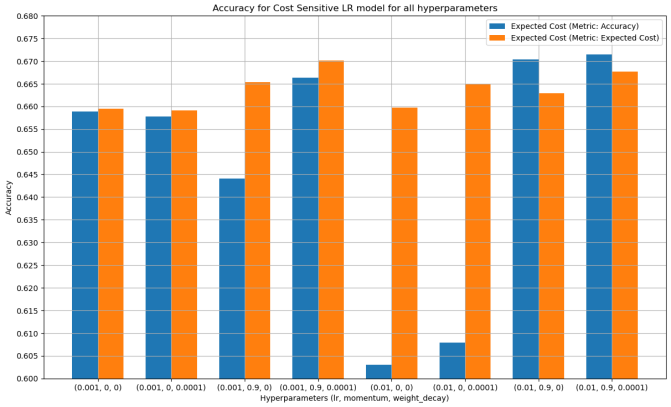


Fig. 5: Accuracy of Logistic Regression

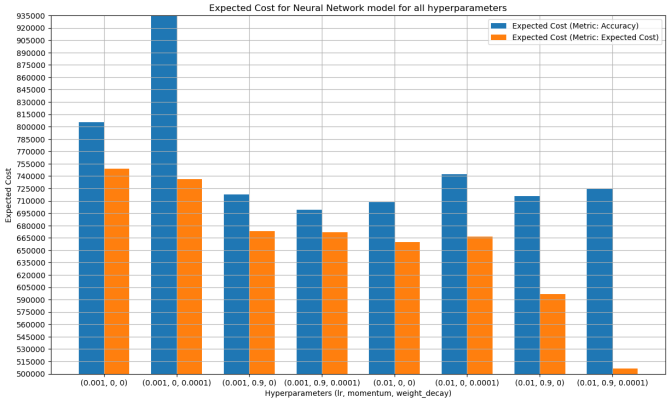


Fig. 6: Expected Cost of Logistic Regression

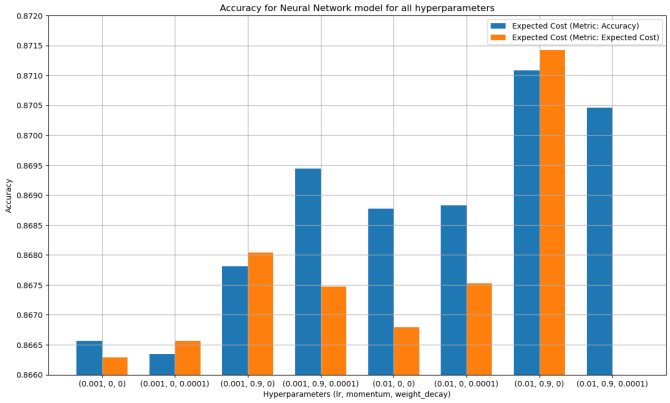


Fig. 7: Accuracy of Logistic Regression

We will now compare each of the best models and draw conclusions and observations.

Fig. 11, 10 shows the comparison of best model results in terms of both expected cost and the accuracy of prediction on the test dataset.

We can observe that even though the accuracy of cost sensitive models is much lower than the accuracy of cost insensitive models, we can surely gain a lower expected cost for cost sensitive models as compared to cost insensitive models.

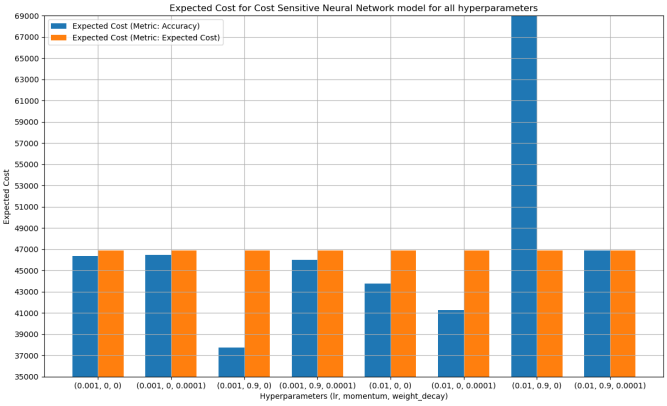


Fig. 8: Expected Cost of Logistic Regression

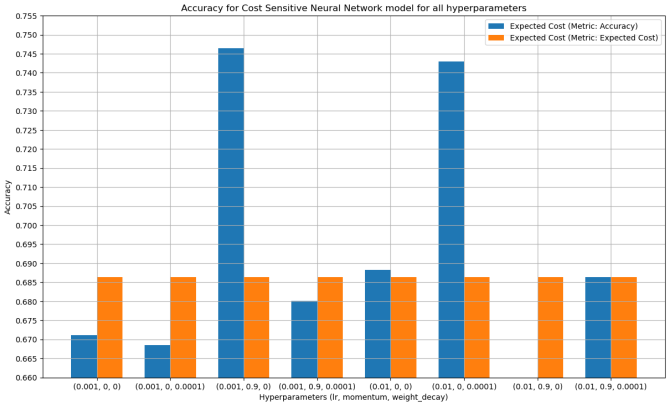


Fig. 9: Accuracy of Logistic Regression

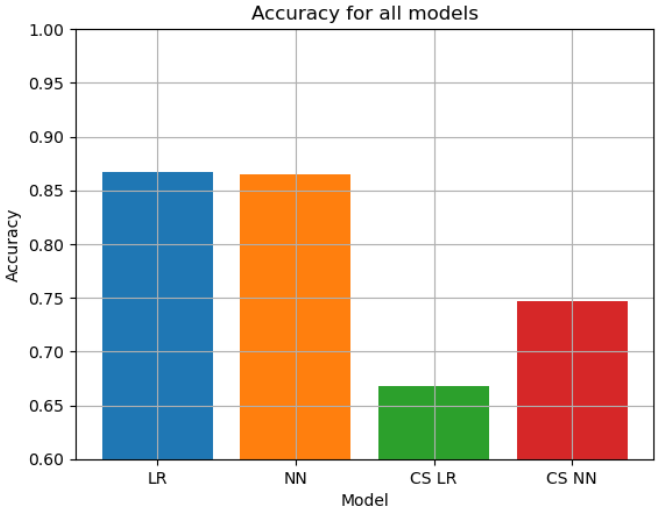


Fig. 10: Model Comparison for Accuracy on Test Data

VI. Conclusion

This report presented the development and evaluation of a cost-sensitive logistic regression model for binary classifi-

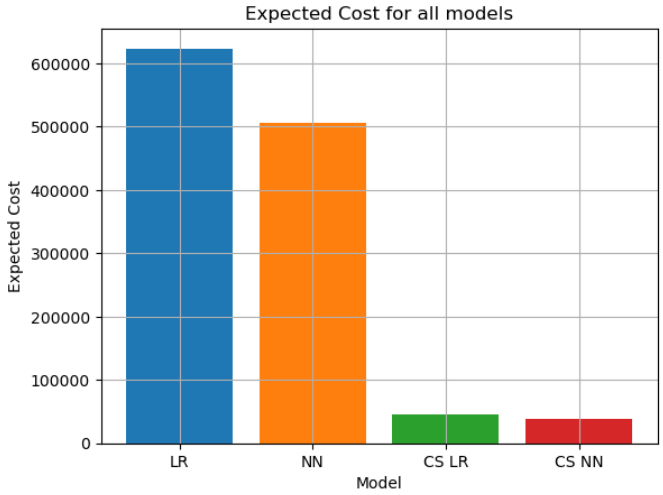


Fig. 11: Model Comparison for Expected Cost on Test Data

cation using a dataset with varying false negative costs and constant costs for false positive, true positive, and true negative classification errors.

As discussed in all the sections, we can surely conclude from the results that a cost sensitive model has an expected cost much lesser than cost insensitive model but has a lower accuracy.

This is due to the fact that the model is able to detect high cost samples correctly and can classify the samples with lower costs incorrectly. The trade off here is between *Expected Cost v/s. Accuracy*.

The model was able to achieve satisfactory performance on both test data, indicating its potential for real-world applications where cost-sensitive classification is required.

We can also conclude that Neural Network model performs better in both situations when the model training was cost sensitive and cost insensitive.

As a future direction, the model could be further refined by exploring alternative cost-sensitive learning techniques, such as genetic algorithms or ensemble methods. Additionally, the model's performance could be evaluated on a broader range of datasets, further validating its robustness.

References

[1] PyData Berlin 2015: CostCla Library

[2] Bahnsen, Alejandro Correa, Djamila Aouada and Björn E. Ottersten. Ensemble of Example-Dependent Cost-Sensitive Decision Trees, (2015)

[3] A. C. Bahnsen, D. Aouada and B. Ottersten, Example-Dependent Cost-Sensitive Logistic Regression for Credit Scoring, 2014 13th International Conference on Machine Learning and Applications

[4] Shen, F., Wang, R., Shen, Y. (2020). A cost-sensitive logistic regression credit scoring model based on multi-objective optimization approach. Technological and Economic Development of Economy, 26(2), 405-429.

[5] Example-Dependent Cost-Sensitive Classification with Applications in Financial Risk Modeling and Marketing Analytics

[6] Sterner, P., Goretzko, D., Pargent, F. (2021, December 7). Everything has its Price: Foundations of Cost-Sensitive Learning and its Application in Psychology

[7] Ling, Charles Sheng, Victor. (2010). Cost-Sensitive Learning and the Class Imbalance Problem, Encyclopedia of Machine Learning.

[8] Le Borgne, Yann-Ael, Wissam, Siblini, Bertrand, Lebichot, and Gianluca, Bontempi. Reproducible Machine Learning for Credit Card Fraud Detection - Practical Handbook. Universite Libre de Bruxelles, 2022.

[9] Fraud detection with cost-sensitive machine learning

Appendix

This section contains all the plots that have been referred to before but not included in the main text.

The validation accuracy plots of the Logistic Regression Model for metrics *accuracy, cost* are given in Fig. 12.

The validation accuracy plots of the Cost-Sensitive Logistic Regression Model for metrics *accuracy, cost* are given in Fig. 14.

The validation accuracy plots of the Neural Network Model for metrics *accuracy, cost* are given in Fig. 15.

The validation accuracy plots of the Cost-Sensitive Neural Network Model for metrics *accuracy, cost* are given in Fig. 16.

The code for the implementation and running of the experiments can be found in the ipynb file provided with the submission.

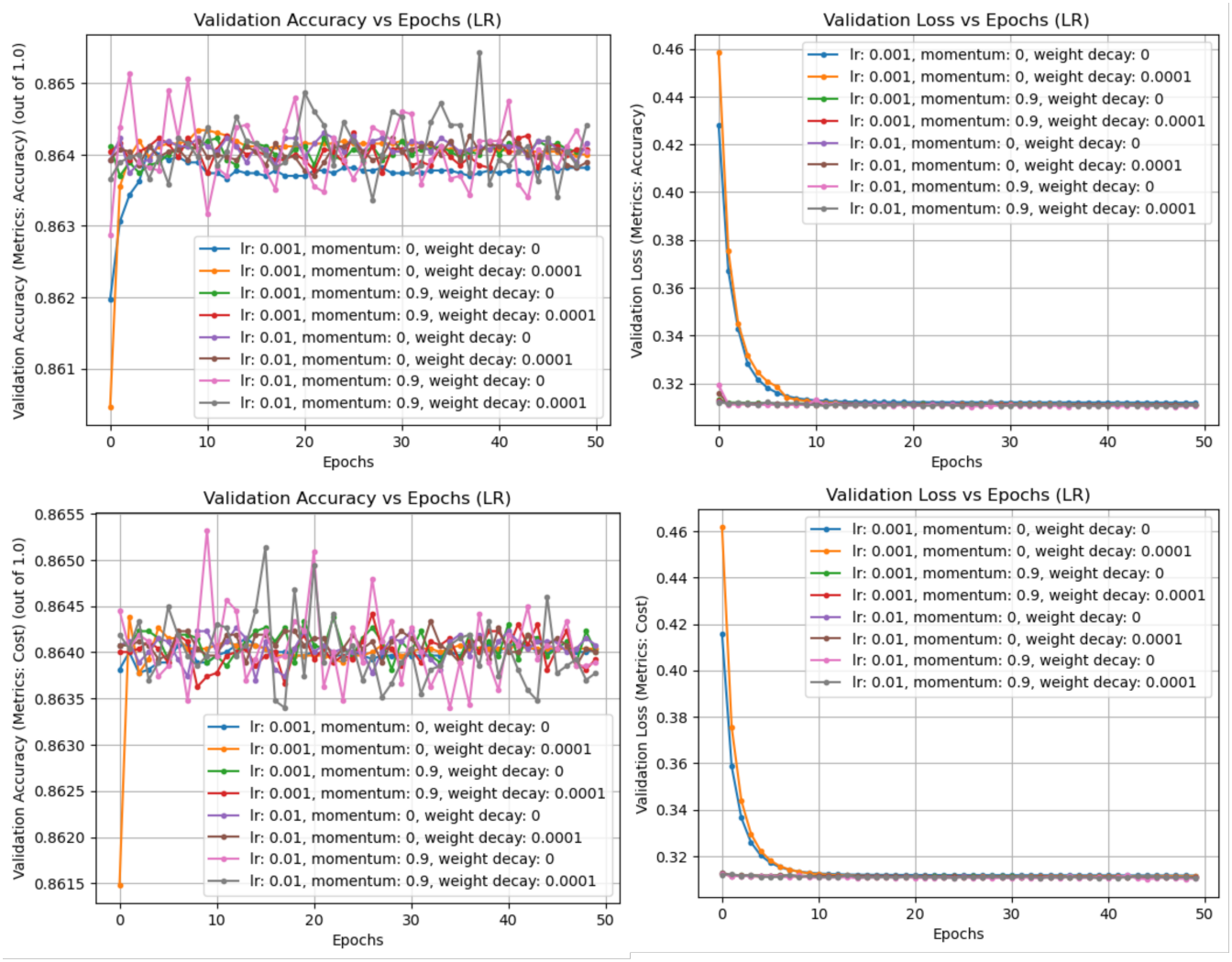
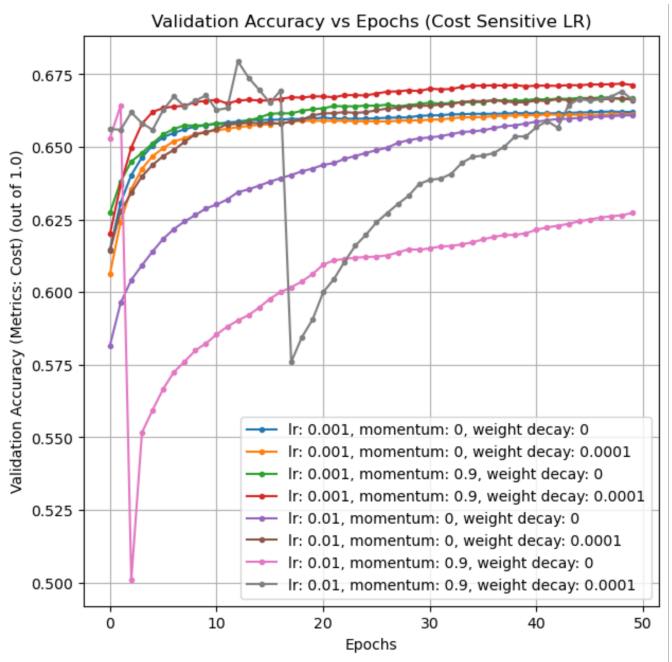


Fig. 12: Logistic Regression Accuracy and Loss Plots



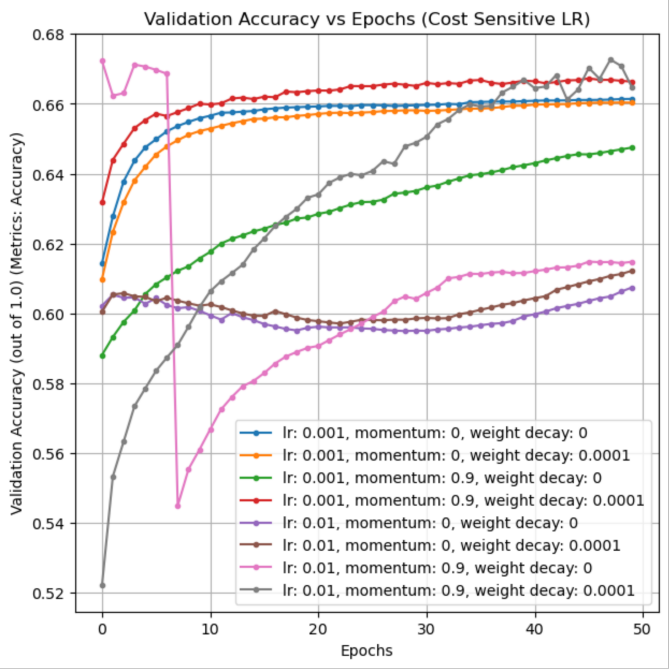
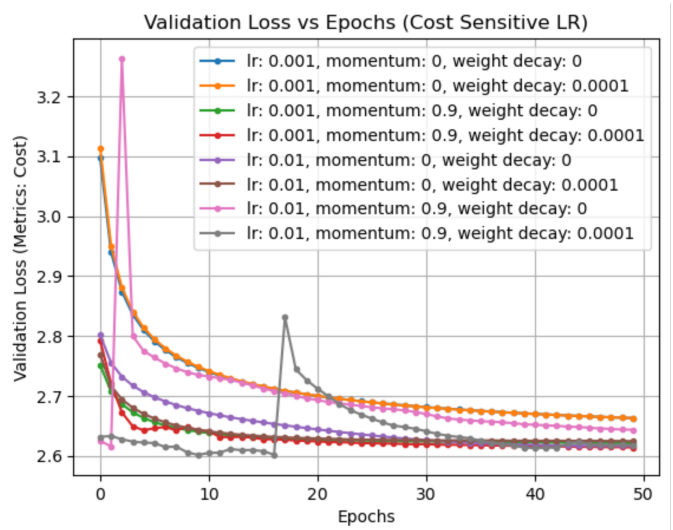


Fig. 13: Cost Sensitive Logistic Regression Accuracy Plot



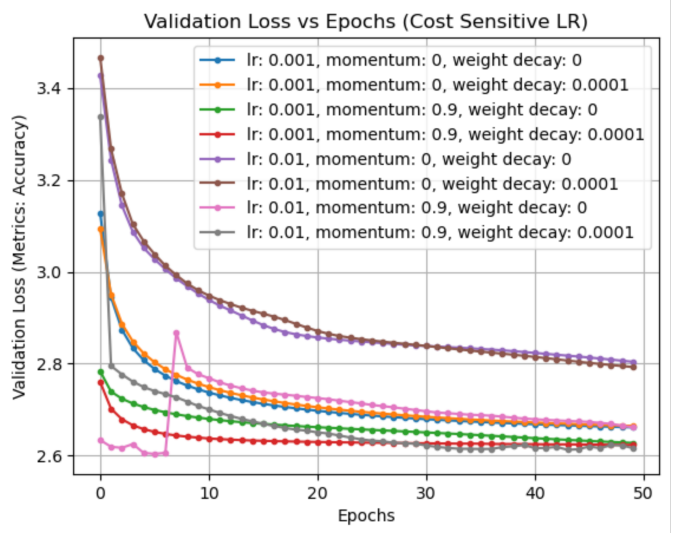


Fig. 14: Cost Sensitive Logistic Regression Loss Plot

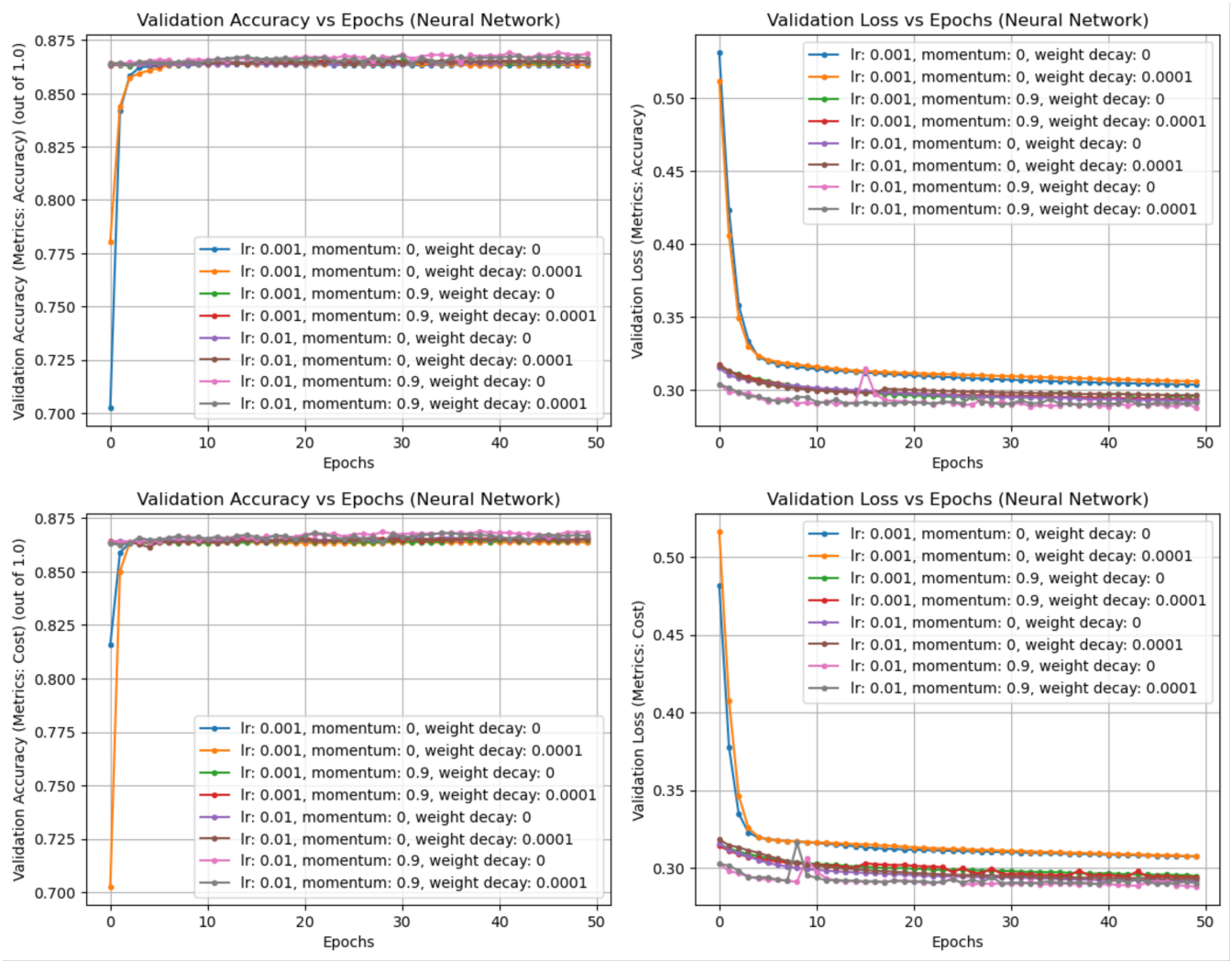


Fig. 15: Neural Network Accuracy and Loss Plots

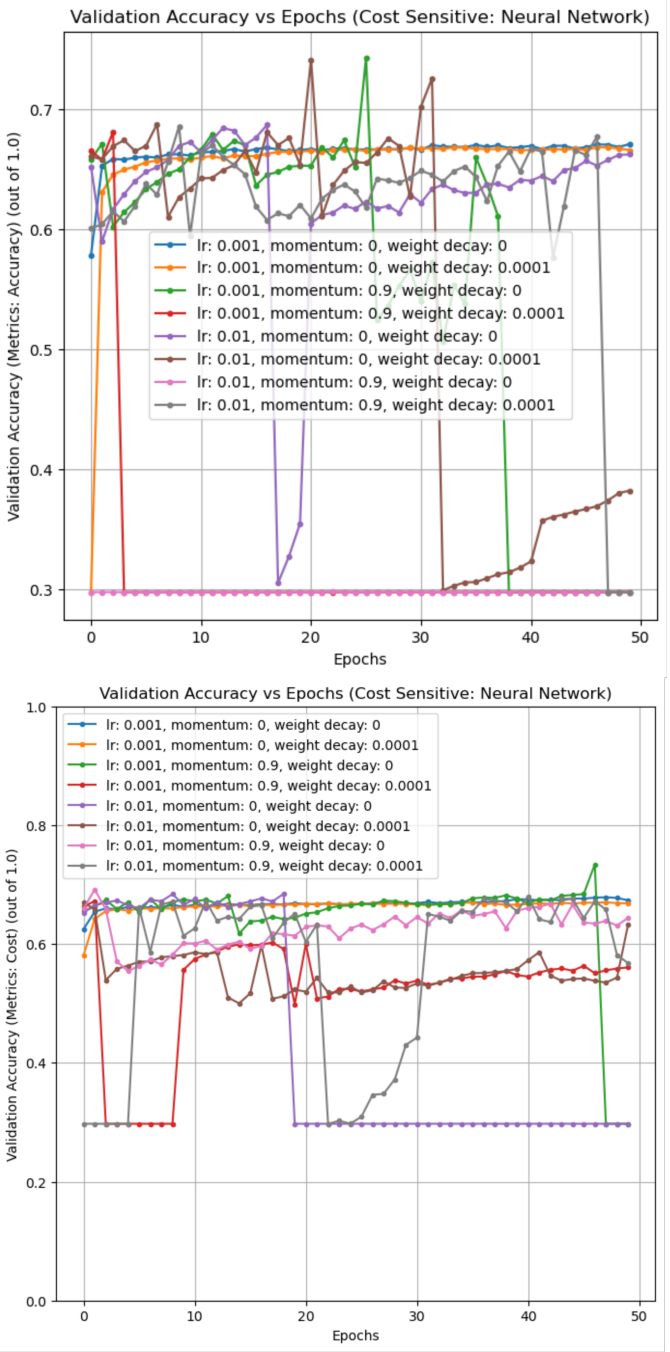


Fig. 16: Cost Sensitive Neural Network Accuracy Plot

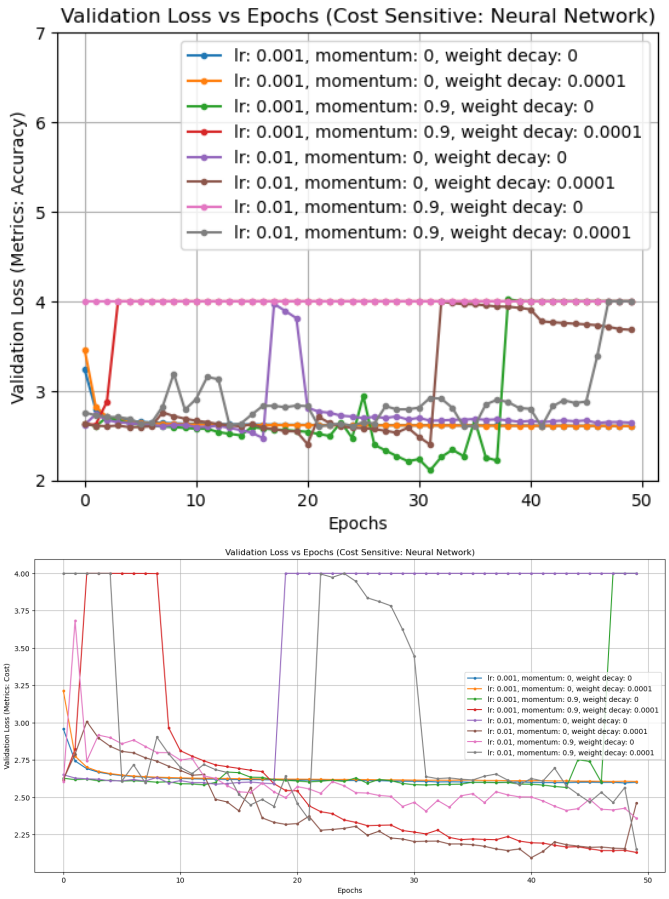


Fig. 17: Cost Sensitive Neural Network Loss Plot