Практична робота №6

Регресія. Прогноз ціни Ethereum

Мета роботи: набути практичних навичок роботи у побудові моделі для прогнозування цін на валютному ринку блокчейну

Стек технологій:

TensorFlow 2.x - ***побудова та навчання LSTM-моделі***

Keras Sequential API - ***конструювання нейромережі (LSTM, Dense, Dropout)***

Google Colaboratory - ***хмарне середовище для виконання роботи***

Pandas - ***завантаження, злиття та обробка датасетів (blockchain_data.csv, historical_data.csv, twitter_sentiments.csv)***

NumPy - ***числові операції та підготовка масивів даних***

Matplotlib - ***побудова графіків (ціна/дата, результати прогнозу)***

Scikit-learn - ***метрики якості: MAE та MSE (sklearn.metrics)***

Kaggle Datasets - ***джерело даних для Bitcoin (Завдання 3)***

Blockchain.com / Binance - ***джерела вхідних даних для ETH***

Теоретичний матеріал

Оригінальна концепція Ethereum була представлена у 2013 році Віталіком Бутеріним під час випуску документу про Ethereum, а в 2015 році платформу Ethereum запустили Бутерін і Джозеф Лубін разом із кількома іншими співзасновниками. Ethereum позиціонує себе як електронну програмовану мережу, яку кожен може побудувати для запуску криптовалют і децентралізованих програм. На відміну від біткойну, максимальний тираж якого становить 21 мільйон монет, кількість ETH, яку можна створити, необмежена, хоча час, необхідний для обробки блоку ETH, обмежує кількість ефіру. Ще одна відмінність між Ethereum і Bitcoin полягає в тому, як мережі обробляють комісію за обробку транзакцій. Ці збори відомі як «газ» у мережі Ethereum і сплачуються учасниками транзакцій Ethereum.

[https://www.blockchain.com/explorer/assets/eth]

Набори даних:

Для роботи використано три набора даних ***blockchain_data.csv*** - дані були зібрані з крипто-валютної біржи Blockchain, ***historical_data.csv*** – дані були зібрані з крипто-валютної біржs Binance, ***twitter_sentiments.csv***-наведено сентиментальний аналіз твітів стосовно валюти(позитивні, негативні, нейтральні)

У межах роботи буде досліджено застосування рекурентних нейронних мереж (RNN) архітектури LSTM для аналізу часових рядів. На відміну від стандартних RNN, LSTM ефективно вирішує проблему затухання градієнта завдяки системі вентилів (забування, входу та виходу), що регулюють оновлення стану клітини (cell state).

Ключові аспекти методології:

Тип задачі. Регресія (передбачення конкретного числового значення вартості).

Метрики оцінки. Для аналізу точності використано MSE (чутлива до значних відхилень) та MAE (стійка до викидів).

Підготовка даних:

Zero-base нормалізація. Перерахунок значень як відносних змін щодо першого елемента вікна ($x_{norm} = x / x_0 - 1.$). Це нівелює різницю в масштабах ціни та обсягів торгів.

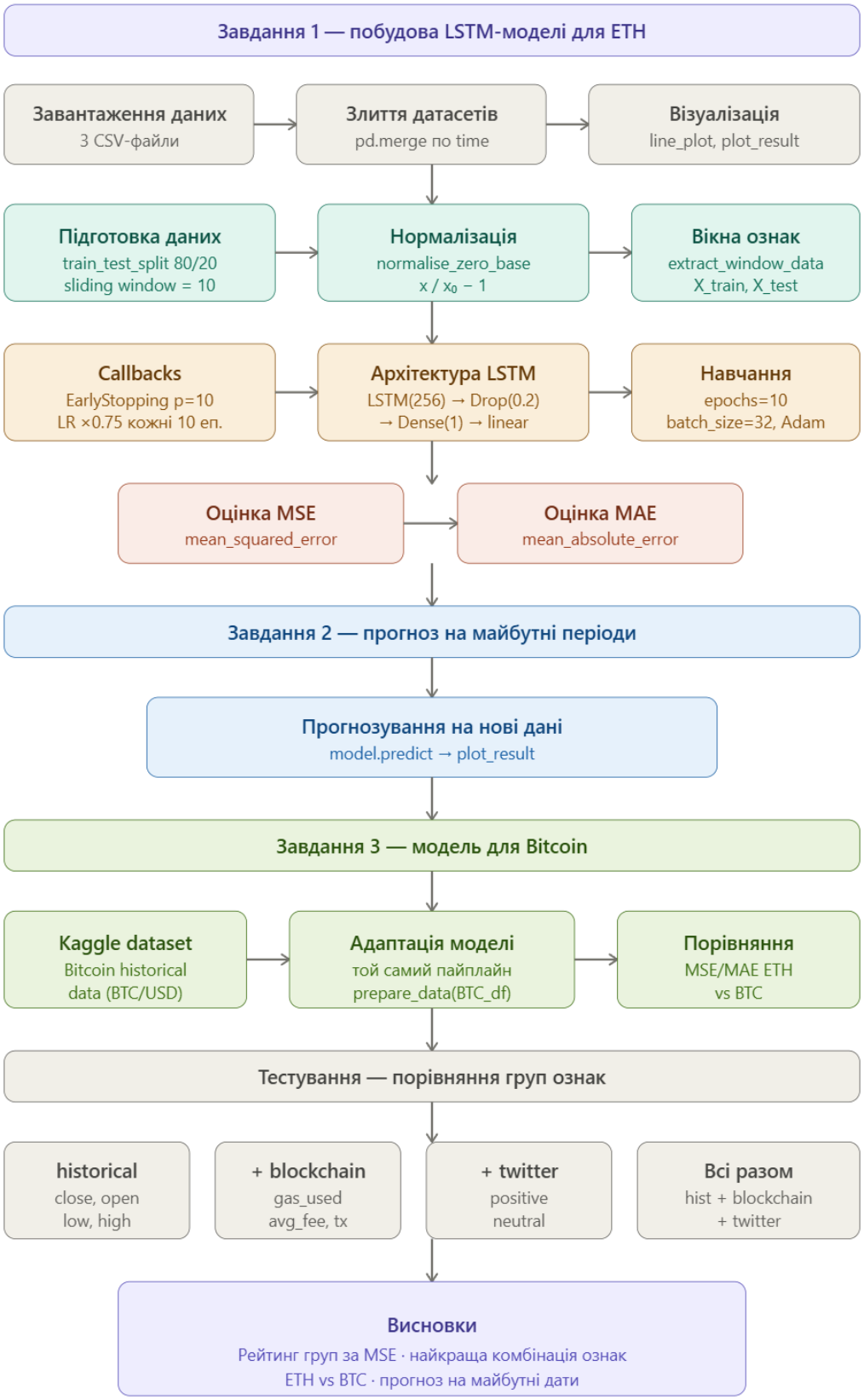
Метод ковзного вікна. Дані розділено на послідовні сегменти (window_len = 10), де кожен фрагмент слугує базою для прогнозу наступного значення.

Оптимізація навчання (Callbacks):

EarlyStopping. Запобігає перенавчанню, зупиняючи процес при стабілізації помилки.

LearningRateScheduler. Динамічно знижує швидкість навчання (на 25% кожні 10 епох) для точного досягнення мінімуму функції втрат.

Схема виконання практичної роботи:



У роботі використовується ***inner merge***, тобто до фінального датасету потрапляють лише ті дати, для яких є дані в усіх трьох джерелах одночасно.

Зміст роботи

Завдання 1. *Побудувати модель для прогнозу ціни криптовалюти ефір(ETH).*

– Завдання рекомендується виконувати в Google Colaboratory.

– Бібліотеки, які знадобляться для роботи:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Activation, Dense, Dropout, LSTM
from sklearn.metrics import mean_squared_error
from datetime import datetime
```

– Завантажити файли (використати назви дата фреймів prices_df, blockchain_df, twitter_sentiments).

– Продивитися перші n рядків файлів.

– Створити функції для макетів графіків:

```
def line_plot(train, test, label1 = None, label2 = None, title='', lw=2):
    fig, ax = plt.subplots(1, figsize=(13, 7))
    if 'time' in train.keys():
        ax.plot(train['time'].apply(lambda x: datetime.fromtimestamp(x)),
train['close'], label=label1, linewidth=lw)
        ax.plot(test['time'].apply(lambda x: datetime.fromtimestamp(x)), t
est['close'], label=label2, linewidth=lw)
    else:
        ax.plot(train['close'], label = label1, linewidth = lw)
        ax.plot(test['close'], label = label2, linewidth = lw)
    ax.set_ylabel('Ціна [USD]', fontsize = 14)
    ax.set_title(title, fontsize = 16)
    ax.legend(loc='best', fontsize = 16)
    plt.show()

def plot_result(actual, predicted):
    fig, ax = plt.subplots(1, figsize=(13, 7))
    ax.plot(actual, label = 'actual', linewidth = 1)
    for key in predicted.keys():
        ax.plot(predicted[key], label = key, linewidth = 1)
    ax.set_ylabel('Ціна [USD]', fontsize = 14)
    ax.legend(loc='best', fontsize = 16)
    plt.show()
```

– Об’єднати набори даних:

```
merged_df = pd.merge(pd.merge(blockchain_df, prices_df, left_on = 'time',
right_on = 'close_time'), twitter_sentiments, on = 'time')
```

– Видалити непотрібні данні. Відсортувати за часом.

– Продивитися перші n рядків отриманого набору даних.

Підготувати дані для машинного навчання.

– Підготувати функції: для розбиття набору даних на тренувальні та тестові дані; для нормалізації даних; для вилучення даних; prepare_data() – використовується для завантаження підготовлених даних. Завантаження та збереження даних за допомогою кількох процесів (distributed settings) призведе до пошкодження даних.

```
def train_test_split(df, test_size = 0.2):
    split_row = len(df) - int(test_size * len(df))
    train_data = df.iloc[:split_row]
    test_data = df.iloc[split_row:]
    return train_data, test_data

def normalise_zero_base(df):
    coefs = df.iloc[0].values
    for i in range(len(coefs)):
        if coefs[i] == 0:
            coefs[i] = 1

    return df / coefs - 1

def extract_window_data(df, window_len):
    window_data = []
    for idx in range(len(df) - window_len):
        data = normalise_zero_base(df[idx: (idx + window_len)].copy())
        window_data.append(data.values)
    return np.array(window_data)

def prepare_data(df, window_len, test_size = 0.2):
    target_col = 'close'

    train_data, test_data = train_test_split(df, test_size = test_size)
    X_train = extract_window_data(train_data, window_len)
    X_test = extract_window_data(test_data, window_len)
    y_train = train_data[target_col][window_len:].values
    y_test = test_data[target_col][window_len:].values
    y_train = y_train / train_data[target_col][:-window_len].values - 1
    y_test = y_test / test_data[target_col][:-window_len].values - 1

    return train_data, test_data, X_train, X_test, y_train, y_test
```

Тестувати побудованої моделі завжди потрібно на даних, які не брали участь у навчанні. Такі дані називають тестовими. Дані, що беруть участь у навчанні, називаються навчальними. Розбиття набору даних на навчальні та тестові дані можна за допомогою функції train_test_split.

– Виконати розбиття набору даних на навчальні та тестові дані. Застосуємо цю функцію до даних, до тестової частини відносимо 20%, до навчальної — решта 80%. Оскільки були використані стандартні параметри, вихідні дані спочатку були перемішані, а потім розділені.

– Отримати інформацію про кількість строк і рядків в навчальній і тестову вибірки.

– Побудувати графік відповідності ціна - дата тренувального і тестового наборів даних.

```
line_plot(train, test,'тренувальний набір','тестувальний набір', title='')
```

Почати навчання

– Припинити навчання, коли показник, що відстежується, перестав покращуватися:

```
early_stopping_callback = tf.keras.callbacks.EarlyStopping(patience = 10,
restore_best_weights = True)

def lr_scheduler(epoch, lr):
    if epoch > 0 and epoch % 10 == 0:
        return lr * 0.75
    return lr

lr_scheduler_callback = tf.keras.callbacks.LearningRateScheduler(lr_schedu
ler)
```

– Функція для побудувати LSTM моделі. Нейронна мережа складається з рівня LSTM, за яким слідує рівень 20% Dropout і щільний шар із лінійною функцією активації.

```
def build_lstm_model(input_data, neurons, dropout, optimizer):
    model = Sequential()
    model.add(LSTM(neurons, input_shape = (input_data.shape[1], input_data
.shape[2])))
    model.add(Dropout(dropout))
    model.add(Dense(units = 1))
    model.add(Activation('linear'))

    model.compile(loss = 'mse', optimizer = optimizer)
    return model
```

– Задати параметри.

```
window_len = 10
neurons = 256
epochs = 10
batch_size = 32
dropout = 0.2
optimizer = 'adam'
```

```
df = merged_df[['close']]
```

– Навчити модель.

```
train, test, X_train, X_test, y_train, y_test = prepare_data(df, window_le
n)
model = build_lstm_model(X_train, neurons, dropout, optimizer)
history = model.fit(np.asarray(X_train).astype('float32'), np.asarray(y_tr
ain).astype('float32'), epochs = epochs, batch_size = batch_size, shuffle
= True, callbacks = [early_stopping_callback, lr_scheduler_callback])
```

– Використати MSE. Mean Squared Error вимірює кількість помилок у статистичних моделях. Зведення різниць у квадрат усуває негативні значення і гарантує, що середня квадратична помилка завжди буде більшою або дорівнює нулю. Тільки ідеальна модель без помилок дає нульовий MSE. А на практиці такого не буває. Якщо взяти квадратний корінь із MSE, отримаємо середню квадратичну помилку (RMSE), яка використовує натуральні одиниці вимірювання. Іншими словами, MSE аналогічно дисперсії, тоді як RMSE схоже на стандартне відхилення.

```
targets = test['close'][window_len:]
preds = model.predict(X_test).squeeze()
print('MSE:', mean_squared_error(preds, y_test))
preds = test['close'].values[window_len:] * (preds + 1)
preds = pd.Series(index = targets.index, data = preds)
plot_result(targets.values, {'price': preds.values})
```

Як показник оцінки також можна використати середню абсолютну похибку (Mean Absolute Error (MAE)) Це середнє за тестовою вибіркою абсолютних відмінностей між фактичними та прогнозованими спостереженнями.

– Використати MAE. Порівняти результати.

Тестування

– Обрати змінні з кожної підгрупи :

```
data_props = {
    'price': ['close'],
    'historical': ['close', 'open', 'low', 'high', 'volume', 'trades_amoun
t'],
    'blockchain': ['max_value', 'average_amount', 'average_fee', 'average_
gas_price', 'gas_used', 'transaction_amount'],
    'twitter': ['positive', 'neutral']
}
```

– Згрупувати дата сет відповідностей:

```
data_sets = [['historical'], ['historical', 'blockchain'], ['historical',
'twitter'], ['historical', 'blockchain', 'twitter']]
```

– Провести навчання на кожному угрупуванні:

```
targets = test['close'][window_len:]
results = {}

mse_results = {}

for item in data_sets[:4]:
    fields = []
    for field_type in item:
        fields.extend(data_props[field_type])

    df = merged_df[fields]
    train, test, X_train, X_test, y_train, y_test = prepare_data(df, windo
w_len)
    current_model = build_lstm_model(X_train, neurons, dropout, optimizer)
    current_model.fit(np.asarray(X_train).astype('float32'), np.asarray(y_
train).astype('float32'), epochs = epochs, batch_size = batch_size, shuffl
e = True, callbacks = [early_stopping_callback, lr_scheduler_callback])

    preds = current_model.predict(X_test).squeeze()
    key = ', '.join(item)
    mse_results[key] = mean_squared_error(preds, y_test)
    preds = test['close'][window_len:] * (preds + 1)
    preds = pd.Series(index = targets.index, data = preds)
    results[key] = preds.values
```

– Побудувати графік отриманого результату

```
plot_result(targets.values, results)
```

– Вивести показники оцінки по кожному угрупуванню:

```
rating = sorted(mse_results.items(), key = lambda x: x[1])
for item in rating:
    print(item[0], '-', item[1])
```

– Зробити висновки

Завдання 2. *Мультикрокове прогнозування.* Замість прогнозу на 1 крок вперед реалізувати прогноз на 7 і 30 днів. Порівняти точність MAE/MSE при збільшенні горизонту прогнозу.

Завдання 3. *Аналіз та інтерпретація:*

– *Аналіз важливості ознак.* Послідовно прибирати кожну групу ознак (*historical, blockchain, twitter*) та фіксувати зміну *MSE*. Побудувати рейтинг їх впливу на якість.

– *Аналіз помилок у часі.* Побудувати графік абсолютної похибки $|y - \hat{y}|$ по датах. Визначити, в які ринкові події (різкі злети/падіння) модель помиляється найбільше.

– *Інтервальне прогнозування.* Реалізувати *Monte Carlo Dropout* - запускати модель 100 разів із увімкненим *dropout* і будувати довірчий інтервал прогнозу.

Завдання 4. Побудувати модель для прогнозу ціни криптовалюти *bitcoin* (обрати дата сет на https://www.kaggle.com).

Контрольні запитання

1. За допомогою якої функції можна розбити набір даних на навчальні та тестові дані? Чому важливо не перемішувати часові ряди перед розбиттям (на відміну від класифікації)?

2. Для чого використовується *LSTM*? У яких задачах *LSTM* має перевагу над звичайними нейронними мережами прямого поширення (*MLP*)?

3. Як працює *LSTM*? Опишіть роль трьох вентилів: *forget gate, input gate, output gate*. Що зберігається у *cell state*, а що у *hidden state*?

4. Які існують методи виміру точності регресійної моделі? Назвіть не менше чотирьох метрик та поясніть, коли кожна з них є пріоритетною.

5. Що таке середня абсолютна похибка (*MAE*)? Запишіть формулу. Чому *MAE* стійкіша до викидів, ніж *MSE*?

6. Що таке середньоквадратична похибка (*MSE*)? Запишіть формулу. У чому полягає відмінність між *MSE* та *RMSE* і коли *RMSE* зручніша для інтерпретації?

7. Що таке метод ковзного вікна (*sliding window*) у контексті часових рядів? Як параметр *window_len* впливає на якість прогнозу?

8. Поясніть метод нормалізації *normalise_zero_base*. Яку формулу він реалізує і чому цей підхід підходить для фінансових даних краще, ніж стандартна мін-макс нормалізація?

9. Чому при роботі з часовими рядами не можна застосовувати стандартну функцію *train_test_split* зі *sklearn* з параметром *shuffle=True*?

10. Які три джерела даних використовуються в роботі? Опишіть, яку інформацію містить кожен із датасетів: historical_data.csv, blockchain_data.csv, twitter_sentiments.csv.

11. Що таке сентиментальний аналіз? Як позитивні та негативні твіти можуть впливати на прогноз ціни криптовалюти?

12. Поясніть роль *EarlyStopping* у навчанні нейронної мережі. Що означає параметр *patience=10* і що робить *restore_best_weights=True*?

13. Як працює *LearningRateScheduler* у роботі? Чому зменшення *learning rate* під час навчання покращує збіжність моделі?

14. Що таке *Dropout* у нейронних мережах? Яку проблему він вирішує і який відсоток нейронів вимикається в роботі?

15. Поясніть вибір функції активації *linear* у вихідному шарі. Чому для задачі регресії не можна використовувати *softmax* або *sigmoid*?

16. Що означає параметр *shuffle=True* у функції *model.fit*? Чи безпечно його використовувати для часових рядів і чому?

17. Як інтерпретувати графік *plot_result*? На що звертати увагу при порівнянні фактичних та передбачених значень ціни?

18. Яка комбінація груп ознак (*historical / blockchain / twitter*) дала найкращий результат у вашому експерименті? Поясніть, чому саме ця комбінація виявилась найефективнішою.

19. Що таке перенавчання (*overfitting*) у контексті прогнозування часових рядів? Як його виявити на графіку *loss* і які методи допомагають з ним боротися?

20. Порівняйте задачі класифікації та регресії в машинному навчанні. Чим відрізняються функції втрат, метрики якості та вихідні шари моделі?