Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

О Т Ч Е Т

по лабораторной работе № 4

Название: Реконструкция модели цифрового двойника человека-оператора в киберфизической системе

Дисциплина: Дистанционный мониторинг сложных систем и процессов

Студент ИУ6-12М (Группа) _____ (Подпись, дата) С.В. Астахов (И.О. Фамилия)

Преподаватель _____ (Подпись, дата) Ю.А. Вишневская (И.О. Фамилия)

Москва, 2023

Введение

Цель работы: изучение особенностей построения алгоритма реконструкции математической модели человека-оператора по временному ряду.

Задание 2 (альтернативное): При выполнении лабораторной работы студентам необходимо разработать цифровую модель эксперта, решающего задачу прогнозирования процесса (ситуации) по временному ряду. В этом случае вместо временного ряда биосигнала необходимо взять любой реальный временной ряд и создать систему прогнозирования исходного временного ряда на будущий период времени.

Ход выполнения

Исходные данные: в качестве рассматриваемого временного ряда взяты биржевые котировки компании StarBucks за 2020 год. Фрагмент датасета представлен на рисунке 1.

Date	Open	High	Low	Close	Adj Close	Volume	MA	STD
2019-12-11	86.260002	86.870003	85.849998	86.589996	79.847794	4921900	0.0	0.000000e+00
2019-12-12	88.000000	88.889999	87.540001	88.209999	81.341652	10282100	0.0	0.000000e+00
2019-12-13	88.019997	88.790001	87.580002	88.669998	81.765823	6714100	0.0	0.000000e+00
2019-12-16	89.139999	89.300003	88.430000	88.779999	81.867256	6705600	0.0	0.000000e+00
2019-12-17	88.870003	88.970001	87.470001	88.129997	81.267868	7296900	7184120.0	1.947931e+06
2019-12-18	88.389999	88.849998	87.820000	87.989998	81.138786	5859200	7371580.0	1.705858e+06
2019-12-19	87.830002	88.589996	87.580002	88.519997	81.627502	6022100	6519580.0	5.831501e+05

Рисунок 1 — фрагмент использованного датасета

Рассчитаем скользящее среднее и среднеквадратическое отклонение и визуализируем этот датасет. Исходный код приведен в листинге 1. Результаты визуализации показаны на рисунке 2.

```
import numpy as np
import pandas as pd

df = pd.read_csv('SBUX.csv', index_col = 'Date', parse_dates=True)
df['MA'] = df['Volume'].rolling(window=5).mean() # скользящее среднее
df['STD'] = df['Volume'].rolling(window=5).std() # отклонение
```

```
df['MA'] = df['MA'].fillna(0)     # заполнить 0 вместо NaN для краевых значений
df['STD'] = df['STD'].fillna(0)

from matplotlib import pyplot as plt
import seaborn as sns

# визуализация данных
plt.style.use('ggplot')
plt.plot(df.index, df['Volume'], label='VOL', color='red')
plt.plot(df.index, df['MA'], label= 'MA', color='blue')
plt.plot(df.index, df['STD'], label= 'STD', color='green')

plt.legend()
plt.show()
```

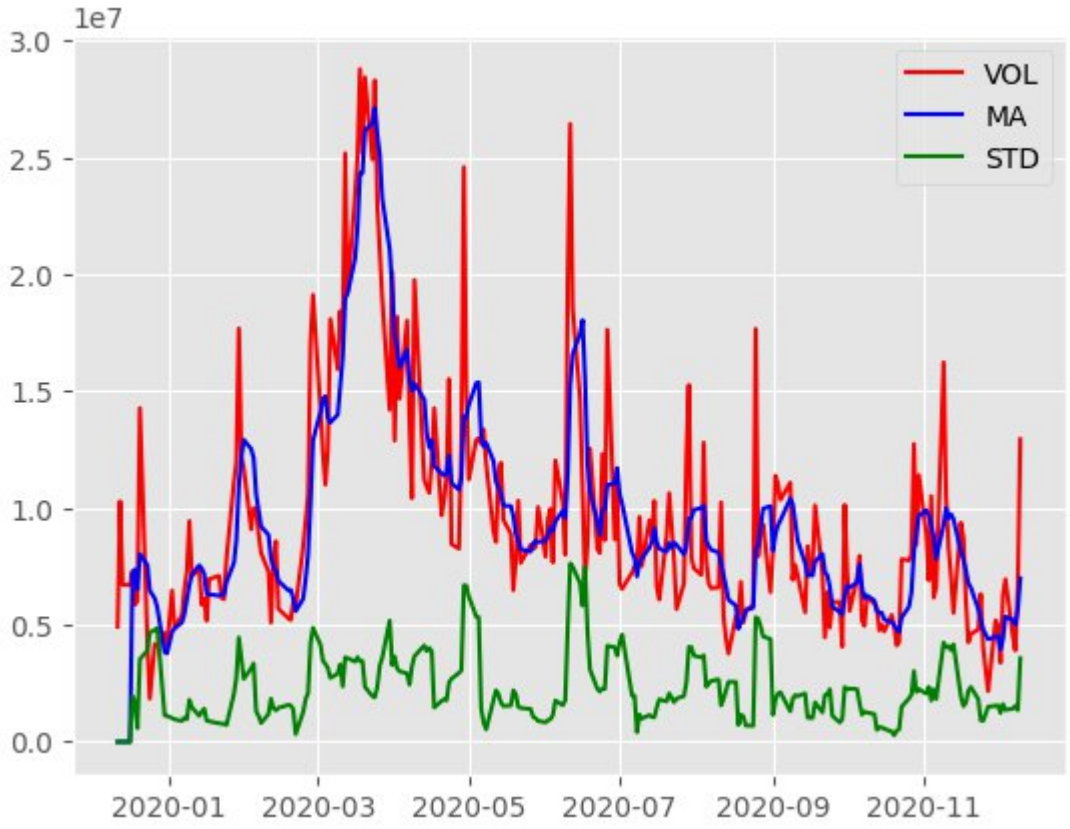

Рисунок 2 — капитализация компании, скользящее среднее и отклонение

Так как капитализация компании имеет высокую волатильность, предположим, что нас интересуют среднесрочные инвестиции и поэтому мы можем прогнозировать поведение скользящего среднего от капитализации. Предварительная обработка данных показана в листинге 2.

Листинг 2 — предобработка данных

```
df['feature'] = df['MA']
```

```
X = df.iloc[:, :5]   # пусть величина зависит от остальных 5 в прошлом
y = df[['feature']]

from sklearn.preprocessing import StandardScaler, MinMaxScaler

mm = MinMaxScaler()
ss = StandardScaler()


X_ss = ss.fit_transform(X) # нормализация значений
y_mm = mm.fit_transform(y)

X_train = X_ss[:200, :]    # отделение тренировочных примеров
X_test = X_ss[200:, :]

y_train = y_mm[:200, :]
y_test = y_mm[200:, :]

import torch #pytorch
import torch.nn as nn
from torch.autograd import Variable

# преобразование данных в формат, совместимый с бибилотекой
X_train_tensors = Variable(torch.Tensor(X_train))
X_test_tensors = Variable(torch.Tensor(X_test))

y_train_tensors = Variable(torch.Tensor(y_train))
y_test_tensors = Variable(torch.Tensor(y_test))

X_train_tensors_final = torch.reshape(X_train_tensors,   (X_train_tensors.shape[0],
1, X_train_tensors.shape[1]))

X_test_tensors_final = torch.reshape(X_test_tensors,  (X_test_tensors.shape[0], 1,
X_test_tensors.shape[1]))
```

В листинге 3 описывается классы LSTM-сети — нейросети долгой краткосрочной памяти — рекуррентной нейронной сети, способный обучаться долгосрочным зависимостям.

Листинг 3 — описание класса LSTM-сети

```
class LSTM1(nn.Module):
    # инициализация параметров
    def __init__(self, num_classes, input_size, hidden_size, num_layers,
seq_length):
        super(LSTM1, self).__init__()
        self.num_classes = num_classes
        self.num_layers = num_layers
        self.input_size = input_size
        self.hidden_size = hidden_size
        self.seq_length = seq_length
```

```
        self.lstm = nn.LSTM(input_size=input_size, hidden_size=hidden_size,
                            num_layers=num_layers, batch_first=True)
        self.fc_1 =  nn.Linear(hidden_size, 128)
        self.fc = nn.Linear(128, num_classes)

        self.relu = nn.ReLU()

    def forward(self,x):
        # скрытое состояние
        h_0 = Variable(torch.zeros(self.num_layers, x.size(0), self.hidden_size))
        # внутреннее состояние
        c_0 = Variable(torch.zeros(self.num_layers, x.size(0), self.hidden_size))
        # распространение сигнала по LSTM
        output, (hn, cn) = self.lstm(x, (h_0, c_0))
        hn = hn.view(-1, self.hidden_size)
        out = self.relu(hn)
        out = self.fc_1(out) #first Dense
        out = self.relu(out) #relu
        out = self.fc(out) #Final Output
        return out
```

Обучение нейросети показано в листинге 4.

Листинг 4 — обучение нейросети

```
num_epochs = 1000 # количество циклов обучения
learning_rate = 0.001 # скорость обучения

input_size = 5
hidden_size = 10
num_layers = 1

num_classes = 1

lstm1 = LSTM1(num_classes, input_size, hidden_size, num_layers,
X_train_tensors_final.shape[1])

criterion = torch.nn.MSELoss()    # среднеквадратичная функция ошибки для обучения
optimizer = torch.optim.Adam(lstm1.parameters(), lr=learning_rate)

# обучение 1000 итераций
for epoch in range(num_epochs):
  outputs = lstm1.forward(X_train_tensors_final)
  optimizer.zero_grad()

  # функция ошибки
  loss = criterion(outputs, y_train_tensors)

  loss.backward()

  # обучение
  optimizer.step()
```

Код, отвечающий за предсказание поведения акций и отрисовку соответствующего графика приведен в листинге 5. График эталонных и предсказанных значений представлен на рисунке 3.

Листинг 5 — предсказание поведения акций

```
df_X_ss = ss.transform(X) # перевод полного набора данных в нужный формат
df_y_mm = mm.transform(y)

df_X_ss = Variable(torch.Tensor(df_X_ss))
df_y_mm = Variable(torch.Tensor(df_y_mm))

df_X_ss = torch.reshape(df_X_ss, (df_X_ss.shape[0], 1, df_X_ss.shape[1]))

# предстказание
train_predict = lstm1(df_X_ss)
data_predict = train_predict.data.numpy()
dataY_plot = df_y_mm.data.numpy()

# обратное масштабирование результатов
data_predict = mm.inverse_transform(data_predict)
dataY_plot = mm.inverse_transform(dataY_plot)
plt.figure(figsize=(10,6))

# конец обучающего набора
plt.axvline(df.index[200], c='r', linestyle='--')

df['predicted'] = data_predict

# визуализация данных
plt.plot(df['feature'], label='Actuall Data') #actual plot
plt.plot(df['predicted'], label='Predicted Data') #predicted plot
plt.title('Time-Series Prediction')
plt.legend()
plt.show()
```

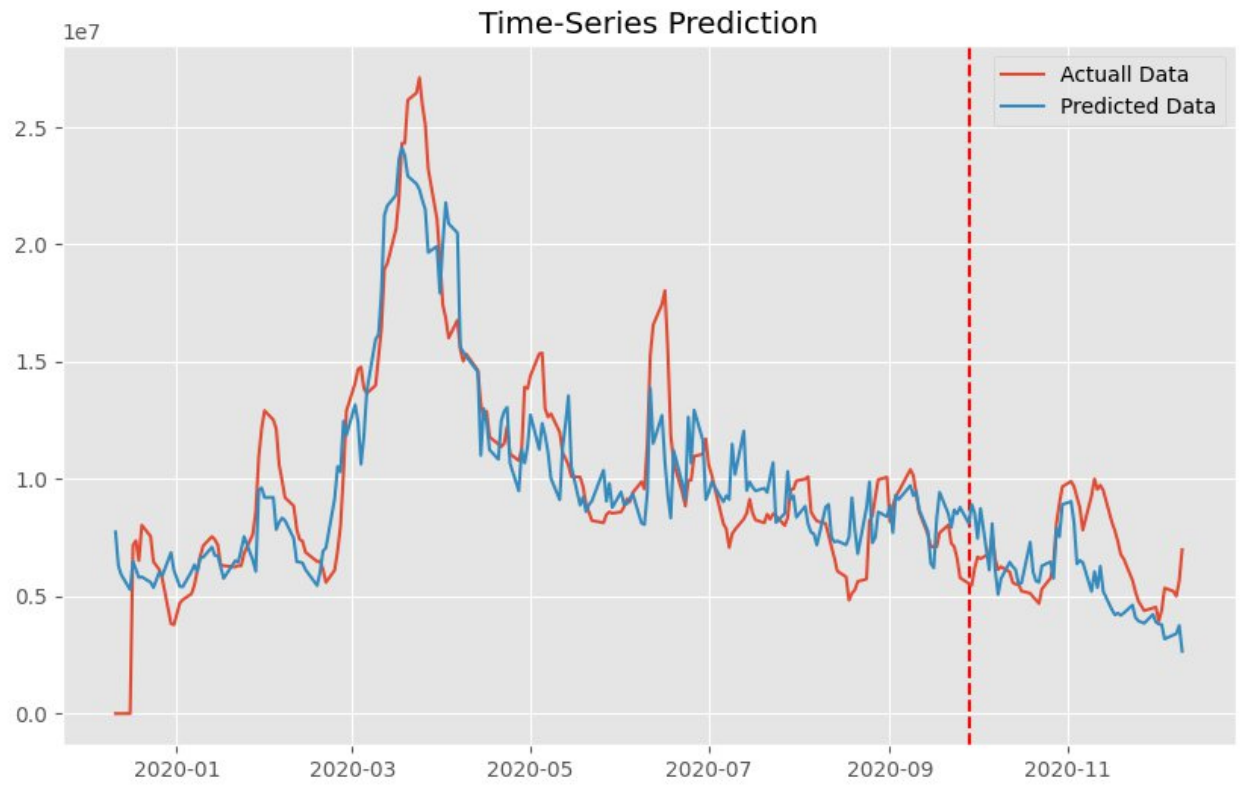

Рисунок 3 — график эталонных и предсказанных значений

Отрисовка графиков, определяющих качетсво модели описана в листинге 6. Сами графики представлены на рисунках 4-6.

Листинг 6 — отрисовка вспомогательных графиков

```
# график ядерной оценки плотности
sns.kdeplot(data=df[['feature', 'predicted']])

import pylab
import scipy.stats as stats

# график квантиль-квантиль
# реальные значения - в тонах синего
fig = plt.figure()
ax = fig.add_subplot(111)
x = df['feature']
res = stats.probplot(x, plot=plt)
ax.get_lines()[0].set_marker('x')
ax.get_lines()[0].set_markerfacecolor('c')
ax.get_lines()[0].set_color('c')
ax.get_lines()[1].set_color('b')
ax.get_lines()[1].set_linestyle(':')

# предсказанные значения - в тонах красного
x = df['predicted']
res = stats.probplot(x, plot=plt)
ax.get_lines()[2].set_marker('.')
ax.get_lines()[2].set_markerfacecolor('r')
```

```
ax.get_lines()[2].set_color('m')
ax.get_lines()[3].set_color('r')
ax.get_lines()[3].set_linestyle('--')
plt.show()

# коррелограмма для разности предсказанных и реальных значений
df['diff'] = df['feature'] - df['predicted']

import statsmodels.api as sm

LAGS=20

fig, ax = plt.subplots(2,1,figsize=(10,10))
sm.graphics.tsa.plot_acf(df['diff'].values.squeeze(), lags=LAGS, ax=ax[0])
plt.show()
```

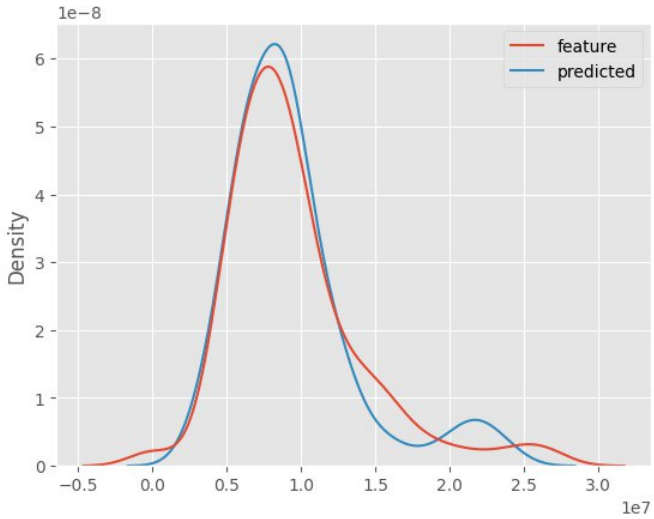

Рисунок 4 — распределение значений

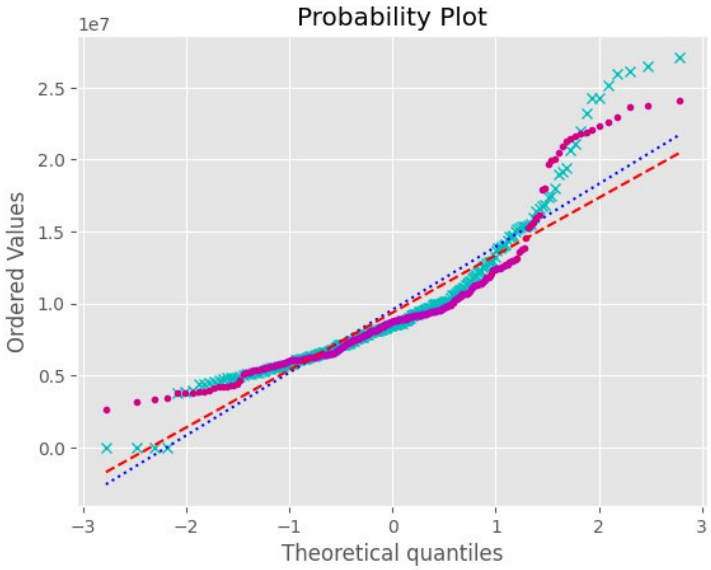

Рисунок 5 — график квантиль-квантиль

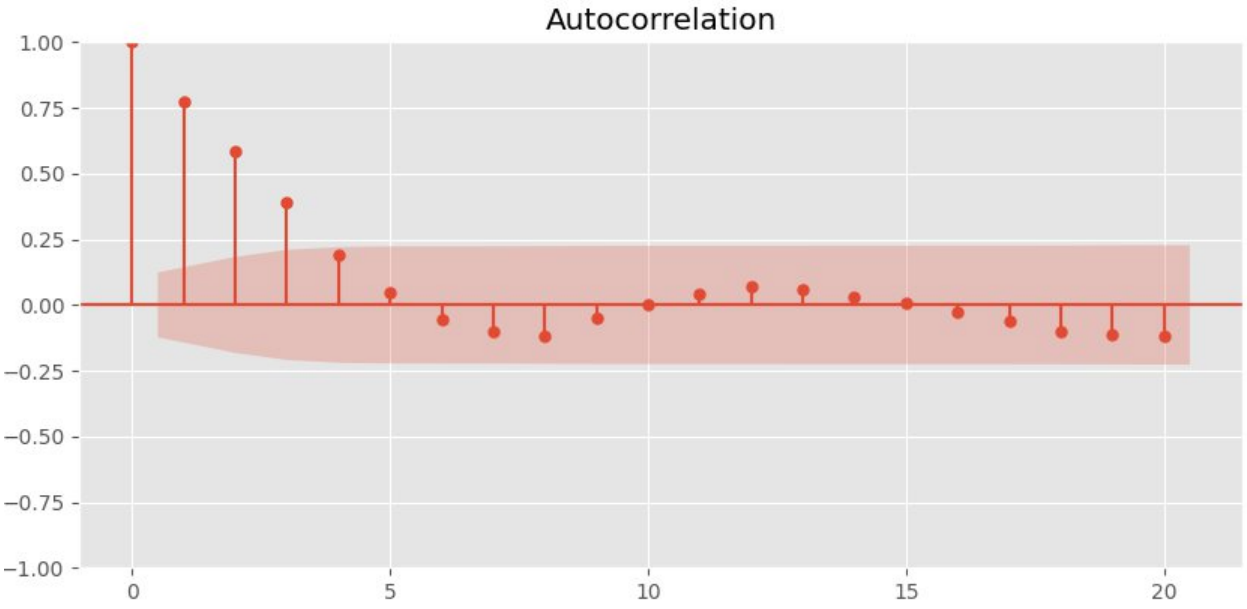

Рисунок 6 — коррелограмма

Как видно из графиков, KDE линия близка к линии нормального распределения, набор данных близок к нормальному распределению – точки на графике квантиль-квантиль лежат близко к диагонали. Большинство точек на коррелограмме попадают в 95% доверительный интервал.

Предсказанные данные сравниваются с показателями реальных данных за 24 месяца. Полученные данные в результате сравнения показывает достаточно низкое отклонение, из чего можно сделать вывод, что предсказание является точным.

Вывод: в ходе лабораторной работы были изучены особенности построения алгоритма реконструкции математической модели человека-оператора по временному ряду.

Контрольные вопросы

1. Приведите примеры временных рядов при создании цифровых двойников;

- биосигналы: частота пульса и дыхания, концентрация различных гормонов и веществ;
- социально-экономические процессы: изменения объемов ВВП, котировок валют и ценных бумаг, изменения рождаемости и смертности, безработицы, изменение потока клиентов;
- природные: изменение атмосферного давления, проход потоков метеоров, изменения температуры;
- и др.+

2. Что понимают под цифровым двойником эксперта?

Цифровой двойник — это цифровая (виртуальная) модель эксперта, принимающая такие же решения, как эксперт в заданной области.

3. В чем состоит задача прогнозирования временных рядов?

Задача прогнозирования временных рядов сводится к выведению зависимости будущих состояний системы/процесса/сигнала от прошлых.

4. Что понимается под реконструкцией математической модели системы? Какова цель реконструкции ММС?

Процесс реконструкции – это получение математической модели системы (ММС) по экспериментальному временному ряду (ВР) a_i(iΔt)=a_i, i=1,…, N. Ее целью является получение ММС в виде уравнений, решение которых с заданной степенью точности воспроизводит исходный ВР a(t).

5. Что такое «переменная состояния» системы? Приведите примеры.

Одна из множества переменных, которые используются для описания математического "состояния" динамической системы. Интуитивно состояние системы описывает достаточно о системе, чтобы определить ее будущее поведение в отсутствие каких-либо внешних сил, воздействующих на систему.

6. Перечислите основные этапы реконструкции математической модели системы;

- постановка задачи;
- определение задачи;
- составление математической модели задачи;
- оценка математической модели / эксперимент;
- выдача результатов.

7. Как оценить адекватность разработанной модели?

- сравнить результаты расчетов по модели с реальным поведением системы в различных ситуациях;
- использовать графики ядерной оценки плотности, график квантиль-квантиль и коррелограмму.