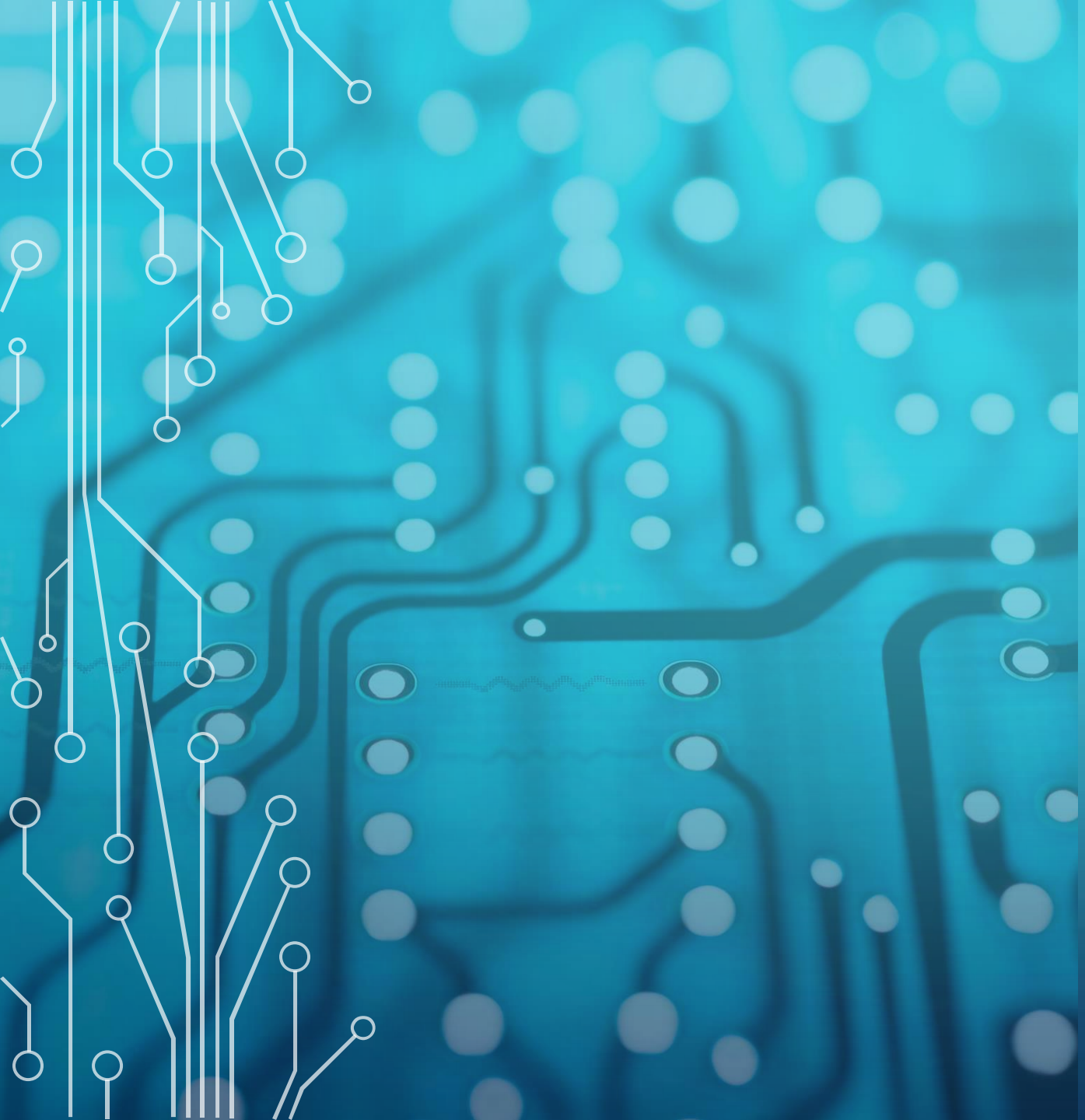
DIABETES PREDICTION

KATARZYNA SZCZEPAŃCZYK

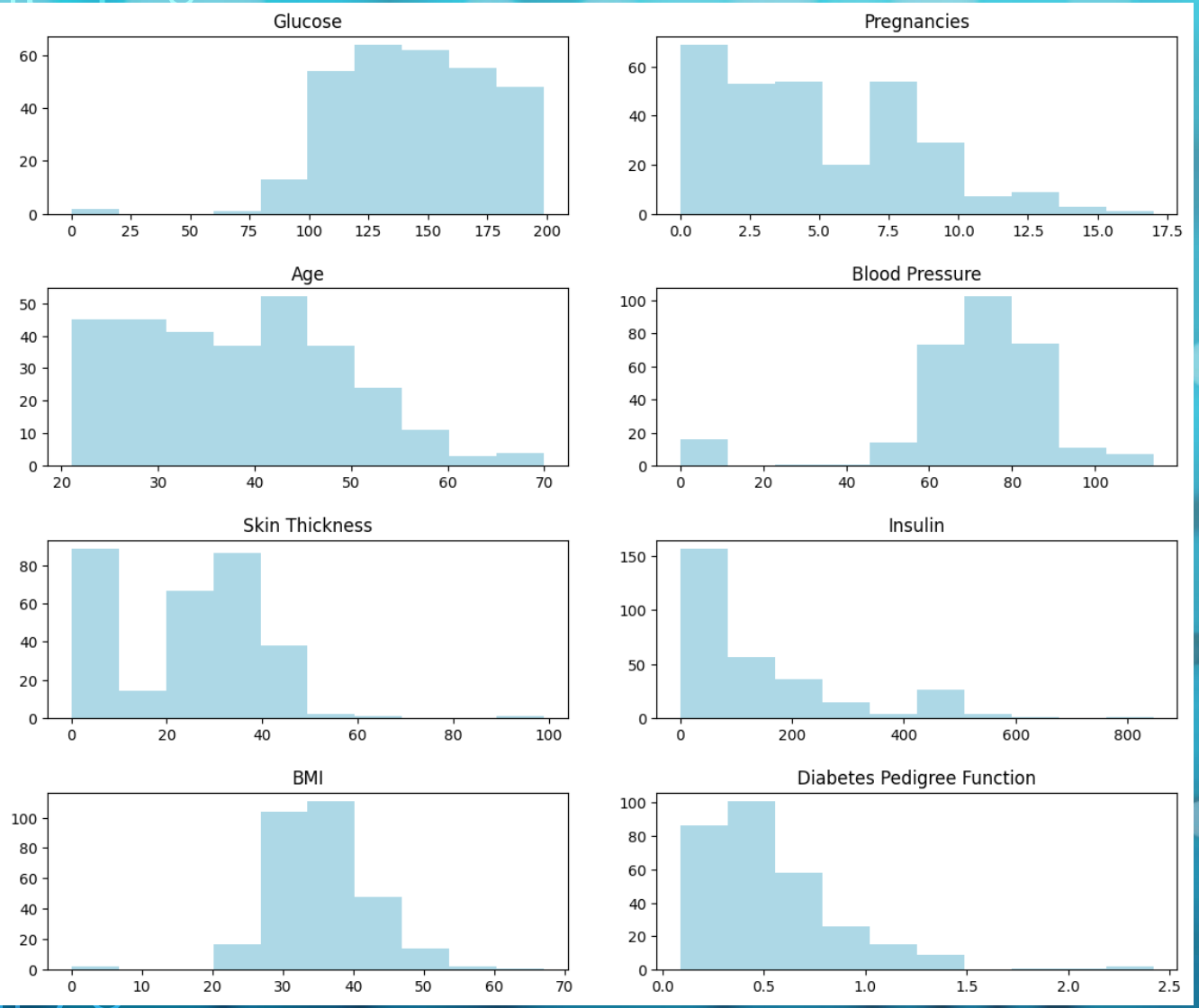
SZYMON WALASIK

05.06.2023



MAIN PROJECT GOALS

- Analyze medical data in order to predict if someone has diabetes
- Load diabetes dataset into program
- Create neural network
- Train neural network
- Get outcome from neural network
- Construct report about the results of our program



LOADING DATASET

- For that project we used dataset from www.kaggle.com.
- This dataset contains following medical measurements :

1. Pregnancies: Number of imes a person was pregnant
2. Glucose: Plasma glucose concentration at 2 hours in an oral glucose tolerance test
3. Blood Pressure: Diastolic blood pressure (mm Hg)
4. Skin Thickness: Triceps skin fold thickness (mm)
5. Insulin: 2-Hour serum insulin (mu U/ml)
6. BMI: Body mass index (weight in kg/(height in m)^2)
7. Diabetes Pedigree Function
8. Age (years)
9. Outcome: Class variable (0 or 1)

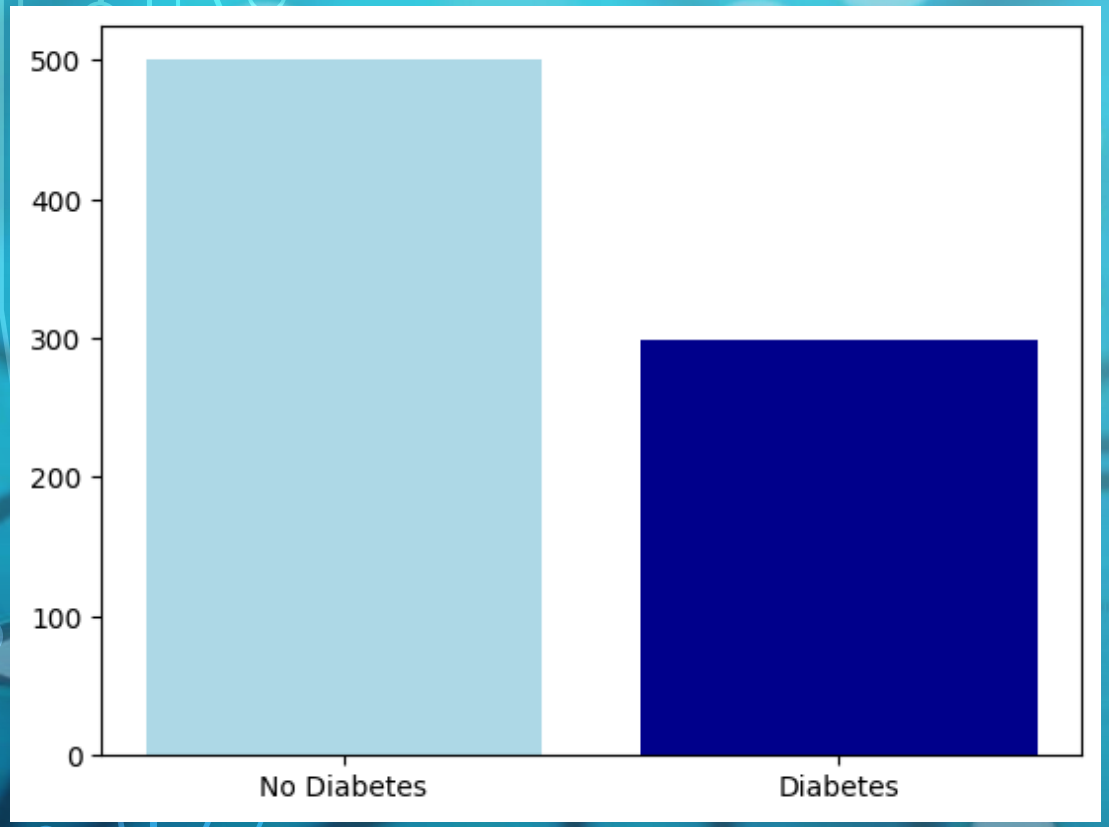
DIABETES PEDIGREE FUNCTION

- The "diabetes pedigree function," also known as the "diabetes pedigree index," is a numerical score that assesses the genetic risk of developing diabetes mellitus. It is commonly used in medical genetics and research to evaluate the likelihood of a person developing diabetes based on their family history.

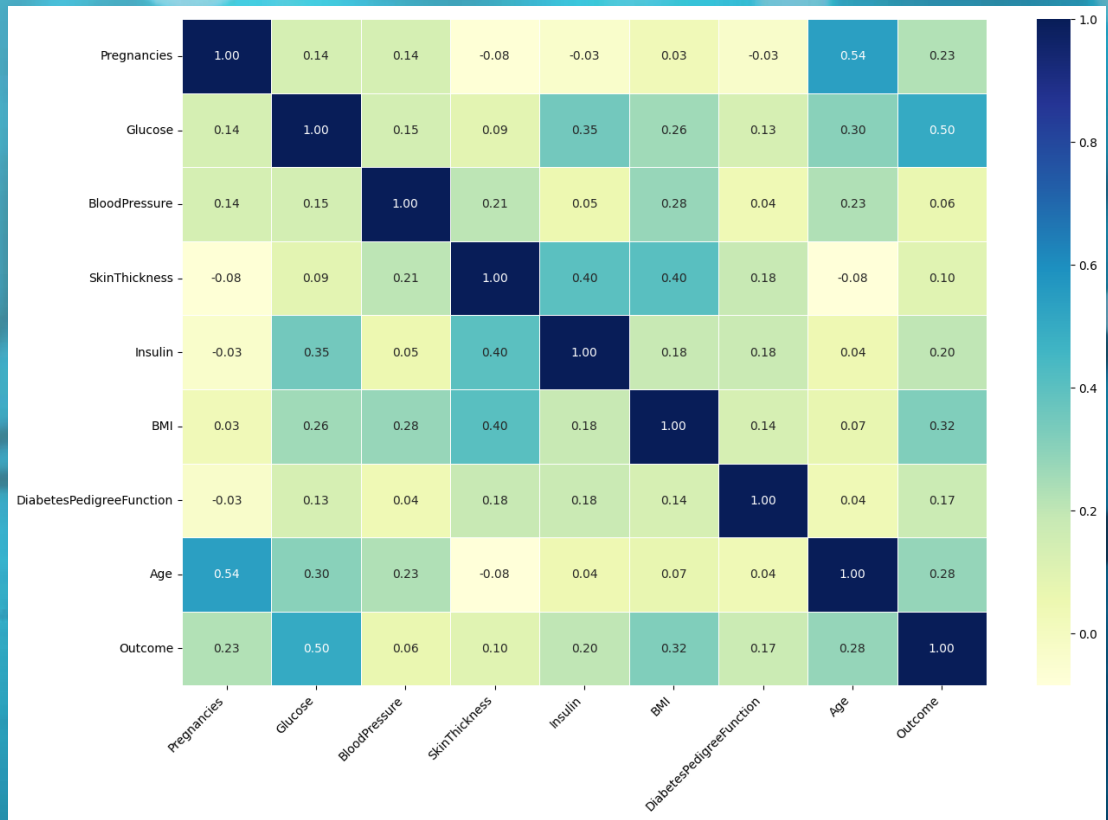
- The diabetes pedigree function takes into account the presence of diabetes in an individual's relatives, including parents, siblings, and grandparents. Each affected relative is assigned a weight or value based on their degree of relatedness and the age at which they were diagnosed with diabetes. The function calculates a score by summing up these weights for all affected relatives.

- The formula for calculating the diabetes pedigree function varies slightly depending on the specific study or application. However, a common approach is to assign a value of 0.1 for each affected sibling, 0.2 for each affected parent, and 0.3 for each affected child. The scores are then summed up to determine the overall risk.

DATASET OUTCOMES



CORRELATION MAP



```
# Definition of neural network model
model = Sequential()
model.add(BatchNormalization(input_shape=(X.shape[1],)))
model.add(Dense(20, input_dim=X.shape[1], activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(15, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(5, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
```

MODEL CREATION

During the creation of our model we used three types of layers:

- Batch normalization layer
- Dense layer
- Dropout layer

BATCH NORMALIZATION LAYER

Batch normalization is a technique commonly used in neural networks to improve the training speed and stability of the model. It normalizes the activations of a given layer by subtracting the batch mean and dividing by the batch standard deviation. This normalization process is applied independently to each feature dimension.

The formula for batch normalization can be expressed as follows:

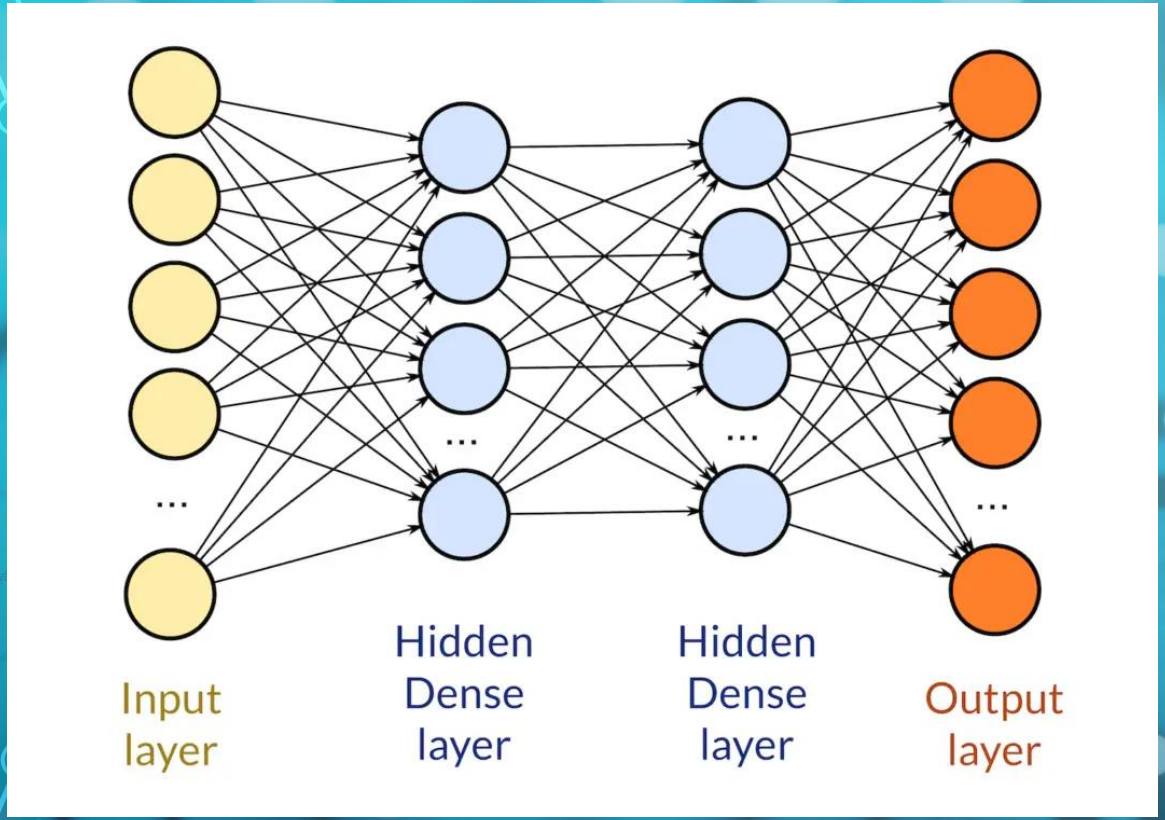
$$\mathrm{BN}(x) = \gamma \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta$$

where:

- x is the input to the batch normalization layer.
- μ is the batch mean.
- σ^2 is the batch variance
- γ is a learnable scale parameter.
- β is a learnable shift parameter.
- ϵ is a small constant (e.g., 10^-5) added to the denominator for numerical stability.

The benefits of using batch normalization include:

- Improved training speed
- Stabilized training
- Reduced sensitivity to hyperparameters
- Generalization improvement

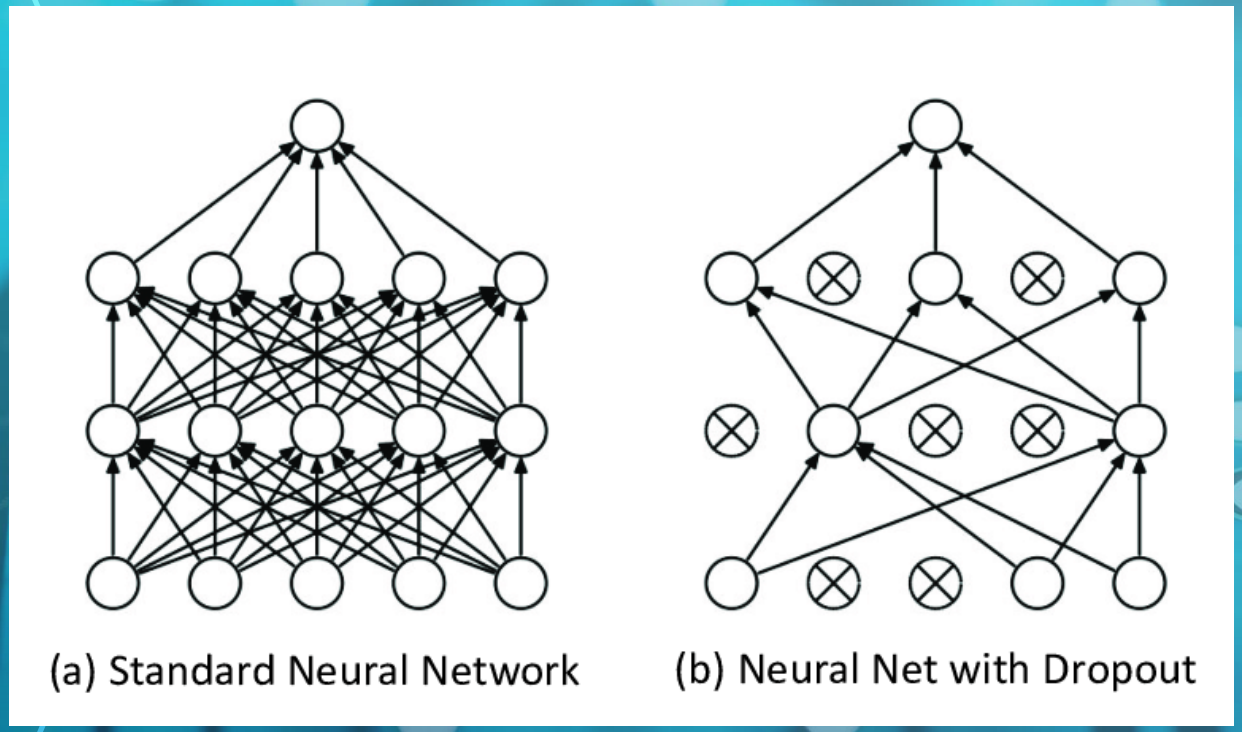


DENSE LAYER

A dense layer, also known as a fully connected layer, is one of the basic types of layers used in neural networks. In a dense layer, every neuron is connected to every neuron in the previous layer, creating a fully connected network between the layers.

Mathematically, a dense layer transforms an input feature vector of size n into an output feature vector of size m. Each neuron in the dense layer computes a weighted sum of its inputs and passes it through an activation function to generate an output.

DROPOUT LAYER



The term “dropout” refers to dropping out the nodes (from input and hidden layer) in a neural network. All the forward and backwards connections with a dropped node are temporarily removed, which creates a new network architecture out of the parent network. The nodes are dropped by a dropout probability of p which is a hyperparameter typically set between 0.2 and 0.5.

HOW WE TESTED OUR MODEL?

```
# Kompilacja modelu
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Trenowanie modelu
early_stopping = EarlyStopping(monitor='val_loss', patience=100)

history = model.fit(X_train, y_train, epochs=100, batch_size=10, validation_split=0.2, callbacks=[early_stopping])

# Testowanie modelu
_, accuracy = model.evaluate(X_test, y_test)
print('Accuracy: %.2f' % (accuracy*100))
```

RESULTS

```
import pandas as pd
metrics = pd.DataFrame(history.history)
metrics['epoch'] = history.epoch
metrics
```

	loss	accuracy	val_loss	val_accuracy	epoch
0	0.655591	0.592476	0.646900	0.6500	0
1	0.639157	0.608150	0.628504	0.7250	1
2	0.632632	0.645768	0.618612	0.6750	2
3	0.620723	0.630094	0.613049	0.7000	3
4	0.627020	0.683386	0.606259	0.7375	4
...	...	...	...	...	...
95	0.529131	0.714734	0.522163	0.7500	95
96	0.535221	0.695925	0.521345	0.7500	96
97	0.552190	0.670846	0.522877	0.7750	97
98	0.487411	0.727273	0.525803	0.7750	98
99	0.538572	0.699060	0.527312	0.7625	99

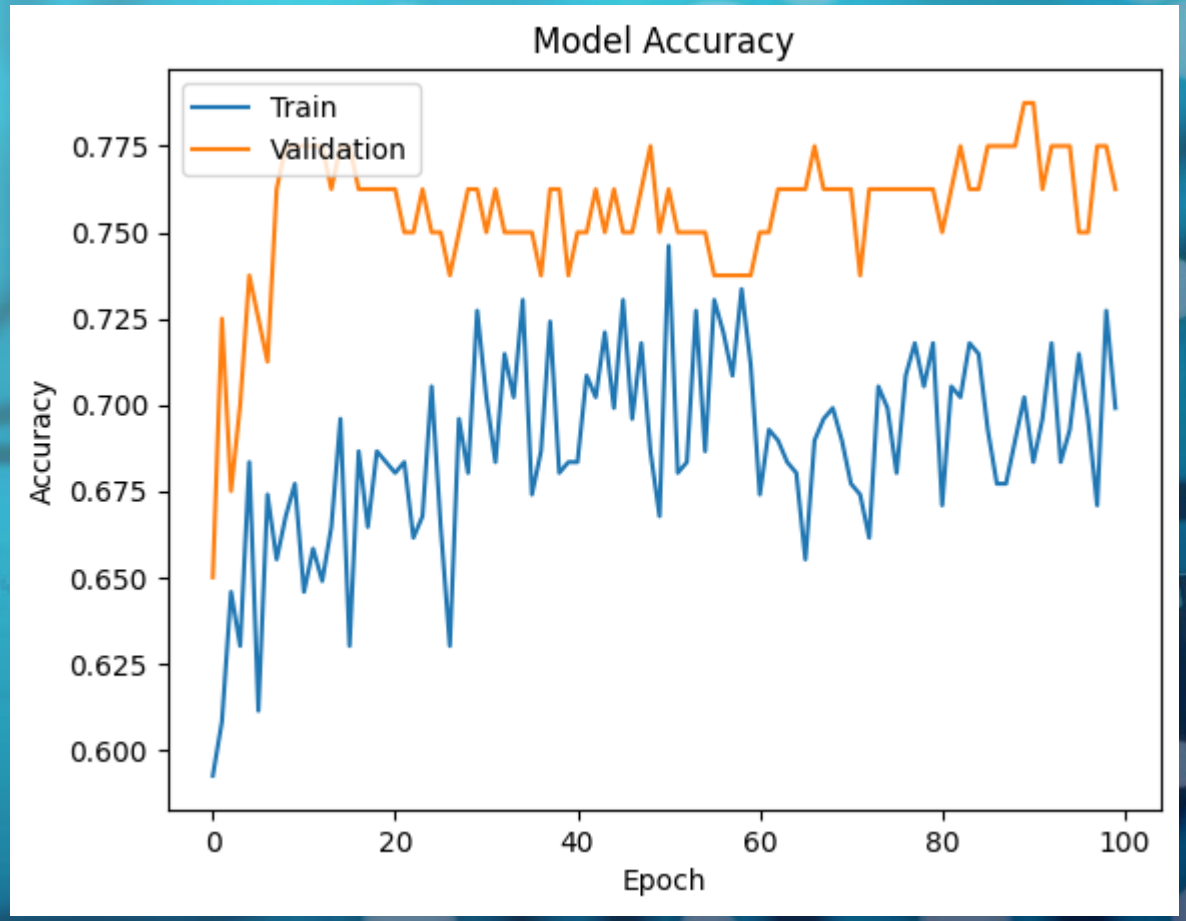
100 rows × 5 columns

MODEL ACCURACY

```python
import matplotlib.pyplot as plt


# Wyświetlanie wykresu dokładności treningu i walidacji

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

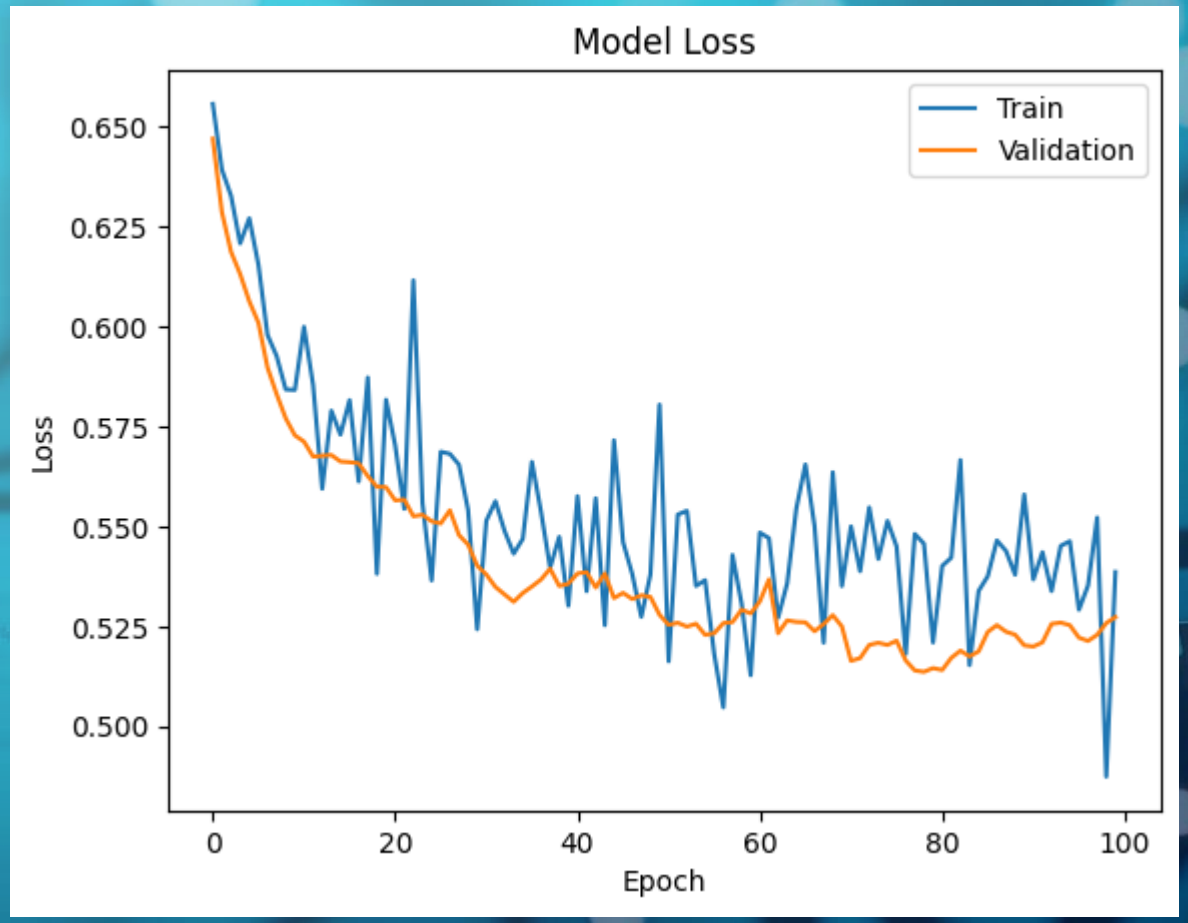


MODEL LOSS

```python
import matplotlib.pyplot as plt

# Wyświetlanie wykresu straty

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper right')
plt.show()
```



THANK YOU FOR YOUR ATTENTION!

KATASZC870@STUDENT.POLSL.PL

SZYMWAL050@STUDENT.POLSL

05.06.2023