DEEP LEARNING AND REINFORCEMENT LEARNING

PIMA DIABETES DATASET

TING CHONG NA
30.05.2023

DATASET

Topic:

Pima Diabetes Dataset

Objective:

Compare and see how different network structures affect the performance, training time, and level of overfitting (or underfitting).

Dataset:

The UCI Pima Diabetes Dataset which has 8 numerical predictors and a binary outcome.

Use a neural network to predict diabetes using the Pima Diabetes Dataset.

y=0	y=1
do not have diabetes	have diabetes

DATASET

	times_pregnant	glucose_tolerance_test	blood_pressure	skin_thickness	insulin	bmi	pedigree_function	age	has_diabetes
626	0	125	68	0	0	24.7	0.206	21	0
639	1	100	74	12	46	19.5	0.149	28	0
52	5	88	66	21	23	24.4	0.342	30	0
487	0	173	78	32	265	46.5	1.159	58	0
436	12	140	85	33	0	37.4	0.244	41	0

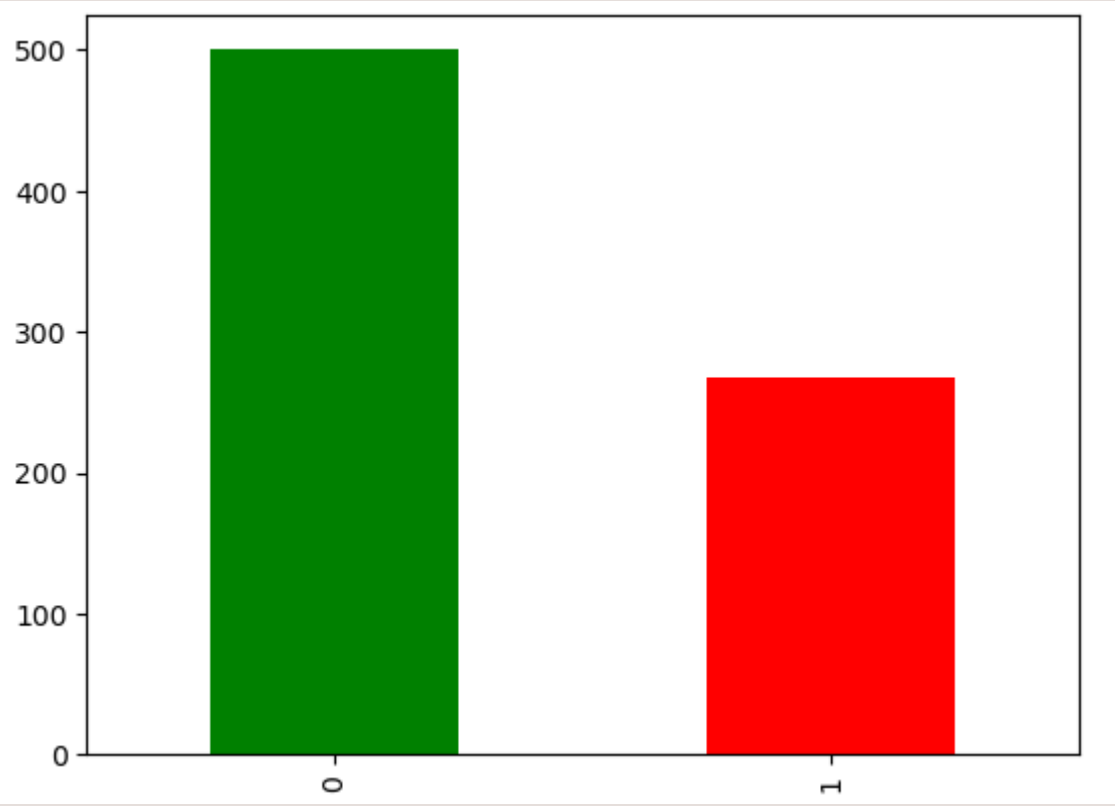
768 observations

9 features

The features available from the dataset are:

1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)

DATA ANALYSIS For all Numeric variables



Imbalanced Class

```
diabetes_df["has_diabetes"].value_counts()
```

```
0    500
1    268
Name: has_diabetes, dtype: int64
```

```
diabetes_df["has_diabetes"].value_counts(normalize=True)
```

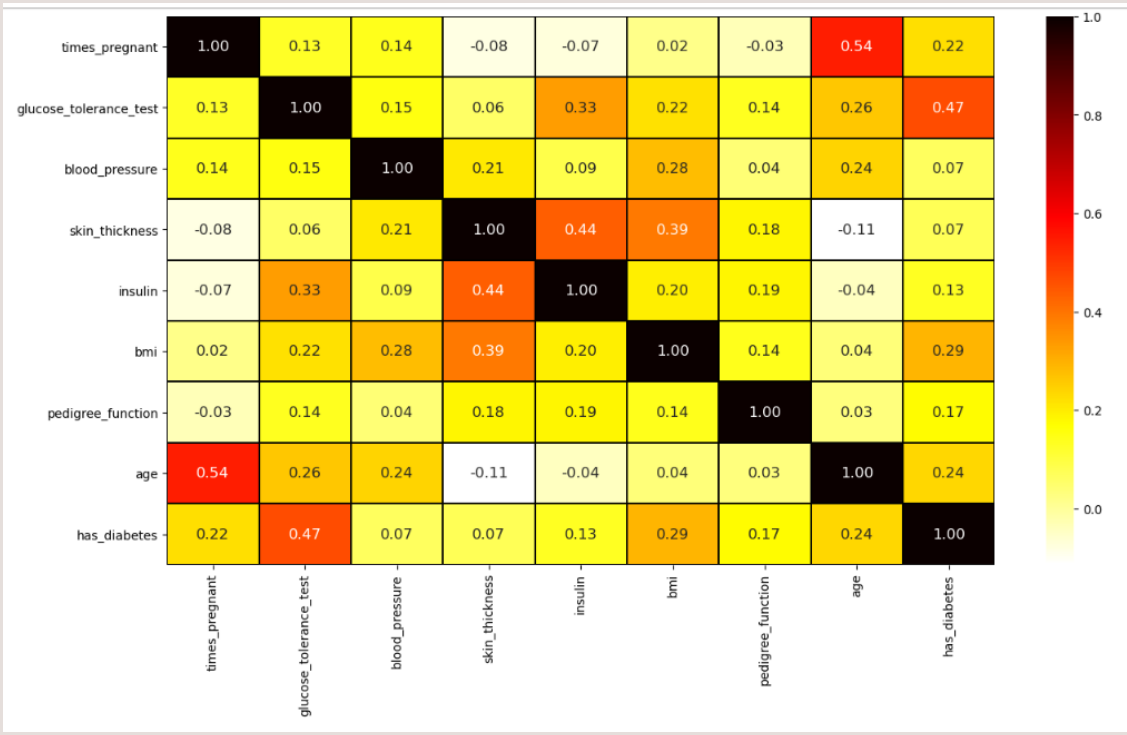
```
0    0.651042
1    0.348958
Name: has_diabetes, dtype: float64
```

35% of the patients have diabetes, while 65% do not

We can get an accuracy of 65% without any model - just declare that no one has diabetes

DATA ANALYSIS For all Numeric variables

Perform Multivariate Analysis by creating a pairplot of all the columns



Using Heatmap to explore which variables are positively or negatively correlated with one another

DATA ANALYSIS For all Numeric variables

Train Test Split

- Split the data to Train and Test (75%, 25%)

ROC-AUC score

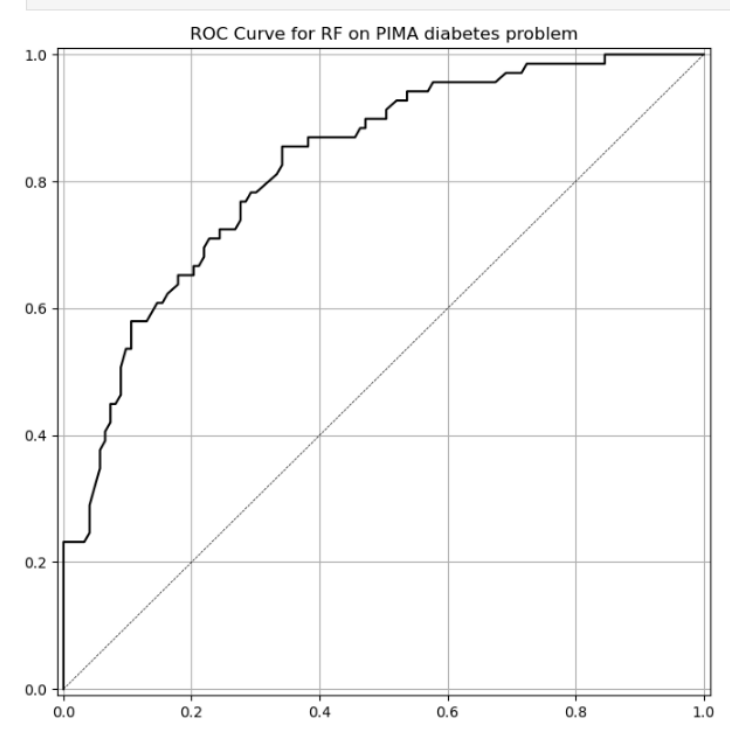
- to evaluate performance of our model

Model

- Random Forest
- Single Hidden Layer Neural Network
- Two Hidden Layer Neural Network

Model: Random Forest

Train a Random Forest model with 200 trees on the training data.



ROC CURVE

```
rf_model = RandomForestClassifier(n_estimators=200)
rf_model.fit(X_train, y_train)
```

```
accuracy is 0.766
roc-auc is 0.826
```

Random Forest

Accuracy: 0.766

ROC: 0.826

Model: Single Hidden Layer Neural Network

Build a Single Hidden Layer Neural Network with 200 epoches.

STANDARDSCALER
Providing Numerical Stability

SEQUENTIAL MODEL
Single Hidden Layer Neural Network

COMPILE MODEL
SGD(lr=.003),accuracy, binary_crossentropy

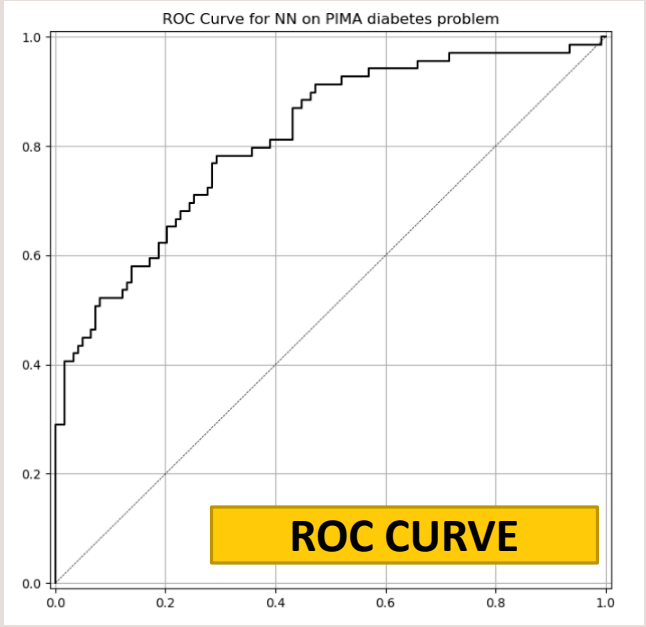
ROC CURVE
Model Performance

Training Loss and Validation Loss

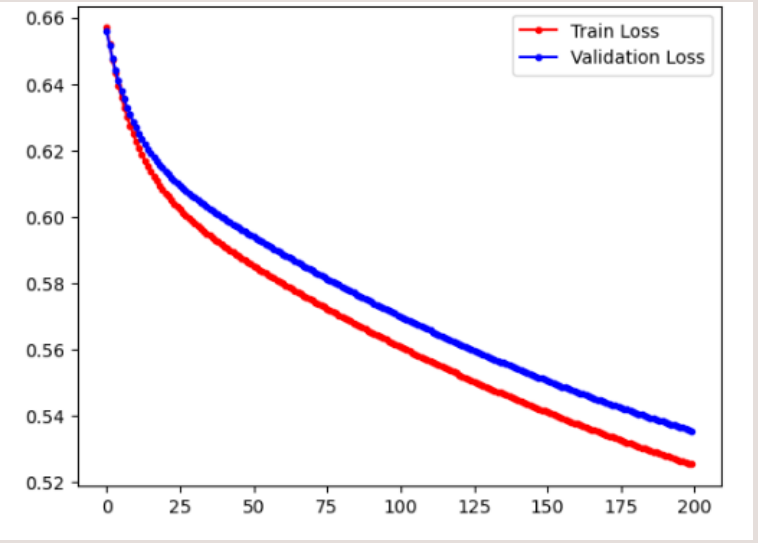
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 12)                108
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 13
=================================================================
Total params: 121
Trainable params: 121
Non-trainable params: 0
```

Sequential Model

```
accuracy is 0.766
roc-auc is 0.816
```



ROC CURVE



Train Loss & Val Loss

Model: Compare results

Random Forest	Accuracy: 0.766	ROC: 0.826
Single Hidden Layer NN	Accuracy: 0.766	ROC: 0.816

In **Single Hidden Layer Neural Network** model

There are some variation in exact numbers due to randomness, but get results in the same ballpark as the **Random Forest**:

- between 75% and 85% Accuracy
- between 80%and 90% for ROC-AUC

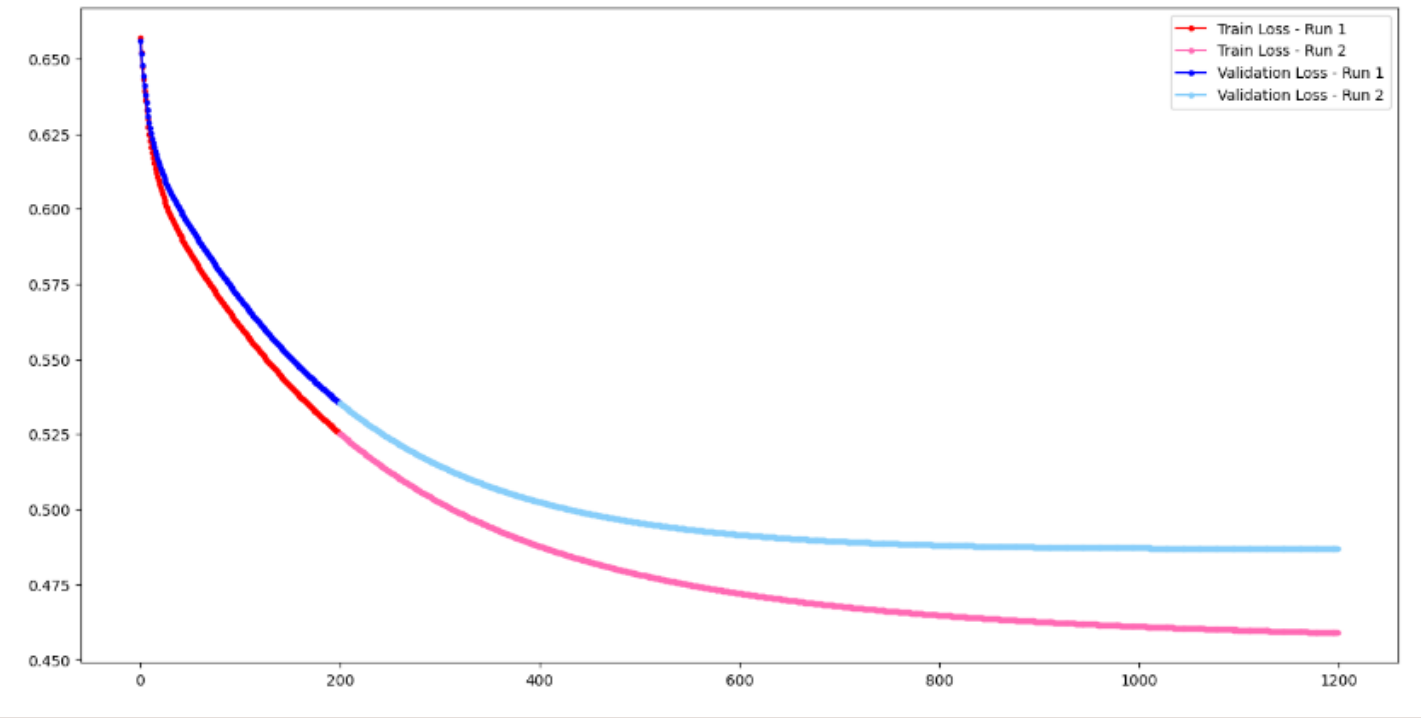
The losses are still going down on both the training set and the validation set. This suggests that the model might benefit from further training.

NEXT: Train for 1000 more epochs.

Model: Single Hidden Layer Neural Network

Train for 1000 more epochs.

```
run_hist_1b = model_1.fit(X_train_norm, y_train, validation_data=(X_test_norm, y_test), epochs=1000)
```



Train Loss & Val Loss

While the training loss is still going down, it looks like the validation loss has stabilized (or even gotten worse!).

This suggests that our network will **NOT** benefit from further training.

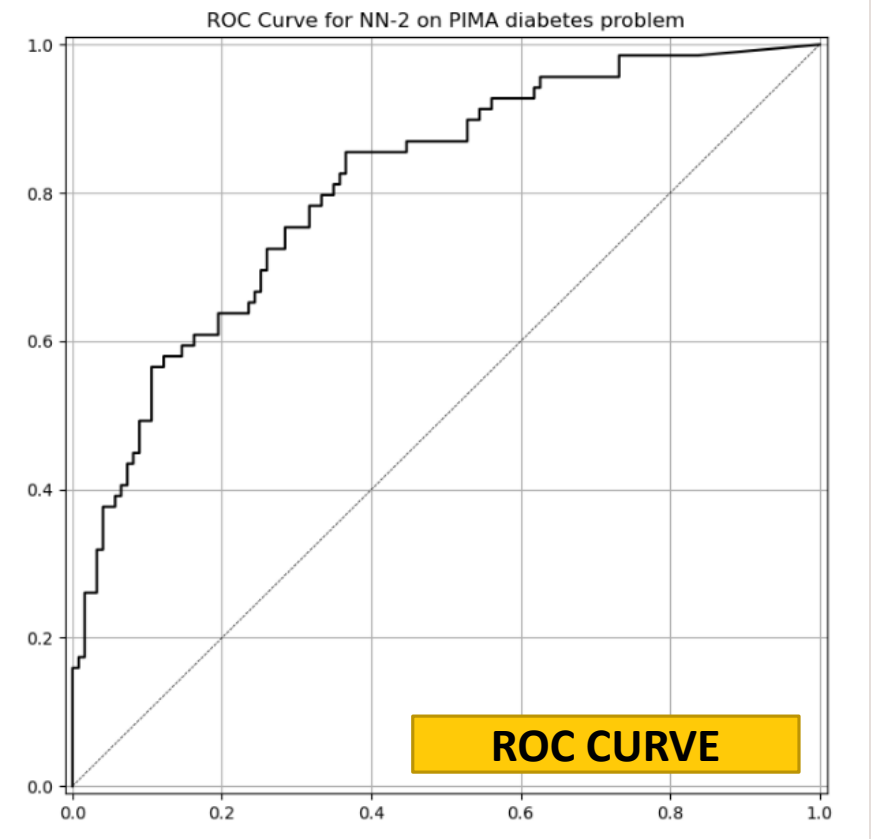
Model: 2 Hidden Layer Neural Network

Build 2 Hidden Layer Neural Network with 1500 epoches.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 6)                 54
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 42
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 7
=================================================================
Total params: 103
Trainable params: 103
Non-trainable params: 0
_________________________________________________________________
```

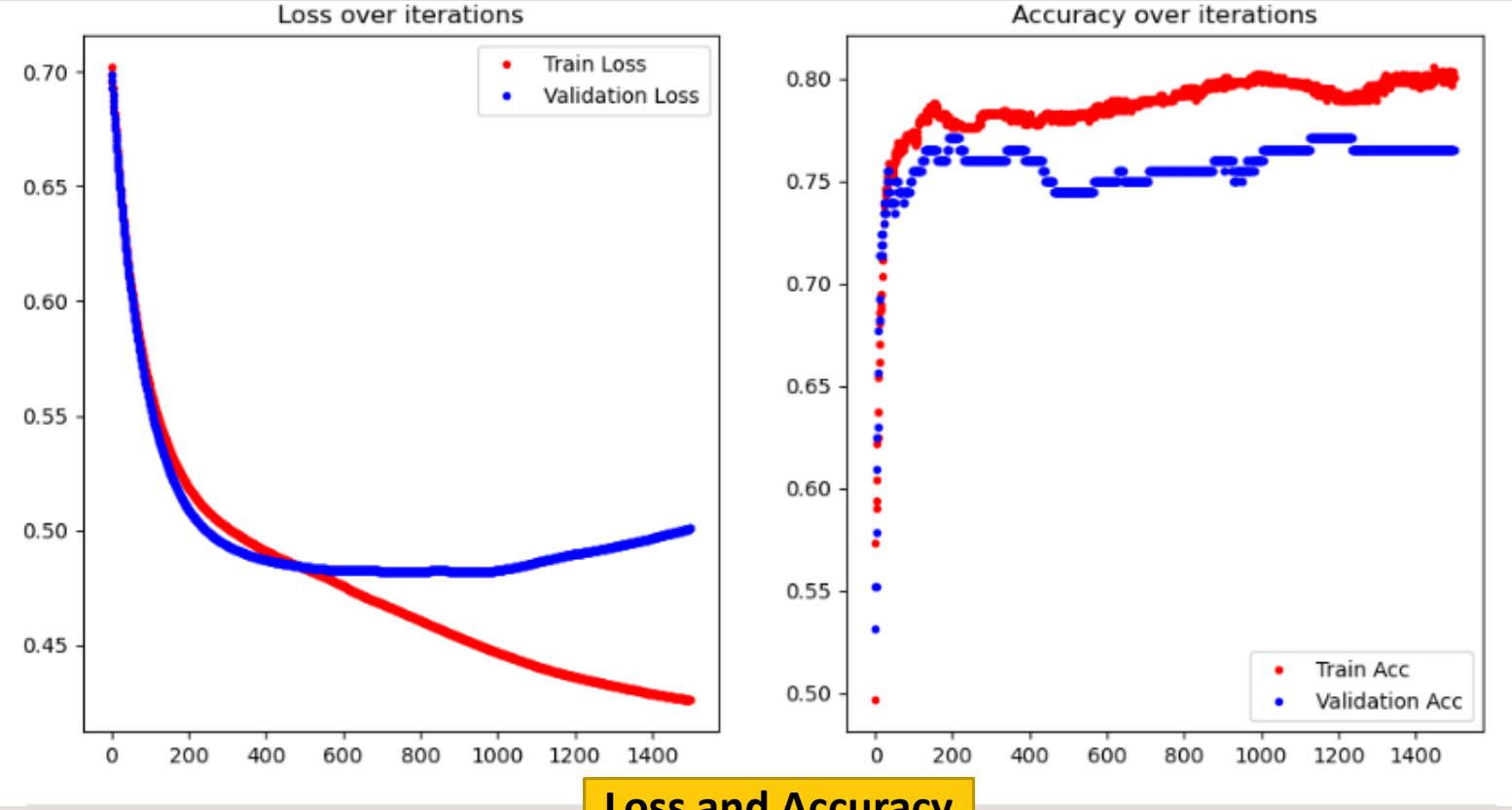
Sequential Model

```
accuracy is 0.766
roc-auc is 0.810
```



Model: 2 Hidden Layer Neural Network

Build 2 Hidden Layer Neural Network with 1500 epoches.



Loss and Accuracy

Conclusion

Random Forest	Accuracy: 0.766	ROC: 0.826
Single Hidden Layer NN	Accuracy: 0.766	ROC: 0.816
Two Hidden Layer NN	Accuracy: 0.766	ROC: 0.810

In conclusion, **Random Forest** seems to be the **best method** as it has the Highest ROC-AUC values.

POSSIBLE FLAW IN MODEL

ROC-AUC metrics

- ROC analysis uses True Positive Rate (TPR or Recall) and False Positive Rate (FPR)
- Precision-Recall analysis exchanges FPR for Precision
- While ROC uses all the cells (TP, FP, TN, FN) of the Confusion Matrix,
  Precision-Recall disregards the True Negatives, which have a high impact on an **imbalanced** problem
- Precision-Recall gives more weight to the minority class (the positive class) than the ROC,
  so **Precision-Recall is more suitable** for this dataset which has imbalanced problems

Regularization Techniques

- **Add more regularization** to model
- L2 (Ridge) regularization, L1 (Lasso) regularization, Dropout, Batch Normalization, Data shuffling