**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 «Технології Data Science»**

**Тема:**

**РЕАЛІЗАЦІЯ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ**

**(Artificial Neural Networks)**

**Виконав:**

Студент 4 курсу кафедри ІПІ ФІОТ,

Навчальної групи ІП-11

Трикош І. В.

**Перевірив:**

Професор кафедри ОТ ФІОТ

Писарчук О.О.

**Київ 2024**

## **І. Мета:**

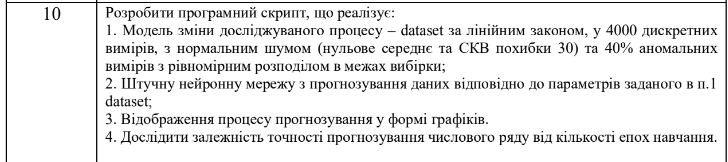
Виявити дослідити та узагальнити особливості підготовки різних типів даних, синтезу, навчання та застосування штучних нейронних мереж (Artificial Neural Networks).

## **ІІ. 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):

**ІІІ 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).



Як dataset використано дані про індекс S&P 500, які було отримано під час парсингу в ході виконання ЛР1.

## **ІІІ. Результати виконання лабораторної роботи.**

* 1. **Синтезована математична модель**

Відповідно до умов задачі синтезовано математичну модель, що реалізує штучну нейронну мережу з прогнозування даних про індекс S&P 500.

Штучна нейронна мережа — це обчислювальна модель, що імітує принципи роботи біологічних нейронних мереж і використовується для вирішення задач машинного навчання та штучного інтелекту. Основні елементи:

1. Вхідний шар — приймає вхідні дані, причому кількість нейронів відповідає кількості ознак даних.
2. Приховані шари — складаються з певної кількості нейронів залежно від задачі, формуючи внутрішнє представлення даних.
3. Вихідний шар — генерує результати на основі обробленої інформації з прихованих шарів.
4. Ваги або зв’язки — визначають, як вхідні дані впливають на кожен нейрон. Значення ваг змінюються в процесі навчання.
5. Функція активації — перетворює зважену суму вхідних сигналів у вихід нейрона.
6. Функція втрат — вимірює різницю між прогнозом моделі та фактичними значеннями. Завдання мережі — мінімізувати це значення.
7. Оптимізатор — алгоритм, який коригує ваги для зменшення помилки, визначеної функцією втрат.

Штучна нейронна мережа використовуватиме два шари LSTM з 20 та 10 нейронами відповідно. LSTM (long short-term memory) – це рекурентна нейронна мережа, що добре підходить у тому числі для прогнозу часових рядів. Вона складається з трьох вентилів: вхідний, вихідний, забуття. Вхідний вентиль вирішує, яку нову інформацію додати до стану. Вихідний вентиль вирішує, яка частина стану передається у прихований стан. Вентиль забуття вирішує, яку частину інформації треба забути. Ваги, що є в цій мережі, використовуються для спрямування дії вентилів. Цю мережу тренують з використанням зворотного поширення в часі. Щоб використовувати LSTM, скористаємося бібліотекою keras.

* 1. **Результати архітектурного проектування та їх опис**

На рис. 1 зображена блок-схема до програмного скрипту:

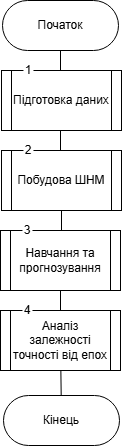


Рисунок 1 – Блок схема алгоритму

Робота алгоритму розпочинається зі зчитування даних із файлу та їх попередня обробка і підготовка. Це включає нормалізацію та створення вхідних і вихідних даних (для прогнозу наступного значення будемо використовувати сім попередніх значень). Ці дії виконуються у блоці 1 блок-схеми алгоритму рис.1.

У блоці 2 будується штучна нейронна мережа з двома шарами LSTM (по 20 та 10 нейронів), з оптимізатором adam та середньоквадратичною помилкою.

У блоці 3 ШНМ навчається та прогнозує значення індексу S&P 500. Також оцінюється точність прогнозу та візуалізуються результати.

У блоці 4 відбувається аналіз залежності точності від кількості епох, а саме виведення графіка помилок та кількості епох. На цьому виконання алгоритму завершене.

* 1. **Опис структури проекту програми**

Для реалізації розроблених алгоритмів мовою програмування Python з використанням можливостей інтегрованого середовища PyCharm сформовано проєкт.

Проєкт базується на лінійній бізнес-логіці функціонального програмування та має таку структуру.

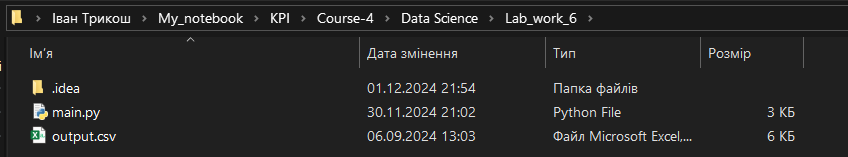


Рисунок 2 – Структура проєкту

Lab\_work\_6 – головний каталог проєкту

main.py – файл програмного коду

output.csv – дані за рік про значення індексу S&P 500

* 1. **Результати роботи програми відповідно до завдання (допускається у формі скриншотів)**

Результатом роботи програми є сукупність послідовності графічних вікон, що реалізують умови завдання лабораторної роботи.

1. На рисунках 3-9 зображено результати роботи алгоритму:

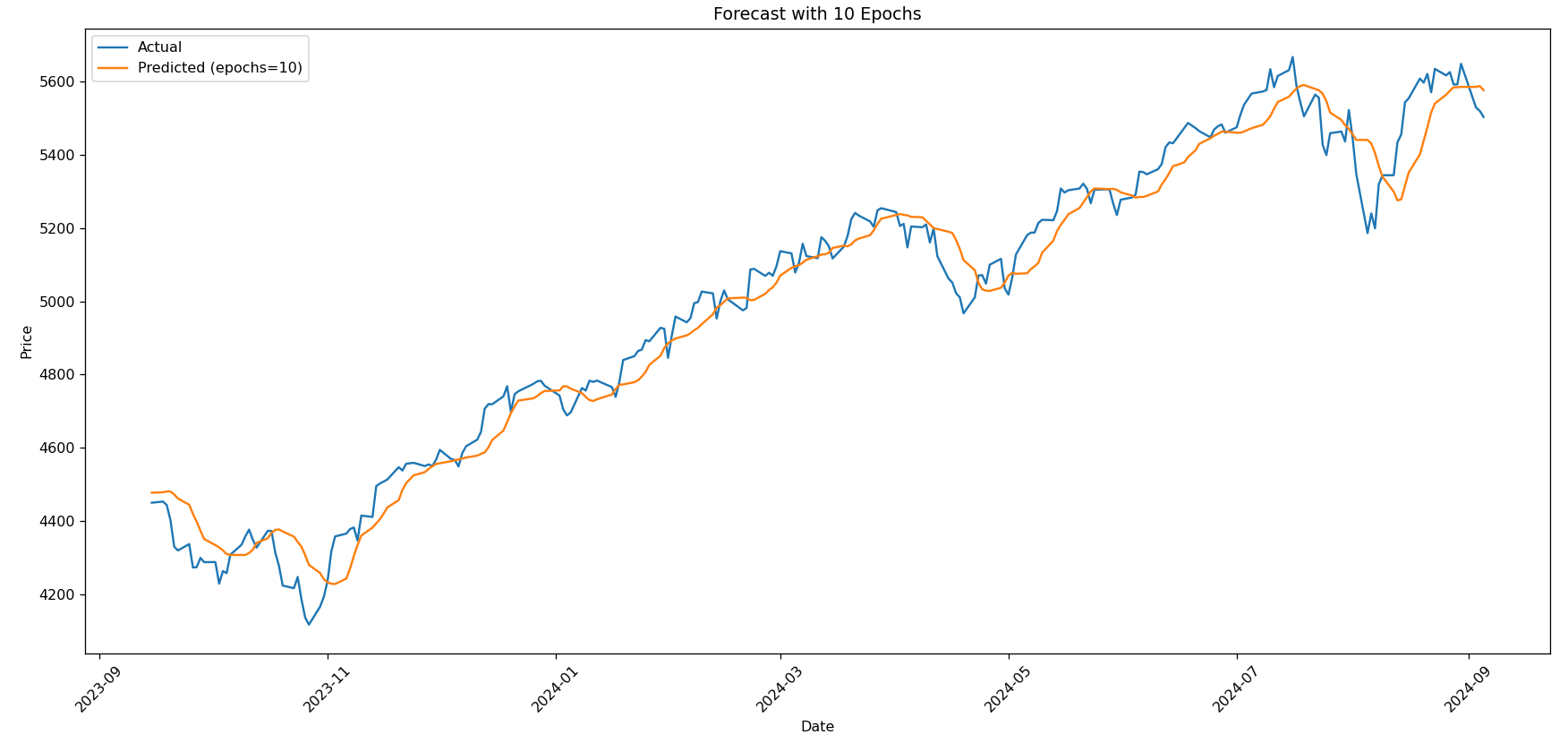


Рисунок 3 – Прогноз моделі після навчання при 10 епохах

Бачимо, що уже на 10 епохах модель прогнозує доволі непогано. Варто зауважити, що прогноз наступного значення відбувається на основі 7 попередніх значень, при цьому всі значення нормалізовані. Це зроблено з метою кращого навчання моделі.

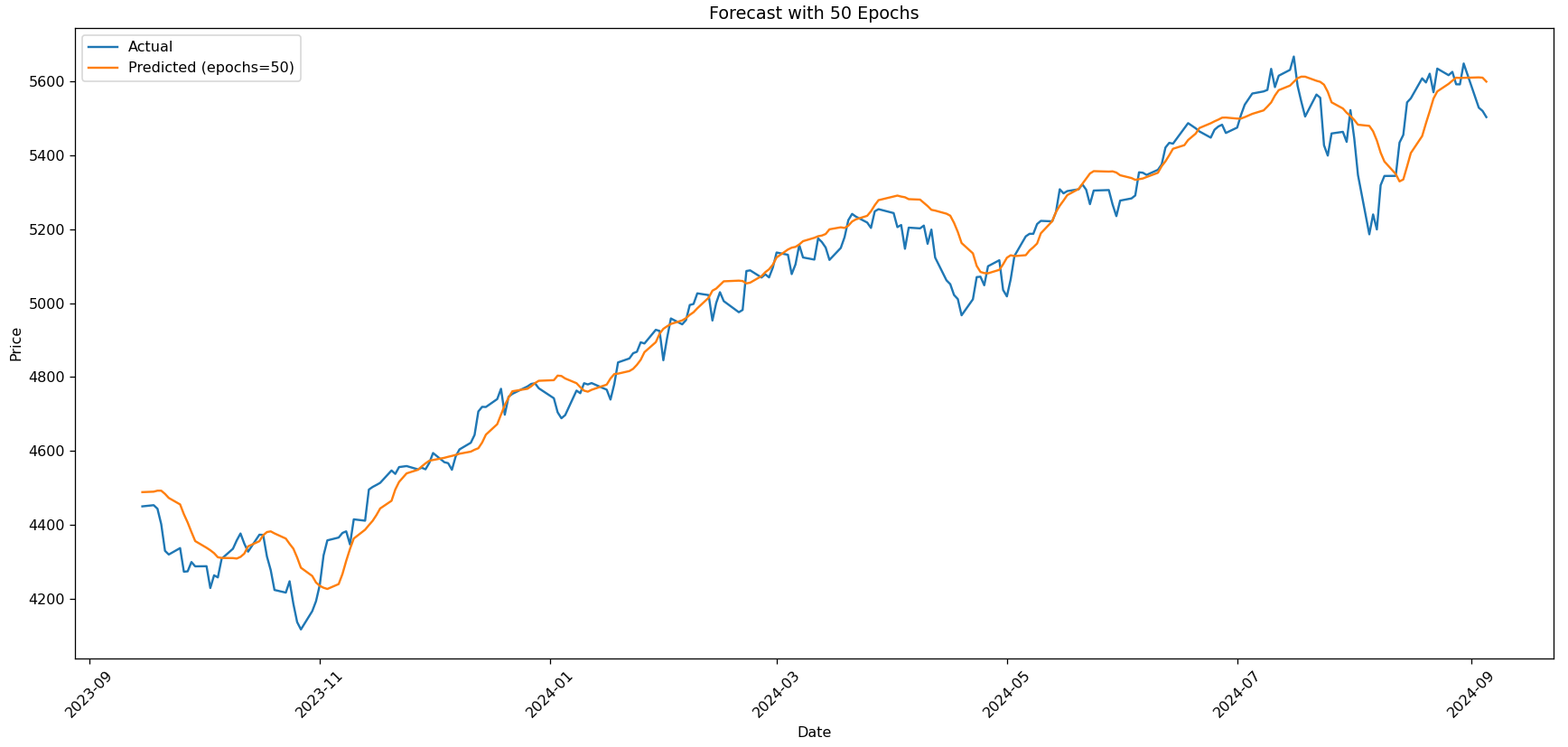


Рисунок 4 – Прогноз моделі після навчання при 50 епохах

При 50 епохах великих змін немає.

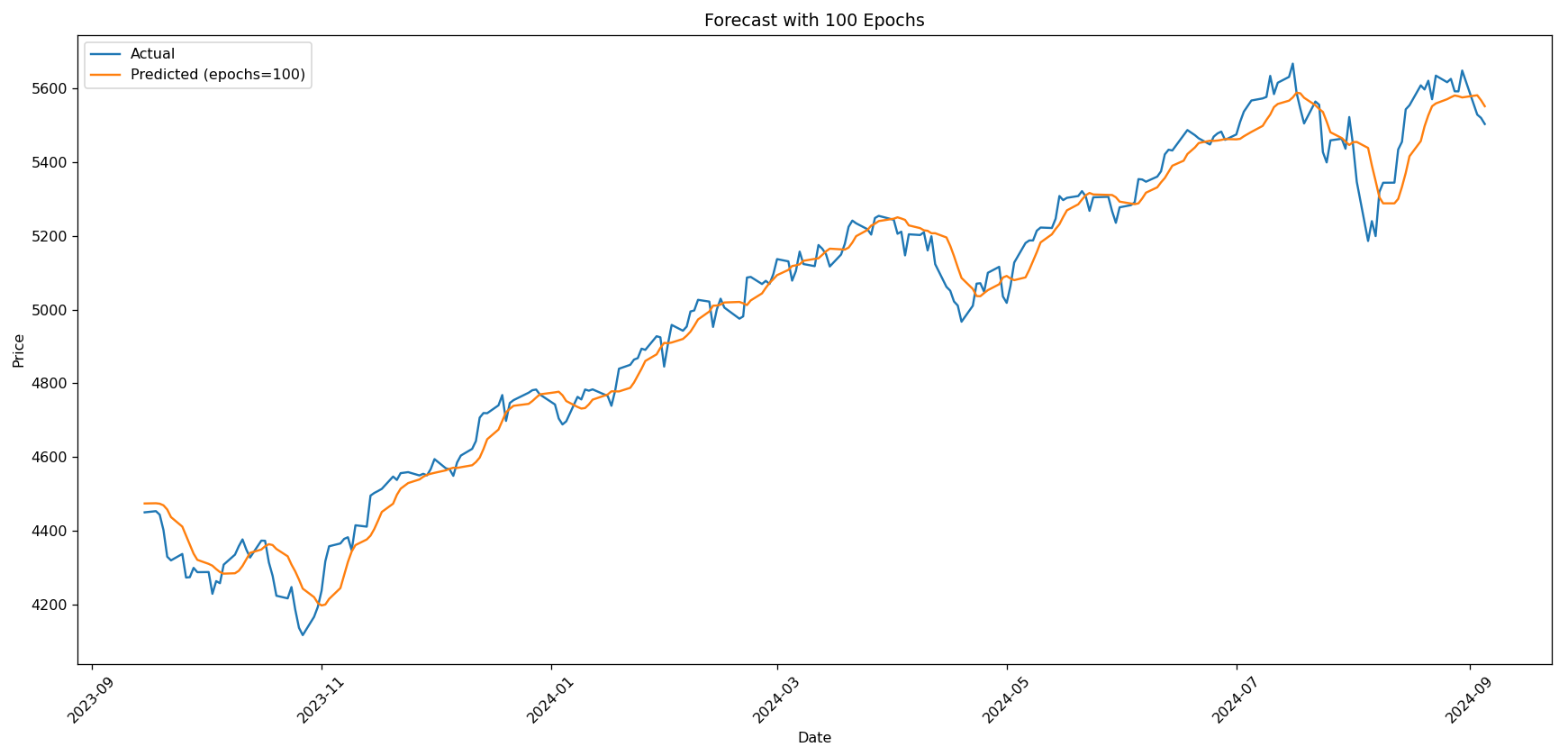


Рисунок 5 – Прогноз моделі після навчання при 100 епохах

На 100 епохах модель прогнозує дещо краще.

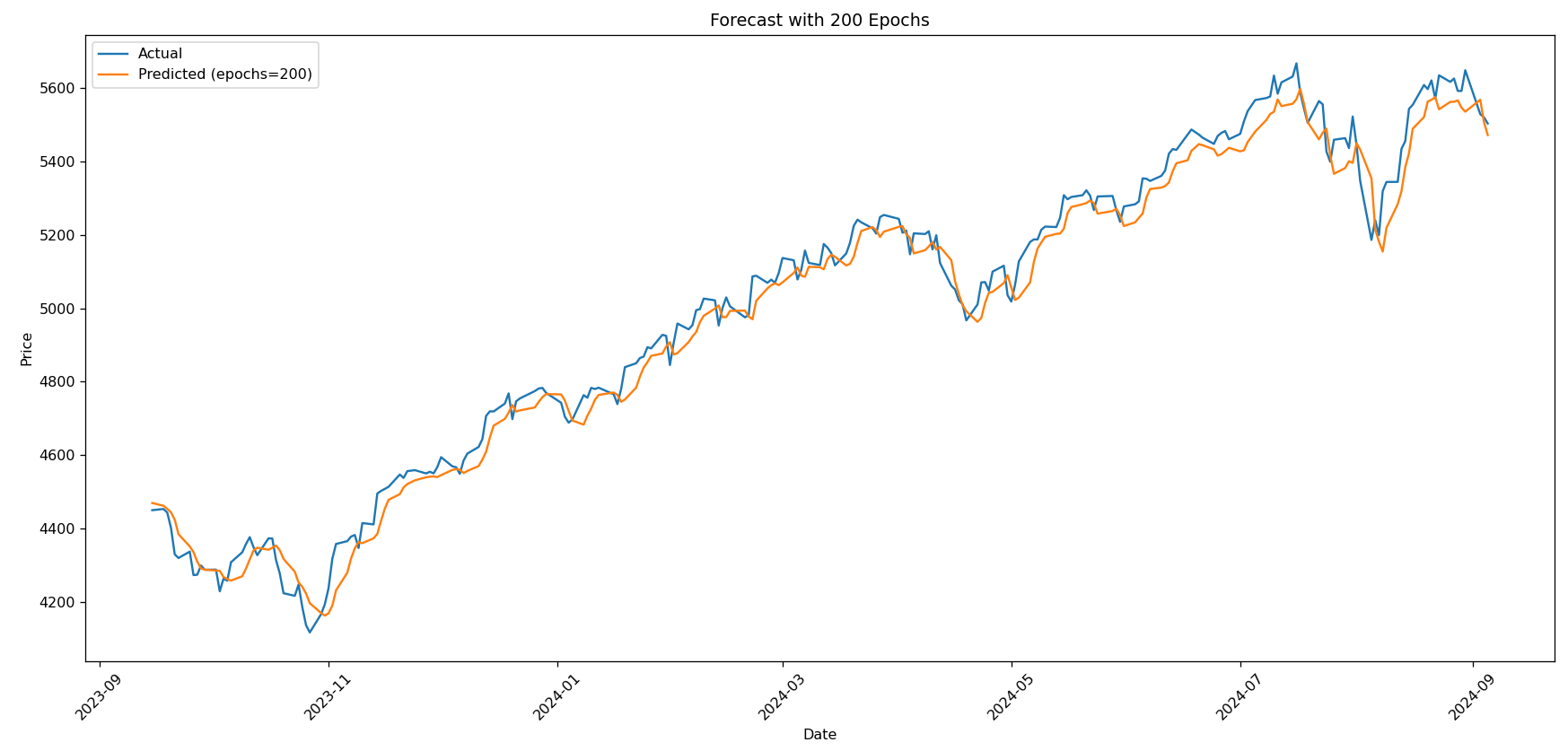


Рисунок 6 – Прогноз моделі після навчання при 200 епохах

При 200 епохах бачимо помітне покращення у прогнозуванні.

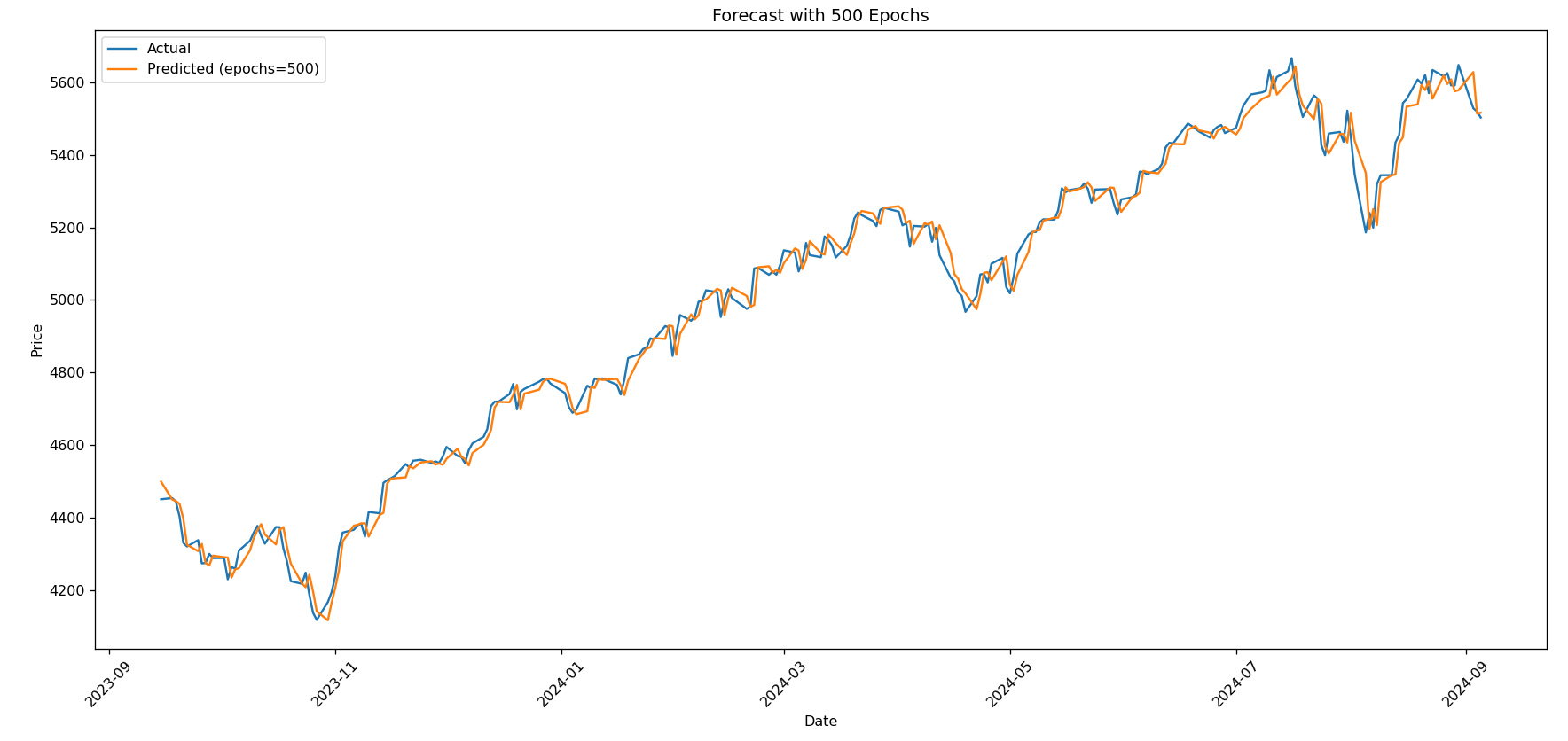


Рисунок 7 – Прогноз моделі після навчання при 500 епохах

При 500 епохах модель уже доволі точно прогнозує дані.

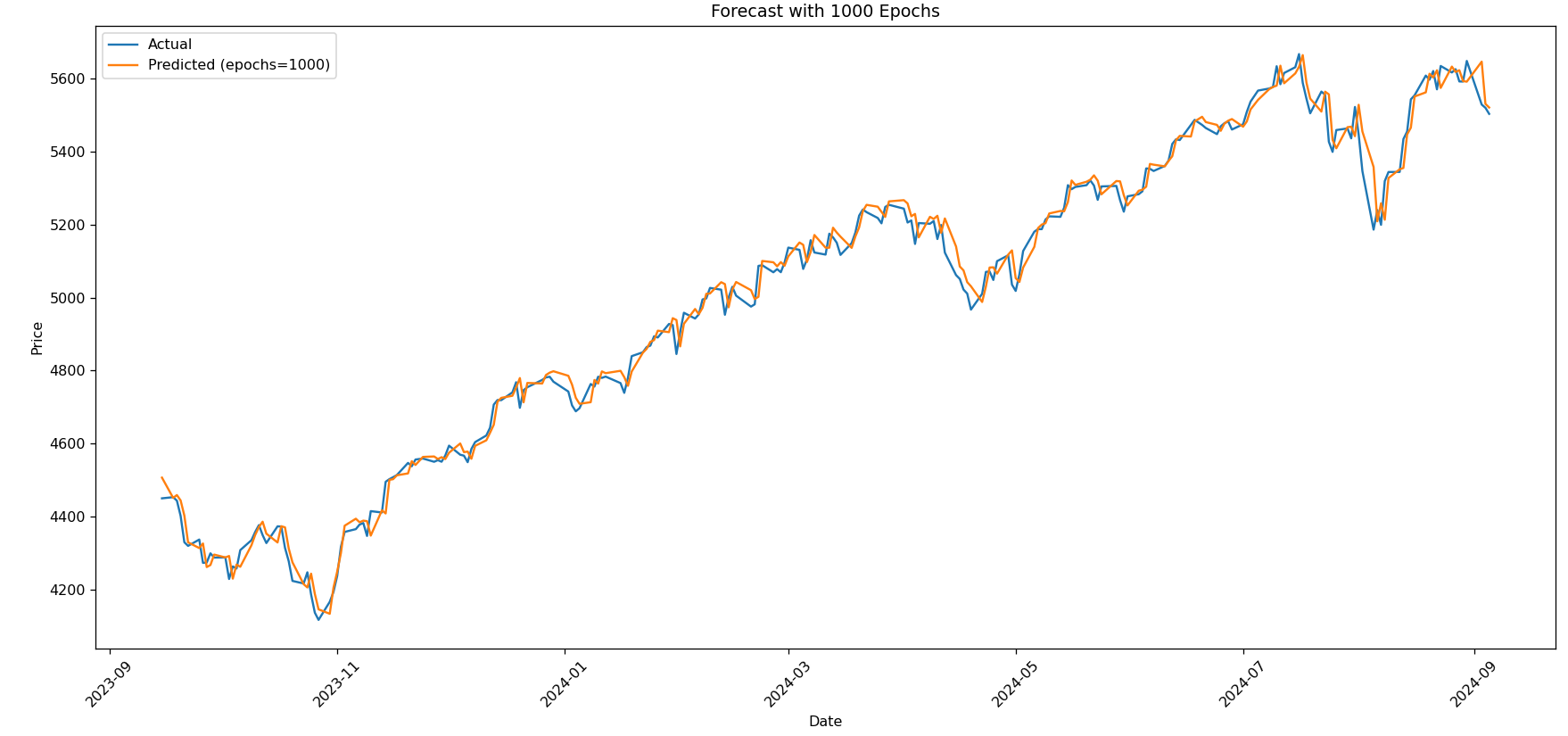


Рисунок 8 – Прогноз моделі після навчання при 1000 епохах

При 1000 епохах візуального покращення немає. Треба переглянути середньоквадратичну похибку для цих епох.

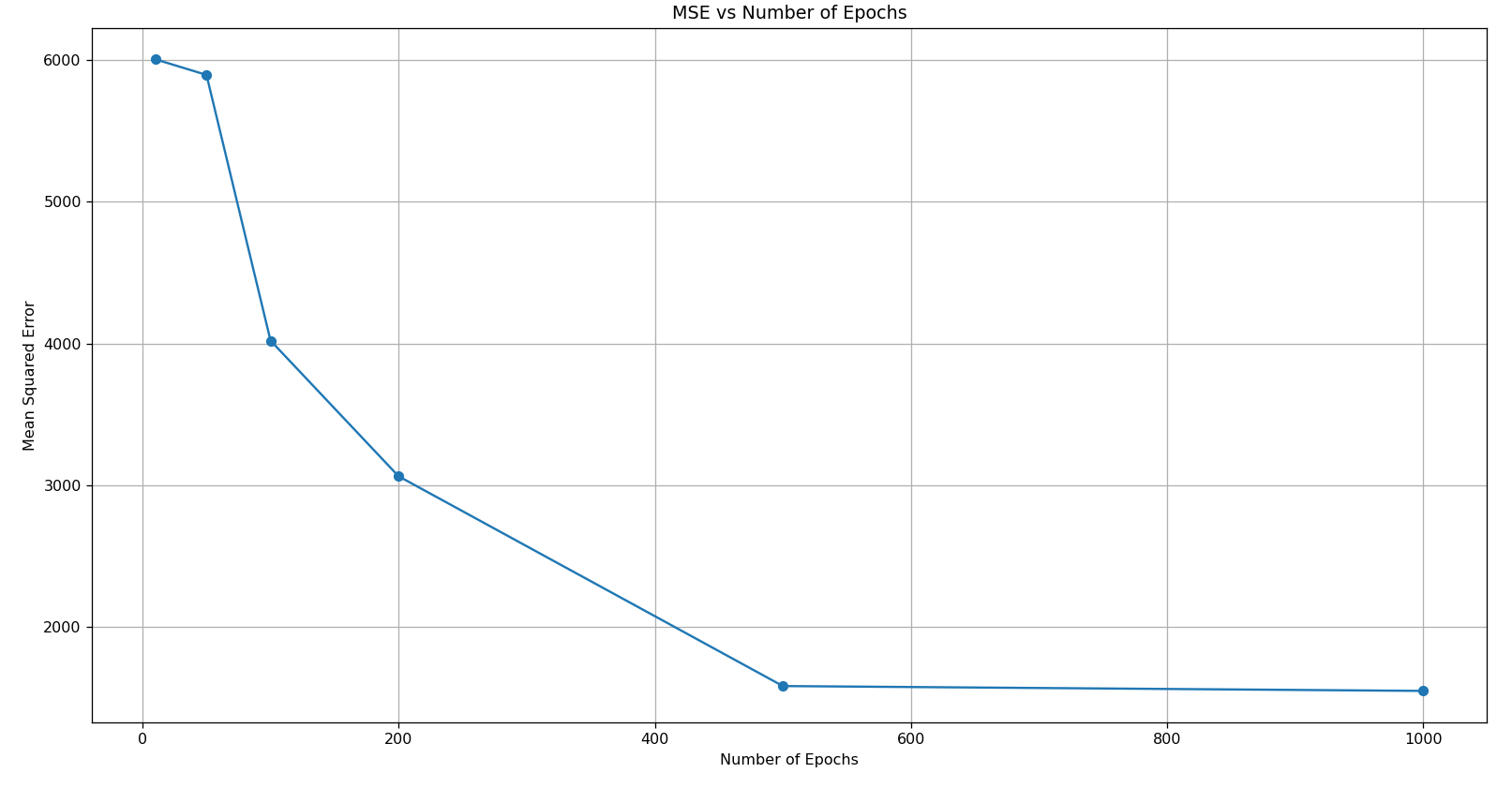


Рисунок 9 – Залежність середньоквадратичної помилки від кількості епох

По графіку бачимо, що до 500 епохи помилка стрімко падає, а після 500 епох помилка майже не зменшується.

* 1. **Програмний код, що забезпечує отримання результату (допускається у формі скриншотів)**

Програмний код послідовно реалізує алгоритм на рис.1 та спрямований на отримання результатів, поданих на рис.3-9.

Для спрощення програмного коду і раціоналізації обчислень застосовано функціональні механізми створення підпрограм.

При цьому використано можливості Python бібліотек: numpy, pandas, matplotlib, tensorflow keras, sklearn.

Контекстні коментарі пояснюють сутність окремих скриптів наведеного коду програми.

main.py

*import* numpy *as* np  
*import* pandas *as* pd  
*import* matplotlib.pyplot *as* plt  
*from* tensorflow.keras.models *import* Sequential  
*from* tensorflow.keras.layers *import* LSTM, Dense  
*from* sklearn.preprocessing *import* MinMaxScaler  
*from* sklearn.metrics *import* mean\_squared\_error  
  
*# 1. Завантаження та підготовка даних*df = pd.read\_csv('output.csv')  
df['Date'] = pd.to\_datetime(df['Date'])  
df.sort\_values('Date', inplace=*True*)  
prices = df['Close/Last'].values.reshape(-1, 1)  
  
*# Нормалізація даних*scaler = MinMaxScaler(feature\_range=(0, 1))  
prices\_scaled = scaler.fit\_transform(prices)  
  
  
*# Створення вхідних та вихідних даних  
def* create\_dataset(data, look\_back):  
 X, y = [], []  
 *for* i *in* range(len(data) - look\_back):  
 X.append(data[i:(i + look\_back), 0])  
 y.append(data[i + look\_back, 0])  
 *return* np.array(X), np.array(y)  
  
  
look\_back = 7 *# Використовуємо сім попередніх значень для передбачення наступного*X, y = create\_dataset(prices\_scaled, look\_back)  
X = X.reshape((X.shape[0], X.shape[1], 1)) *# Перетворення для LSTM  
  
  
# 2. Побудова штучної нейронної мережі  
def* build\_model():  
 model = Sequential([  
 LSTM(20, input\_shape=(look\_back, 1), return\_sequences=*True*),  
 LSTM(10, return\_sequences=*False*),  
 Dense(1)  
 ])  
 model.compile(optimizer='adam', loss='mean\_squared\_error')  
 *return* model  
  
  
*# 3. Навчання та прогнозування*epochs\_list = [10, 50, 100, 200, 500, 1000]  
results = {}  
  
*for* epochs *in* epochs\_list:  
 model = build\_model()  
 model.fit(X, y, epochs=epochs, batch\_size=10, verbose=0)  
  
 *# Прогнозування* predictions = model.predict(X)  
 predictions = scaler.inverse\_transform(predictions) *# Повернення до вихідного масштабу* real\_values = scaler.inverse\_transform(y.reshape(-1, 1))  
  
 *# Оцінка точності* mse = mean\_squared\_error(real\_values, predictions)  
 results[epochs] = {  
 "mse": mse,  
 "predictions": predictions.flatten()  
 }  
  
 *# Візуалізація прогнозів* plt.figure()  
 plt.plot(df['Date'][look\_back:], real\_values, label="Actual")  
 plt.plot(df['Date'][look\_back:], predictions, label=f"Predicted (epochs={epochs})")  
 plt.legend()  
 plt.title(f"Forecast with {epochs} Epochs")  
 plt.xlabel("Date")  
 plt.ylabel("Price")  
 plt.xticks(rotation=45)  
 plt.tight\_layout()  
 plt.show()  
  
*# 4. Аналіз залежності точності від кількості епох*epochs = list(results.keys())  
mse\_values = [results[ep]["mse"] *for* ep *in* epochs]  
  
plt.figure()  
plt.plot(epochs, mse\_values, marker='o')  
plt.title("MSE vs Number of Epochs")  
plt.xlabel("Number of Epochs")  
plt.ylabel("Mean Squared Error")  
plt.grid()  
plt.show()

## **IV. Висновки.**

У ході виконання лабораторної роботи було досліджено та узагальнено особливості підготовки різних типів даних, синтезу, навчання та застосування штучних нейронних мереж (Artificial Neural Networks). Було використано LSTM для прогнозування часових рядів, а саме значень індексу S&P 500. Також було проведено аналіз залежності середньоквадратичної похибки від кількості епох.

Виконав: студент Трикош І. В.