МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тюменский индустриальный университет»

Институт сервиса и отраслевого управления

Кафедра «Бизнес-информатики и математики»

**КУРСОВОЙ ПРОЕКТ**

**Дисциплина:** «Объектно-ориентированный анализ и программирование»

**Тема:** «Разработка нейросети на основе пакетов Python»

**Выполнили:**студент 1 курса направление 38.03.05 «Бизнес – информатика» группа БИб-19-1

Лисовский А.И.

студент 1 курса направление 38.03.05 «Бизнес – информатика» группа БИб-19-1

Михеенков А.С.

**Проверил:**

к.т.н., доцент Сергеев В.В.

Тюмень – 2020

Оглавление

[Введение 3](#_Toc60400860)

[1. Теоретические основы объектно-ориентированного программирования и нейронных сетей 4](#_Toc60400861)

[1.1. История развития и основные понятия нейронных сетей и ООП 4](#_Toc60400862)

[1.2. Основы объектно-ориентированного програмирования 4](#_Toc60400863)

[1.3. Нейронные сети 5](#_Toc60400864)

[1.4. Язык программирования python 6](#_Toc60400865)

[2. Практическая часть 7](#_Toc60400866)

[2.1. Выбор пакета для работы с нейронными сетями 7](#_Toc60400867)

[2.2. Выбор темы и поиск информации 7](#_Toc60400868)

[2.3. Создание и обучение модели. 8](#_Toc60400869)

[2.4. UML диаграмма 9](#_Toc60400870)

[Заключение 11](#_Toc60400871)

[Список литературы 12](#_Toc60400872)

[Приложение 13](#_Toc60400873)

# Введение

В наше время повсеместно начинают использоваться нейронные сети. А язык программирования Python является одним из самых распространённых. Неудивительно что существуют пакеты для работы с нейронными сетями на этом языке. Существует огромное количество способов их применения, в этой работе мы попробуем сделать модель для предсказания курса акций.

Объект проекта: пакет для создания и обучения нейронных сетей.

Предмет исследования: нейронная сеть как модель для прогнозирования.

Цель: спроектировать нейронную сеть с помощью пакета для языка программирования Python.

Задачи:

* Выбрать пакет для создания нейронных сетей
* Изучить выбранный пакет
* Выбрать область для создания нейронной сети
* Собрать и обработать данные для обучения и тестирования модели
* Спроектировать и создать нейронную сеть
* Проанализировать качество работы сети

# Теоретические основы объектно-ориентированного программирования и нейронных сетей

## История развития и основные понят ия нейронных сетей и ООП

Идея объектно-ориентированного программирования сформировалась 70-80 годы XX века. Основными понятиями ООП являются класс и объект (экземпляр класса).

Впервые понятие Нейронной сети было сформулировано У. Маккалоком и У. Питтсом. Их основной сутью является имитация нейронных связей мозга животных. Основными понятиями являются нейрон, связь и функция активации. Причём в этой области ООП может раскрыться во всей красе. Можно создать класс нейронов и нейронной сети. Однако современные пакеты облегчают жизнь разработчикам и скрывают многие тонкости, давая программистам удобный интерфейс создания и контроля нейронной сети.

## Основы объектно-ориентированного програмирования

Существует четыре основных принципа ООП:

* Абстракция, выделение в объекте изучения основных признаков и свойств
* Инкапсуляция, разделение кода на части и ограничение доступа к свойствам и полям
* Наследование, создание дочерних классов, наследующих характеристики класса родителя
* Полиморфизм, создание абстрактных классов, методы которых реализуются в дочерних классах

Достоинства ООП:

* Создание понятий близких к предметной области, по сути, можно расширить язык программирования своими понятиями
* Возможность создания сложных и комплексных программ при помощи инкапсуляции
* Возможность использовать один код несколько раз, при помощи наследования
* Относительная лёгкость в модификации программы
* Возможность создания и использования своих библиотек классов

Недостатки ООП:

* Небольшое снижение быстродействия программы
* Порог вхождения увеличивается, ООП сложнее в понимании чем процедурное программирование
* Неквалифицированное применение ООП может привести к ухудшению программы

## Нейронные сети

Нейронные сети – это часть более общего направления, машинного обучения, задачей которого является создание моделей прогнозирования. Нейронные сети – это один из мощнейших инструментов данной области.

Также более объёмные и сложные нейронные сети входят в понятие глубокого обучения, одно из самых востребованных направлений в наши дни. Нейронные сети состоят из двух элементов: нейроны и связь между ними. У нейрона есть своего рода «порты» входа и выхода информации. Нейрон получает по связям информацию от других нейронов, преобразует её с помощью функции активации и отправляет следующим нейронам. В нейронной сети нейроны расположены по слоям, где каждый нейрон из одного слоя связан с каждым нейроном следующего слоя. Также имеются входной и выходной слои служащие для взаимодействия с нейронной сетью. Остальные слои называются скрытыми и зачастую на диаграммах их заменяют одним «слоем». Это базовая конструкция нейронной сети. Существуют другие более сложные конструкции как, например свёрточные или рекурсивные сети. При проектировании нейронных сетей обычно сталкиваются с двумя проблемами: переобучение сети и недообучение сети. В результате переобучения модель слишком подстраивается под обучающую выборку, и в результате не способна эффективно работать с генеральным множеством. В случае с недообучением модель не способна выделить существенные признаки в обучающей выборке, это означает то, что она либо слишком проста, либо обучающей выборки недостаточно и её надо расширять или изменять.

## Язык программирования python

Язык программирования python – это объектно-ориентированный язык, в котором почти каждый элемент является своего рода частью этой концепции. Основным преимуществом данного языка является его читабельность и низкий порог вхождения, что позволяет выбирать его в качестве первого языка. Однако не существует идеального инструмента, за высокую скорость написания кода приходится платить производительностью. Python – интерпретируемый язык, а значит по своей скорости он в значительной степени уступает таким компилируемым языкам как C++ или C#. Однако Python можно расширить при помощи пакетов, написанных другими программистами, и на данный момент уже существует огромное количество пакетов для разных целей. Например, для написания 2D игр существует пакет pygame, для создания сайтов – Flask, Django, и для нейронных сетей – Keras, построенный как интерфейс-окружение для Tensorflow.

## Пакеты Tenzorflow и Keras

Tenzorflow – комплексная платформа для работы с машинным обучением. Данная система разрабатывается компанией Google. Её код находится в открытом доступе, что позволяет всем заинтересованным в машинном обучении лицам опробовать данную систему. Также разработан пакет Tensorflow для Python позволяющий взаимодействовать с этой платформой через язык программирования Python.

# Практическая часть

## Выбор пакета для работы с нейронными сетями

Существует много пакетов для работы с нейронными сетями на языке Python: Theano, TensorFlow, Keras, Lasange, Neon и другие. Для данного проекта мы выбрали пакет Keras так, как он представляет своего рода надстройку для TensorFlow и Theano. Таким образом он включает в себя возможности этих двух пакетов и удобный интерфейс для создания и тестирования нейронных сетей.

## Выбор темы и поиск информации

Для данного проекта мы хотим попытаться сделать модель для предсказания курса акций на некоторый промежуток времени. Мы навряд ли сможем найти готовые наборы информации, поэтому мы обратились к сайту Московской биржи для поиска информации. На данном сайте можно узнать результаты торгов для многих компаний. Поэтому была написана программа для сбора данной информации. В первую очередь мы получили коды фирм, чьи акции представлены на данном сайте. Затем мы создали программу, которая каждые полсекунды отправляет запрос на сайт Московской биржи и получает результаты торгов на сто дней от указанной даты. В конечном итоге один элемент набора данных представляет собой два массива в каждом из которых находится нормализованная разница в ценах на момент закрытия торгов. Общая длинна этих двух массивов равна 100. Первый массив будет подаваться нейронной сети на вход, а второй будет выступать в качестве результата, который мы хотим получить.

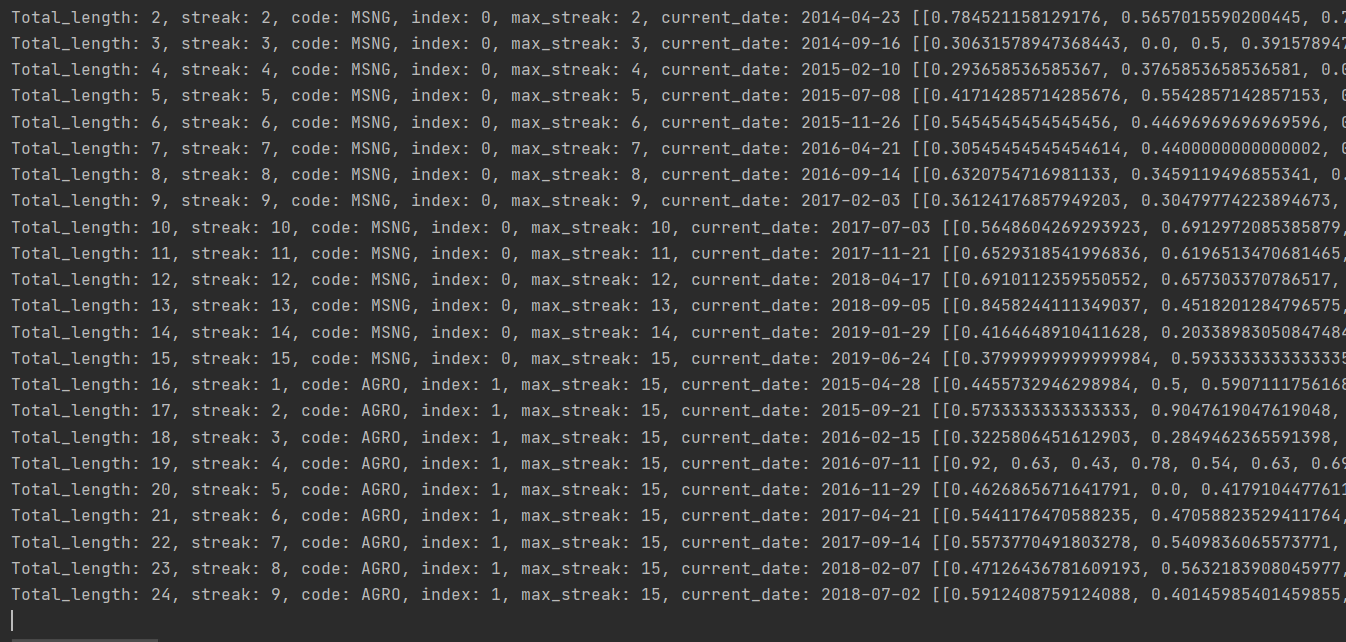


Рисунок 1 Процесс сбора и обработки информации

В итоге был получен массив длинной в 2539 элементов, которые состоят из двух массивов, описанных выше. Мы прекрасно понимаем то, что курс акций зависит от многих факторов, которые мы берём в расчёт, и то, что данного набора информации недостаточно для задачи такой сложности. Однако, было бы интересно попробовать сделать модель высокой точности при таких ограничениях.

## Создание и обучение модели.

В результате многократных экспериментов, в процессе которых мы изменяли характеристики сети с целью найти наиболее оптимальный вариант, была получена нейронная сеть со следующей структурой: первый входной слой с 93 слоями, затем два скрытых слоя с 10 нейронами и гиперболическим тангенсом в качестве функции активации и выходной слой с 7 нейронами и логистической функцией в качестве функции активации.

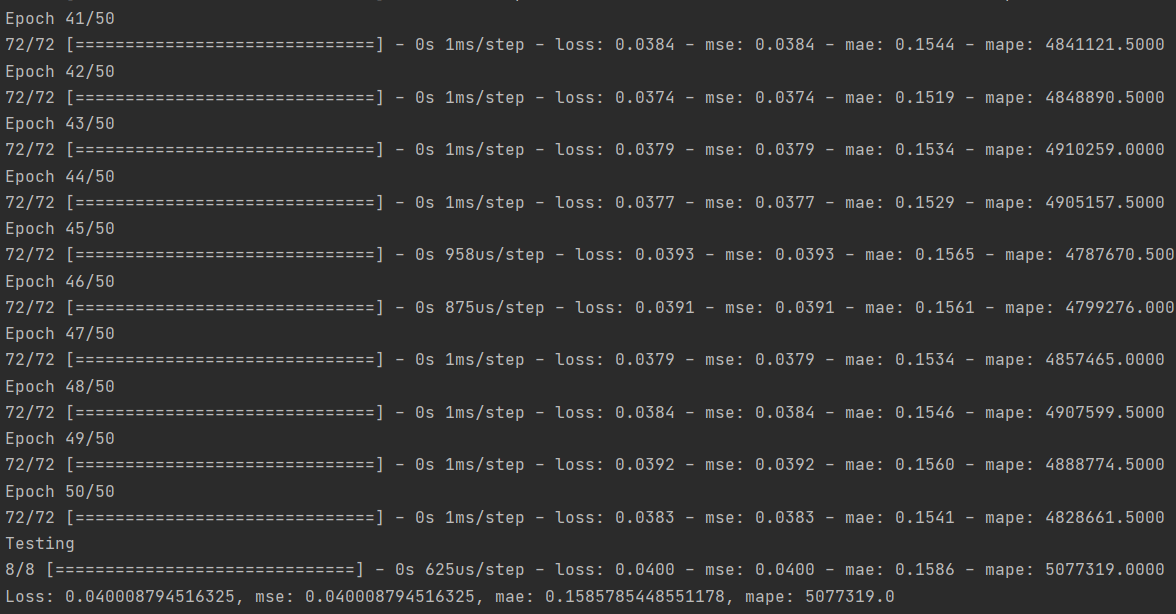


Рисунок 2 Результат обучения и тестирования

Изначально модель создаётся со случайными значениями весов, поэтому значение среднеквадратической ошибки сильно колеблется от 3 сотых до 5 сотых, что не является таким ужасным результатом, но и хорошим назвать его сложно. Но с помощью такой модели уже можно делать относительные предсказания. Значения среднеквадратичной ошибки во время тестирования и обучения не сильно отличается. Из этого мы можем сделать вывод, что основной проблемой является нехватка набора данных, их просто не хватает для полноценного обучения модели, и при этом задача довольно сложная, потому что даже при многократном обучении модель справляется сравнительно также при тестировании. Таким образом происходит недообучение.

## UML диаграмма

Увы, но из-за готового интерфейса Keras довольно проблематично создать красочный класс. Но всё же в процессе работы над данным проектом был создан класс Network, схему которого вы можете увидеть ниже.

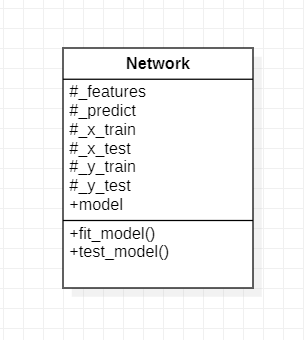


Рисунок 3 UML диаграмма класса Network

# Заключение

В процессе выполнения данного проекта был изучен пакет Keras, используемый для создания нейронных сетей. А также была спроектирована модель для прогнозирования курса акций Московской биржи. Хотя данная модель не идеальна с её помощью можно делать относительные прогнозы акций. Было очень интересно искать информацию и обрабатывать её. Ведь сейчас самым сложным в машинном обучении является поиск и обработка информации для обучения модели.

# Список литературы

* Keras API reference [Электронный рессурс].- Режим доступа URL: <https://keras.io/api/> (дата обращения : 18.12.2020)
* Рашка С. Python и машинное обучение / пер. с англ. А. В. Логунова. - М.: ДМК Пресс, 2017. - 418 с.: ил.
* Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. — СПб.: Питер, 2018. — 480 с.: ил. — (Серия «Библиотека программиста»).
* Джоши, Пратик. Искусственный интеллект с примерами на Python. : Пер. с англ. - СПб. : ООО "Диалектика", 2019. - 448 с. - Парал. тит. англ.

# Приложение

1. Листинг кода из файла gather\_SECID.py

import requests  
import pandas as pd  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 req = requests.get(  
 'https://iss.moex.com/iss/engines/stock/markets/shares/boardgroups/57/securities.jsonp?iss.meta=off&iss.json=extended&callback=angular.callbacks.\_k&security\_collection=3&sort\_column=SHORTNAME&sort\_order=asc&lang=ru&\_=1608203668291')  
  
 text = req.text  
 key = '"SECID": "'  
 code\_list = []  
 while text.count(key) > 0:  
 code = text[text.index(key) + 10:text.index(key) + 15]  
 if code[-1] == '"':  
 code = code[:-1]  
 if code not in code\_list:  
 code\_list.append(code)  
 text = text.replace(key+code+'"', '0'\*15, 2)  
  
 df = pd.DataFrame(code\_list, columns=['SECID'])  
 df.to\_csv('data/SECID.csv', index=False)

1. Листинг кода из файла gather\_datasets.py

import requests  
import pandas as pd  
import datetime  
import json  
import time  
  
# import random  
  
df = pd.read\_csv('data/SECID.csv')  
print(df)  
total\_data = {'features': [], 'predict': []}  
streak = 0  
max\_streak = 0  
previous\_code = df.at[0, 'SECID']  
data\_length = 100  
data\_test\_length = 7  
  
start\_counting\_time = datetime.datetime.now()  
  
for index, code\_i in df.iterrows():  
  
 code = code\_i['SECID']  
 streak = 0  
  
 date = datetime.date(day=1, month=1, year=2005)  
  
 req = requests.get(f'https://iss.moex.com/iss/history/engines/stock/markets/shares/boards/TQBR/securities/'  
 f'{code}.jsonp?from={date.strftime("%Y-%m-%d")}&iss.meta=off&iss.json=extended&callback=angular.callbacks.\_s&\_=1608126939154')  
  
 text = req.text  
 text = text[22:-2]  
  
 my\_data = json.loads(text)  
 start\_price = my\_data[1]['history'][0]['LEGALCLOSEPRICE']  
 current\_date = datetime.datetime.strptime(my\_data[1]['history'][0]['TRADEDATE'], '%Y-%m-%d').date()  
  
 while (current\_date + datetime.timedelta(days=600)) < datetime.date.today():  
 data = [[], []]  
  
 while (len(data[0]) + len(data[1])) < data\_length:  
 time.sleep(0.5)  
 req = requests.get(f'https://iss.moex.com/iss/history/engines/stock/markets/shares/boards/TQBR/securities/'  
 f'{code}.jsonp?from={current\_date.strftime("%Y-%m-%d")}&iss.meta=off&iss.json=extended&callback=angular.callbacks.\_s&\_=1608126939154')  
  
 text = req.text  
 text = text[22:-2]  
 my\_data = json.loads(text)  
  
 if (current\_date + datetime.timedelta(days=150)) > datetime.datetime.now().date():  
 break  
  
 if len(my\_data[1]['history']) == 0:  
 current\_date += datetime.timedelta(days=50)  
 continue  
 previous\_day = my\_data[1]['history'][0]  
 counter = 1  
 while (previous\_day['LEGALCLOSEPRICE'] is None) and (counter < len(my\_data[1]['history'])):  
 previous\_day = my\_data[1]['history'][counter]  
 counter += 1  
  
 if counter == len(my\_data[1]['history']):  
 current\_date += datetime.timedelta(days=100)  
 continue  
  
 for i in range(counter, len(my\_data[1]['history'])):  
 if (len(data[0]) + len(data[1])) == data\_length:  
 break  
 day = my\_data[1]['history'][i]  
 current\_date = datetime.datetime.strptime(day['TRADEDATE'], '%Y-%m-%d').date()  
 if day['LEGALCLOSEPRICE'] is None:  
 continue  
  
 if len(data[0]) >= (data\_length - data\_test\_length):  
 data[1].append(day['LEGALCLOSEPRICE']-previous\_day['LEGALCLOSEPRICE'])  
 else:  
 data[0].append(day['LEGALCLOSEPRICE']-previous\_day['LEGALCLOSEPRICE'])  
 previous\_day = day  
 if code == previous\_code:  
 streak += 1  
 else:  
 streak = 1  
 if streak > max\_streak:  
 max\_streak = streak  
 if (data[0].count(0) + data[1].count(0)) > (data\_length//2):  
 continue  
 if (len(data[0]) + len(data[1])) == data\_length:  
 max\_dif = max(max(data[0]), max(data[1]))  
 min\_dif = min(min(data[0]), min(data[1]))  
 if (max\_dif-min\_dif) == 0:  
 max\_dif += max\_dif/10  
 for i in range(len(data[0])):  
 data[0][i] = (data[0][i] - min\_dif)/(max\_dif-min\_dif)  
 for i in range(len(data[1])):  
 data[1][i] = (data[1][i] - min\_dif)/(max\_dif-min\_dif)  
  
 total\_data['features'].append(data[0])  
 total\_data['predict'].append(data[1])  
  
 print(f'Total\_length: {len(total\_data["features"])}, streak: {streak},'  
 f' code: {code}, index: {index}, max\_streak: {max\_streak}, current\_date: {current\_date}', data)  
 previous\_code = code  
write\_data = pd.DataFrame(total\_data)  
write\_data.to\_csv('data/DATASETS.csv', mode='w', index=False)  
print(f'finished in {datetime.datetime.now() - start\_counting\_time}')

1. Листинг кода из файла main.py

import pandas as pd  
import os  
os.environ["TF\_CPP\_MIN\_LOG\_LEVEL"] = "2" # Ограничение на использовании графического процессора  
from tensorflow import keras  
from tensorflow.keras import layers  
from sklearn.model\_selection import train\_test\_split  
import numpy as np  
  
  
class Network:  
 def \_\_init\_\_(self, dataset):  
 self.\_features = np.empty((0, len([float(x) for x in dataset['features'][0][1:-1].split(', ')])), int)  
 self.\_predict = np.empty((0, len([float(x) for x in dataset['predict'][0][1:-1].split(', ')])), int)  
  
 for index, row in dataset.iterrows():  
 self.\_features = np.vstack(  
 (self.\_features, np.asarray([float(x) for x in row['features'][1:-1].split(', ')]).astype(np.float32)))  
 self.\_predict = np.vstack(  
 (self.\_predict, np.asarray([float(x) for x in row['predict'][1:-1].split(', ')]).astype(np.float32)))  
  
 self.\_x\_train, self.\_x\_test, self.\_y\_train, self.\_y\_test = train\_test\_split(self.\_features, self.\_predict,  
 test\_size=0.1)  
  
 self.model = keras.Sequential([  
 layers.InputLayer(input\_shape=(len(self.\_x\_train[0]),)),  
 layers.Dense(10, activation=keras.activations.tanh),  
 layers.Dense(10, activation=keras.activations.tanh),  
 layers.Dense(len(self.\_y\_train[0]), activation=keras.activations.sigmoid)  
 ])  
 self.model.compile(loss='mse', optimizer=keras.optimizers.Adam(0.1), metrics=['mse', 'mae', 'mape'])  
  
 def fit\_model(self):  
 return self.model.fit(self.\_x\_train, self.\_y\_train, epochs=50, verbose=1)  
  
 def test\_model(self):

print('Testing')  
 return self.model.evaluate(self.\_x\_test, self.\_y\_test, verbose=1)  
  
  
df = pd.read\_csv('data\_gazering/data/DATASETS.csv')  
  
network = Network(df)  
history = network.fit\_model()  
result = network.test\_model()  
print(f'Loss: {result[0]}, mse: {result[1]}, mae: {result[2]}, mape: {result[3]}')