Міністерство освіти і науки України
Національний технічний університет України «КПІ» імені Ігоря Сікорського
Кафедра обчислювальної техніки ФІОТ

ЗВІТ
з лабораторної роботи №6
з навчальної дисципліни «Вступ до технології Data Science»

Тема:

РЕАЛІЗАЦІЯ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ
(Artificial Neural Networks)

Виконав:
Студент 3 курсу кафедри ІПІ ФІОТ,
Навчальної групи ІП-11
Сідак К.І.

Перевірив:
Професор кафедри ОТ ФІОТ
Писарчук О.О.

Київ 2023

I. Мета роботи:

Виявити дослідити та узагальнити особливості підготовки різних типів даних, синтезу, навчання та застосування штучних нейронних мереж (Artificial Neural Networks).

II. Завдання:

Розробити програмний скрипт мовою Python що реалізує обчислювальний алгоритм за технологіями штучних нейронних мереж (Artificial Neural Networks): підготовка даних; конструювання нейромережі; навчання штучної нейронної мережі; застосування нейромережі (класифікація / ідентифікація / прогнозування):

III рівень складності 9 балів за самостійним вибором напрямку:

1. Відповідно до технічних умов, табл.2 додатку, але в якості Data Set – обрати реальні дані у форматі числових / часових рядів, наприклад, як результат виконання лабораторних робіт із статистичного навчання (парсинг самостійно обраного сайту).

III. Виконання лабораторної роботи.

3.1. Зчитування даних та їх нормалізація

Для виконання даної лабораторної роботи я використав реальні дані по щоденному курсу франку до долара. Оскільки ці дані вже були завантажені у відповідний csv файл, то для отримання даних я зчитав цей файл у датафрейм.

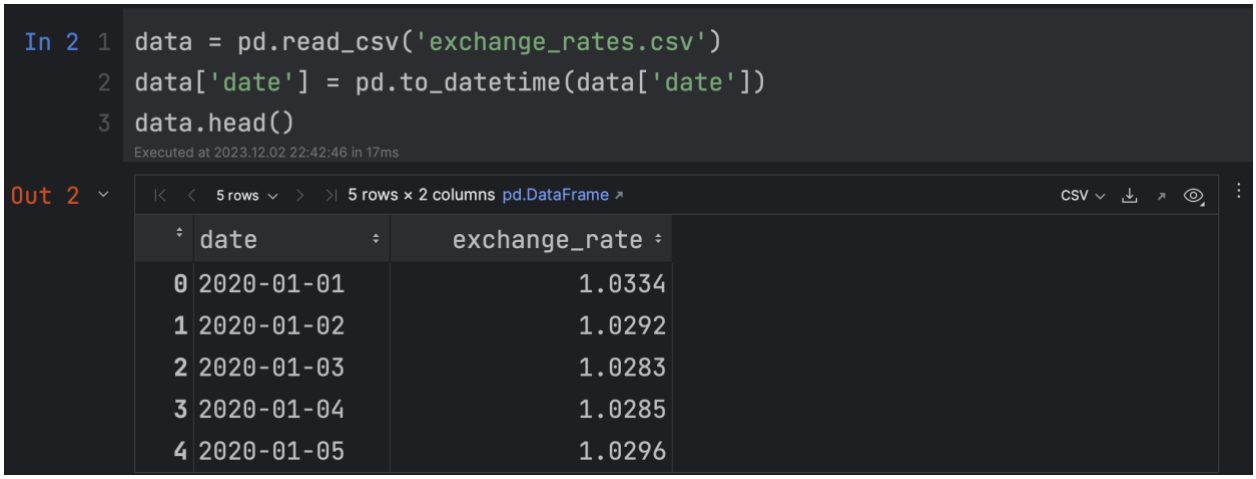

Рис 3.1 – Зчитування даних у датафрейм

Наступним кроком я нормалізував дані, а саме курс франку до долара, для покращення ефективності навчання нейронної мережі та зберіг цей нормалізований масив в окремому стовпці.

```
In 3
data['normalized_rate'] = MinMaxScaler().fit_transform
 (data[['exchange_rate']])
data.head()
```
Executed at 2023.12.02 22:42:46 in 10ms

Out 3

5 rows × 3 columns pd.DataFrame ↗ CSV

	date	exchange_rate	normalized_rate
0	2020-01-01	1.0334	0.259755
1	2020-01-02	1.0292	0.236343
2	2020-01-03	1.0283	0.231327
3	2020-01-04	1.0285	0.232441
4	2020-01-05	1.0296	0.238573

Рис. 3.2 – Нормалізація даних

3.2. Створення навчальної та тестової вибірки

Моя нейронна мережа буде прогнозувати курс франку в конкретний день на основі даних за попередні 7 днів (тиждень). Я обрав саме тиждень, адже, враховуючи невеликий розмір датасету, такий відносно невеликий проміжок не буде сприяти перенавчанню моделі та при цьому є більш логічним брати саме тиждень, а не, скажімо, 5 днів чи 8 днів, які не є цілою кількістю тижнів, місяців і т.д. На основі даної умови я створив відповідні навчальні та тестові вибірки у відношенні 80% до 20% відповідно.

```
In 4
def create_dataset(dataset, look_back):
    X, Y = [], []
    for i in range(len(dataset) - look_back - 1):
        X.append(dataset[i:(i + look_back)])
        Y.append(dataset[i + look_back])
    return np.array(X), np.array(Y)


look_back = 7
train_size = int(len(data) * 0.8)
train_data, test_data = data[0:train_size], data[train_size:len(data)]
X_train, y_train = create_dataset(train_data['normalized_rate'].values, look_back)
X_test, y_test = create_dataset(test_data['normalized_rate'].values, look_back)
```
Executed at 2023.12.03 00:37:02 in 12ms

Рис. 3.3 – Створення навчальної та тестової вибірки

3.3. Створення архітектури нейронної мережі та її навчання

Для прогнозування курсу франку я створив нейронну мережу з двома прихованими прошарками, що містять 8 та 4 нейрони відповідно. Вхідний прошарок містить 5 елементів, вихідний – 1 нейрон. В якості функцій активації я обрав ReLU через те, що вона сприяє швидкому навчанню шляхом градієнтного спуску та цільова змінна є неперервною. Вихідний прошарок використовує лінійну функцію активації, адже ми маємо задачу прогнозування (регресії). Кількість нейронів у прихованих прошарках я брав невелику (менше, ніж 2 * розмірність вхідного прошарку) через малий об’єм даних для зменшення ризику перенавчання, а кількість нейронів другого прошарку брав меншою, ніж для першого, що є досить частим рішенням у подібних задачах. Крім того, для зручності ці кількості нейронів є степенями двійки.

Наступним кроком я скомпілював модель, використовуючи функцію в якості функцію втрат MSE (адже задача є прогнозування неперервної змінної), тобто,

використовував норму мінімізації квадрату невязки. В якості коефіцієнту навчання я взяв значення 0.001.

Навчання моделі я здійснював з використанням mini-batch gradient descent, тобто для розповсюдження (propagation) на кожній ітерації використовується лише частина всього навчального датасету (32 у моєму випадку, що є значенням за замовчуванням та після експериментів з різними значеннями (16, 64, 256) я виявив, що воно є оптимальним). Всього я використовую 100 епох для навчання моделі (на кожній епосі відбувається декілька ітерацій з відповідним batch_size, щоб усі дані з навчальної вибірки в результаті були використані), проте варто зазначити, що навчання може закінчитись раніше, якщо після певної епохи наступні 20 епох не будуть давати покращення цільової функції втрат на тестовому наборі (це реалізовано за допомогою early_stopping). Також я зберігаю модель з найнижчим значенням цільової функції втрат на тестовому наборі у відповідному файлі.

```
In 5
model = Sequential()
model.add(Dense(8, input_dim=look_back, activation='relu'))
model.add(Dense(4, activation='relu'))
model.add(Dense(1))

model.compile(optimizer=Adam(learning_rate=0.001), loss='mean_squared_error')
early_stopping = EarlyStopping(monitor='val_loss', patience=20)
model_checkpoint = ModelCheckpoint('best_model.h5', monitor='val_loss', save_best_only=True)

history = model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test, y_test),
                    verbose=1,
                    callbacks=[early_stopping,
                               model_checkpoint])
Executed at 2023.12.03 14:26:54 in 4s 59ms

Epoch 1/100
 1/35 [..............................] - ETA: 4s - loss: 0.1423

2023-12-03 14:26:50.102047: W tensorflow/tsl/platform/profile_utils/cpu_utils.cc:128] Failed to get CPU frequency: 0 Hz
Epoch 96/100
35/35 [==============================] - 0s 1ms/step - loss: 7.6156e-04 - val_loss: 8.1569e-04
Epoch 97/100
35/35 [==============================] - 0s 1ms/step - loss: 7.4337e-04 - val_loss: 8.2580e-04
Epoch 98/100
35/35 [==============================] - 0s 783us/step - loss: 7.4164e-04 - val_loss: 8.2801e-04
Epoch 99/100
35/35 [==============================] - 0s 1ms/step - loss: 7.4272e-04 - val_loss: 8.1431e-04
Epoch 100/100
35/35 [==============================] - 0s 1ms/step - loss: 7.2879e-04 - val_loss: 7.9413e-04
```

Рис. 3.4 – Створення та навчання нейронної мережі

Після навчання моделі я візуалізував залежність точності прогнозування числового ряду від кількості епох навчання.

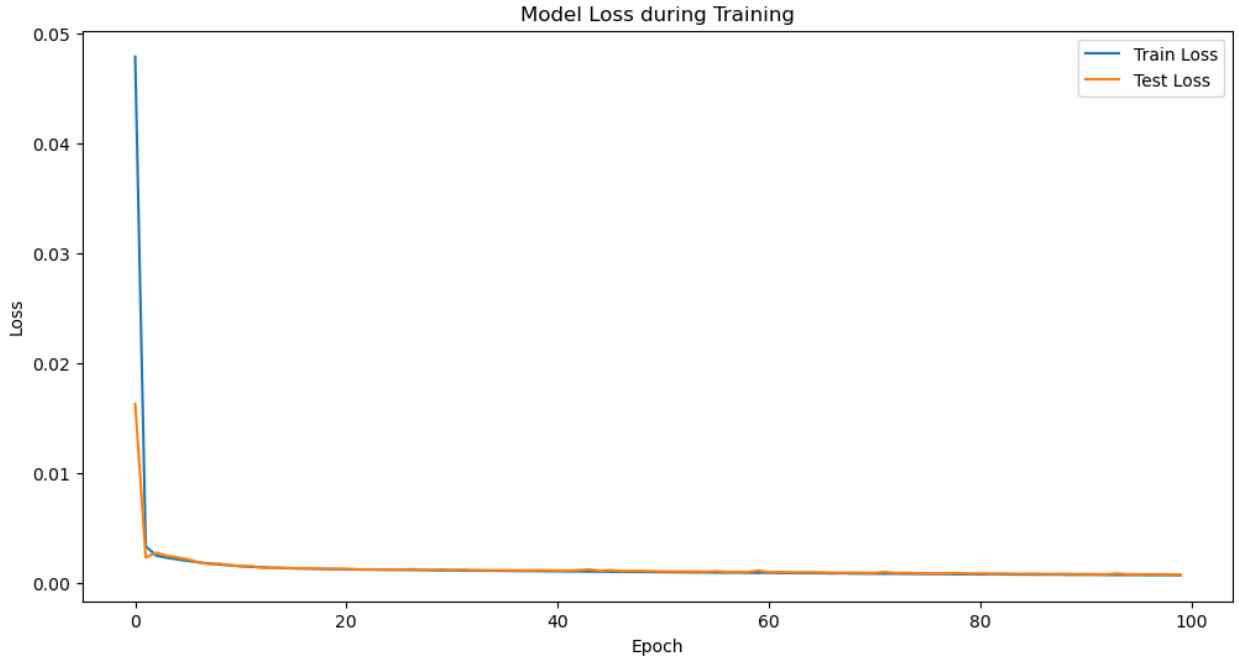

Рис. 3.5 – Залежність точності прогнозування числового ряду від кількості епох навчання

Можна побачити, що на перших епохах відбувається суттєве покращення з кожною наступною епохою. Надалі зменшення значення цільової функції є не таким суттєвим, проте все одно цей тренд спостерігається. Зокрема на 60-х епохах спостерігається покращення (рис 3.6) та далі до останньої 100-ї епохи (рис. 3.4) (саме тому навчання відбувалося всі 100 епох та не зупинилось раніше).

```
Epoch 62/100
35/35 [==============================] - 0s 942us/step - loss: 9.4212e-04 - val_loss: 0.0010
Epoch 63/100
35/35 [==============================] - 0s 1ms/step - loss: 9.2970e-04 - val_loss: 9.9978e-04
Epoch 64/100
35/35 [==============================] - 0s 951us/step - loss: 9.2528e-04 - val_loss: 9.9849e-04
Epoch 65/100
35/35 [==============================] - 0s 861us/step - loss: 9.2419e-04 - val_loss: 0.0010
Epoch 66/100
35/35 [==============================] - 0s 1ms/step - loss: 9.1895e-04 - val_loss: 9.9012e-04
Epoch 67/100
35/35 [==============================] - 0s 1ms/step - loss: 8.9615e-04 - val_loss: 9.6839e-04
Epoch 68/100
35/35 [==============================] - 0s 1ms/step - loss: 9.1403e-04 - val_loss: 9.5917e-04
Epoch 69/100
35/35 [==============================] - 0s 667us/step - loss: 8.8849e-04 - val_loss: 9.6228e-04
Epoch 70/100
35/35 [==============================] - 0s 1ms/step - loss: 8.8304e-04 - val_loss: 9.5665e-04
Epoch 71/100
35/35 [==============================] - 0s 1ms/step - loss: 8.7653e-04 - val_loss: 9.3754e-04
```

Рис. 3.6 – Частина історії навчання

Крім того, як на графіку, так і з історії видно, що різниця між значенням функції втрат на навчальному та тестовому наборах не є суттєвою, що свідчить про відсутність перенавчання.

Завантаживши найкращу модель з файлу, я вивів значення функції втрат цієї моделі на тестовому наборів та переконався, що це значення співпадає зі значення на 100 епосі., тобто протягом уснаього навчання точність прогнозування моделі покращувалась.

```
In 8  1 best_model = load_model('best_model.h5')
      2 val_loss = best_model.evaluate(X_test, y_test)
      3 print(f'Validation loss: {val_loss:.5f}')
Executed at 2023.12.03 14:26:54 in 85ms

  9/9 [==============================] - 0s 556us/step - loss: 7.9413e-04
  Validation loss: 0.00079
```

Рис. 3.7 – Значення функції втрат найкращої моделі

3.4. Відображення процесу прогнозування у формі графіків

Останнім кроком я візуалізував прогнозовані значення курсу франку та реальні значення курсу для тестового набору. Варто зазначити, що модель прогнозує нормалізовані значення (від 0 до 1), і в аналогічній шкалі значення тестового набору, тому я використав вже навчений об’єкт MinMaxScaler, щоб перетворити дані в оригінальну шкалу.

```
In 9  1 def plot_predictions(model, X_data, Y_data):
      2     plt.figure(figsize=(12, 6))
      3     y_pred_scaled = model.predict(X_data)
      4     y_pred = min_max_scaler.inverse_transform(y_pred_scaled)
      5     actual_rate = min_max_scaler.inverse_transform(Y_data.reshape(-1, 1))
      6     plt.plot(actual_rate, label='Actual Rate')
      7     plt.plot(y_pred, label='Predicted Rate')
      8     plt.legend()
      9     plt.show()
Executed at 2023.12.03 14:26:54 in 6ms

In 10 1 plot_predictions(best_model, X_test, y_test)
Executed at 2023.12.03 14:26:54 in 197ms
```

Рис. 3.8 – Функція для відображення процесу прогнозування

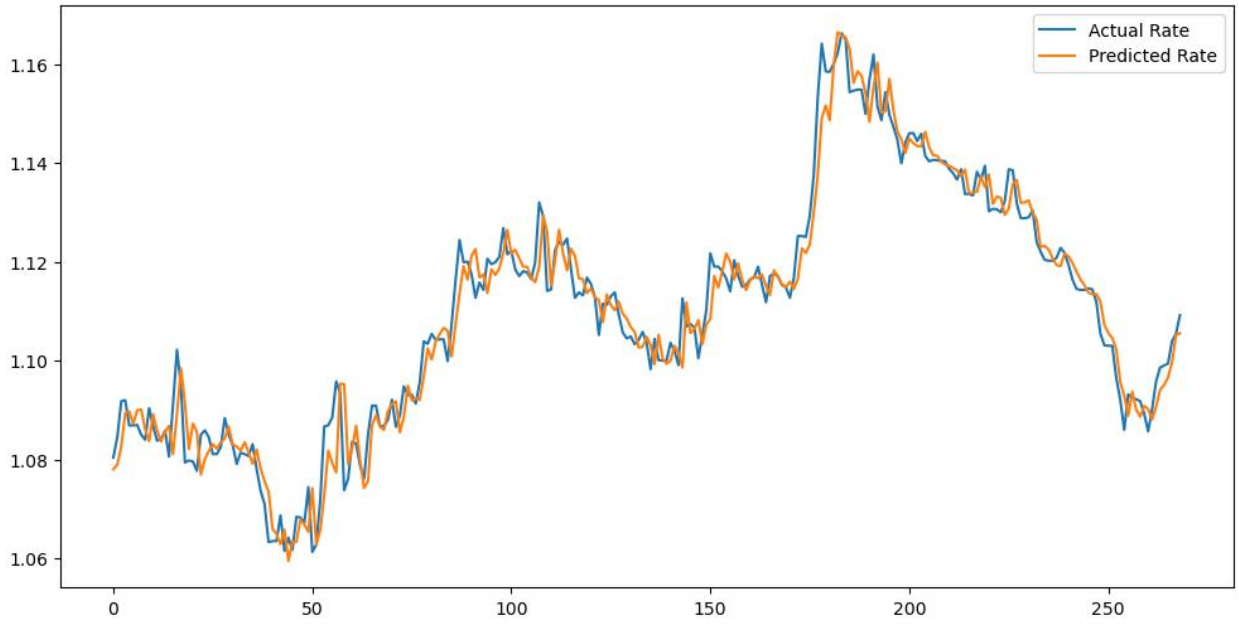

Рис. 3.9 – Візуалізація прогнозованого та реального курсу

IV. Висновок

Отже, в ході даної лабораторної я застосував на практиці навички трансформації даних та проектування архітектури та навчання штучної нейронної мережі для прогнозування часових рядів на основі попередніх даних за певний період. Крім того, я візуалізував залежність точності прогнозування (значення функції втрат) від кількості епох навчання, а також прогнозовані значення курсу франку та реальні для тестового набору. Отримані результати свідчать, що модель не є перенавченою та показує досить непогані результати на моїх даних. Варто зазначити, що через випадкову ініціалізацію вагових коефіцієнтів для прошарків нейромережі, наявність випадкової складової в процесі навчання з mini-batch gradient descent та відносно невеликий об’єм даних нейронна мережа буде показувати певну мінливість результатів з кожним запуском програми, проте загалом вона все одно дає непоганий результат при швидкому навчанні та не є перенавченою.