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 домашнему заданию № 1

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

Дисциплина: Дистанционный мониторинг сложных систем и процессов

Студент	ИУ6-12М		С.В. Астахов
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			Ю.А. Вишневская
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2023

Введение

Цель работы: Разработать и исследовать цифрового двойника человека-оператора киберфизической системы.

Ход выполнения

Исходные данные: в качестве задания было выбрано разработать цифрового двойника биржевого трейдера. Двойник реализует две основные задачи: предсказывает поведение цены акций и реализует прибыльную стратегию торговли ими. В качестве исходных данных взяты биржевые котировки компании StarBucks за 2020 год. Фрагмент датасета представлен на рисунке 1.

Date	Open	High	Low	Close	Adj Close	Volume
2019-12-11	86.260002	86.870003	85.849998	86.589996	79.847794	4921900
2019-12-12	88.000000	88.889999	87.540001	88.209999	81.341652	10282100
2019-12-13	88.019997	88.790001	87.580002	88.669998	81.765823	6714100
2019-12-16	89.139999	89.300003	88.430000	88.779999	81.867256	6705600
2019-12-17	88.870003	88.970001	87.470001	88.129997	81.267868	7296900

Рисунок 1 — фрагмент использованного датасета

Рассчитаем скользящее среднее и среднеквадратическое отклонение и визуализируем этот датасет. Так же добавим колонку с будущим значением скользящего среднего (ее значение мы и будем предсказывать). 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)

NUM_FEATURES_N = 4
HIDDEN_SIZE_N = 7
SHIFT_N = -15

df['MA'] = df['Volume'].rolling(window=5, min_periods=1).mean() # скользящее
среднее
df['STD'] = df['Volume'].rolling(window=5, min_periods=1).std() # отклонение

df['MA_lead_15'] = df['MA'].shift(SHIFT_N)
df['MA_lead_15'] = df['MA_lead_15'].fillna(df['MA'])
```

```
from matplotlib import pyplot as plt
import seaborn as sns

# визуализация данных
plt.plot(df.index, df['Open'], label='Open', color='red')
plt.plot(df.index, df['MA'], label= 'MA', color='blue')
plt.plot(df.index, df['MA_lead_15'], label= 'MA_lead_15', color='cyan')
plt.plot(df.index, df['STD'], label= 'STD', color='green')

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

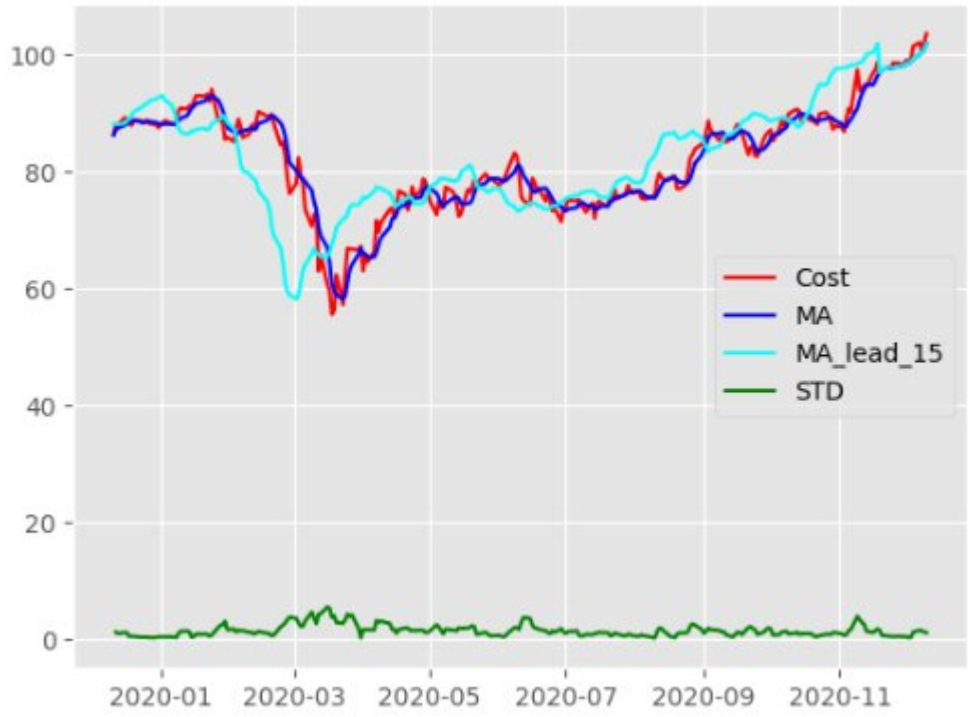

Рисунок 2 —стоимость акции, скользящее среднее и отклонение

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

Листинг 2 — предобработка данных

```
df['feature'] = df['MA_lead_15']

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()
```

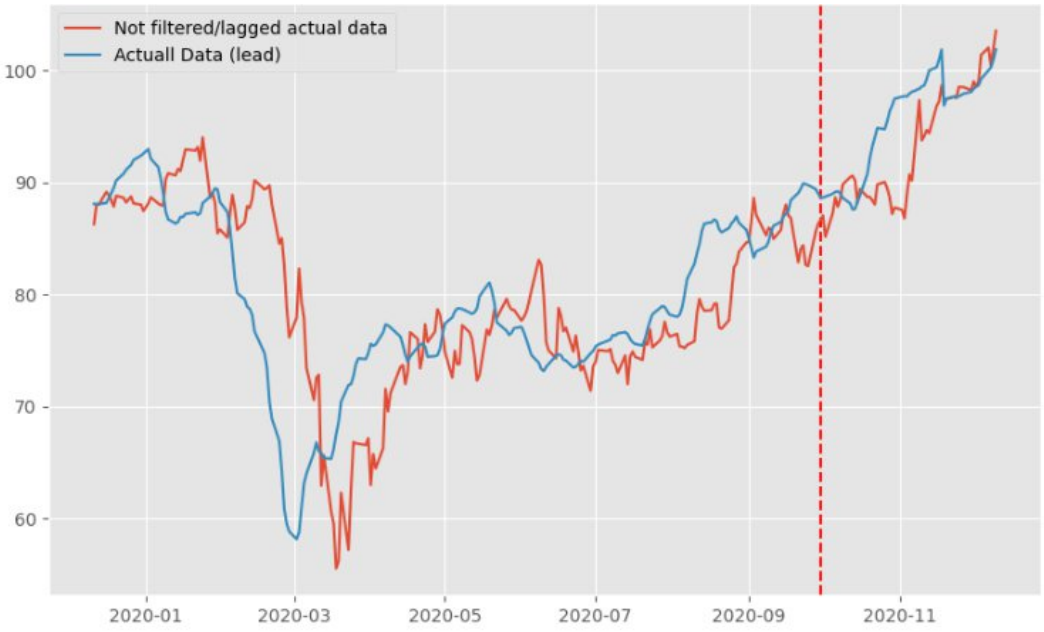

Рисунок 3 — график эталонных и предсказанных значений (со смещением)

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()
```

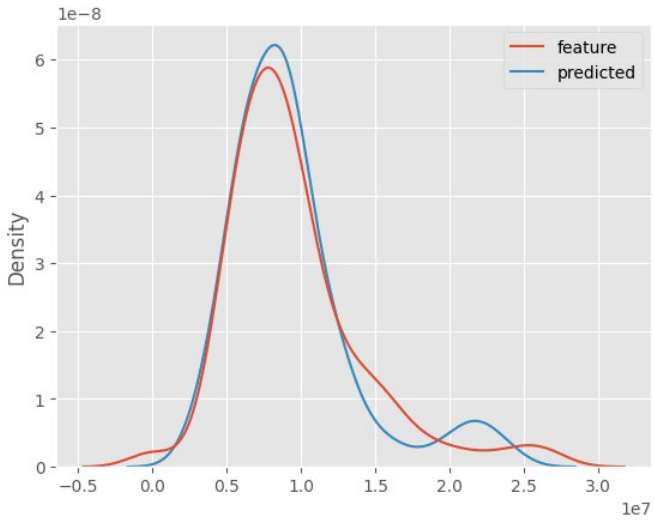

Рисунок 4 — распределение значений

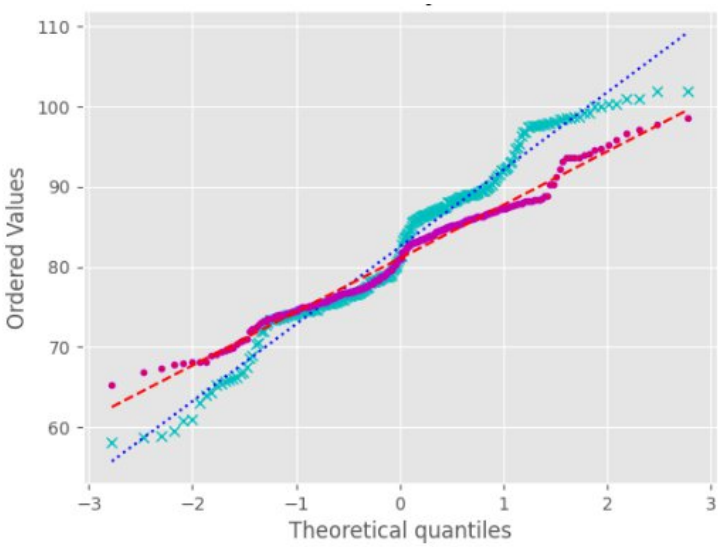

Рисунок 5 — график квантиль-квантиль

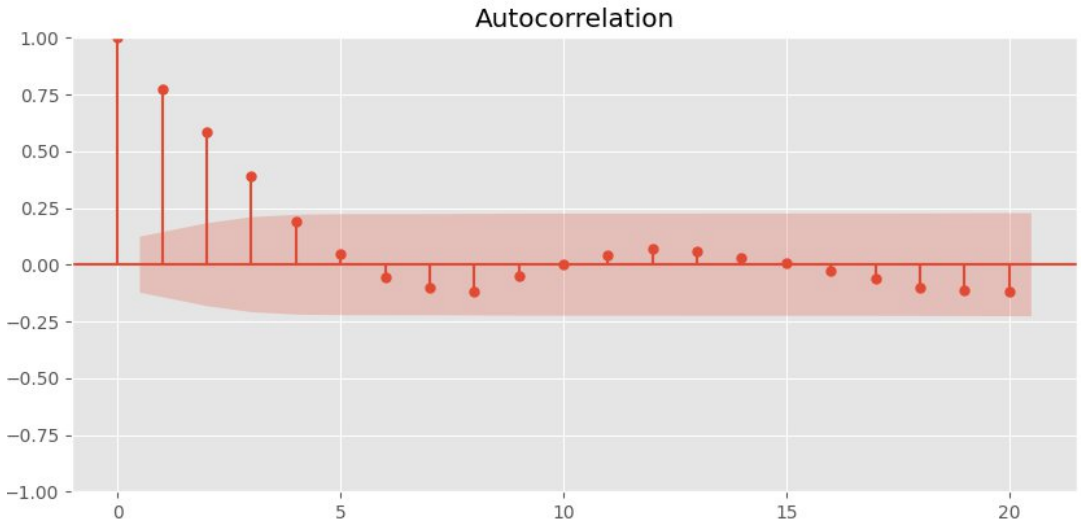

Рисунок 6 — коррелограмма

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.

Скопируем датафрейм и добавим колонку с предсказанием относительного изменения цены акций (листинг 6). Отобразим текущие значение цены, предсказание (со смещением) и разницу между ними (рисунок 7).

Листинг 6 — предсказываемые изменения

```
df2 = df.copy()

# относительное изменение цены
df2['diff_cost_relative'] = (df2['predicted'] - df2['Open']) / (0.5 *
(df2['predicted'] + df2['Open']))

# визуализация предсказываемых изменений
plt.plot(df2['Open'], label='Actuall Data')
plt.plot(df2['predicted'], label='Predicted Data (lead 15 days)')
plt.plot(df2['diff_cost_relative'] * df2['Open'], label='Diff')
plt.title('Time-Series Prediction')
plt.legend()
plt.show()
```

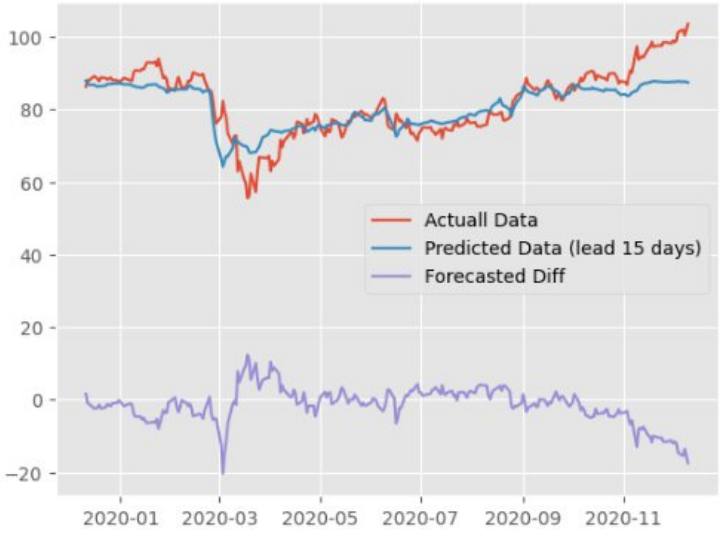

Рисунок 7 — предсказываемые изменения

Разработаем правила торговли акциями. Пусть, если предсказан рост цены акций (более определенного значения), то необходимо купить пропорциональный пакет акций, если падение — продать. Изначально на счетах двойника 100000 денежных единиц и 0 акций. Смоделируем заданные правила (листинг 7).

Листинг 7 — автоматическая торговля акциями

```
# начальные значения
cash = [100000]
assets = [0]
overall = [100000] # валюта + рыночная стоимость акций

koeff = 0.5     # “агрессивность” торговли
min_diff = 0.05 # минимальное необходимое изменение
i = 0

for index, row in df2.iterrows():
    # сколько купить
    buy_assets = (koeff * cash[i] * row['diff_cost_relative']) // row['Open']
    # сколько это стоит
    spend_cash = buy_assets * row['Open']
    # нельзя иметь <0 акций или денежных единиц
    if (cash[i] - spend_cash >= 0) and (assets[i] + buy_assets >= 0) and
abs(row['diff_cost_relative']) > min_diff:
        cash.append(cash[i] - spend_cash)
        assets.append(assets[i] + buy_assets)
    else:
        cash.append(cash[i])
        assets.append(assets[i])
    i += 1
    overall.append(cash[i] + assets[i] * row['Open'])

print(overall[-1])
# (stdout) 142638.98
```

Как видно, конечная стоимость портфеля составила 142638 денежных единиц, что почти в полтора раза больше изначальной. Визуализируем историю операций над портфелем (листинг 8, рисунок 8).

Листинг 8 — история операций над портфелем

```
plt.plot(df2['overall'], label='Overall capitalization')
plt.plot(df2['cash'], label='Cash')
plt.plot(df2['assets_cost'], label='Assets capitalization')
plt.plot(df2['diff_cost_relative'] * 100000, label='Predicted diff (x1000)')
plt.hlines([-min_diff * 100000, min_diff * 100000], df2.index[0], df2.index[-1])

plt.title('Time-Series Prediction')
```

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

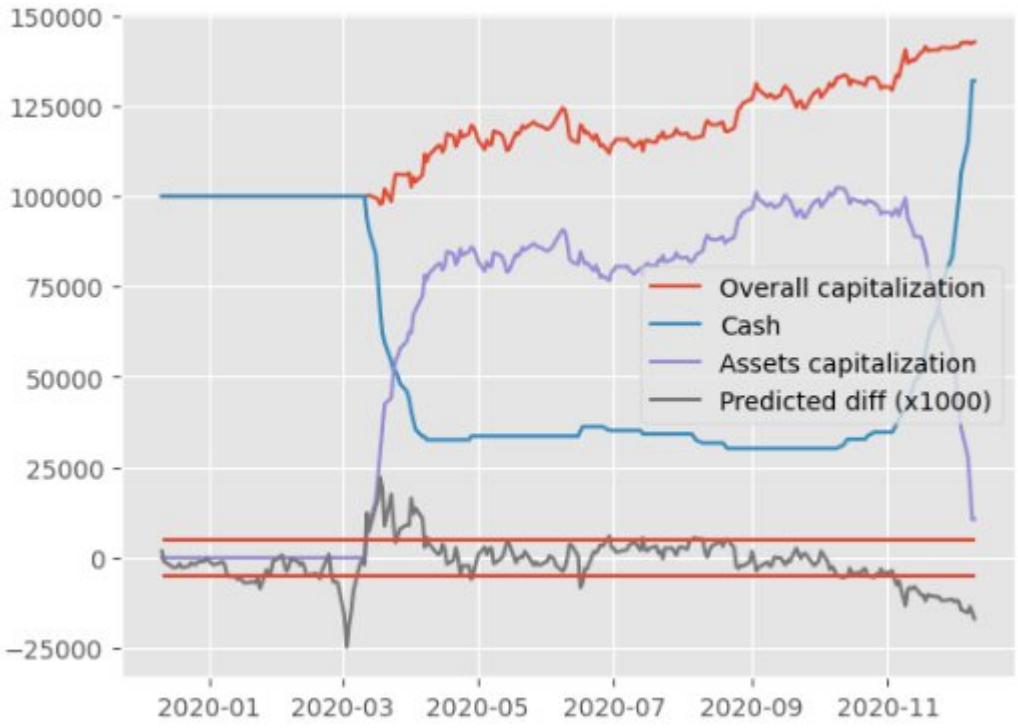

Рисунок 8 — история операций над портфелем

Как видно из графика, капитализация портфеля относительно монотонно росла в рассматриваемом периоде, что говорит о корректности поведения разработанного цифрового двойника.

Вывод: в ходе лабораторной работы были повторены особенности построения алгоритма реконструкции математической модели человека-оператора по временному ряду, разработан и протестирован цифровой двойник биржевого трейдера.

Контрольные вопросы

1. ***С какой целью создаются цифровые двойники?***

   Цифровой двойник эксперта моделирует поведение эксперта с целью поддержки процесса принятия решений. Цифровой двойник технического объекта (устройства, производства, механизма и т.п.) позволяет моделировать различные состояния объекта с целью оптимизации процессов работы с ним, выявления потенциальных угроз и т.п.

2. ***Как формируется структура модели при реконструкции?***

   При реконструкции в машинном обучении структура модели формируется на основе извлечения и отбора признаков, обработки данных и выбора подходящей архитектуры модели. Этот процесс включает в себя несколько этапов:

   - Извлечение и отбор признаков: На этом этапе происходит анализ и выбор наиболее информативных признаков из доступных данных, что позволяет сократить размерность и улучшить производительность модели;

   - Обработка данных: Включает в себя предварительную обработку данных, такую как масштабирование, нормализацию, заполнение пропущенных значений и кодирование категориальных признаков. Цель - подготовить данные для обучения модели;

   - Выбор архитектуры модели: На основе особенностей задачи выбирается подходящая архитектура модели, например, нейронные сети, деревья решений, или другие. Это важный шаг, влияющий на качество и эффективность модели.

3. ***Как оценить адекватность реконструированной модели?***

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

4. ***Укажите возможные аппроксимации области адекватности***

В машинном обучении возможные аппроксимации области адекватности модели включают следующие:

- аппроксимация функции;
- аппроксимация границы;
- аппроксимация вероятности;
- аппроксимация характеристик распределения.

5. ***В чем заключается преимущество структурного подхода при анализе процессов дистанционного мониторинга сложных процессов и систем?*** — Преимущество структурного подхода при анализе процессов дистанционного мониторинга сложных процессов и систем заключается в том, что сложная система разбивается на подсистемы, каждую из которых можно разбить еще на подсистемы (до тех пор, пока это необходимо) и рассмотреть все независимо друг от друга. Но несмотря на это, система сохраняет взаимосвязи между подсистемами, а также получает иерархический вид.
6. ***Укажите этапы построения функциональной модели***

   1. Выделение бизнес-процессов;
   2. Построение контекстной диаграммы;
   3. Декомпозиция необходимых уровней.

7. ***Для чего реализуется процесс декомпозиции?***

   Декомпозиция системы на подсистемы позволяет лучше понять механизмы ее функционирования за счет уменьшения сложности рассматриваемого за раз блока (подсистемы).

8. ***Назовите правила построения IDEF0-модели***

   1. Строится контекстная диаграмма, которая в дальнейшем разбивается на подсистемы (происходит декомпозиция) до тех пор, пока это необходимо;
   2. Каждая диаграмма содержит механизмы (стрелка в блок снизу) и данные (стрелка в блок сверху) управления, а также входные (стрелка в блок слева) и выходные параметры (стрелка из блока справа).

3. Блоки располагаются по диагонали из левого верхнего угла в правый нижний. И из каждый предыдущий блок соединяется соединяется с каждым последующим блоком стрелкой выходных параметров.
4. Уровень декомпозиции должен содержать не менее 3 и не более 6 блоков.

9. С какой целью строится начальная контекстная диаграмма?

Контекстная диаграмма строится с целью дать общую информацию о рассматриваемой системе, показать какие в ней данные и механизмы управления, а такие будут входные и выходные параметры.

10. Зачем необходимо указывать точку зрения на начальной контекстной диаграмме?

Точка зрения показывает с какой стороны будет рассмотрена данная система. А также в каком направлении необходимо будет производить декомпозицию отдельных элементов, необходимых только для выбранной точки зрения.