Example dependent cost-sensitive classification using a deep neural net

Team: Puneet Mangla (CS17BTECH11029)
Yash Khasbarge (CS17BTECH11044)
Rushikesh Tammewar (CS17BTECH11041)

Objective: Implement example-dependent cost-sensitive classification using a deep neural network to classify fraudulent and non-fraudulent taxpayers.
Given a feature vector **x** for each taxpayer and a label **y** which is 1 iff taxpayer is fraudulent else 0, train a Deep neural network $\mathbf{h_\theta(x)}$ (outputs P(y=1|x)) by optimizing the following objective

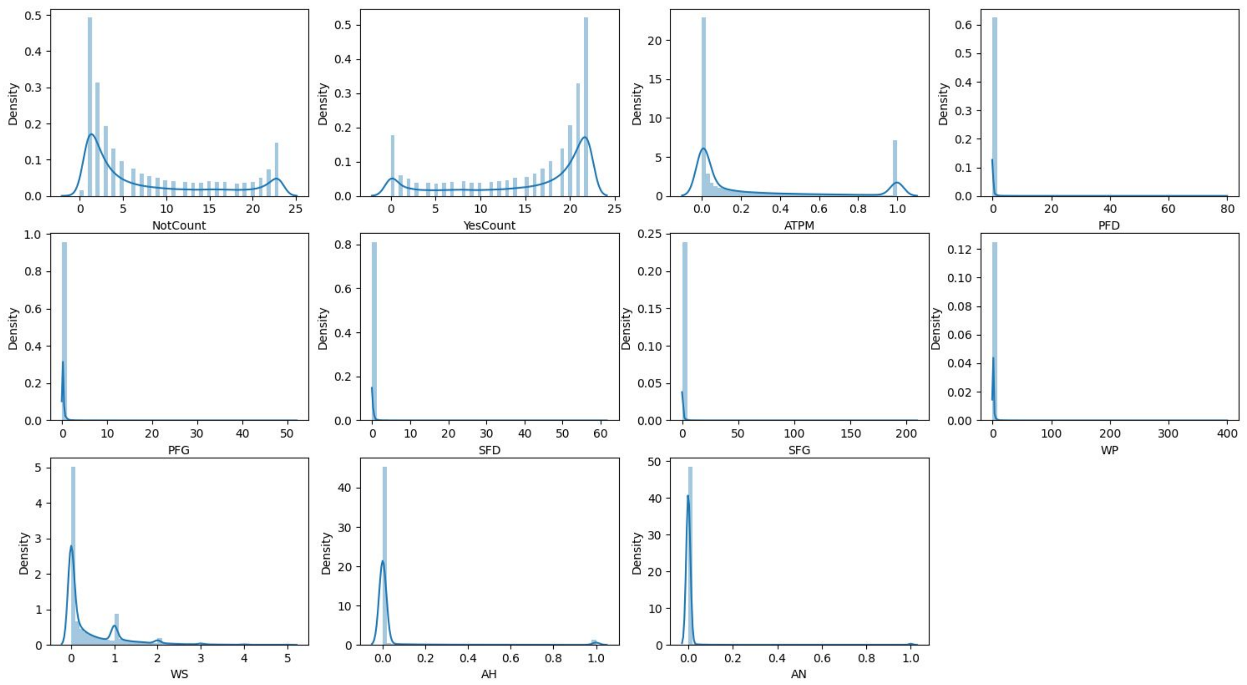
$$J^c(\theta) = \frac{1}{N}\sum_{i=1}^{N}\Big(y_i(h_\theta(\mathbf{x}_i)C_{TP_i} + (1-h_\theta(\mathbf{x}_i))C_{FN_i})$$
$$+(1-y_i)(h_\theta(\mathbf{x}_i)C_{FP_i} + (1-h_\theta(\mathbf{x}_i))C_{TN_i})\Big).$$

Here CTP, CFN, CFP, and CTN are costs for true positive, false negative, false positive, and true negative respectively.
At inference, we make predictions by obtaining probability, $\mathbf{P(y=1|x) = h_\theta(x)}$. If $\mathbf{P(y=1|x) > 0.5}$ we classify it as positive else negative.

Dataset Summary:

- 147636 samples
- "Status" is the dependent variable. Class "0" has 103554 samples and Class "1" has 44082 samples.
- "FNC" gives us false negative cost for each example.
- Rest 11 fields are independent features used to predict.
- <u>Description of each field:</u>
  Status (dependent), NotCount (not filed in time), YesCount (filed in time), ATPM (Average tax per month), PFD (purchase from fraudulent), PFG (purchase from genuine), SFD (Sales to fraudulent), SFG (sales to genuine), WP (purse value in waybill), WS (sales value in waybills), FNC (false negative cost), AH (0ne more independent variable), AN (0ne more independent variable)

- Distribution plots of 11 features



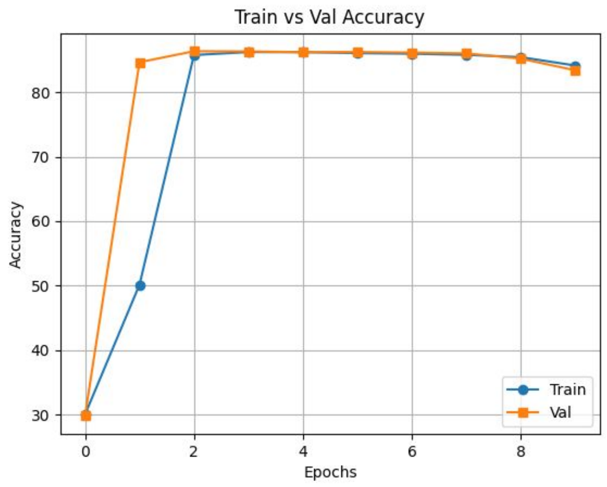
Attribute	**Mean**	**Variance**
NotCount	57.369688	7.7221
YesCount	57.713207	15.221
ATPM	0.134161	0.2532
PFD	0.112566	0.027062
PFG	0.139405	0.050789
SFD	0.088974	0.021648
SFG	1.289939	0.071709
WP	5.916689	0.271927
WS	0.582762	0.493587
AH	0.033224	0.045045
AN	0.010196	0.013474

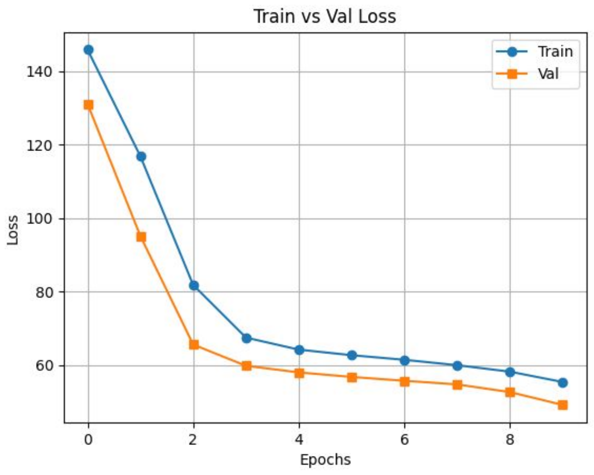
Experimentation Results:

- <u>Implementation Details:</u>
    - FPC, TPC, and TNC = 150.0, 150.0, 0.0
    - Number of epochs: 10
    - Train: 100000 samples, Validation: 15000 samples, and Test: 32636 samples
    - Adam Optimizer with learning rate 1e-3 is used for optimizing
    - Checkpoint with best validation accuracy is used to evaluate on test data
    - 3 Layer MLP with ReLU activations. Below is the architecture

```
nn.Sequential( nn.Linear(11,8),
                                nn.ReLU(),
                                nn.Linear(8,4),
                                nn.ReLU(),
                                nn.Linear(4,1),
nn.Sigmoid())
```

- Evaluation Metrics:
    - Accuracy: 100*(#correct predictions) / (total predictions)
    - Precision: (#True positives)/ (#True positives + #False positives)
    - Recall : (#True positives)/ (#True positives + #False negatives)
- <u>Results:</u>
    - Test accuracy: 86.457 %
    - Test precision: 0.779, Test Recall: 0.756, Test F1 Score: 0.767
    - Train vs Val Accuracy/Loss Curves below


Train vs Val Accuracy
Accuracy
Epochs
Train
Val


Train vs Val Loss
Loss
Epochs
Train
Val