RAIN PREDICTION IN AUSTRALIA

JAN MAKOWIECKI

BARTOSZ PIJET

BIOLOGICALLY INSPIRED ARTIFICIAL INTELLIGENCE - PROJECT

AGENDA

- Project goals
- Data
- Models
- Output analysis
- Conclusions

PROJECT GOALS

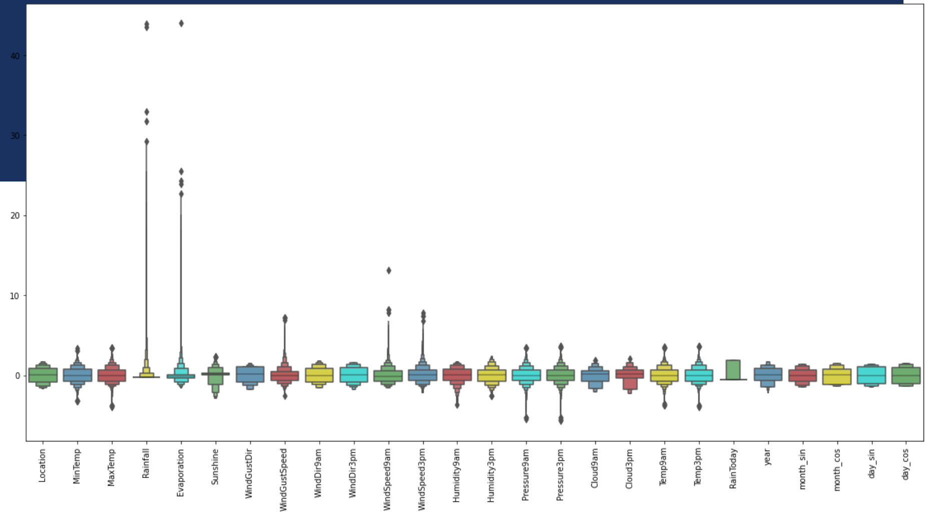
- Predicting next-day rain using a dataset containing 10 years of daily weather ibservations from different locations across Australia

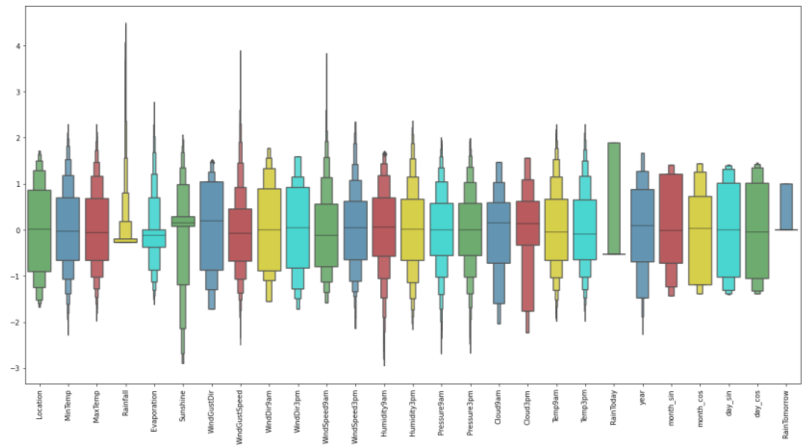
DATA

- The ”Rain in Australia” dataset from kaggle.com was used to train the model
- The dataset contains 145460 entries - about 10 years of daily weather observations from different locations across Australia. Observations were drawn from numerous weather stations.
- In this project, I will use this data to predict whether or not it will rain the next day. There are 23 attributes including the target variable "RainTomorrow", indicating whether or not it will rain the next day or not.

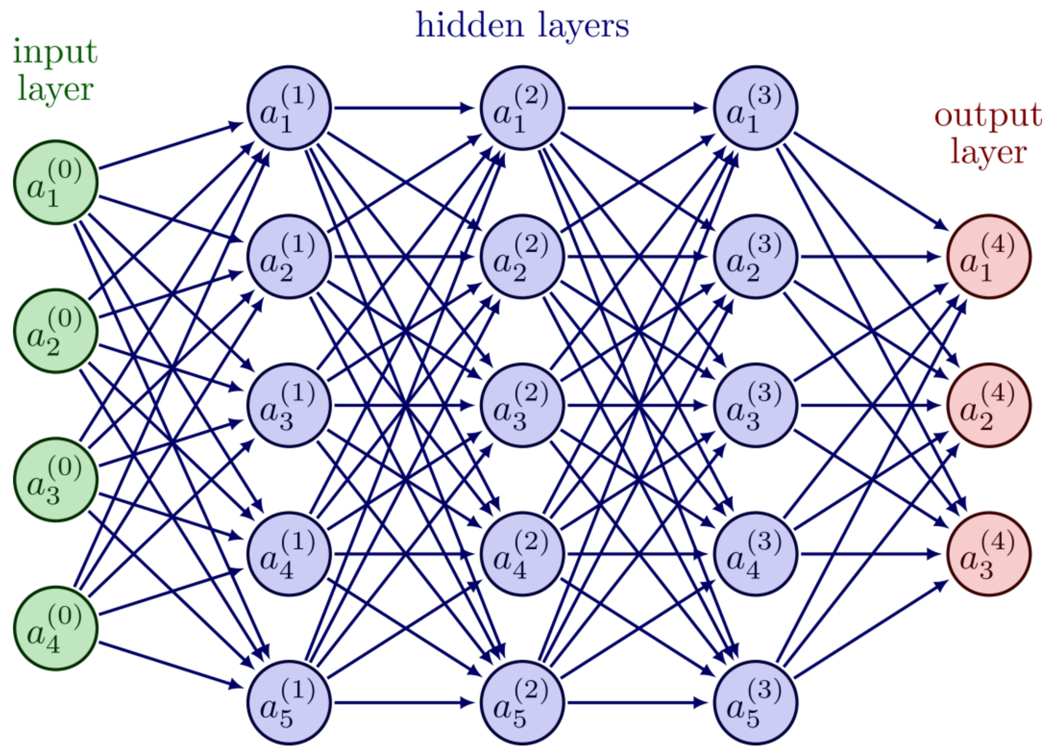
DATA PREPROCESSING

- Before training the model, the collected dataset underwent preprocessing steps to ensure its quality and compatibility with the model
- Preprocessing steps included:
  - Parsing Date into datetime
  - Filling missing values with mode of the column value
  - Label encoding columns with categorical data
  -Performing the scaling of the features
  -Detecting outliers
  -Dropping the outliers based on data analysis





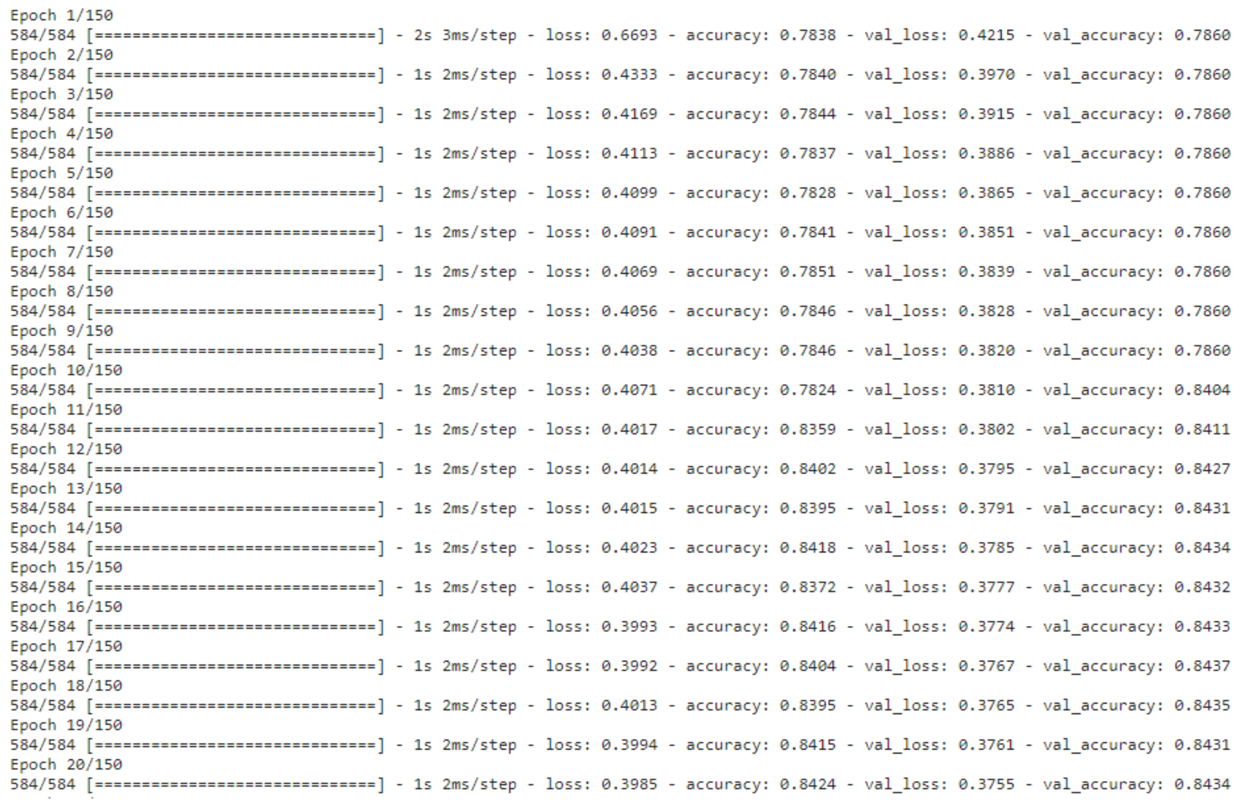
SHORT INTRODUCTION TO NEURAL NETWORKS



Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

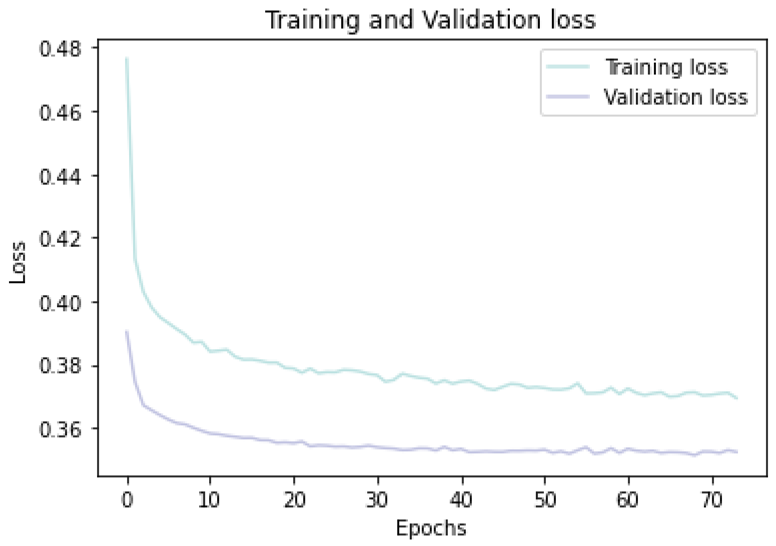
MODEL BUILDING

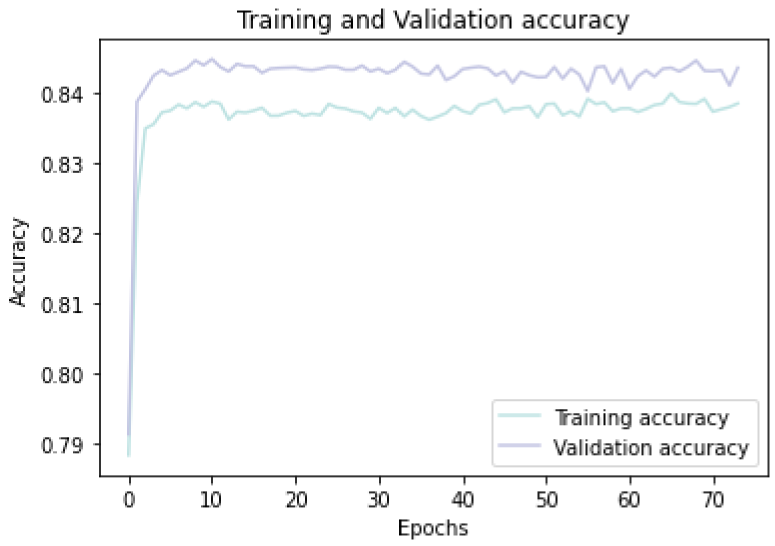
- **In this project, we build an artificial neural network.**
- **Following steps are involved in the model building**
  - Assining X and y the status of attributes and tags
  - Splitting test and training sets
  - Initialising the neural network
  - Defining by adding layers
  - Compiling the neural network
  - Train the neural network

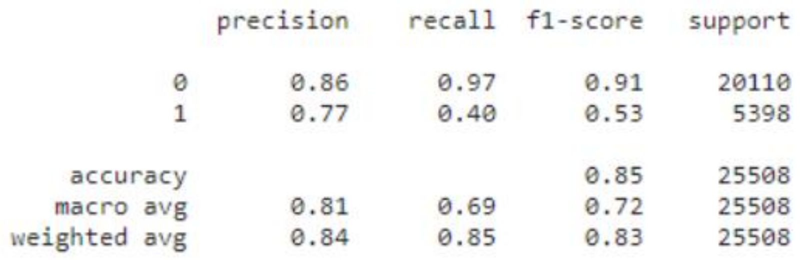


```
Epoch 1/150
584/584 [==============================] - 2s 3ms/step - loss: 0.6693 - accuracy: 0.7838 - val_loss: 0.4215 - val_accuracy: 0.7860
Epoch 2/150
584/584 [==============================] - 1s 2ms/step - loss: 0.4333 - accuracy: 0.7840 - val_loss: 0.3970 - val_accuracy: 0.7860
Epoch 3/150
584/584 [==============================] - 1s 2ms/step - loss: 0.4169 - accuracy: 0.7844 - val_loss: 0.3915 - val_accuracy: 0.7860
Epoch 4/150
584/584 [==============================] - 1s 2ms/step - loss: 0.4113 - accuracy: 0.7837 - val_loss: 0.3886 - val_accuracy: 0.7860
Epoch 5/150
584/584 [==============================] - 1s 2ms/step - loss: 0.4099 - accuracy: 0.7828 - val_loss: 0.3865 - val_accuracy: 0.7860
Epoch 6/150
584/584 [==============================] - 1s 2ms/step - loss: 0.4091 - accuracy: 0.7841 - val_loss: 0.3851 - val_accuracy: 0.7860
Epoch 7/150
584/584 [==============================] - 1s 2ms/step - loss: 0.4069 - accuracy: 0.7851 - val_loss: 0.3839 - val_accuracy: 0.7860
Epoch 8/150
584/584 [==============================] - 1s 2ms/step - loss: 0.4056 - accuracy: 0.7846 - val_loss: 0.3828 - val_accuracy: 0.7860
Epoch 9/150
584/584 [==============================] - 1s 2ms/step - loss: 0.4038 - accuracy: 0.7846 - val_loss: 0.3820 - val_accuracy: 0.7860
Epoch 10/150
584/584 [==============================] - 1s 2ms/step - loss: 0.4071 - accuracy: 0.7824 - val_loss: 0.3810 - val_accuracy: 0.8404
Epoch 11/150
584/584 [==============================] - 1s 2ms/step - loss: 0.4017 - accuracy: 0.8359 - val_loss: 0.3802 - val_accuracy: 0.8411
Epoch 12/150
584/584 [==============================] - 1s 2ms/step - loss: 0.4014 - accuracy: 0.8402 - val_loss: 0.3795 - val_accuracy: 0.8427
Epoch 13/150
584/584 [==============================] - 1s 2ms/step - loss: 0.4015 - accuracy: 0.8395 - val_loss: 0.3791 - val_accuracy: 0.8431
Epoch 14/150
584/584 [==============================] - 1s 2ms/step - loss: 0.4023 - accuracy: 0.8418 - val_loss: 0.3785 - val_accuracy: 0.8434
Epoch 15/150
584/584 [==============================] - 1s 2ms/step - loss: 0.4037 - accuracy: 0.8372 - val_loss: 0.3777 - val_accuracy: 0.8432
Epoch 16/150
584/584 [==============================] - 1s 2ms/step - loss: 0.3993 - accuracy: 0.8416 - val_loss: 0.3774 - val_accuracy: 0.8433
Epoch 17/150
584/584 [==============================] - 1s 2ms/step - loss: 0.3992 - accuracy: 0.8404 - val_loss: 0.3767 - val_accuracy: 0.8437
Epoch 18/150
584/584 [==============================] - 1s 2ms/step - loss: 0.4013 - accuracy: 0.8395 - val_loss: 0.3765 - val_accuracy: 0.8435
Epoch 19/150
584/584 [==============================] - 1s 2ms/step - loss: 0.3994 - accuracy: 0.8415 - val_loss: 0.3761 - val_accuracy: 0.8431
Epoch 20/150
584/584 [==============================] - 1s 2ms/step - loss: 0.3985 - accuracy: 0.8424 - val_loss: 0.3755 - val_accuracy: 0.8434
```

BASE MODEL RESULTS



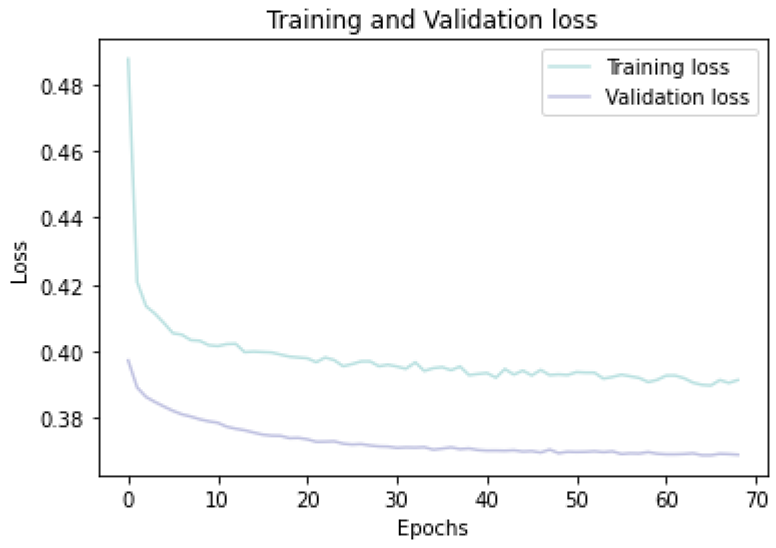


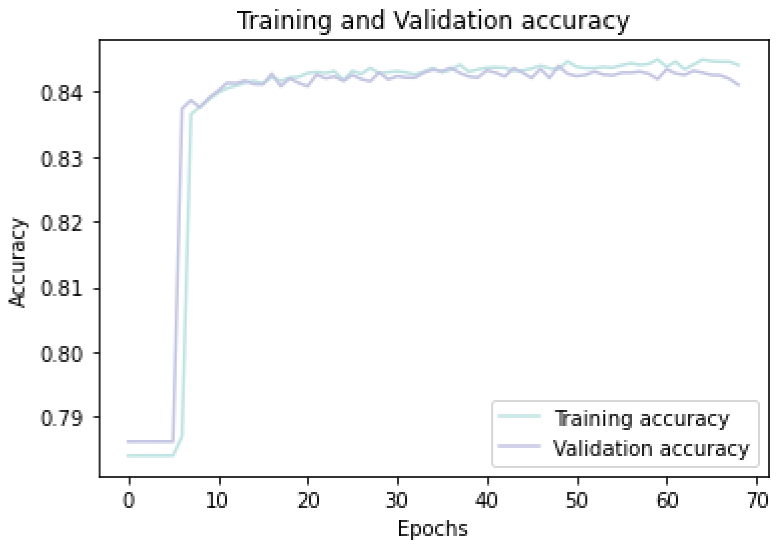


```
              precision    recall  f1-score   support

           0       0.86      0.97      0.91     20110
           1       0.77      0.40      0.53      5398

    accuracy                           0.85     25508
   macro avg       0.81      0.69      0.72     25508
weighted avg       0.84      0.85      0.83     25508
```

OUR MODEL MODIFICATIONS

We tried a variety of changes in model building to see if we can make it better e.g.:
Changing testing size split to 33%.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 42)

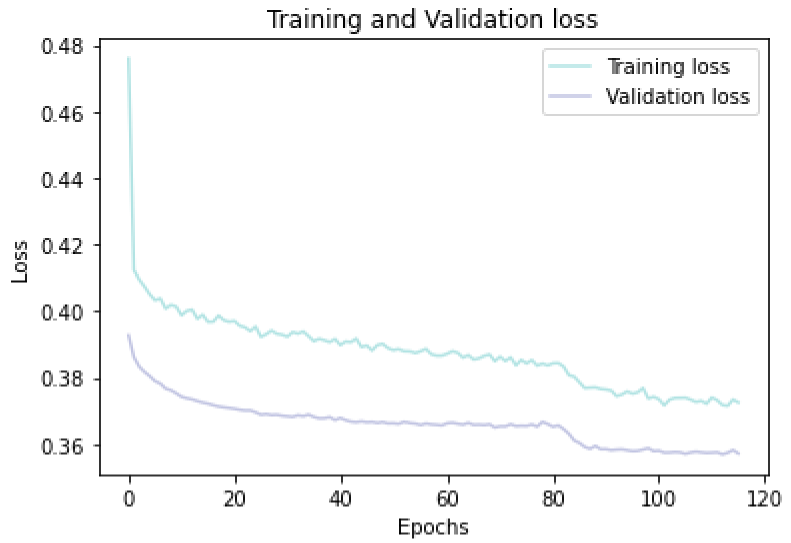


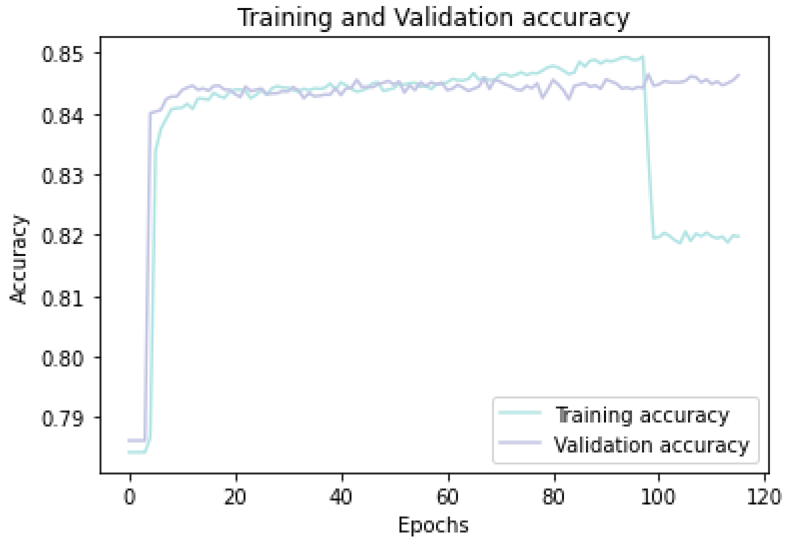


```
              precision    recall  f1-score   support

           0       0.88      0.94      0.91     33131
           1       0.69      0.52      0.59      8956

    accuracy                           0.85     42087
   macro avg       0.78      0.73      0.75     42087
weighted avg       0.84      0.85      0.84     42087
```

OUR MODEL MODIFICATIONS

Changing testing size split to 15%.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.15, random_state = 42)

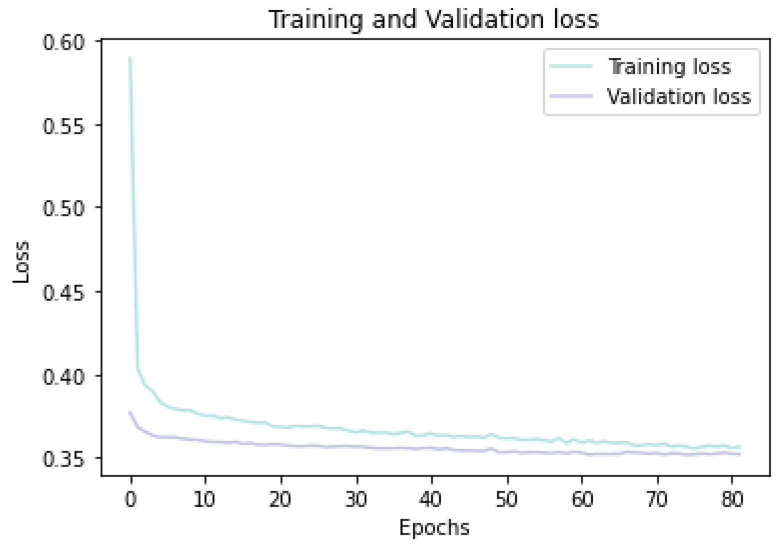


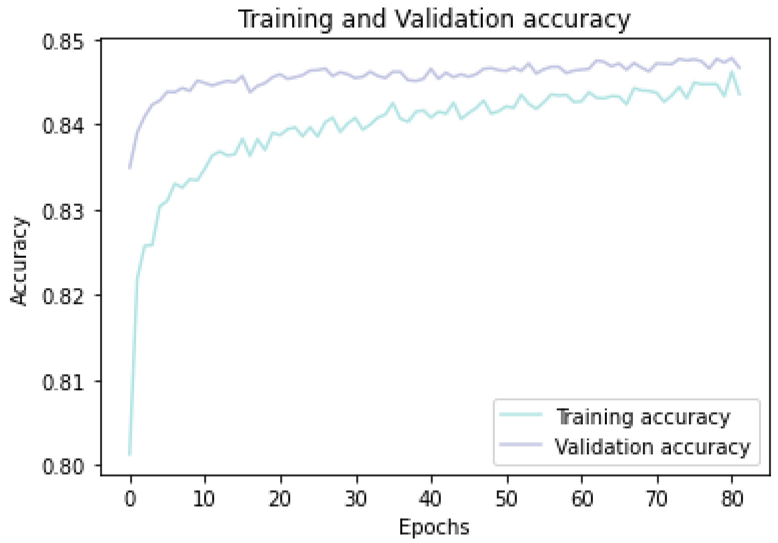


```
              precision    recall  f1-score   support

           0       0.89      0.92      0.90     15099
           1       0.65      0.56      0.61      4032

    accuracy                           0.85     19131
   macro avg       0.77      0.74      0.75     19131
weighted avg       0.84      0.85      0.84     19131
```

OUR MODEL MODIFICATIONS

Changing activation function to tanh.
model.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'tanh'))

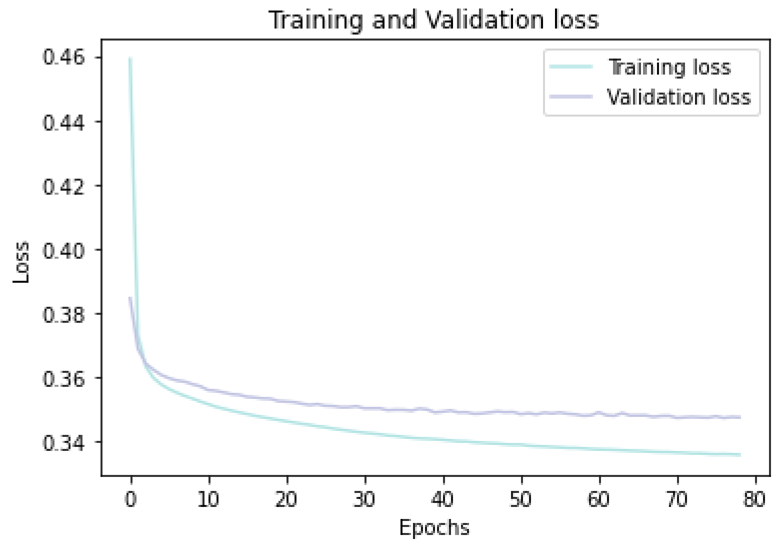


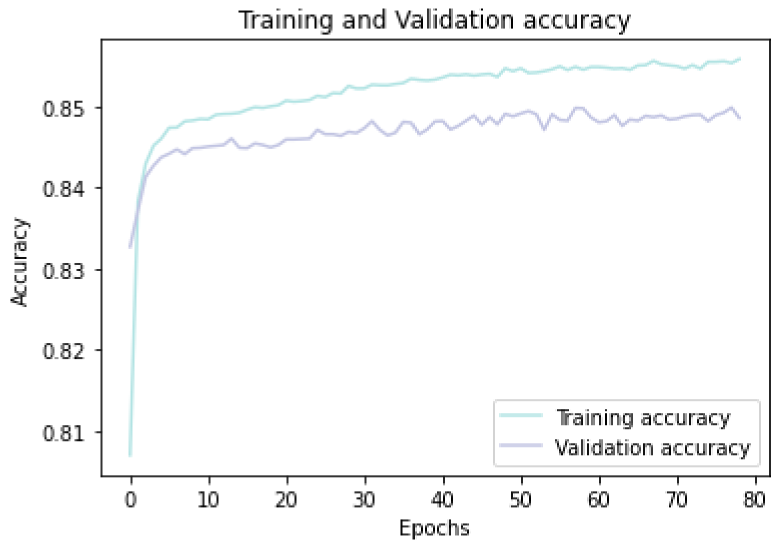


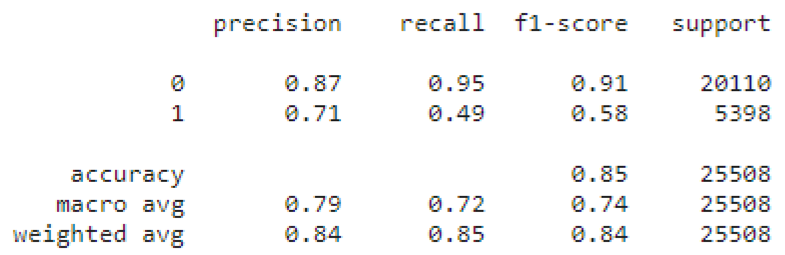
```
              precision    recall  f1-score   support

           0       0.87      0.95      0.91     20110
           1       0.73      0.48      0.58      5398

    accuracy                           0.85     25508
   macro avg       0.80      0.72      0.75     25508
weighted avg       0.84      0.85      0.84     25508
```

OUR MODEL MODIFICATIONS

Deleting droputs.
model.add(Dropout(0.25))



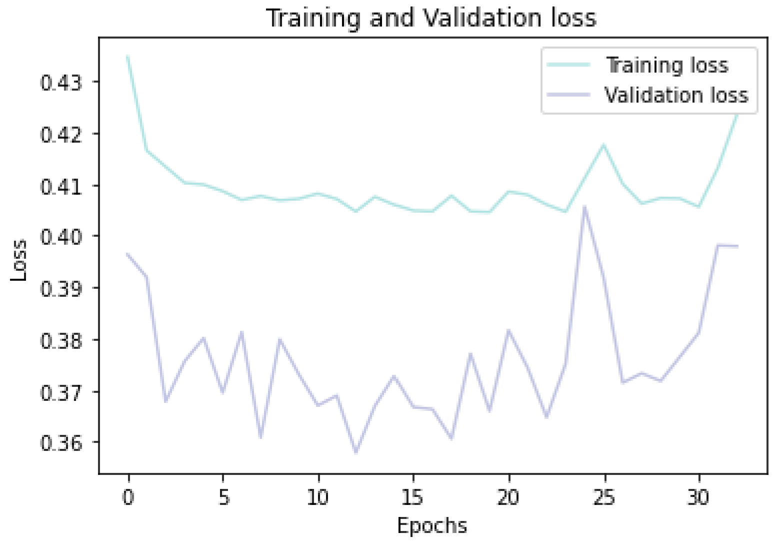


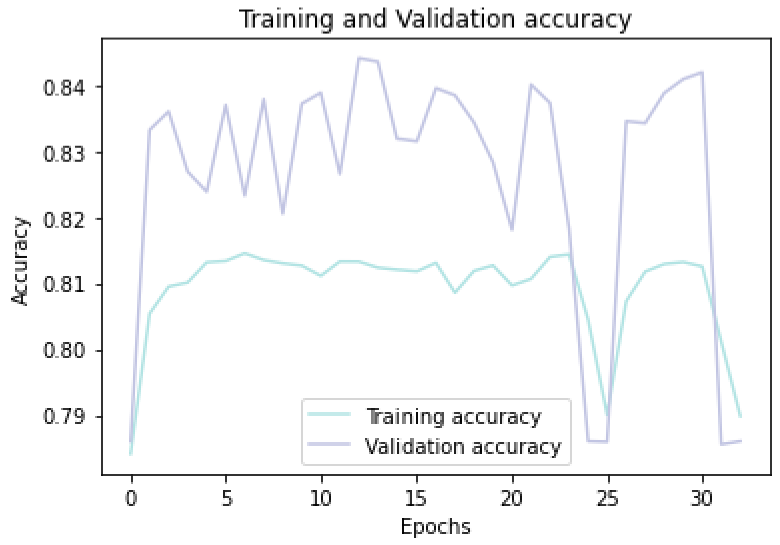


```
              precision    recall  f1-score   support

           0       0.87      0.95      0.91     20110
           1       0.71      0.49      0.58      5398

    accuracy                           0.85     25508
   macro avg       0.79      0.72      0.74     25508
weighted avg       0.84      0.85      0.84     25508
```

OUR MODEL MODIFICATIONS

Changing learning rate.
opt = Adam(learning_rate=0.01)

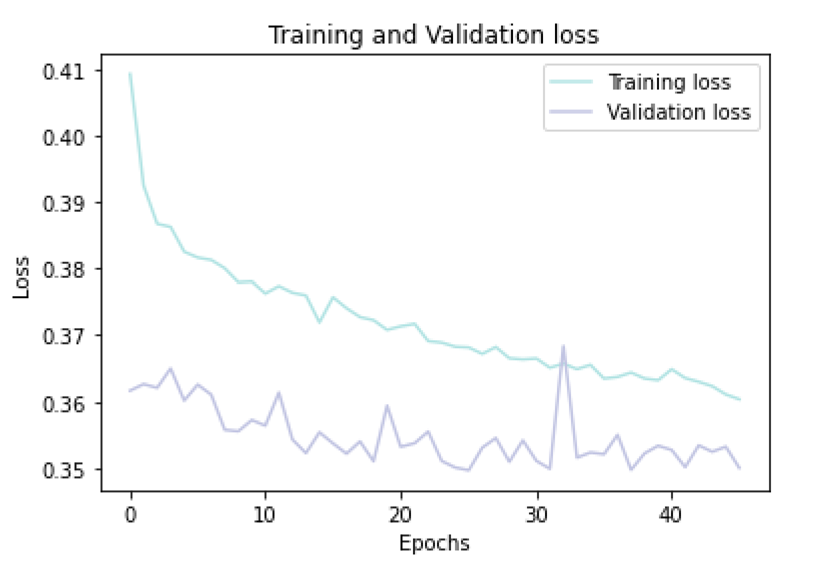


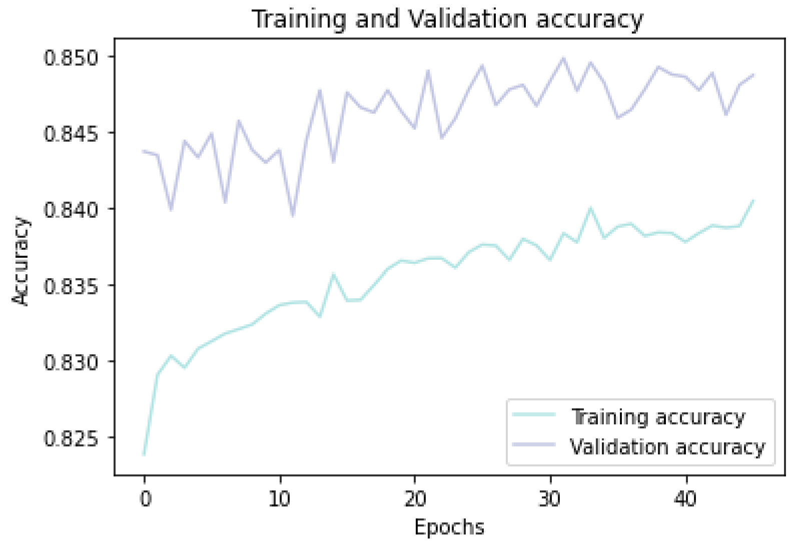


```
              precision    recall  f1-score   support

           0       0.86      0.96      0.91     20110
           1       0.75      0.41      0.53      5398

    accuracy                           0.85     25508
   macro avg       0.80      0.68      0.72     25508
weighted avg       0.83      0.85      0.83     25508
```

OUR MODEL MODIFICATIONS

Changing learning rate.
opt = Adam(learning_rate=0.002)

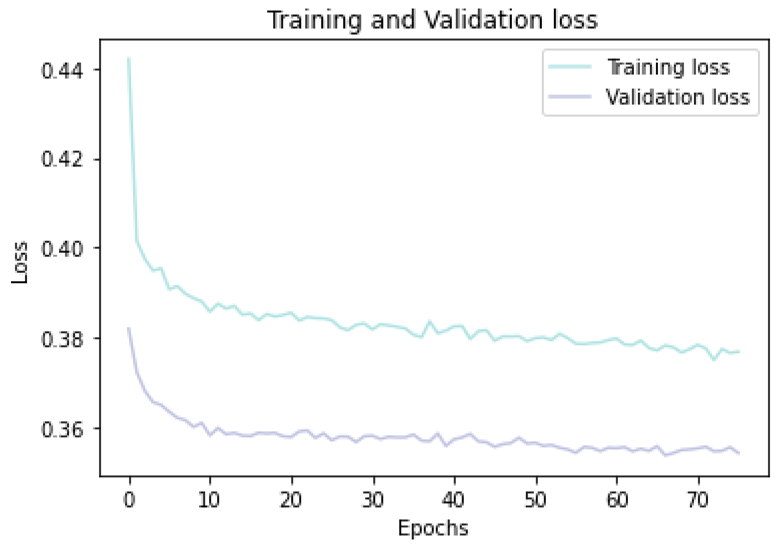


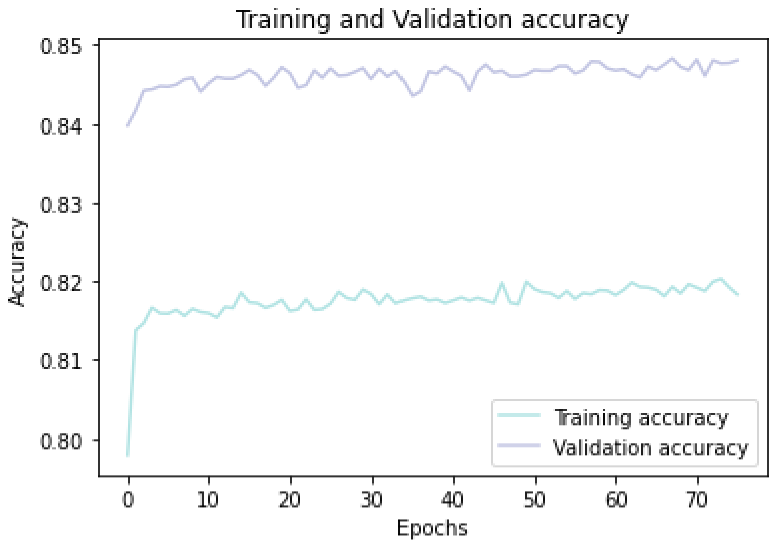


```
              precision    recall  f1-score   support

           0       0.88      0.94      0.91     20110
           1       0.71      0.50      0.59      5398

    accuracy                           0.85     25508
   macro avg       0.79      0.72      0.75     25508
weighted avg       0.84      0.85      0.84     25508
```

OUR MODEL MODIFICATIONS

Changing learning rate.
opt = Adam(learning_rate=0.0002)

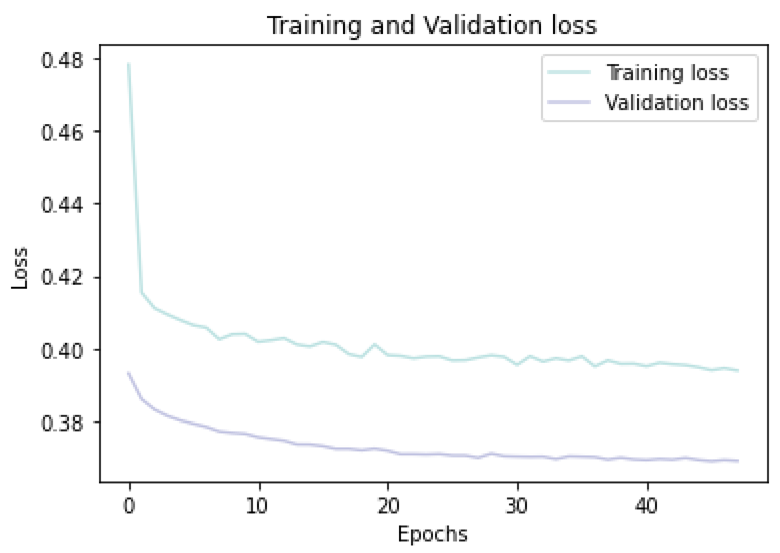


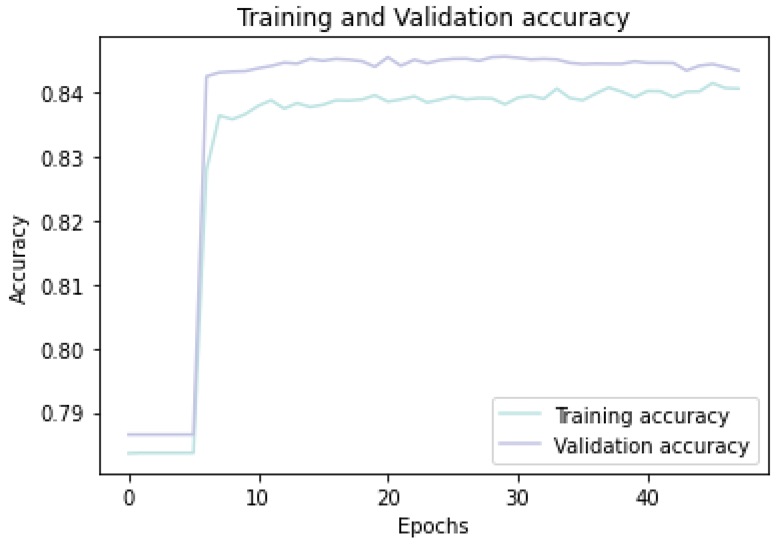


```
              precision    recall  f1-score   support

           0       0.88      0.94      0.91     20110
           1       0.71      0.50      0.59      5398

    accuracy                           0.85     25508
   macro avg       0.79      0.72      0.75     25508
weighted avg       0.84      0.85      0.84     25508
```

OUR MODEL MODIFICATIONS

Changing validation split to 0,3..
history = model.fit(X_train, y_train, batch_size = 32, epochs = 150, callbacks=[early_stopping], validation_split=0.3)



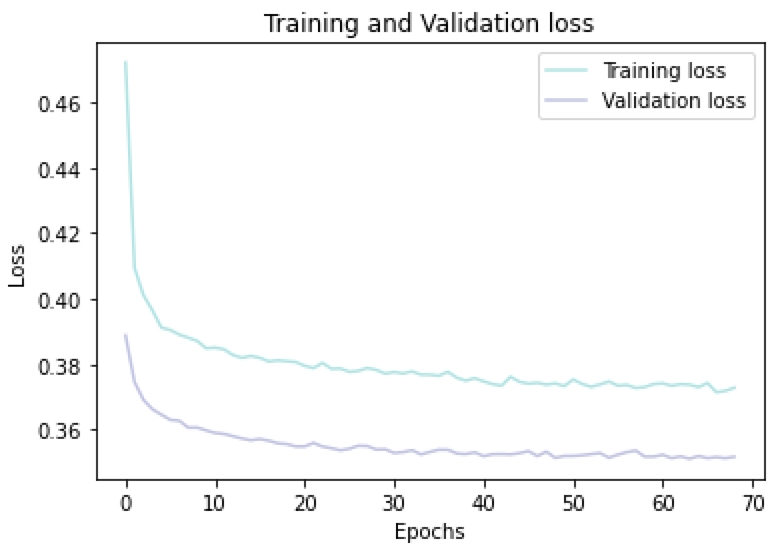


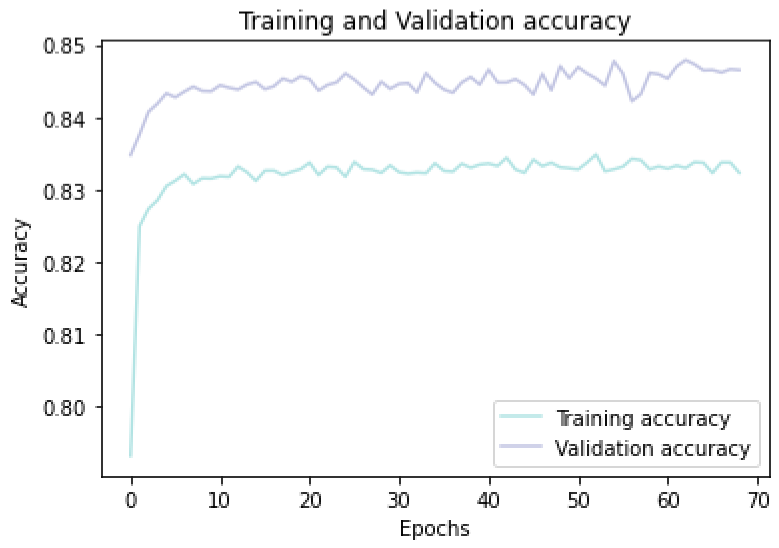
```
              precision    recall  f1-score   support

           0       0.87      0.94      0.91     20110
           1       0.69      0.50      0.58      5398

    accuracy                           0.85     25508
   macro avg       0.78      0.72      0.74     25508
weighted avg       0.84      0.85      0.84     25508
```

OUR MODEL MODIFICATIONS

Changing batch size to 28

history = model.fit(X_train, y_train, batch_size = 28, epochs = 150, callbacks=[early_stopping], validation_split=0.2)

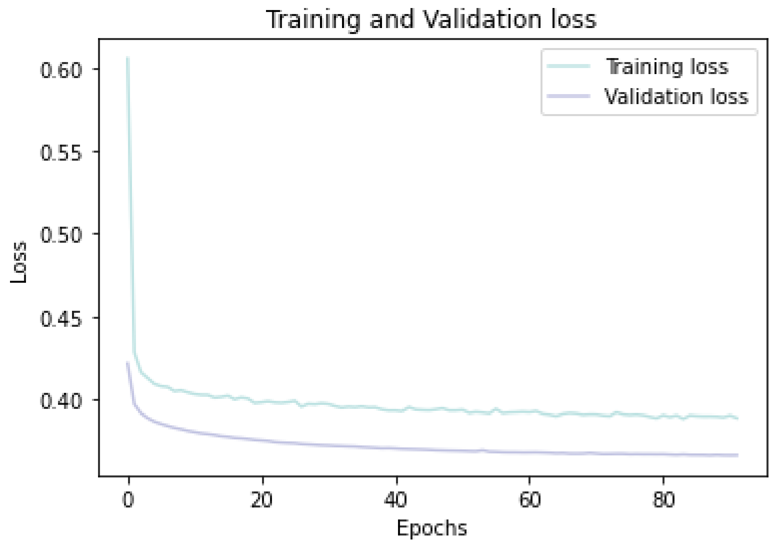


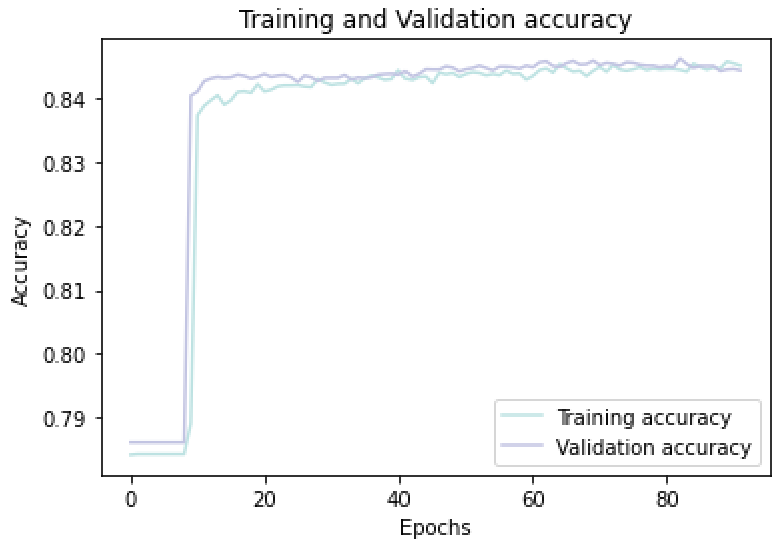


```
              precision    recall  f1-score   support

           0       0.86      0.96      0.91     20110
           1       0.75      0.44      0.55      5398

    accuracy                           0.85     25508
   macro avg       0.81      0.70      0.73     25508
weighted avg       0.84      0.85      0.83     25508
```

OUR MODEL MODIFICATIONS

Changing batch size to 140
history = model.fit(X_train, y_train, batch_size = 140, epochs = 150, callbacks=[early_stopping], validation_split=0.2)





```
              precision    recall  f1-score   support

           0       0.88      0.93      0.91     20110
           1       0.68      0.53      0.60      5398

    accuracy                           0.85     25508
   macro avg       0.78      0.73      0.75     25508
weighted avg       0.84      0.85      0.84     25508
```

SUMMARY

- Thanks to this project we deepened our knowledge about artificial inteligence engineering.
- We also refreshed our data visualization skills
- It is not very easy to upgrade project of Kaggle Grandmaster

SOURCE

https://www.kaggle.com/code/karnikakapoor/rain-prediction-ann

https://www.ibm.com/topics/neural-networks

THANKS!