Rain prediction in Australia

Biologically Inspired Artificial Intelligence - project

Authors:
Jan Makowiecki
Bartosz Pijet

Supervisor:
dr inż. Grzegorz Baron

1. Introduction

The goal of this project was to predict next-day rain using a dataset containing 10 years of daily weather observations from different locations across Australia.

2. Possible approaches

a) Linear regression
   + simple to understand
   + computationally efficient
   - sensitive to outliers
   - limited complexity
   - assumes linearity
   - isn't "biologically inspired"

b) Decision trees
   + simple to understand and visualize
   + works with non-linear data
   + doesn't require normalization of data
   + insensitive to outliers
   - small change in data can cause a large change in structure
   - training is time consuming
   - inadequate for applying regression and predicting continuous values

c) Random forest
   + works well with both categorical and numerical data
   + transformation and scaling is usually not necessary
   + insensitive to outliers
   + works with non-linear data
   - computationally intensive for large datasets
   - hard to interpret
   - little control over the model

d) Neural network
   + self-learning
   + works with non-linear data
   + fault tolerance
   + operating speed
   + biologically inspired
   - computationally expensive
   - requires lots of data to train
   - susceptibility to overfitting
   - requires data preprocessing

We chose a neural network as our prediction model, because it balances simplicity and universality.

3. Data selection

Almost all of the prediction models are using the "weatherAUS" dataset. It contains 10 years of detailed weather data from all over Australia. There are over 140 '000 records with very little missing values. Data was collected by the Australian Commonwealth

Bureau of Meteorology and can be accessed online on their website (http://www.bom.gov.au/climate/data/). It was processed and prepared into a single csv file by Graham Williams. Our version of the file was taken from kaggle.com . We decided to choose it as our training dataset, because of the good quality and amount of data.

Dataset contains 145 ‘460 rows x 23 columns of data. Columns are as follows:

Column	Description	Type
Date	Date of observation	object
Location	Common name of the location of the weather station	object
MinTemp	Minimum temperature in degrees celsius	float64
MaxTemp	Maximum temperature in degrees celsius	float64
Rainfall	Amount of rainfall recorded for the day in mm	float64
Evaporation	So-called Class A pan evaporation (mm) in the 24 hours to 9am	float64
Sunshine	Number of hours of bright sunshine in the day	float64
WindGustDir	Direction of the strongest wind gust in the 24 hours to midnight	object
WindGustSpeed	Speed (km/h) of the strongest wind gust in the 24 hours to midnight	float64
WindDir9am	Direction of the wind at 9am	object
WindDir3pm	Direction of the wind at 3pm	object
WindSpeed9am	Wind speed (km/hr) averaged over 10 minutes prior to 9am	float64
WindSpeed3pm	Wind speed (km/hr) averaged over 10 minutes prior to 3pm	float64
Humidity9am	Humidity (percentages) at 9am	float64
Humidity3pm	Humidity (percentages) at 3pm	float64
Pressure9am	Atmospheric pressure (hpa) reduced to mean sea level at 9am	float64
Pressure3pm	Atmospheric pressure (hpa) reduced to mean sea level at 3pm	float64
Cloud9am	Fraction of sky obscured by cloud at 9am measured in unit of eigths	float64
Cloud3pm	Fraction of sky obscured by cloud at 3pm measured in unit of eigths	float64
Temp9am	Temperature (degrees C) at 9am	float64
Temp3pm	Temperature (degrees C) at 3pm	float64
RainToday	Yes if precipitation exceeds 1mm, otherwise No	object
RainTomorrow	The target variable. Did it rain tomorrow? (Yes/No)	object

4. Tools

Nowadays there are many tools to create and train neural networks. Some of the more popular are:

- Tensorflow
- PyTorch
- Deeplearning4j
- Microsoft Cognitive Toolkit (CNTK)
- Keras
- ONNX
- MXNet
- Caffe

For this project we used Python and Jupyter Notebook as base programs and Keras on top of Tensorflow, because Keras is beginner friendly and many helpful tutorials can be found on the Web.

5. Solution structure

Solution is presented as a jupyter notebook, thus most of the functionalities are encapsulated in cells and not in classes or structures. Furthermore, there isn't a proper user interface - to modify a script one has to change code in a specific cell and run the "Model building" section again.

Notebook is divided in sections:

a) Data preparation - in this section we import all needed libraries and functionalities and also read data from csv to Pandas' dataframe,

b) Data visualization and cleaning - in this section we focus on cleaning data by removing rows without value in the target column and by filling missing data in other columns. We also encode cyclical data (months and days) and plot some charts. Encoding creates 2 new columns from 1 source column with values first scaled to *sin* and *cos* functions and then scaled to have values in range <0, 1>,

c) Data preprocessing - in this section we label categorical data, remove outliers based on z-score and normalize all columns to have values in range <0, 1>,

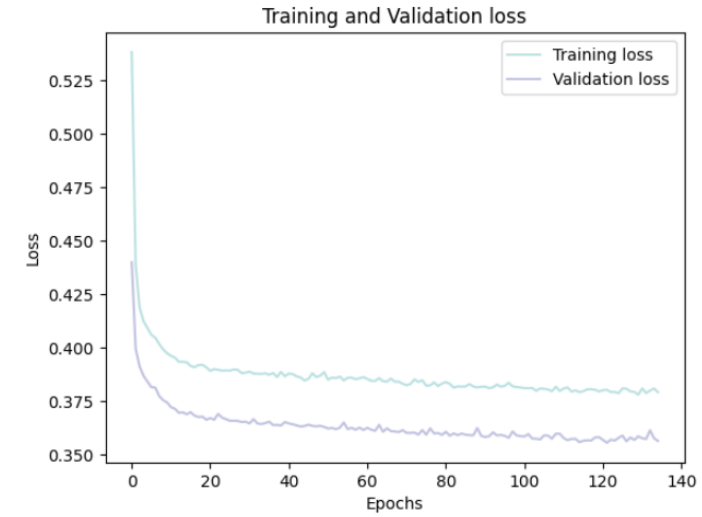
d) Model building - in this section we create an artificial neural network model, train it and check its statistics like training accuracy or loss.

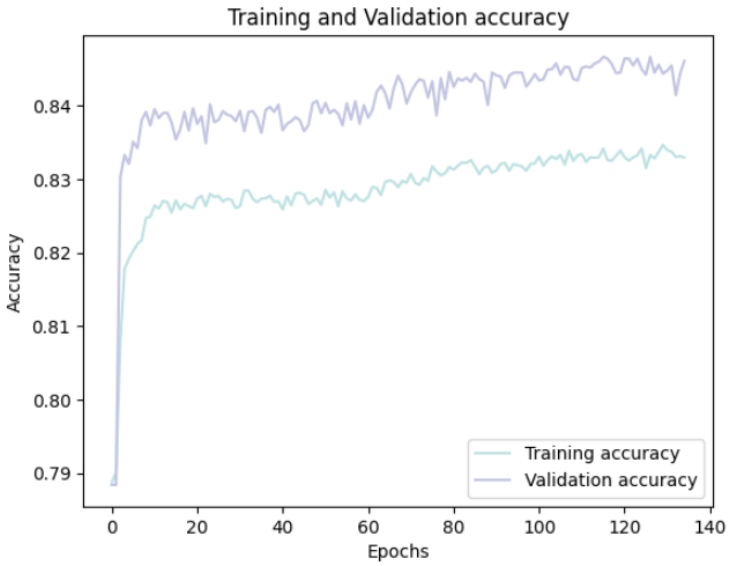
Cell responsible for training (base version):

```
X = features
y = target

# Splitting test and training sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

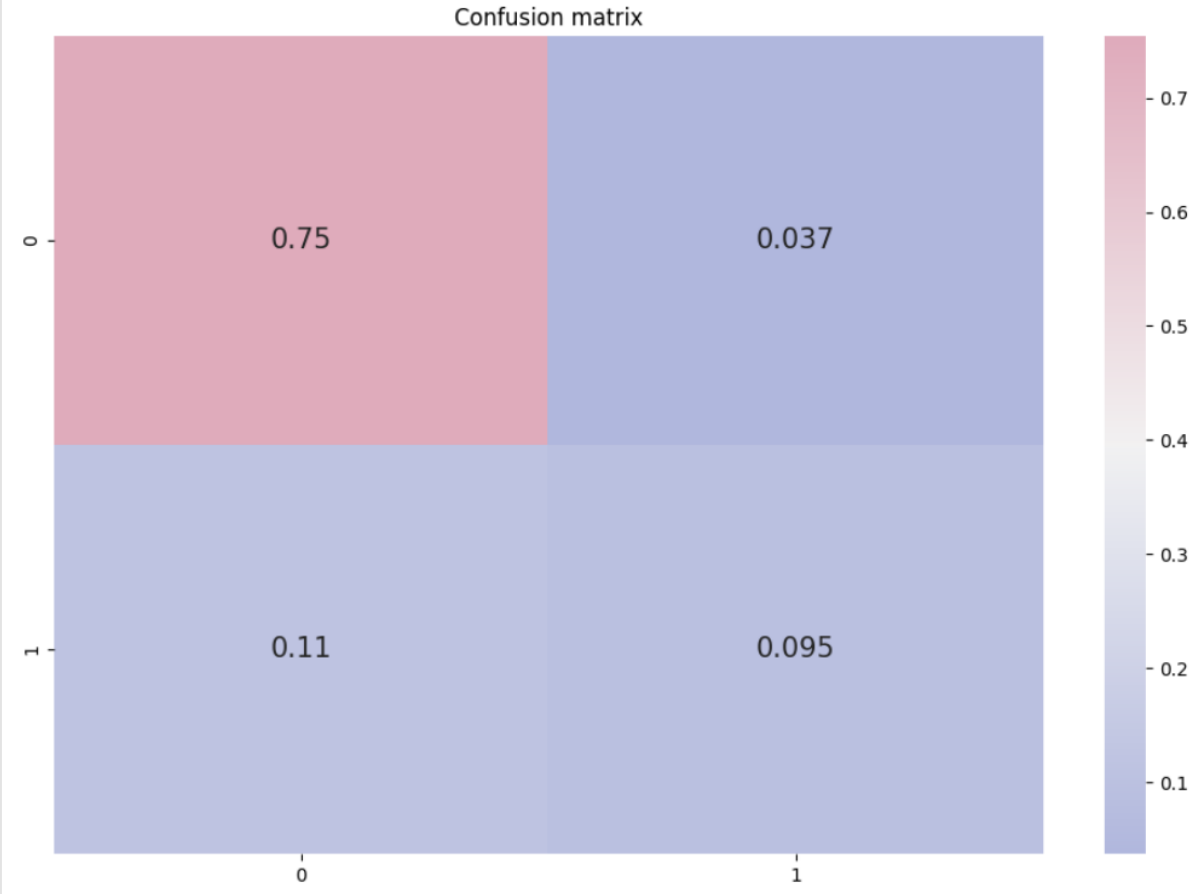
```python
# Early stopping
early_stopping = callbacks.EarlyStopping(
    min_delta=0.001,            # minimium amount of change to count as an improvement
    patience=20,                # number of epochs with no improvement after which training will be stopped
    restore_best_weights=True,  # restore model weights from the epoch with the best value of the monitored quantity
)

# Initialising the NN
model = Sequential()

# Layers
# Dense - standard NN layer
# Dropout - randomly sets input units to 0, which helps prevent overfitting
model.add(Dense(units = 32, kernel_initializer = 'uniform', activation = 'relu', input_dim = X.shape[1]))
model.add(Dense(units = 32, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dense(units = 16, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dropout(0.25))
model.add(Dense(units = 8, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dropout(0.5))
model.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))

# Compiling the ANN
# Adam optimization is a stochastic gradient descent method
# that is based on adaptive estimation of first-order and second-order moments.
opt = Adam(learning_rate=0.00009)
model.compile(optimizer = opt, loss = 'binary_crossentropy', metrics = ['accuracy'])

# Training the ANN
history = model.fit(X_train, y_train, batch_size = 32, epochs = 150, callbacks=[early_stopping], validation_split=0.2)
```

Results for this model were as follows:





	precision	recall	f1-score	support
0	0.87	0.95	0.91	21121
1	0.72	0.45	0.56	5577
accuracy			0.85	26698
macro avg	0.79	0.70	0.73	26698
weighted avg	0.84	0.85	0.83	26698

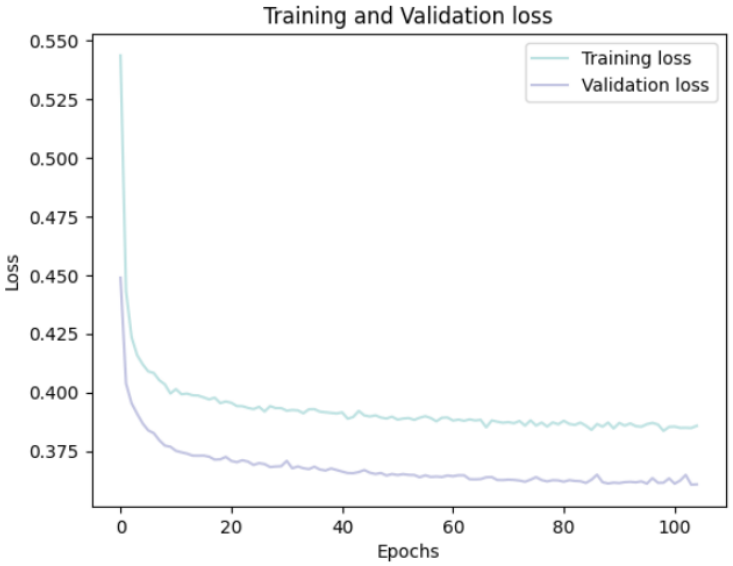


Base model accuracy started from around 82% and rose to around 84% which could be considered pretty good, but not perfect.

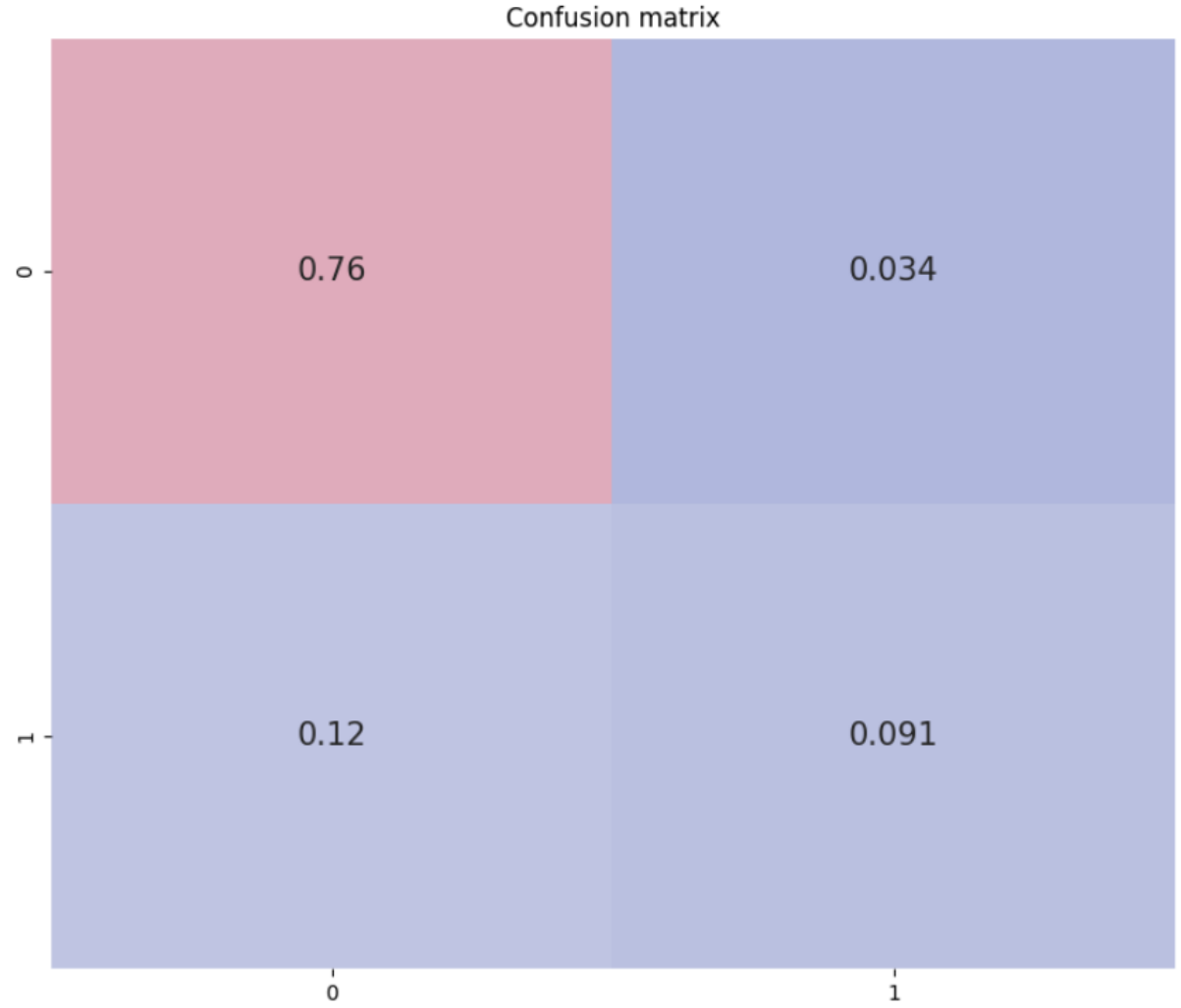
6. Experiments

We then experimented with the base model by changing different parameters and checking how it affects accuracy. Because we are beginners with machine learning and artificial intelligence, we don't know which parameters affect what, so we decided to change one parameter to a different value in every experiment. At the end we tried to change all of the parameters at once and see if accuracy will change.

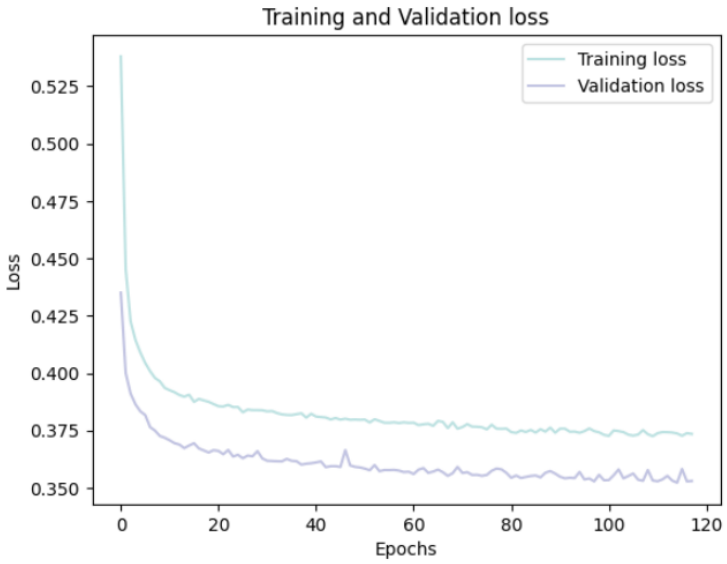
a) test_size: 0.2 -> 0.33



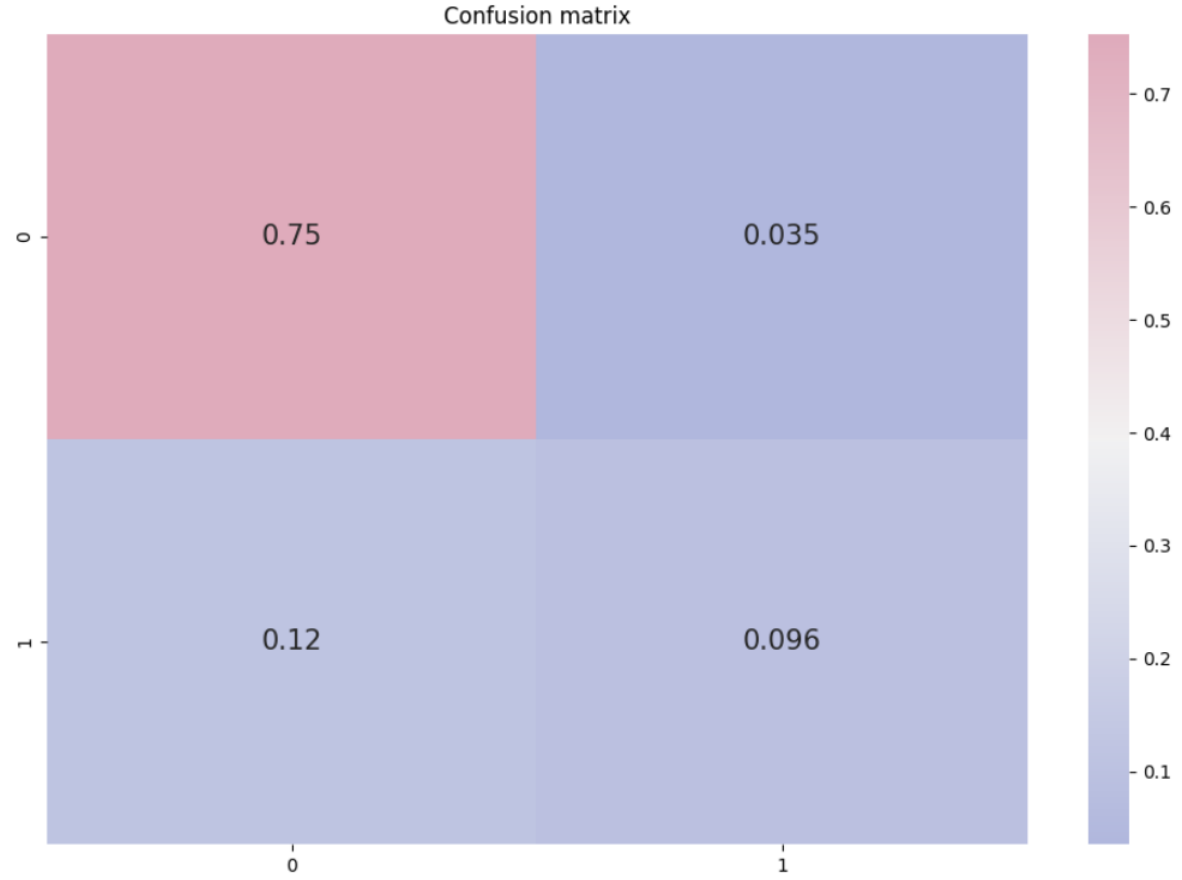
	precision	recall	f1-score	support
0	0.86	0.96	0.91	34813
1	0.73	0.43	0.54	9239
accuracy			0.85	44052
macro avg	0.80	0.70	0.73	44052
weighted avg	0.84	0.85	0.83	44052



b) test_size: 0.2 -> 0.15

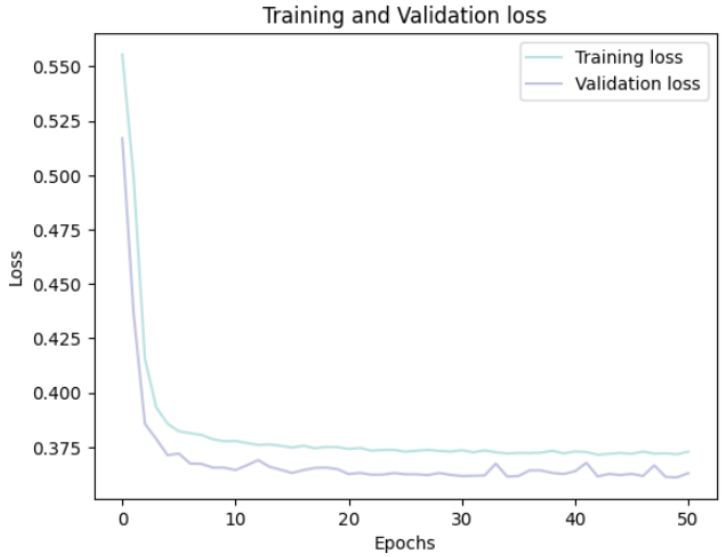


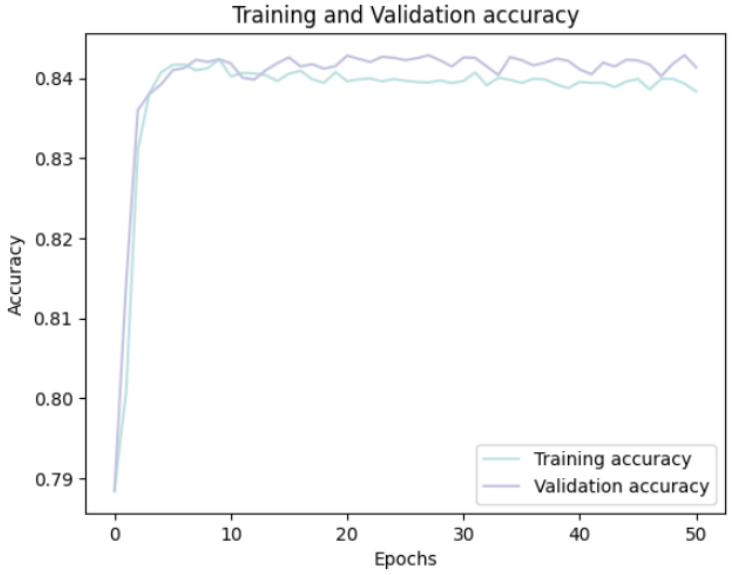
	precision	recall	f1-score	support
0	0.87	0.96	0.91	15774
1	0.73	0.45	0.56	4250
accuracy			0.85	20024
macro avg	0.80	0.70	0.73	20024
weighted avg	0.84	0.85	0.83	20024



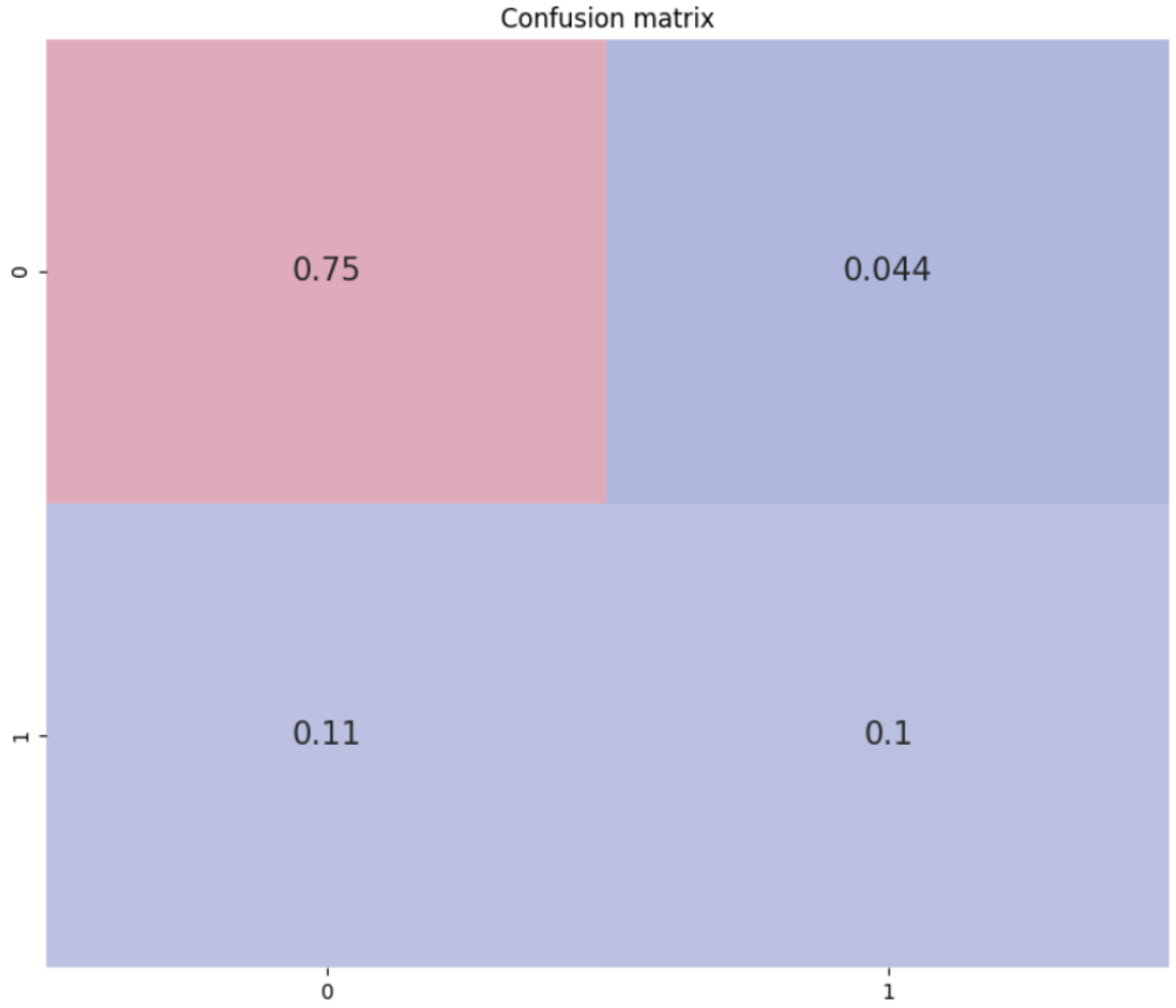
a,b - changing the test sample size did not result in any meaningful changes neither in training accuracy plot nor confusion matrix.

c) layers activation function: relu -> tanh



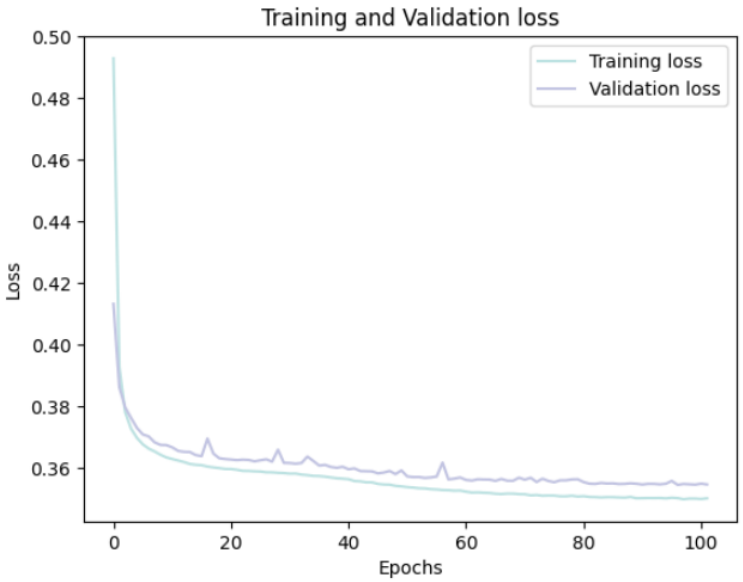


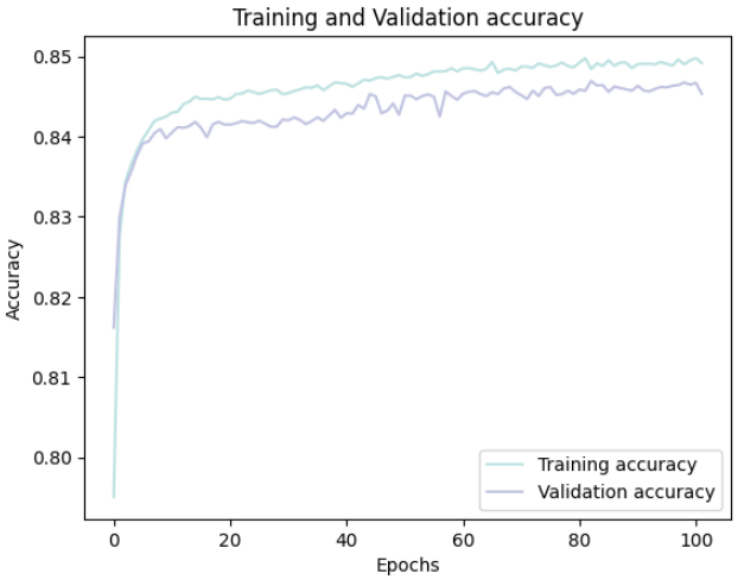
	precision	recall	f1-score	support
0	0.87	0.94	0.91	21121
1	0.70	0.48	0.57	5577
accuracy			0.85	26698
macro avg	0.78	0.71	0.74	26698
weighted avg	0.84	0.85	0.84	26698



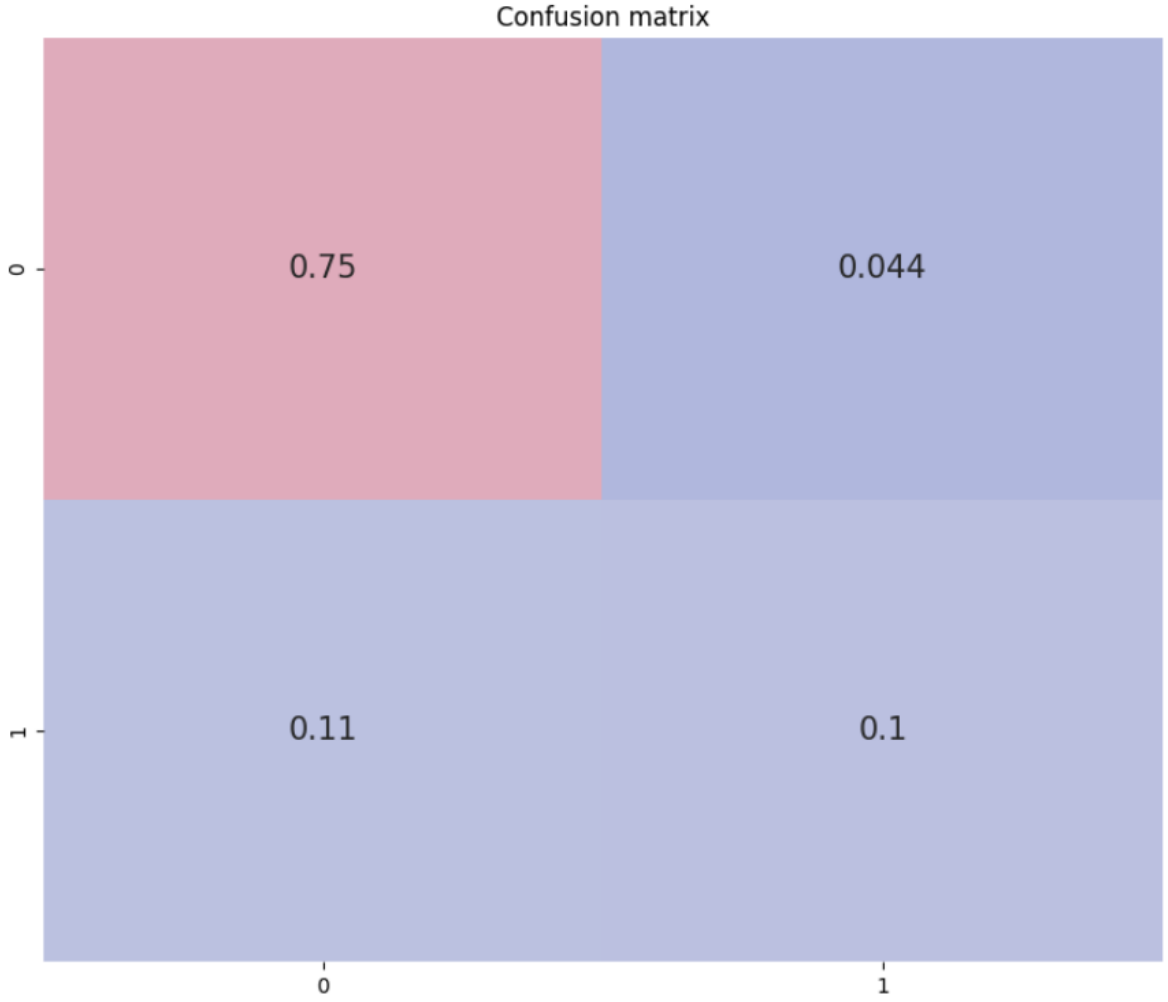
c - changing activation function brought training and validation diagrams closer

d) dropouts: [0.25, 0.5] -> removed



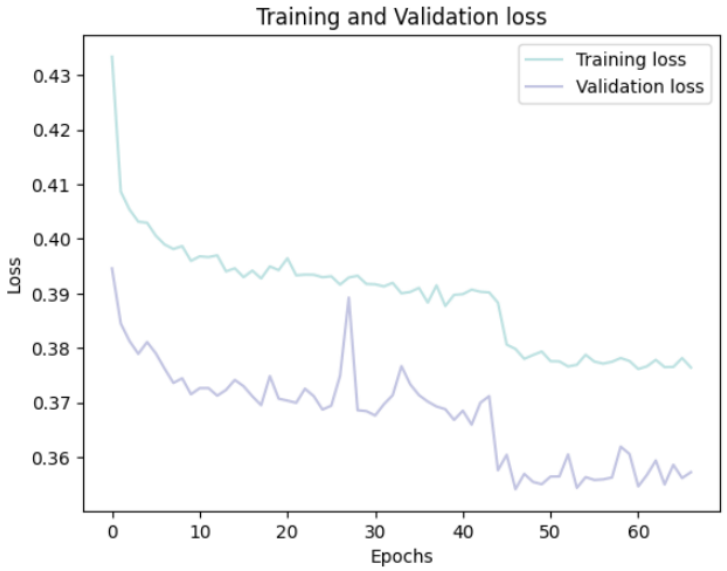


	precision	recall	f1-score	support
0	0.87	0.94	0.91	21121
1	0.70	0.49	0.57	5577
accuracy			0.85	26698
macro avg	0.79	0.72	0.74	26698
weighted avg	0.84	0.85	0.84	26698

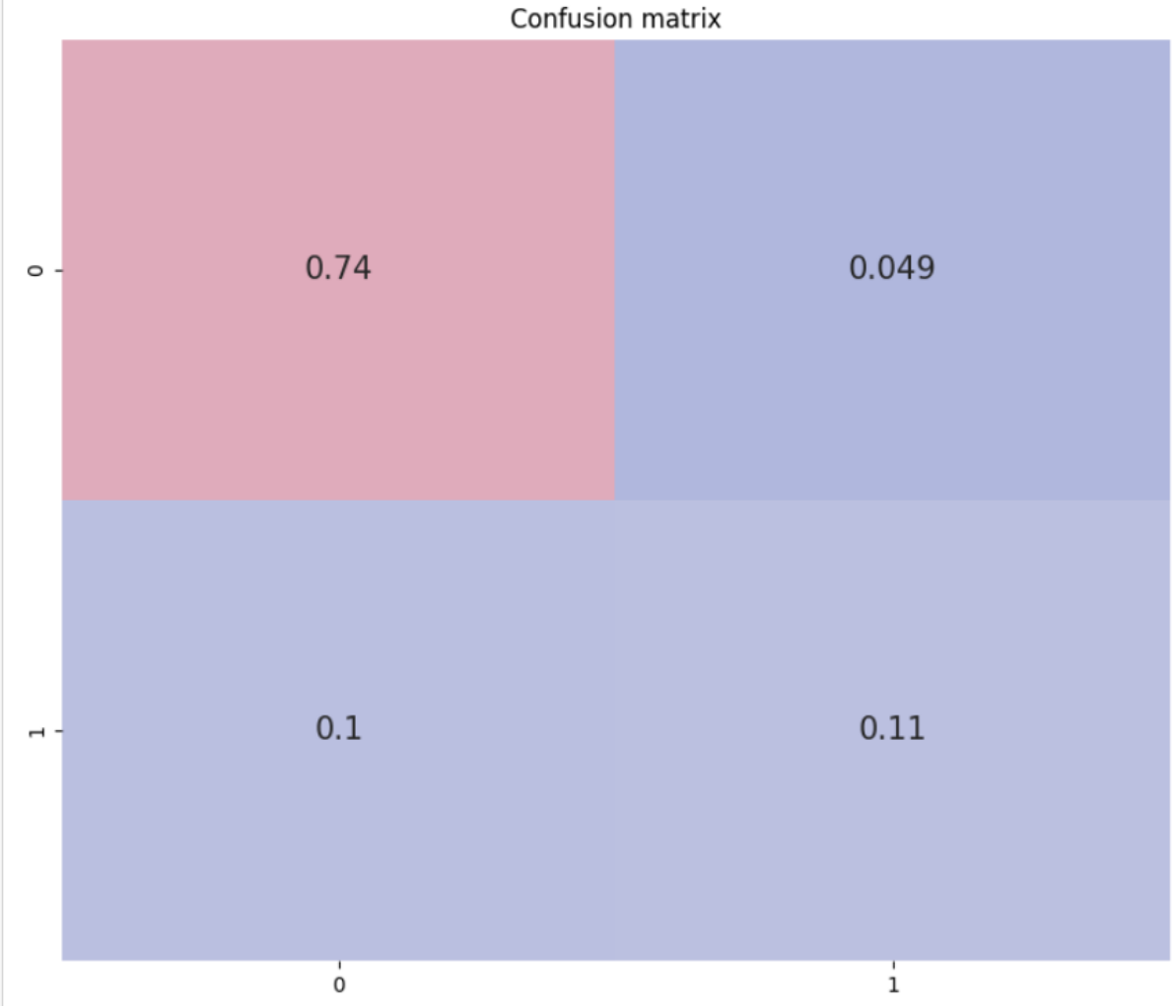


d - with removal of dropouts we achieved minimal overfitting

e) Adam learning rate: 0.00009 -> 0.001

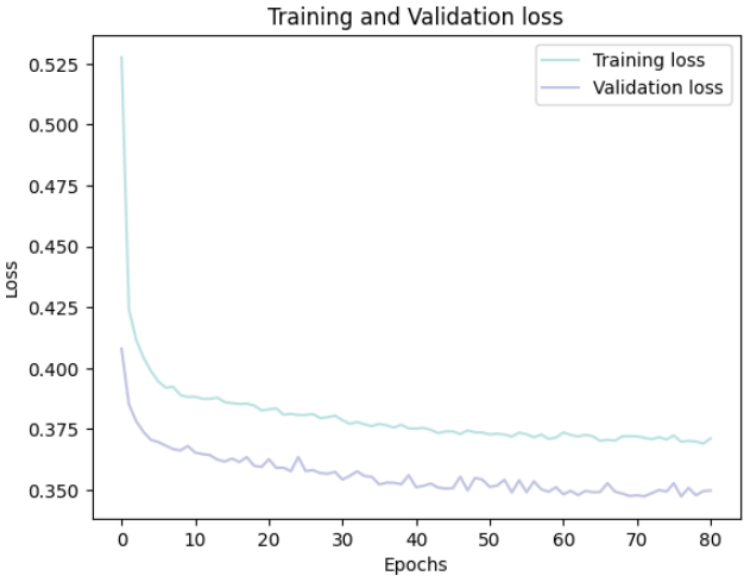


	precision	recall	f1-score	support
0	0.88	0.94	0.91	21121
1	0.69	0.52	0.59	5577
accuracy			0.85	26698
macro avg	0.78	0.73	0.75	26698
weighted avg	0.84	0.85	0.84	26698

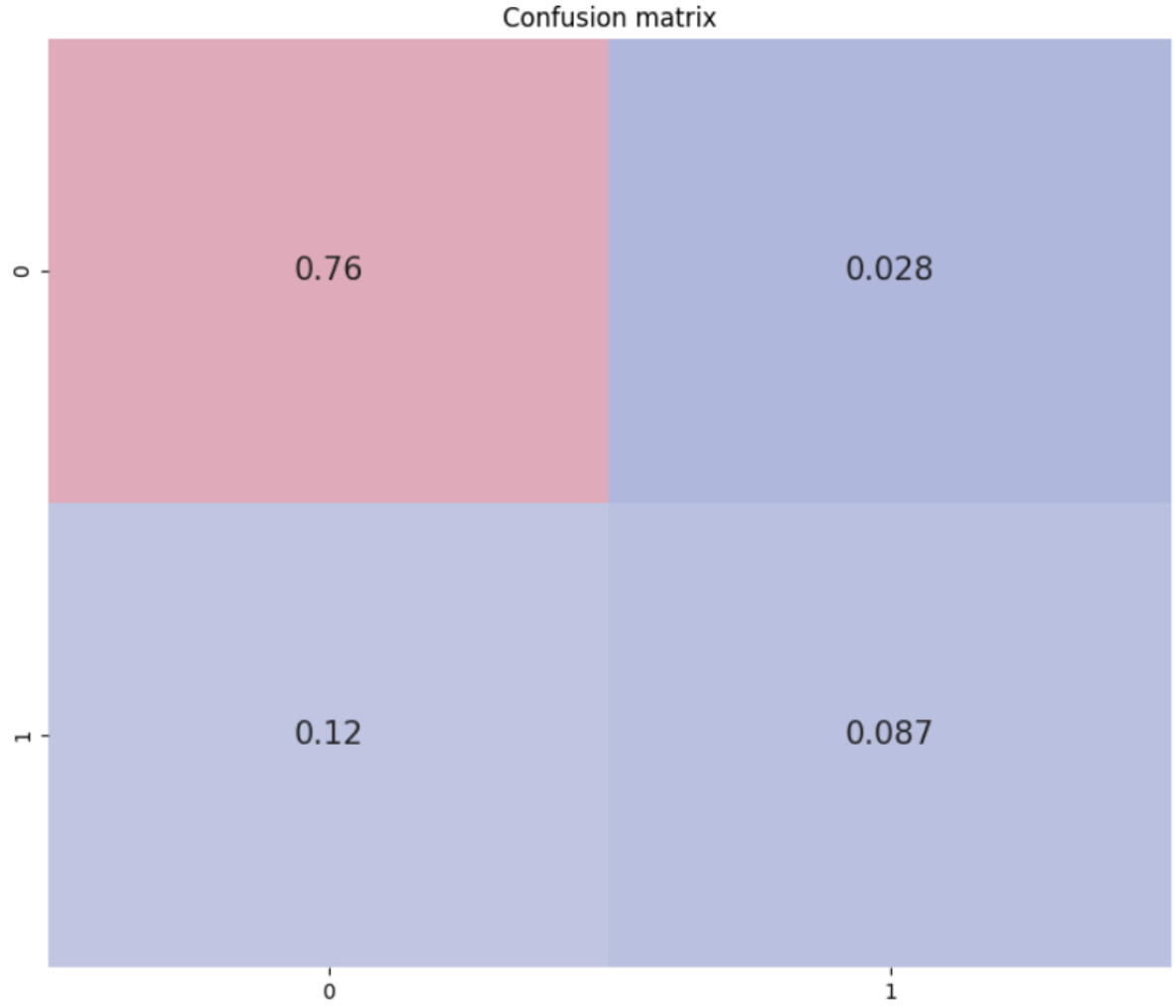


e - this change in learning rate caused the model to learn more slowly (slower value decrease in loss diagram) and at some point training accuracy just fell by 2%p.

f) Adam learning rate: 0.00009 -> 0.0002

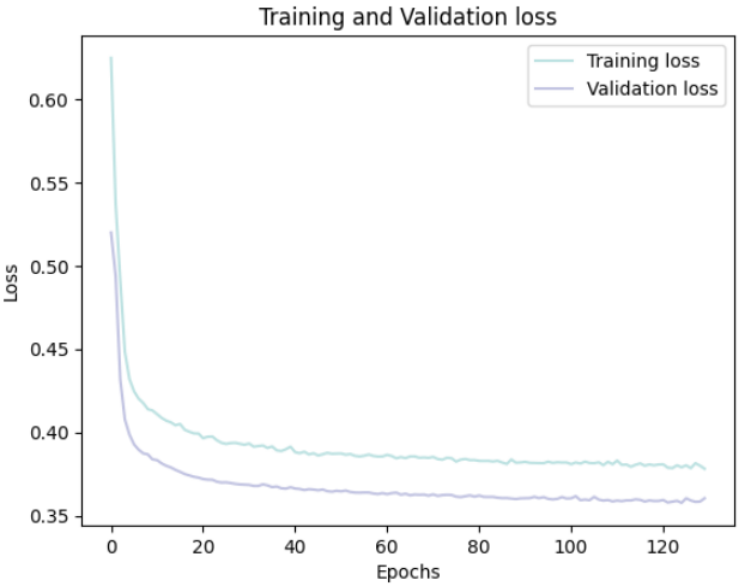


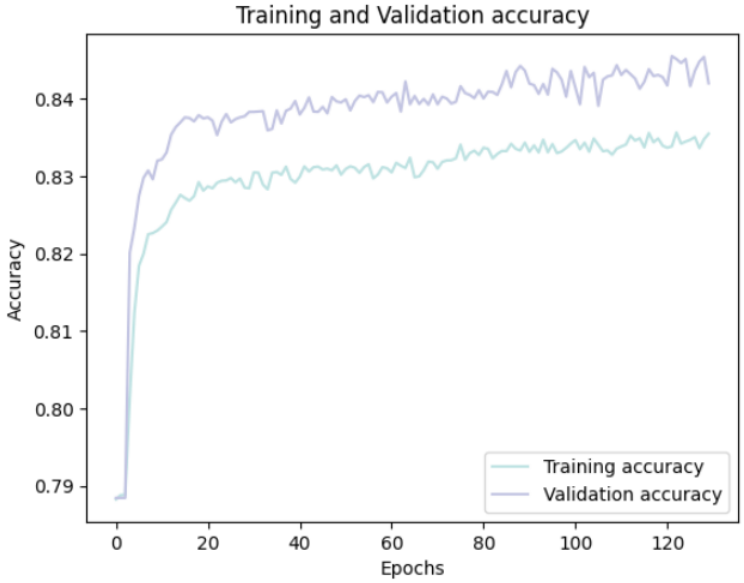
	precision	recall	f1-score	support
0	0.86	0.96	0.91	21121
1	0.76	0.42	0.54	5577
accuracy			0.85	26698
macro avg	0.81	0.69	0.72	26698
weighted avg	0.84	0.85	0.83	26698



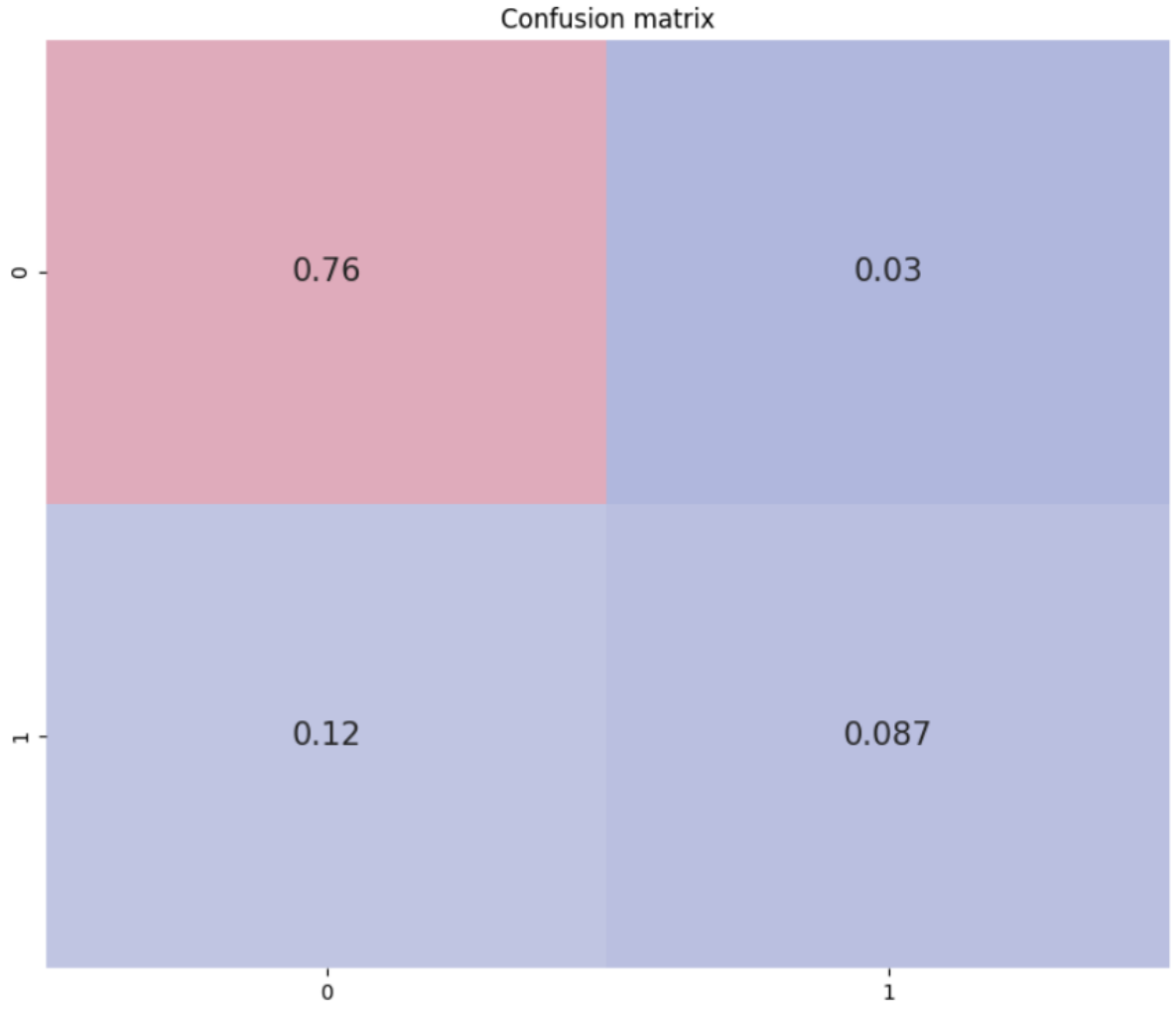
f - this change to learning rate slightly increased fluctuations in accuracy diagram.

g) model fit - batch size: 32 -> 128



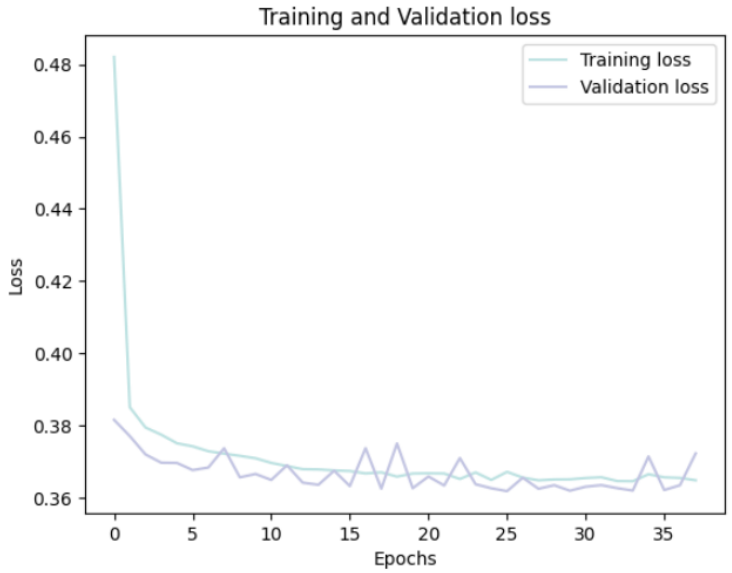


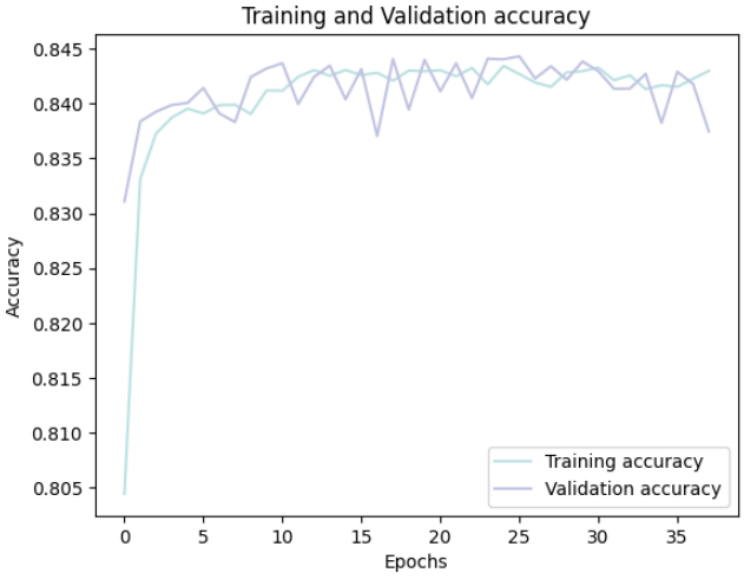
	precision	recall	f1-score	support
0	0.86	0.96	0.91	21121
1	0.74	0.42	0.53	5577
accuracy			0.85	26698
macro avg	0.80	0.69	0.72	26698
weighted avg	0.84	0.85	0.83	26698



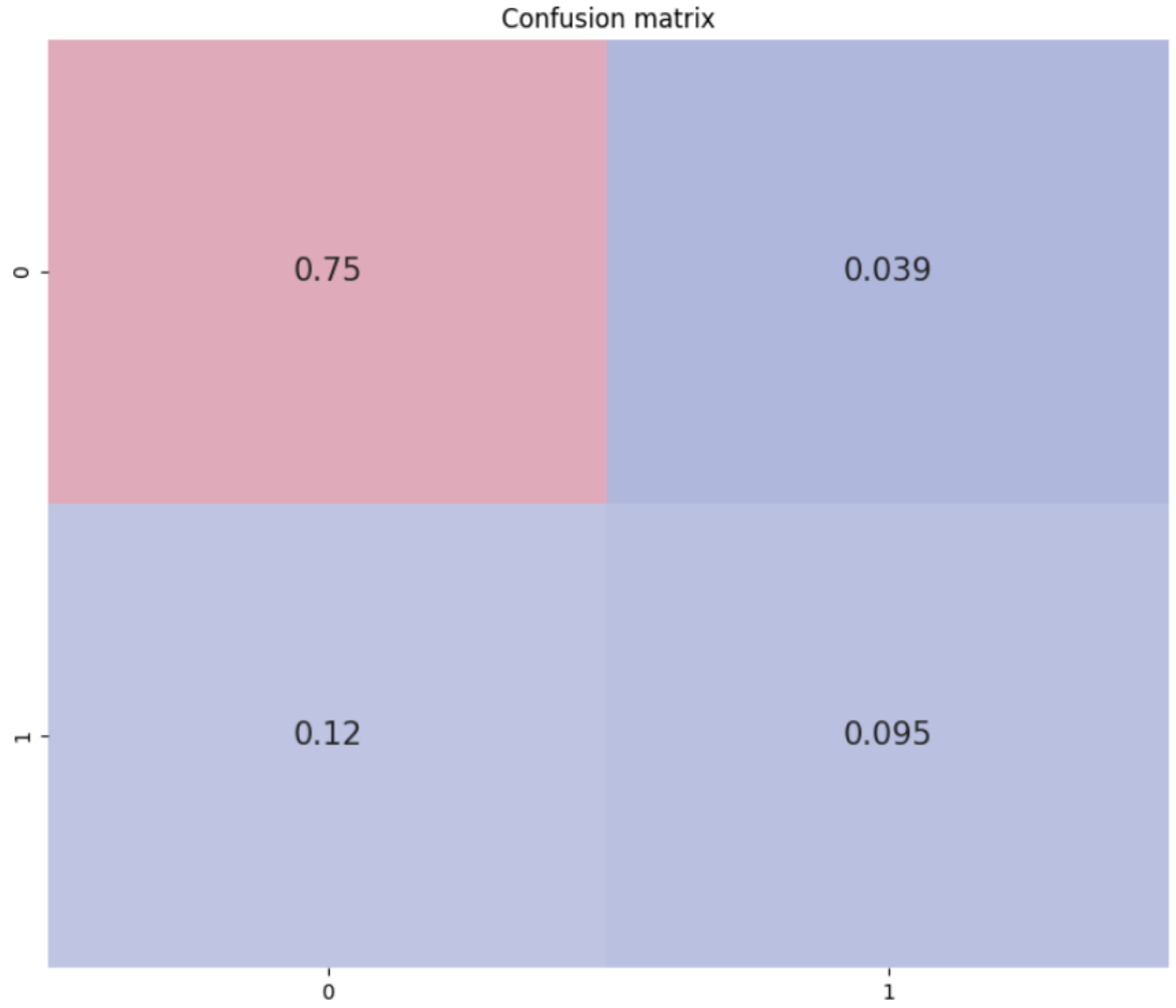
g - changing the batch size did not result in any meaningful changes neither in training accuracy plot nor confusion matrix

h) multiple changes:
- test_size: 0.2 -> 0.1
- 2 first layers activation function: relu -> tanh
- dropouts: [0.25, 0.5] -> [0.25, 0.25]
- Adam learning rate: 0.00009 -> 0.0005
- model fit - batch size: 32 -> 128
- model fit - validation split: 0.2 -> 0.3





	precision	recall	f1-score	support
0	0.87	0.95	0.91	10527
1	0.71	0.45	0.55	2822
accuracy			0.84	13349
macro avg	0.79	0.70	0.73	13349
weighted avg	0.83	0.84	0.83	13349



h - experiment resulted in an unrepresentative validation dataset, which was caused by increasing size of validation dataset.

Before and after normalizing values from every column (to range <0, 1>) [advice gotten during presentation] base model and every modification lead to maximum accuracy of 84% and in some cases it fell to 83%. After experimenting a bit we couldn't improve accuracy past 84%. It probably was caused by the way we handled missing values (median for numerical, mode for categorical or because of the model we chose - neural networks.

7. Conclusions

The aim of this project was to develop a neural network model for predicting rain in Australia. The project utilized a dataset containing historical weather data, including attributes such as temperature, humidity, etc. By leveraging the power of neural networks, the model aimed to capture complex relationships and patterns in the data to somehow make accurate weather predictions.

This project helped us start our adventure with machine learning and artificial intelligence. We learned how neural networks work and how to prepare data for it. We can now create a simple neural network and train it to predict weather data in Australia with over 80% accuracy. There might be some room for improvement depending on what we are trying to improve - model (we chose neural networks, but maybe in this case something different will work better), code quality (“training cell” could be embedded in class or function to ease parametrization or data (other dataset or different approach to missing values).

Overall, this project proved to be an engaging and enlightening experience, enabling us to delve into the intricacies of machine learning and gain hands-on exposure to neural network-based prediction models.

8. Sources:

- https://www.projectpro.io/article/predictive-modelling-techniques/598
- https://medium.com/@satyavishnumolakala/linear-regression-pros-cons-62085314aef0
- https://dhirajkumarblog.medium.com/top-5-advantages-and-disadvantages-of-decision-tree-algorithm-428ebd199d9a
- https://medium.datadriveninvestor.com/random-forest-pros-and-cons-c1c42fb64f04
- https://merehead.com/blog/neural-network-architecture/
- http://www.bom.gov.au/climate/data/
- https://rdrr.io/cran/rattle.data/man/weatherAUS.html
- https://www.kaggle.com/code/karnikakapoor/rain-prediction-ann/input
- https://marutitech.com/top-8-deep-learning-frameworks/
- https://www.kaggle.com/code/avanwyk/encoding-cyclical-features-for-deep-learning

Github: https://github.com/Kadekk3939/australia-rain-prediction