|  |  |
| --- | --- |
|  | **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**

**Отчет**

|  |  |
| --- | --- |
| **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.

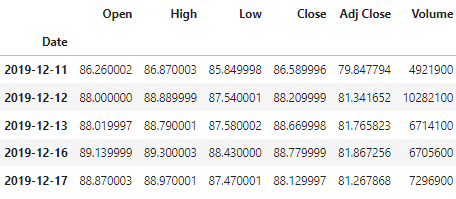


Рисунок 1 — фрагмент использованного датасета

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() |

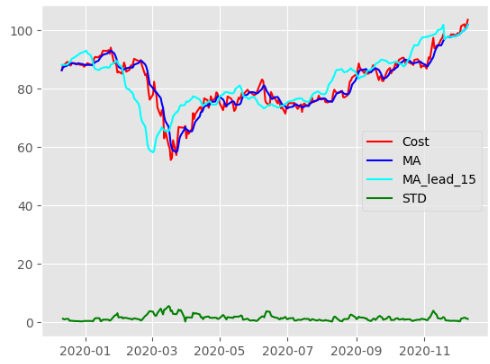


Рисунок 2 —стоимость акции, скользящее среднее и отклонение

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):  # иницилизация 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() |

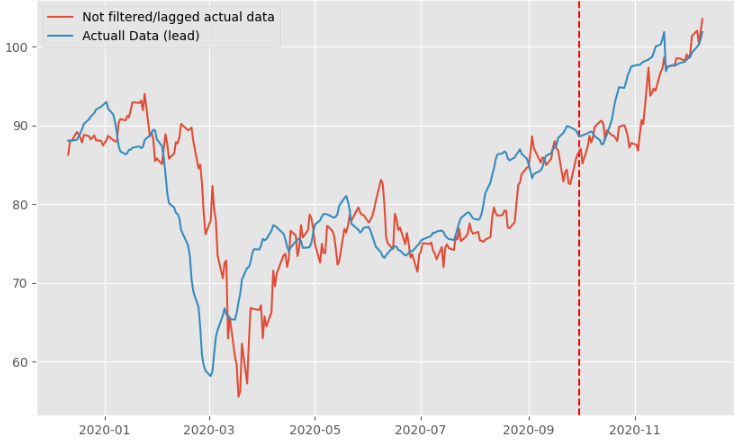


Рисунок 3 — график эталонных и предсказанных значений (со смещением)

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() |

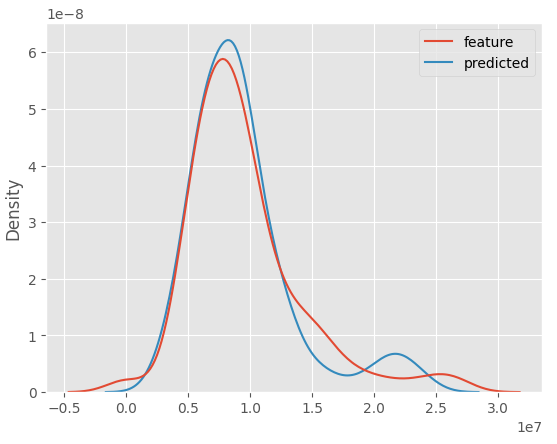


Рисунок 4 — распределение значений

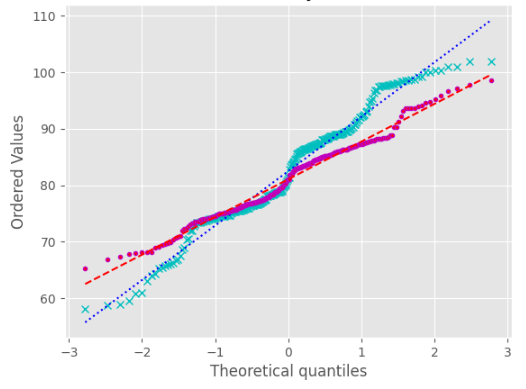


Рисунок 5 — график квантиль-квантиль

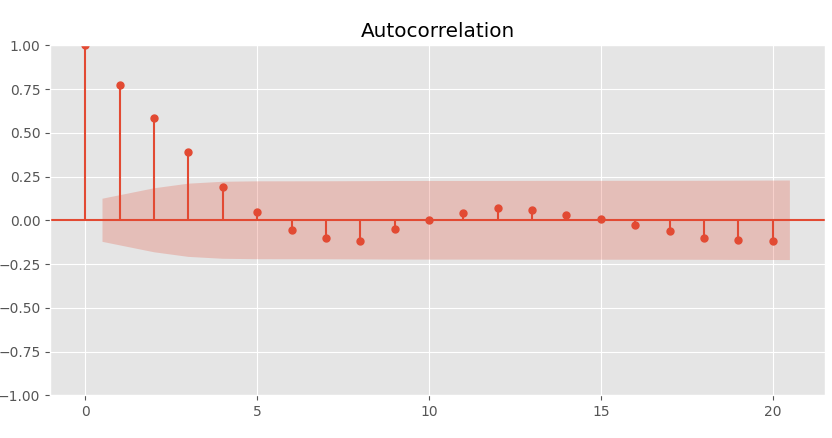


Рисунок 6 — коррелограмма

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() |

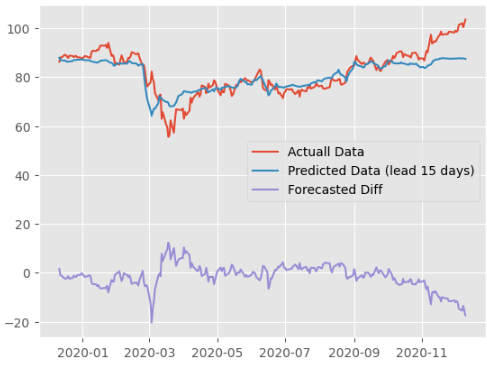


Рисунок 7 — предсказываемые изменения

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() |

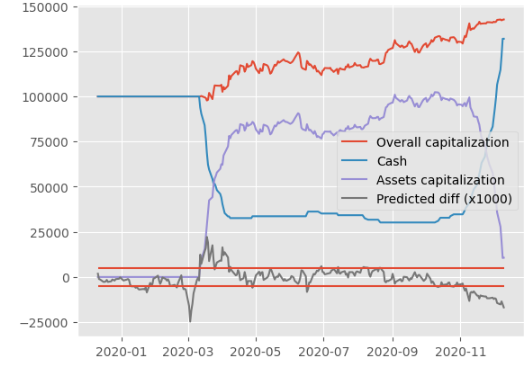


Рисунок 8 — история операций над портфелем

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.

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

1. ***Укажите возможные аппроксимации области адекватности***
2. ***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.
3. ***Укажите этапы построения функциональной модели***

1. Выделение бизнес-процессов;

2. Построение контекстной диаграммы;

3. Декомпозиция необходимых уровней.

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

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

1. ***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.

1. ***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.

1. ***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.