Nick Lutostanski

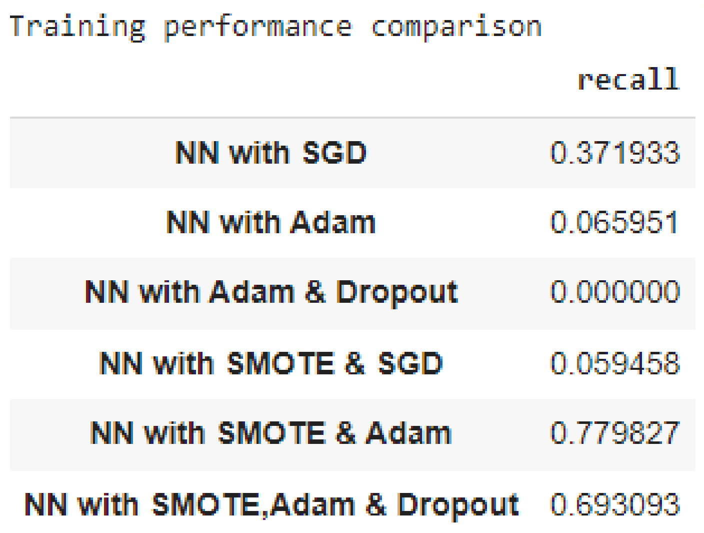
Intro to Neural Networks: AI/ML for Business Applications

4/5/2024

Contents / Agenda

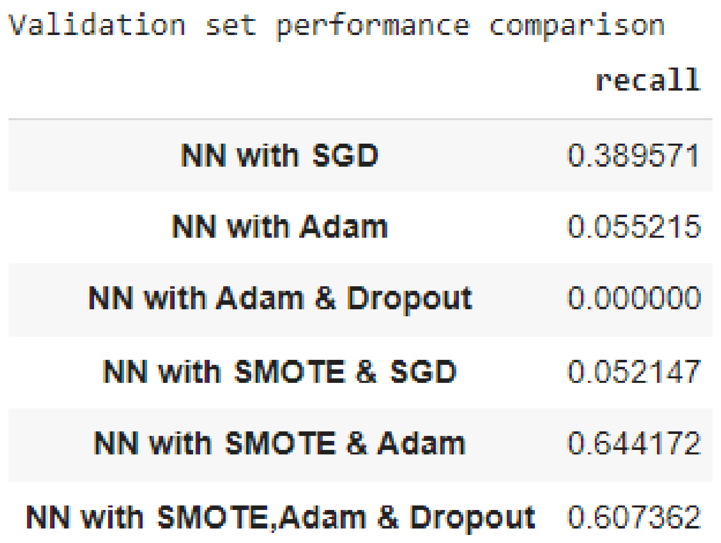
Executive Summary

- The goal of the project was to build 6 different neural networks using different methods of class balancing, weight optimizers, a the dropout method and see which would work better. Our results are below:



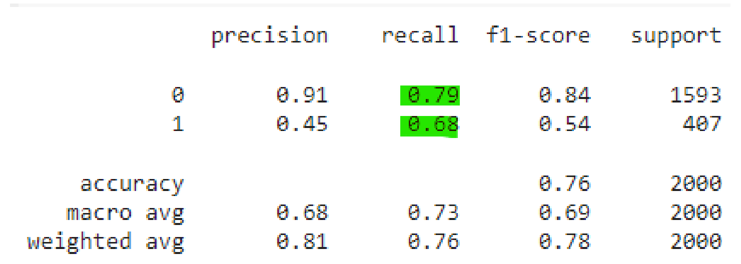
Training performance comparison

	recall
NN with SGD	0.371933
NN with Adam	0.065951
NN with Adam & Dropout	0.000000
NN with SMOTE & SGD	0.059458
NN with SMOTE & Adam	0.779827
NN with SMOTE,Adam & Dropout	0.693093



Validation set performance comparison

	recall
NN with SGD	0.389571
NN with Adam	0.055215
NN with Adam & Dropout	0.000000
NN with SMOTE & SGD	0.052147
NN with SMOTE & Adam	0.644172
NN with SMOTE,Adam & Dropout	0.607362



```
              precision    recall  f1-score   support

           0       0.91      0.79      0.84      1593
           1       0.45      0.68      0.54       407

    accuracy                           0.76      2000
   macro avg       0.68      0.73      0.69      2000
weighted avg       0.81      0.76      0.78      2000
```

Performance of NN w/ SOTE & Adam on unseen test data

- Based on these scores, we would recommend deployment of the Neural network with Smote class balancing and the Adam optimizer for weights to predict which customers will churn and which will not.

- It produced the highest recall score among training/validation and then as well on unseen testing data. Further tinkering with hyperparameters – adding more nodes per player, or an additional layer, might be recommended to finer tune performance.

- We can then run this network on unseen test data and target those that we believe will churn in the next 6 months with special promotions or incentives to stay with the bank.

Business Problem Overview and Solution Approach

- Businesses like banks that provide service have to worry about the problem of 'Customer Churn' i.e. customers leaving and joining another service provider. It is important to understand which aspects of the service influence a customer's decision in this regard. Management can concentrate efforts on the improvement of service, keeping in mind these priorities.
- You as a Data scientist with the bank need to build a neural network-based classifier that can determine whether a customer will leave the bank or not in the next 6 months.
- We will build a neural network that will predict if a customer is likely to churn in the next 6 months starting with a SGD as an optimizer. We will then try and improve the model by using Adam as an optimizer, Adam with the dropout technique. After building these 3 networks, we will balance the sample data using SMOTE (to oversample the minority class) and build them all again to see the potential performance improvements.

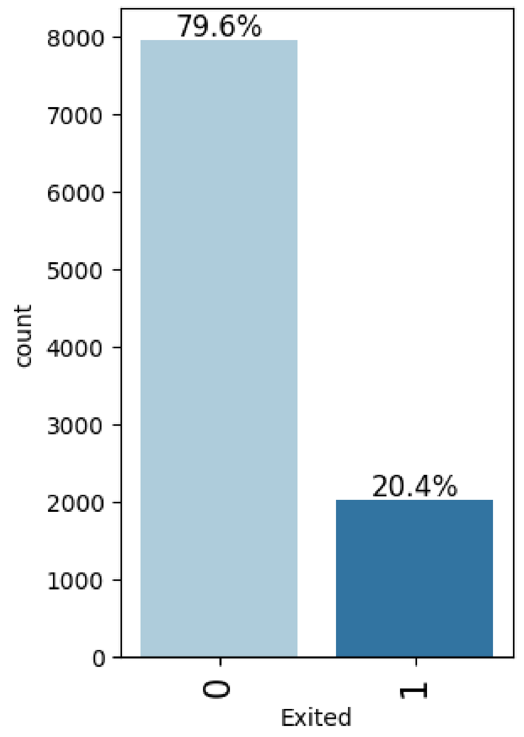
Business Problem Overview - Dataset Information

Data Dictionary

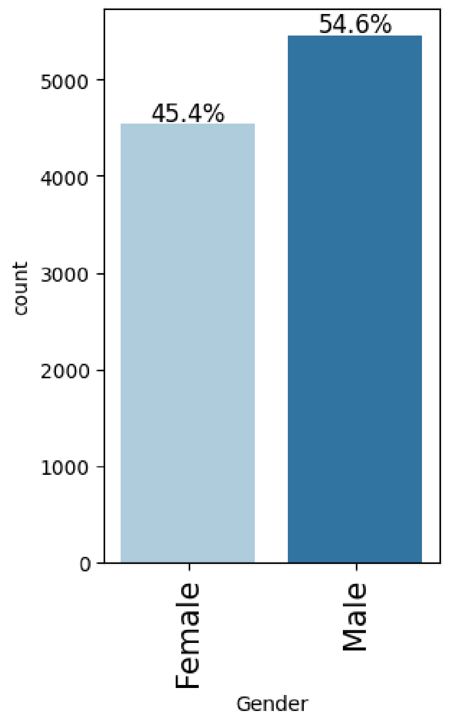
- CustomerId: Unique ID which is assigned to each customer
- Surname: Last name of the customer
- CreditScore: It defines the credit history of the customer.
- Geography: A customer's location
- Gender: It defines the Gender of the customer
- Age: Age of the customer
- Tenure: Number of years for which the customer has been with the bank
- NumOfProducts: refers to the number of products that a customer has purchased through the bank.
- Balance: Account balance
- HasCrCard: It is a categorical variable which decides whether the customer has credit card or not.
- EstimatedSalary: Estimated salary
- IsActiveMember: Is is a categorical variable which decides whether the customer is active member of the bank or not (Active member in the sense, using bank products regularly, making transactions etc)
- Exited : whether or not the customer left the bank within six month. It can take two values
    - 0 = No (Customer did not leave the bank)
    - 1 = Yes (Customer left the bank)

EDA Results – Univariate Analysis

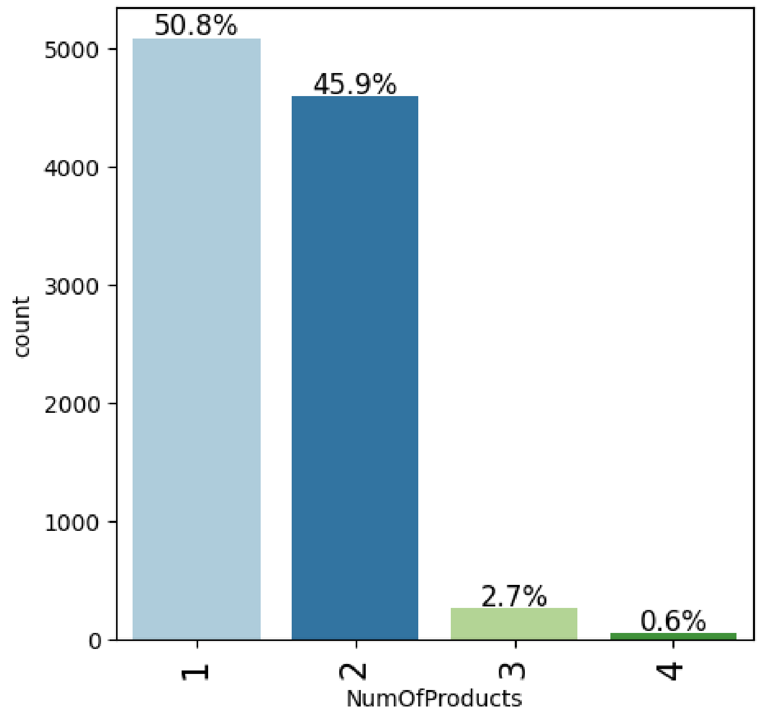
```
labeled_barplot(ds, "Exited", perc=True)
```



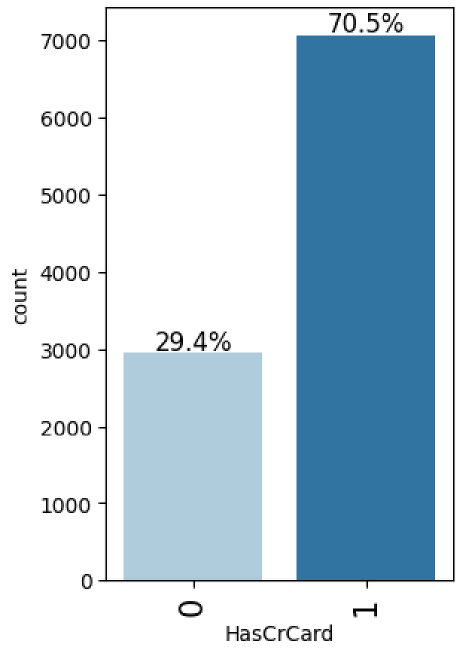
```
labeled_barplot(ds,'Gender', perc=True)
```



```
labeled_barplot(ds, 'NumOfProducts', perc=True)
```

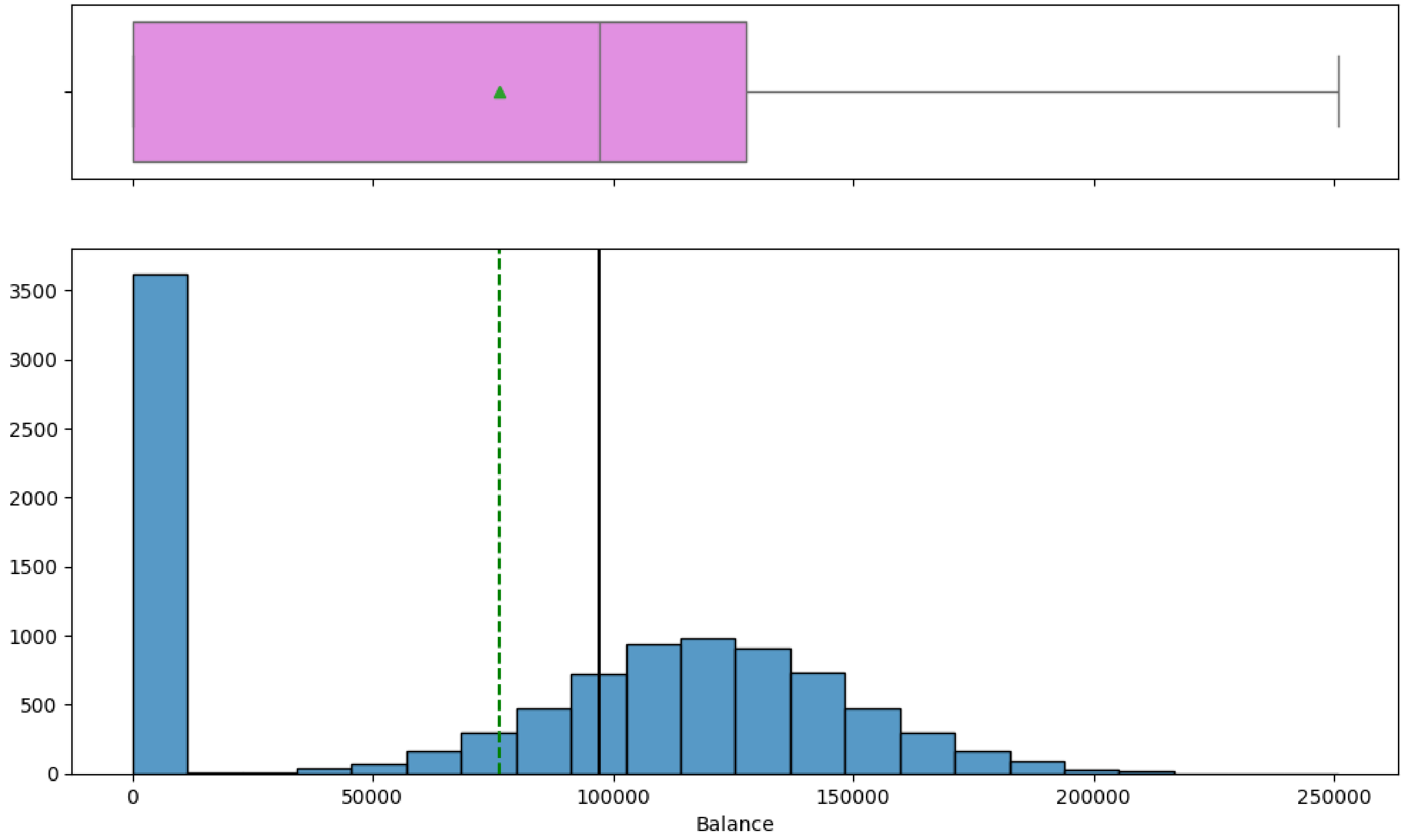


```
labeled_barplot(ds, 'HasCrCard', perc=True)
```



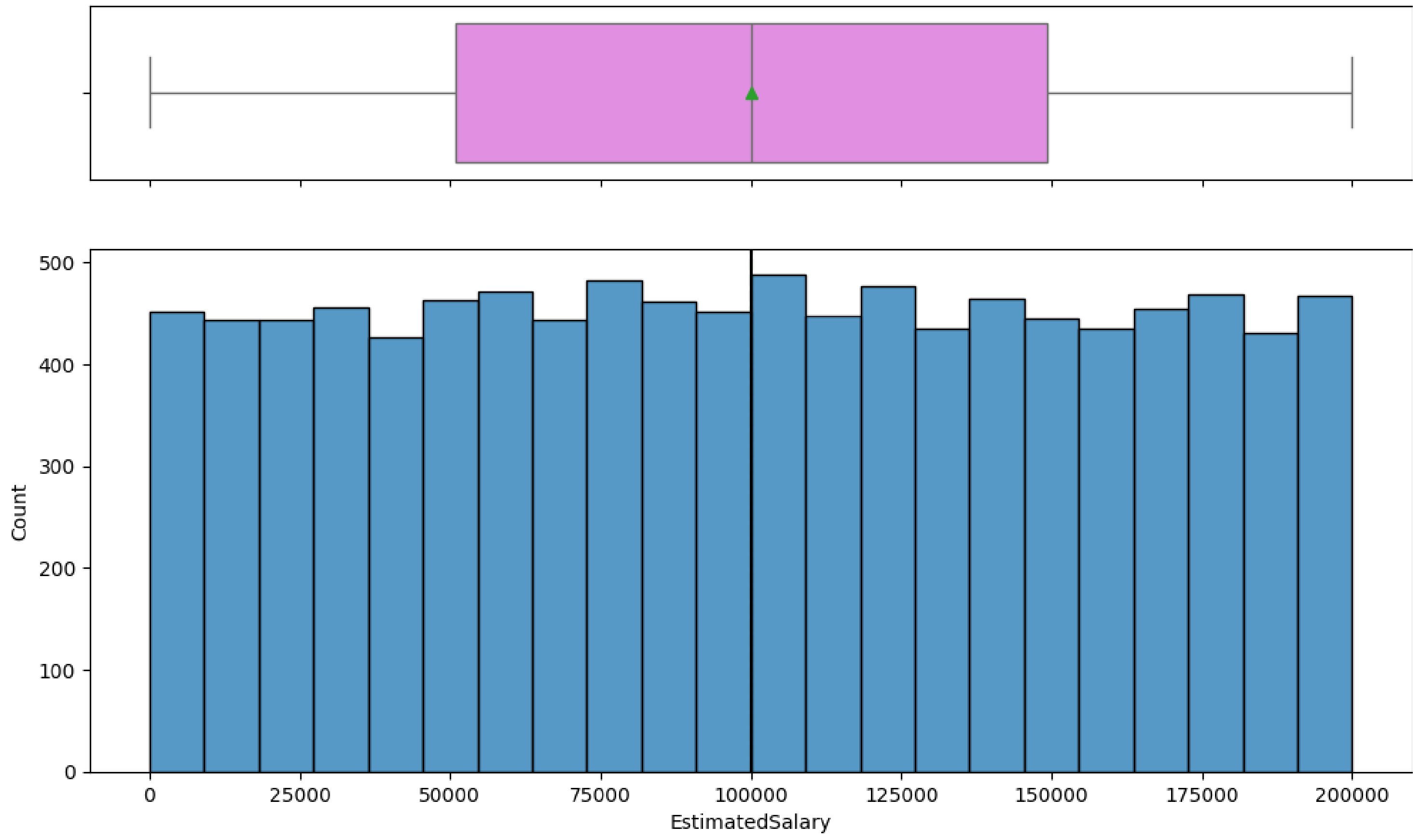
- *In the data set, 79.6% of customers did not churn while 20.4% did. The data set is imbalanced for our target variable. We will balance this using SMOTE in 3 of our 6 NN.*
- *The data set is roughly balanced by gender, with slightly more males*
- *A large majority of customers only have 1-2 products from the bank. A very small percentage have 3+*
- *70.5% of customers in the data set had credit cards.*

EDA Results – Univariate Analysis - Balance



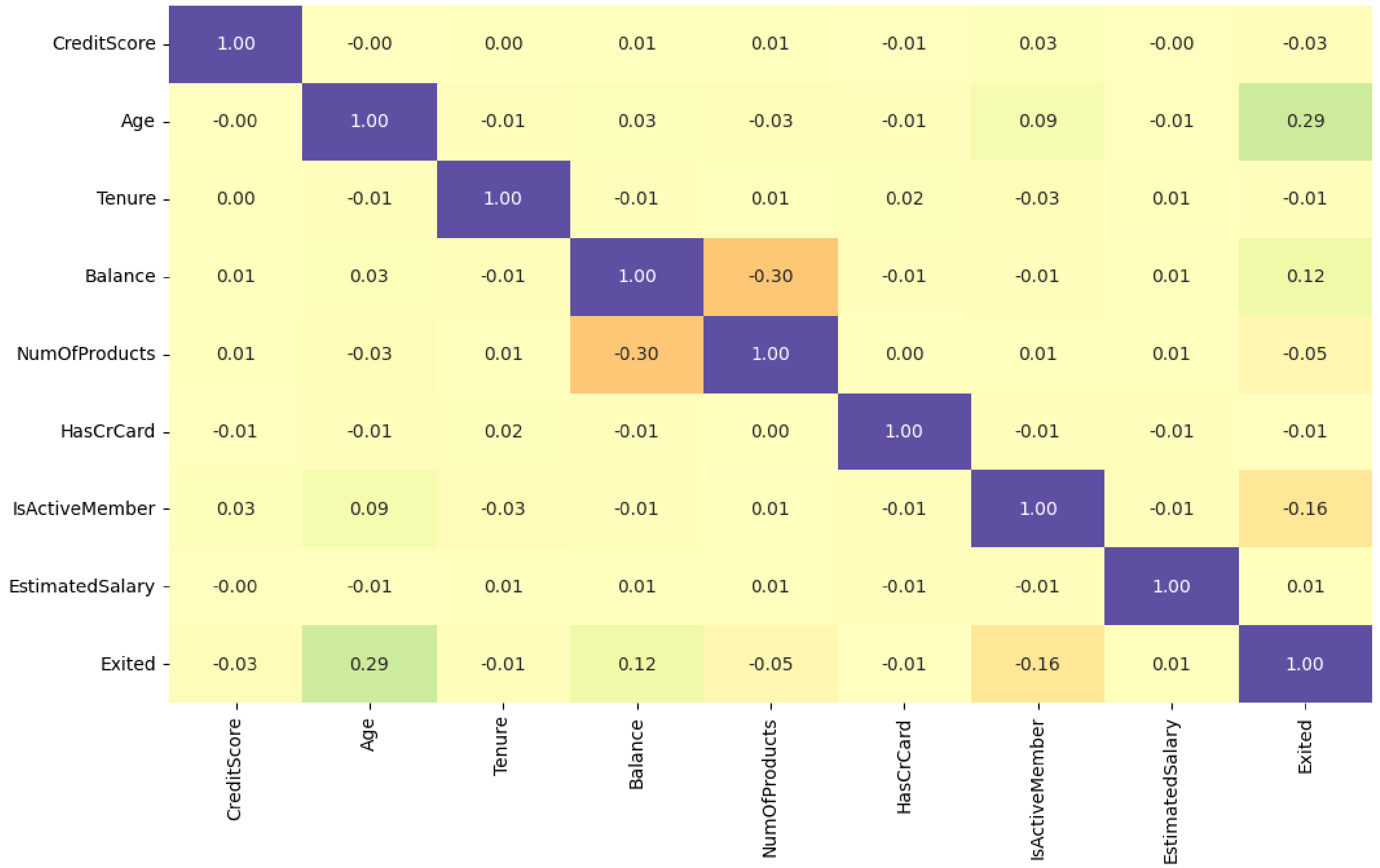
- *The balance almost looks normally distributed if it weren't for the very high number of members that have a zero or very small balance close to zero, as seen on the right-most bin in the histogram.*

EDA Results – Univariate Analysis – Estimated Salary



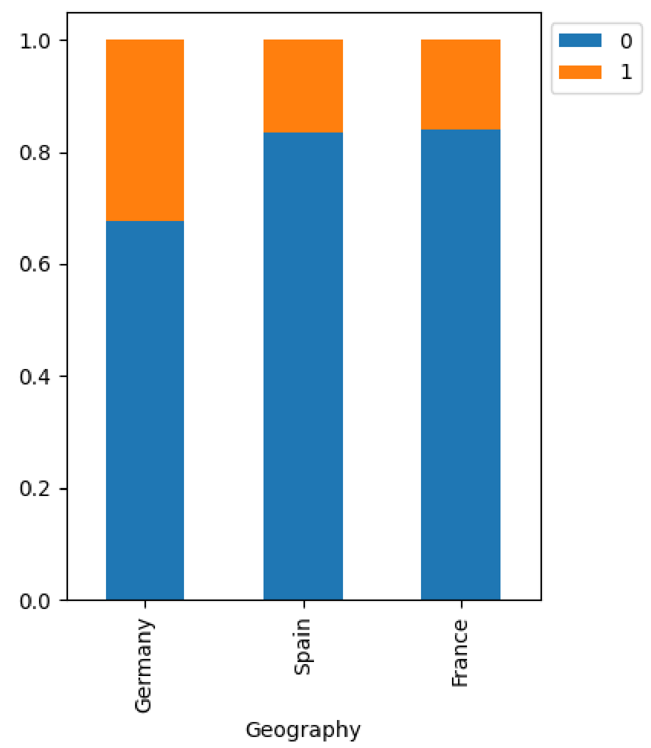
- There is a fairly even distribution among estimated salary range.

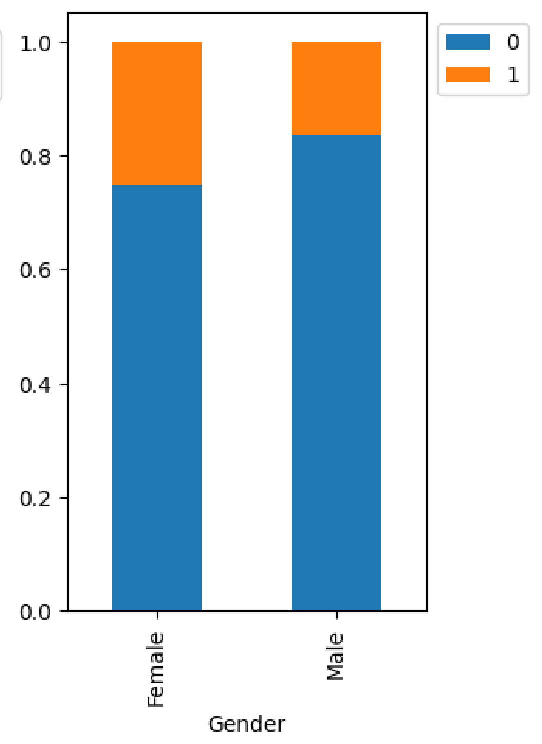
EDA Results – Bivariate Analysis - Heatmap

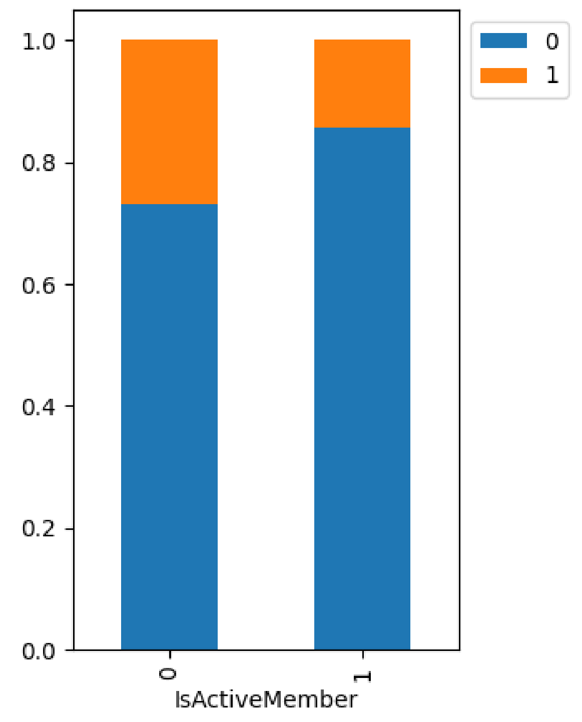


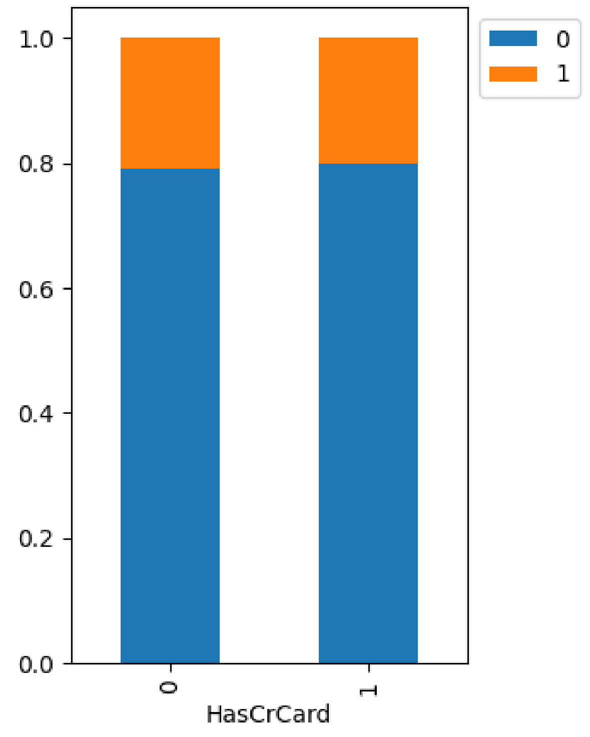
- *Examining the correlation heatmap, we can see that account balance and number of products are slightly negatively correlated and Age seems to be the most positively correlated value relating to the target variable Exited.*

EDA Results – Bivariate Analysis – Geog, Gender,



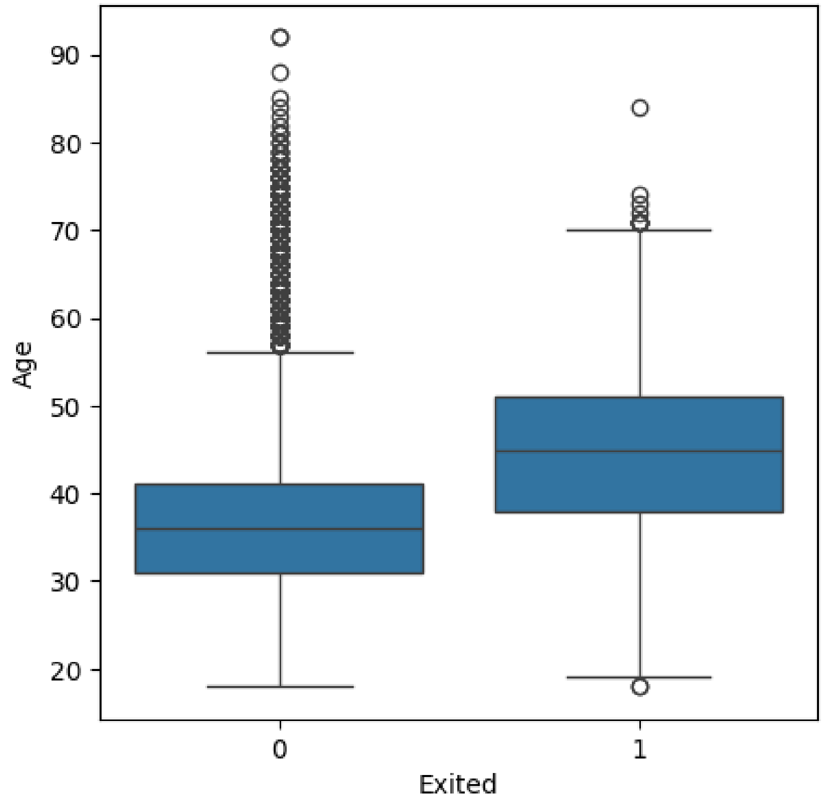


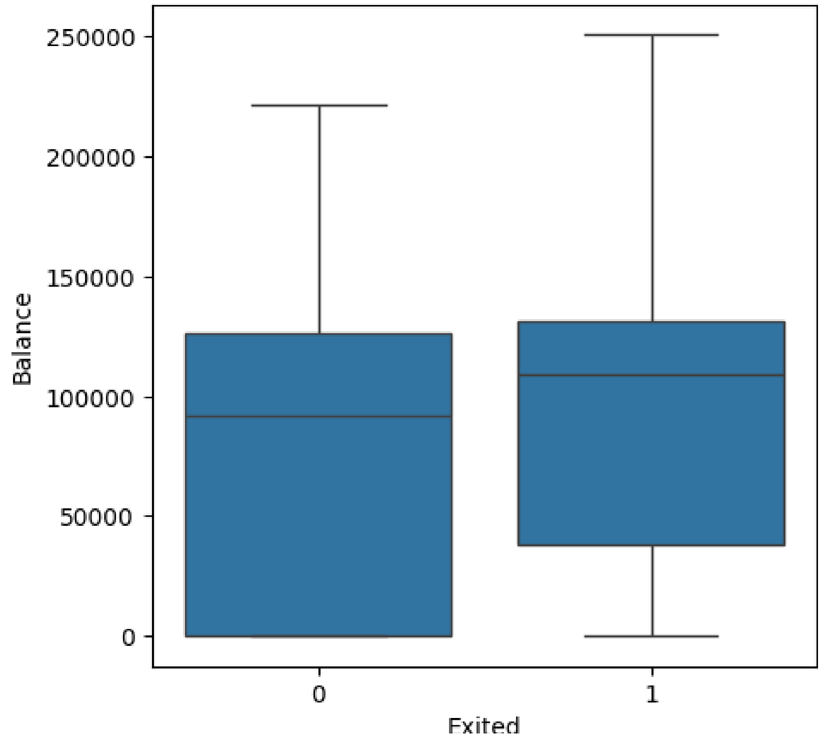




While we can observe slight difference between the categorical variables, it seems like roughly equal distributions between churners and non-churners when you look at Geography, Gender, if they are active members, or if they have a credit card or not.

EDA Results – Bivariate Analysis – Age & Balance





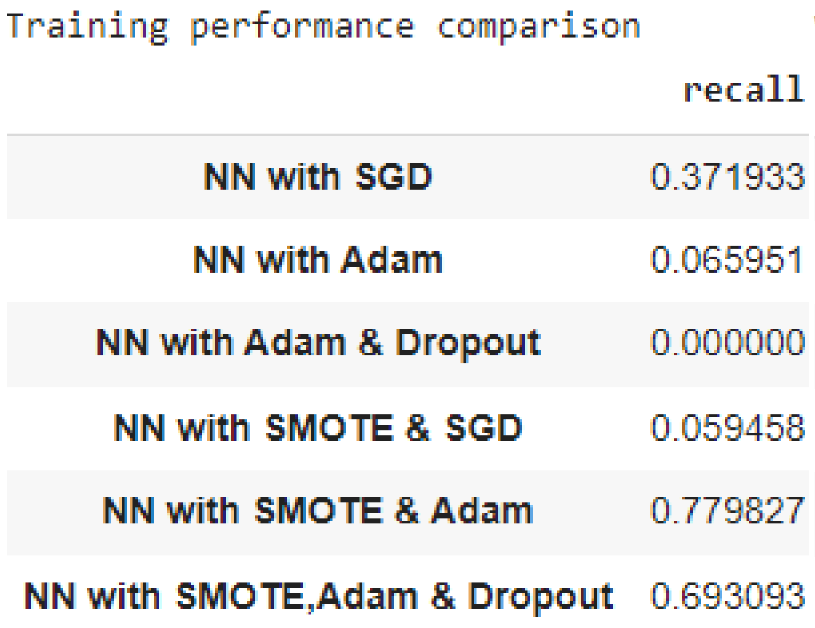
- *As we noted while looking at the heatmap, there is a fairly large difference in distribution of churners versus non churners in age. The median age to churn is about 10 years higher, on average, than those that do not churn, but there are a large number of older outliers among the non-churners as well.*
- *Interestingly, those with lower balances were more likely not to churn compared to those with higher balances.*

Data Preprocessing

- We observed that there were no duplicate values or missing values, but we removed Column Number, Customer number, and Surname since they were unique values and did not contribute any valuable data.
- While there are a large number of outliers in age, this is valuable data that we need to capture for our analysis so we do not treat.
- We encoded Geography and Gender column so that we can use it in the neural network.
- Of the 10,000 rows in the data set, we used 20% for the test set, 64% for the training set, and 16% for the validation set.
- For all neural networks, we chose epochs at 50 and a batch size of 6400 (which is equal to the size of the test set) since computational efficiency isn't an issue with such a small dataset.
- We used StandardScaler() on CreditScore, Age, Tenure, Balance, And Estimated Salary range variables to normalize the distributions.
- For this business scenario, like other churning problems, we would like to use the recall metric to judge a neural networks performance since we would like to minimize false negatives, which in this case would be predicting that a customer will NOT churn, when they actually would.

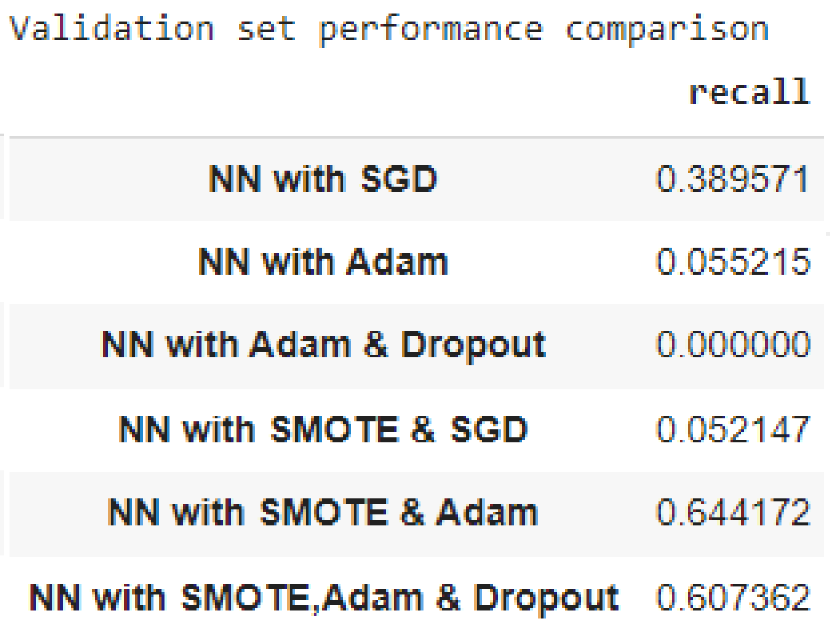
Model Performance Summary

- For all networks, we used the full 6400 test set as the batch_size and 50 epochs. We used sigmoid activation function for all output layers and ReLU activation functions for all hidden layers. When dropout technique is used, we specified .5 value. Our goal was to maximize the recall for performance to minimize misclassifying actual churners as non-churners.



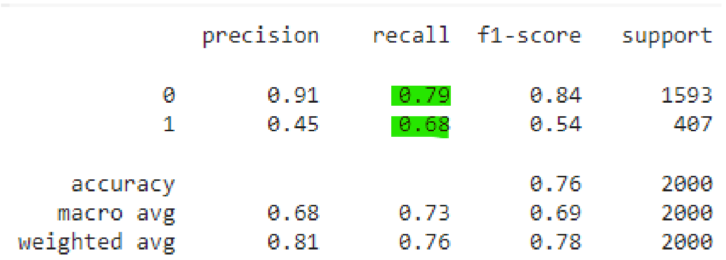
Training performance comparison

	recall
NN with SGD	0.371933
NN with Adam	0.065951
NN with Adam & Dropout	0.000000
NN with SMOTE & SGD	0.059458
NN with SMOTE & Adam	0.779827
NN with SMOTE,Adam & Dropout	0.693093



Validation set performance comparison

	recall
NN with SGD	0.389571
NN with Adam	0.055215
NN with Adam & Dropout	0.000000
NN with SMOTE & SGD	0.052147
NN with SMOTE & Adam	0.644172
NN with SMOTE,Adam & Dropout	0.607362



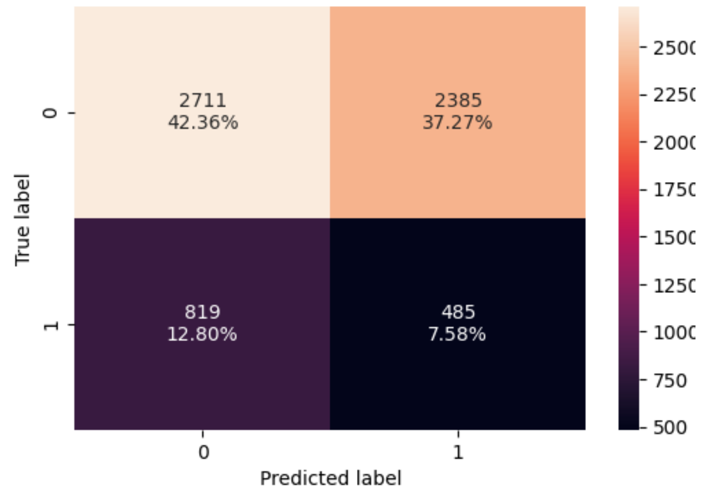
```
              precision    recall  f1-score   support

           0       0.91      0.79      0.84      1593
           1       0.45      0.68      0.54       407

    accuracy                           0.76      2000
   macro avg       0.68      0.73      0.69      2000
weighted avg       0.81      0.76      0.78      2000
```

Based on the results that we saw, it appears that NN with SMOTE & Adam optimizer had the best performance on both the training set and the Validation set. So it would be the best to use among those that we built. Fitting this to the test set produces a .68 recall score.

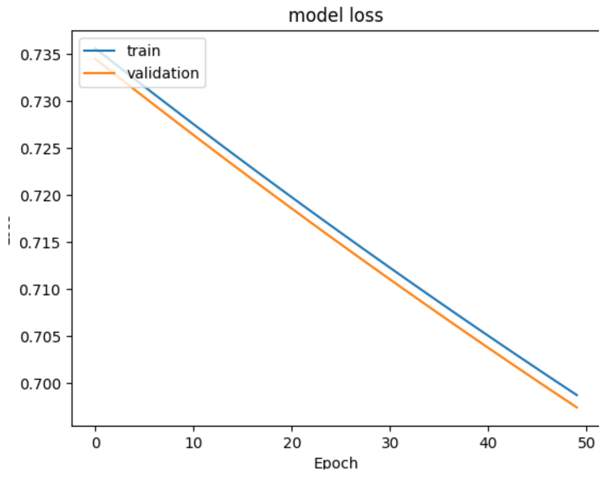
APPENDIX

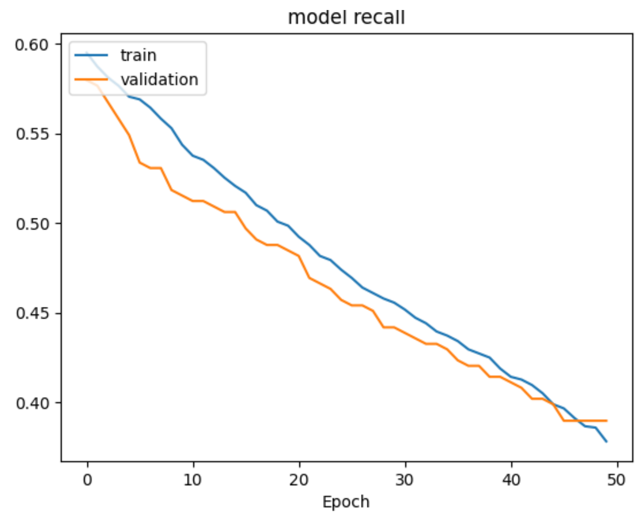
Neural Network with Stochastic Gradient Descent (SGD)

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param
=================================================================
 dense (Dense)               (None, 64)                768

 dense_1 (Dense)             (None, 64)                4160

 dense_2 (Dense)             (None, 1)                 65

=================================================================
Total params: 4993 (19.50 KB)
Trainable params: 4993 (19.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

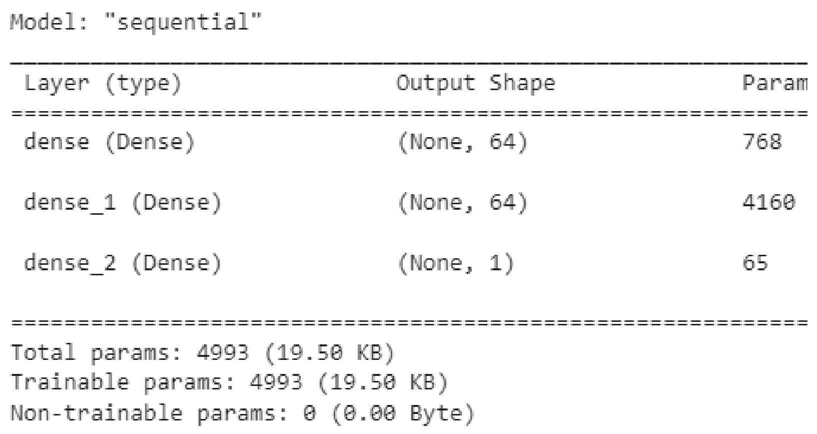


- In this model, the loss function in both training and validation set decline linearly.
- The recall declines somewhat linearly as well.

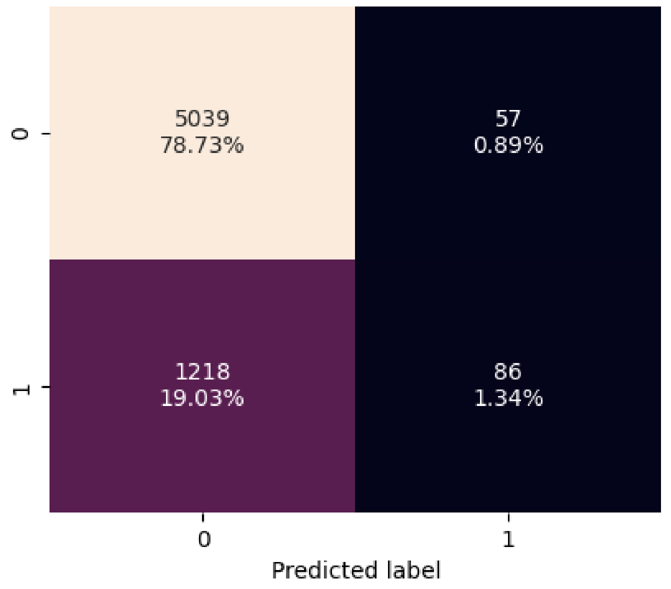




Neural Network with Adam Optimizer

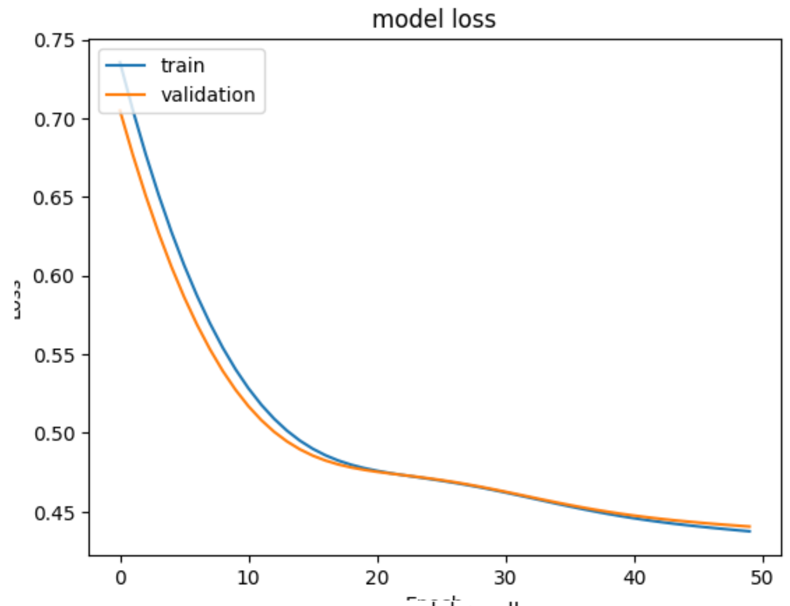


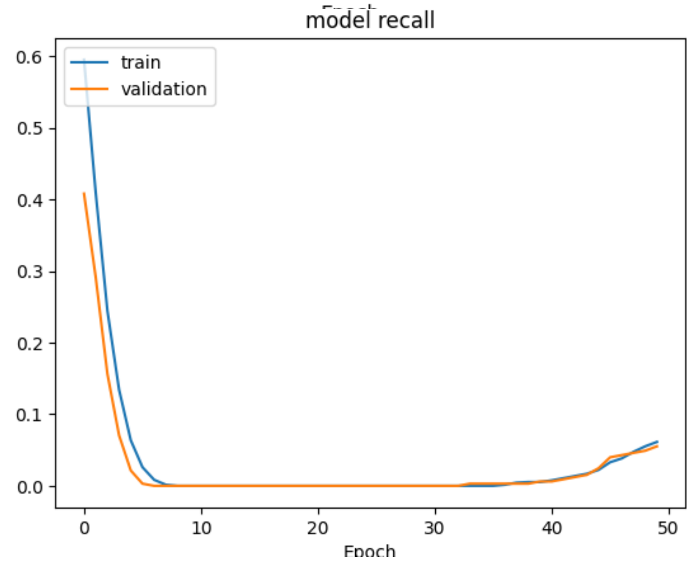
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param
=================================================================
 dense (Dense)               (None, 64)                768

 dense_1 (Dense)             (None, 64)                4160

 dense_2 (Dense)             (None, 1)                 65

=================================================================
Total params: 4993 (19.50 KB)
Trainable params: 4993 (19.50 KB)
Non-trainable params: 0 (0.00 Byte)
```



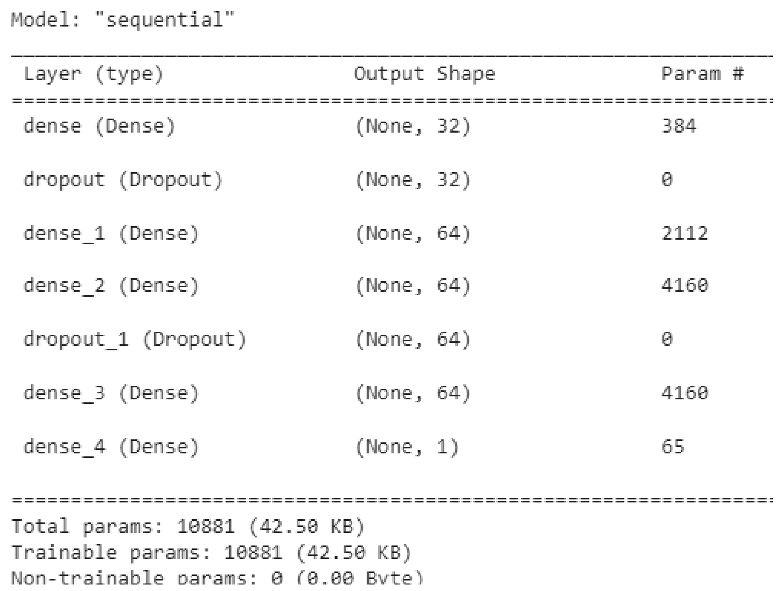
- In this model, the loss function in both training and validation decline and even out at about 30 epochs.

- The recall in both training and validation set follow each other to zero very quickly and stay there, only gradually ascending as the model keeps running.



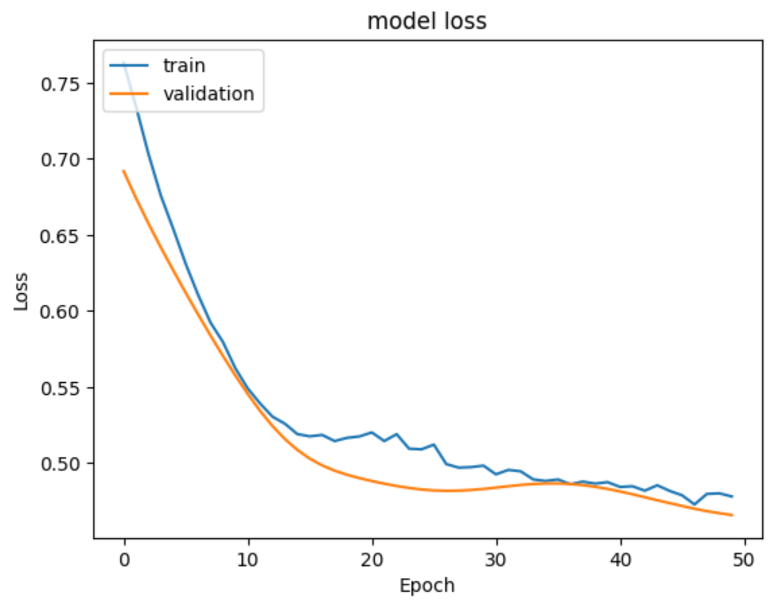


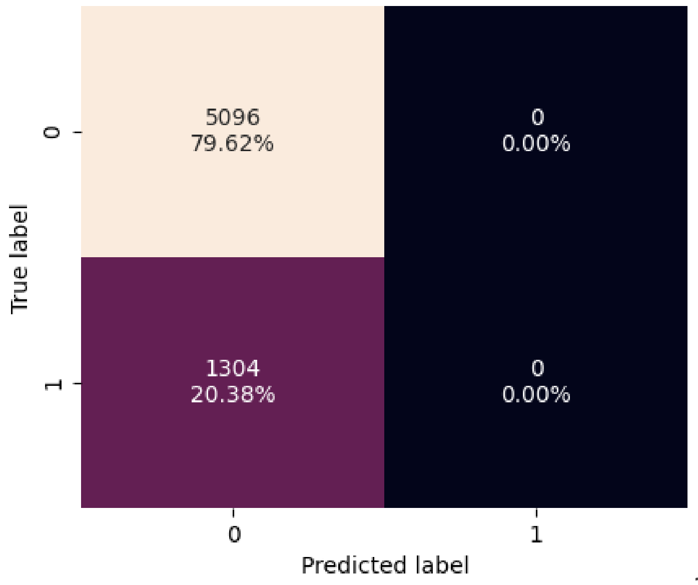
Neural Network with Adam Optimizer & Dropout (.5)

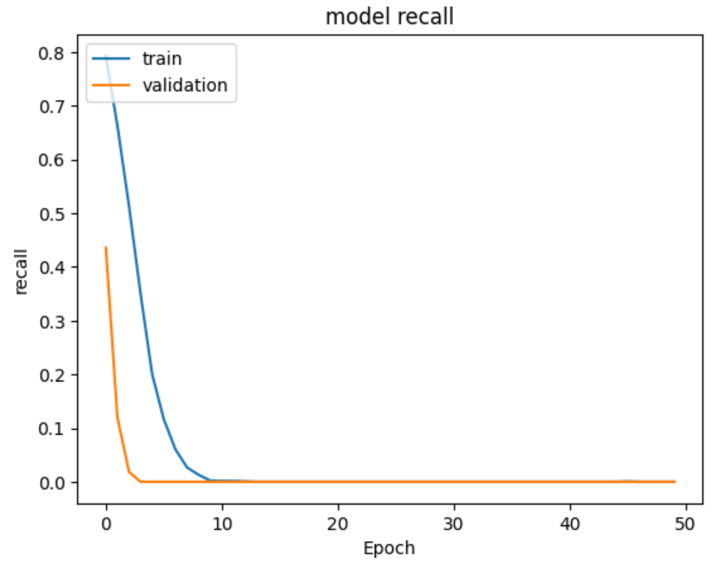


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 32)                384

 dropout (Dropout)           (None, 32)                0

 dense_1 (Dense)             (None, 64)                2112

 dense_2 (Dense)             (None, 64)                4160

 dropout_1 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 64)                4160

 dense_4 (Dense)             (None, 1)                 65

=================================================================
Total params: 10881 (42.50 KB)
Trainable params: 10881 (42.50 KB)
Non-trainable params: 0 (0.00 Byte)
```

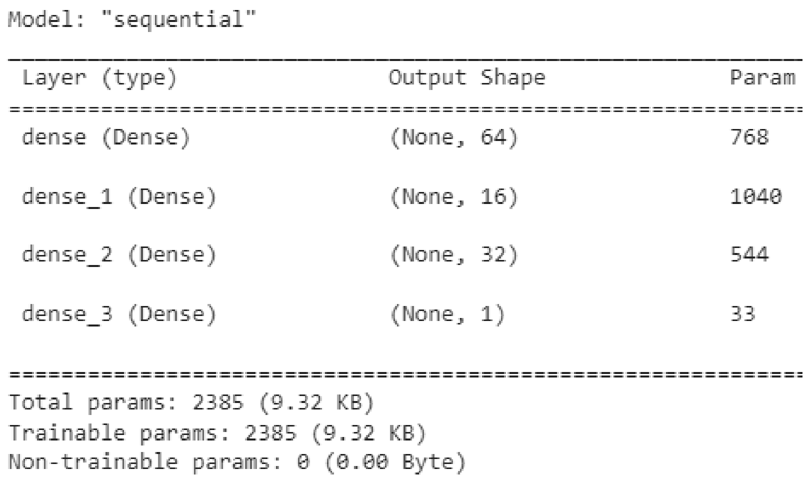
In this model, the model loss function performs OK, but the recall is very poor and quickly drops to zero.



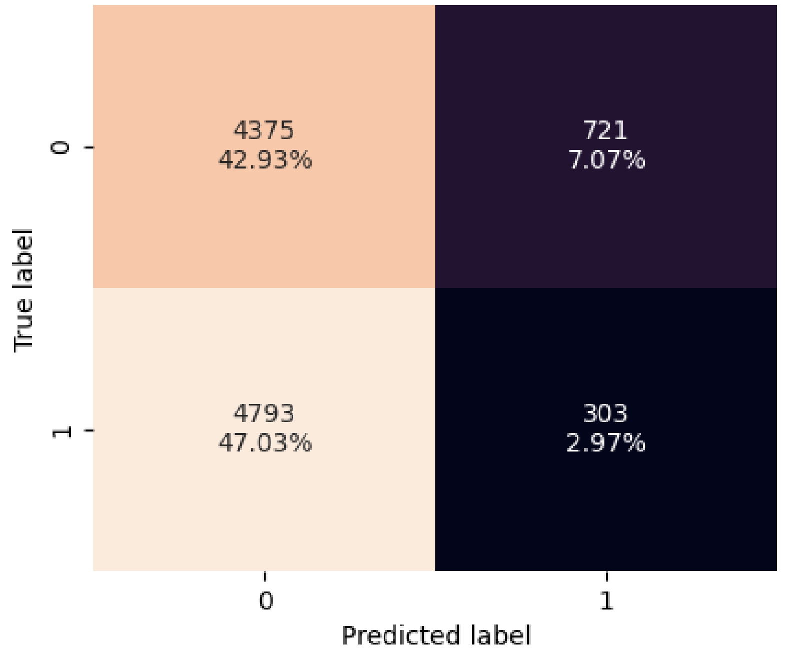




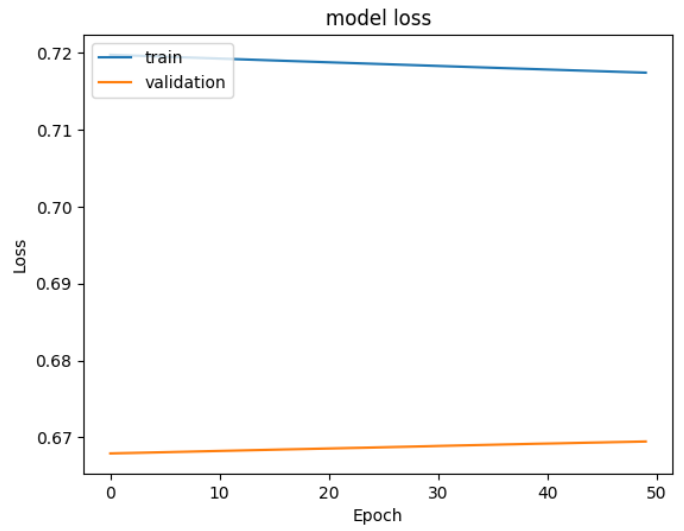
Neural Network with SMOTE balanced data & SGD



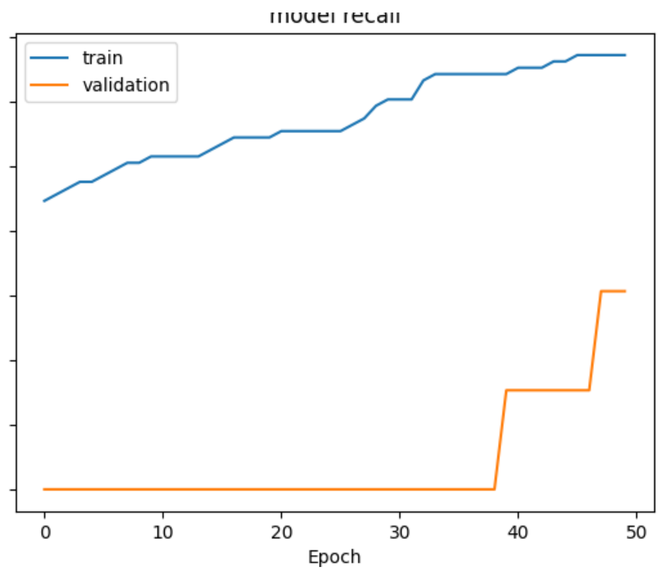
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param
=================================================================
 dense (Dense)               (None, 64)                768

 dense_1 (Dense)             (None, 16)                1040

 dense_2 (Dense)             (None, 32)                544

 dense_3 (Dense)             (None, 1)                 33

=================================================================
Total params: 2385 (9.32 KB)
Trainable params: 2385 (9.32 KB)
Non-trainable params: 0 (0.00 Byte)
```



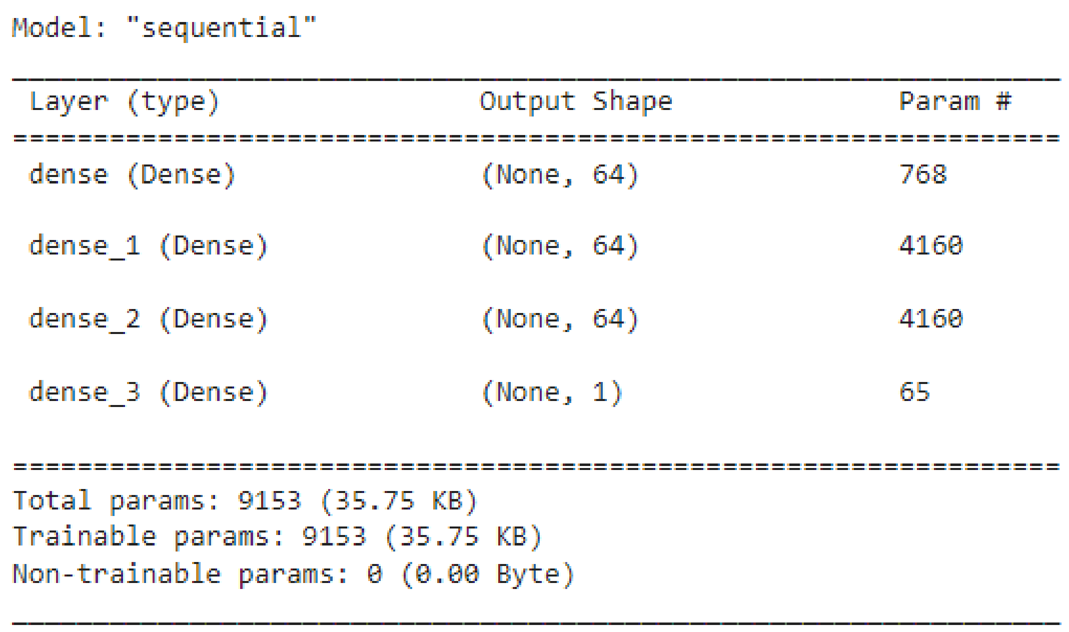
- Using this model, the loss function is very low in the training set and very high in the training set and do not converge.



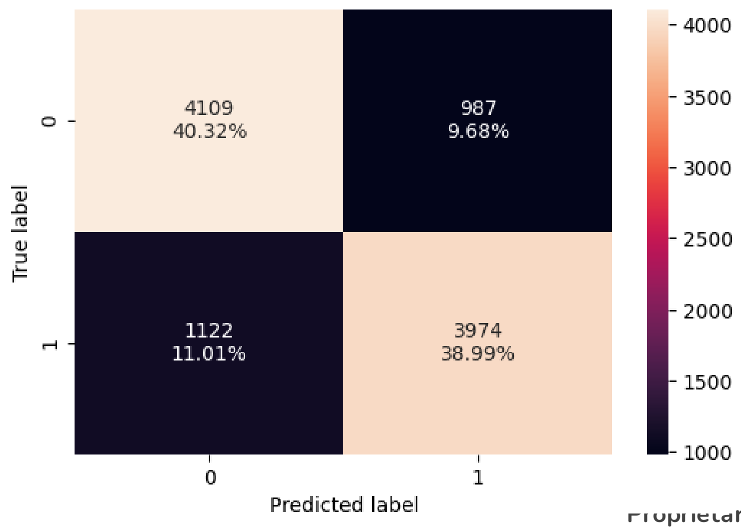
- The recall in the train set is very high, bt the validation set doesn't begin ascending in a stair step fashion until after 40 epochs.



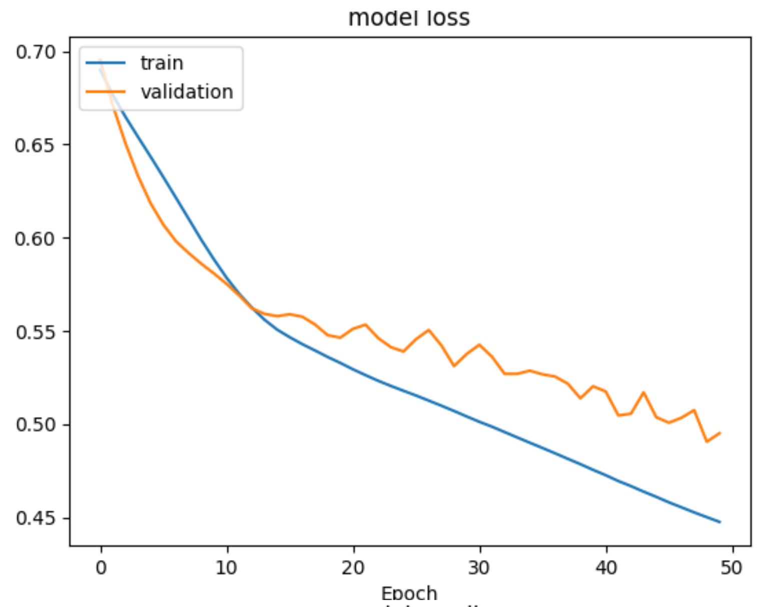
Neural Network with SMOTE balanced data & Adam optimizer

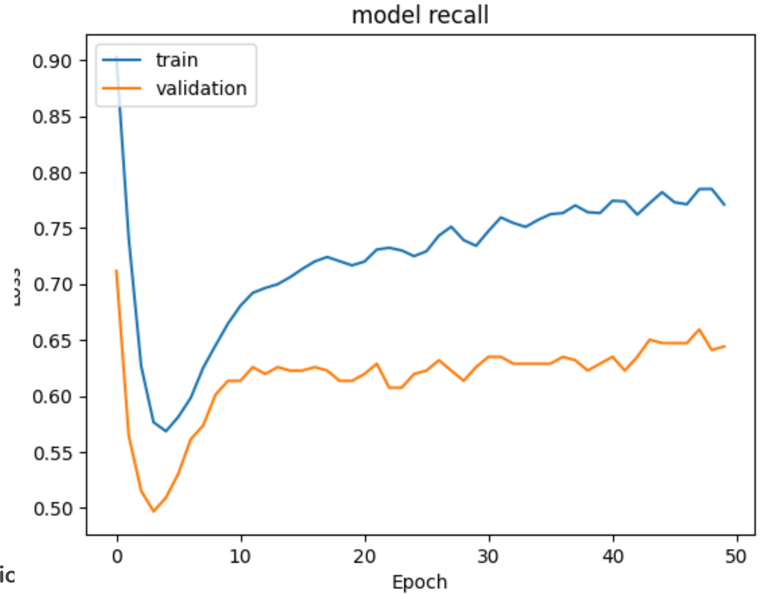


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 64)                768

 dense_1 (Dense)             (None, 64)                4160

 dense_2 (Dense)             (None, 64)                4160

 dense_3 (Dense)             (None, 1)                 65

=================================================================
Total params: 9153 (35.75 KB)
Trainable params: 9153 (35.75 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```



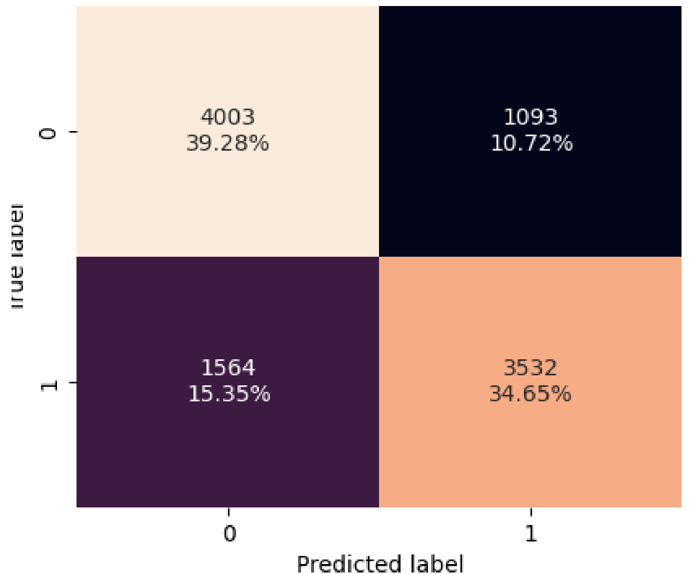
- This model has a slight oscillation in the validation set as the network runs.
- The model recall is actually quite good on both test and validation set.
- This model is our best bet for production compared to the rest



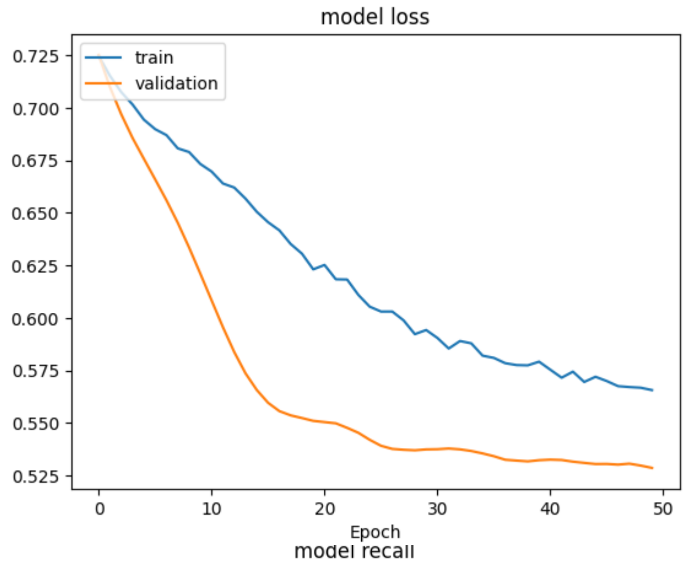


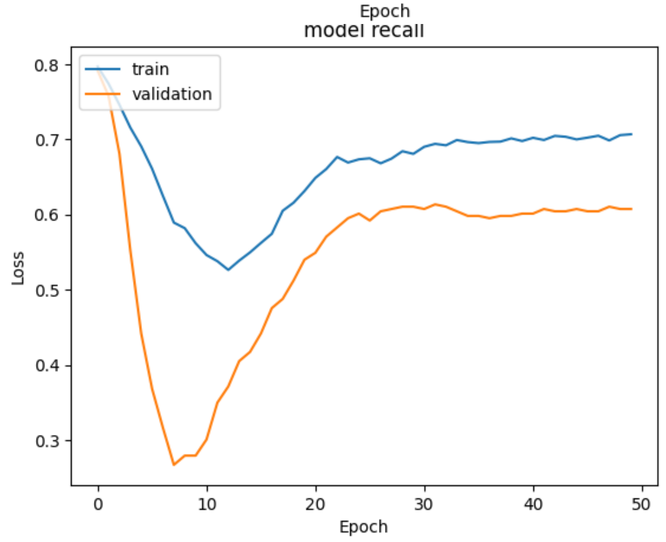
Neural Network with SMOTE & Adam w/ dropout (.5)

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 64)                768

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 64)                4160

 dropout_1 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 8)                 520

 dense_3 (Dense)             (None, 1)                 9

=================================================================
Total params: 5457 (21.32 KB)
Trainable params: 5457 (21.32 KB)
Non-trainable params: 0 (0.00 Byte)
```



- This model has a smoother curve than any of the non-balanced networks we have created so far.
- The model recall is actually quite good on both test and validation set, but not as good as using SMOTE without the dropout feature.





Great Learning

Happy Learning !