Laboratorium 7 – Deep Learning – sztuczne sieci neuronowe

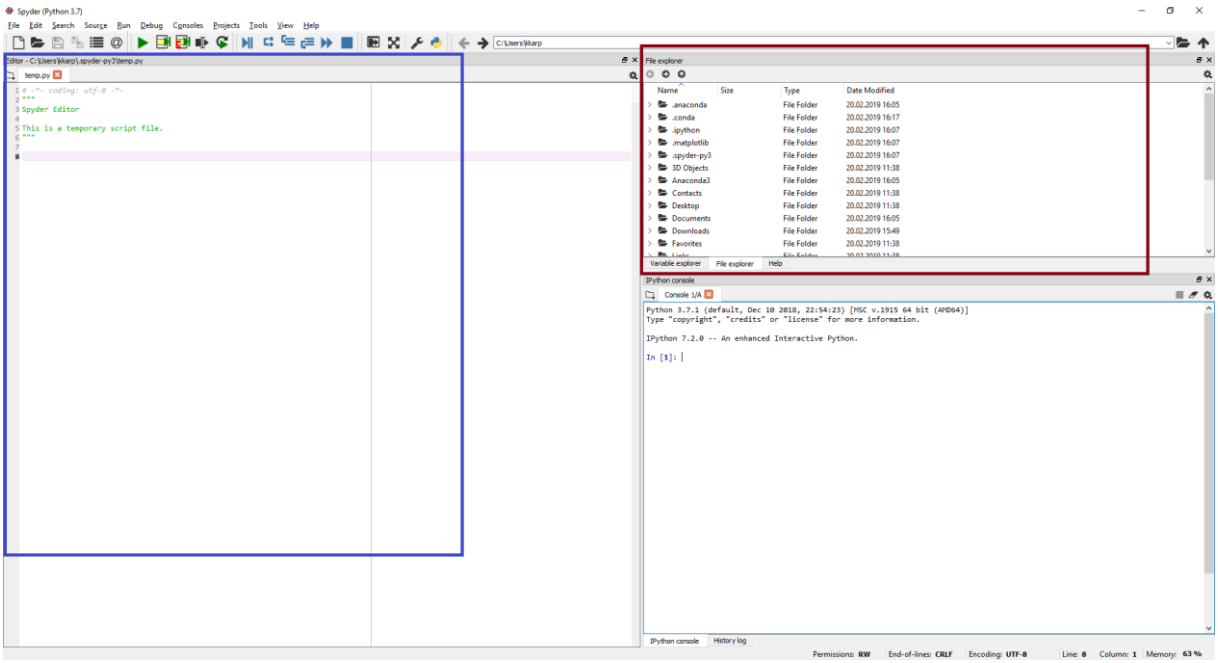
Deep learning to klasa algorytmów uczenia maszynowego, które wykorzystują wiele warstw do stopniowego wyodrębniania cech wyższego poziomu z surowych danych wejściowych. Na przykład w przetwarzaniu obrazu niższe warstwy mogą identyfikować krawędzie, podczas gdy wyższa warstwa może identyfikować elementy znaczące dla człowieka, takie jak cyfry/litery lub twarze.

Zadania laboratoryjne:

1. Przygotowanie środowiska do pracy.
   1.1. Pobrać i zainstalować platformę Anaconda (https://www.anaconda.com/distribution/).
   1.2. Uruchomić platformę Anaconda, a następnie aplikację Spyder.
   1.3. Na komputerze utworzyć nowy folder i zapisać w nim otrzymany od prowadzącego plik z danymi (Mall_Customers.csv).

UWAGA! Pliki mogą być wykorzystywane podczas kolejnych laboratoriów, dlatego po zakończeniu zajęć należy skopiować folder na prywatny nośnik.

   1.4. W programie Spyder ustawić swój folder jako katalog roboczy (obszar zaznaczony czerwonym prostokątem na Rysunku 1.). Zapisać pod dowolną nazwą plik temp.py (widoczny w obszarze zaznaczonym niebieskim prostokątem na Rysunku 1.) w tym samym folderze co plik ze zbiorem danych.



Rysunek 1. Interfejs programu Spyder.

2. **Zadania do wykonania**
   Wykorzystać plik Churn_Modelling.csv do wytrenowania sieci neuronowej tak, aby określała którzy klienci rezygnują lub nie rezygnują z usług banku.
   2.1. Instalacja biblioteki Keras.
   Uruchomić anaconda prompt (Programy > Anaconda > Anaconda Prompt). Wpisać polecenie:

```
conda install -c conda-forge keras
```

   2.2. Zaimportować biblioteki.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

2.3. Wczytać zbiór danych Churn_Modelling.csv. Utworzyć macierz zmiennych niezależnych i wektor zmiennych zależnych.

```
dataset = pd.read_csv('Churn_Modelling.csv')
X = dataset.iloc[:, 3:13].values
y = dataset.iloc[:, 13].values
```

2.4. Zakodować dane kategoryczne (kolumny „Geography" i „Gender").

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X_1 = LabelEncoder()
X[:, 1] = labelencoder_X_1.fit_transform(X[:, 1])
labelencoder_X_2 = LabelEncoder()
X[:, 2] = labelencoder_X_2.fit_transform(X[:, 2])
onehotencoder = OneHotEncoder(categorical_features = [1])
X = onehotencoder.fit_transform(X).toarray()
X = X[:, 1:]
```

2.5. Podzielić zestaw danych na zestawy testowy i trenignowy.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.2, random_state = 0)
```

2.6. Przeskalować (dokonać normalizacji) danych.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

2.7. Zaimportować bibliotekę Keras oraz odpowiednie pakiety.

```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

2.8. Inicjalizacja sieci neuronowej (definiowanie sekwencji warstw lub inaczej: definiowanie grafu).

```
classifier = Sequential()
```

2.9. Dodawanie warstwy wejściowej oraz pierwszej ukrytej warstwy.

```
classifier.add(Dense(output_dim = 6, init = 'uniform',
activation = 'relu', input_dim = 11))
```

2.10. Dodawanie drugiej ukrytej warstwy.

```
classifier.add(Dense(output_dim = 6, init = 'uniform',
activation = 'relu'))
```

2.11. Dodawanie warstwy wyjściowej.

```
classifier.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))
```

2.12. Kompilacja sieci neuronowej.

```
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

2.13. Dopasowanie sieci neuronowej do zestawu treningowego.

```
classifier.fit(X_train, y_train, batch_size = 10, nb_epoch = 100)
```

2.14. Wykorzystanie sieci neuronowej do predykcji danych wyjściowych na podstawie zestawu testowego.

```
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)
```

2.15. Utworzyć macierz błędów.

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)
```

2.16. Wykonać sprawozdanie z realizacji podpunktów 2.1 – 2.15. Przesłać w formie pliku pdf do serwisu moodle. Plik powinien zawierać następujące informacje:

- Należy porównać dane z y_test z danymi przewidzianymi przez sieć neuronową. Czy utworzona siec neuronowa dokonała poprawnej predykcji?
- Zamieścić screen z pierwszych kilku wierszy y_test i y_pred.
- Przeanalizować macierz błędów. Wskazać ilu sieć neuronowa dokonała poprawnych predykcji, że klient odejdzie z banku, a ile błędnych? Analogicznie ilu sieć neuronowa dokonała poprawnych predykcji, że klient nie odejdzie z banku, a ile błędnych?