**Прогнозирование цены акций СБЕРБАНКА с ML в Python**

**Введение**

Одним из наиболее распространенных примеров использования машинного обучения является «Fintech»; большая часть которых находится на фондовом рынке.

Финансовые исследователи данных и ученые в течение последних многих лет были наняты для того, чтобы разобраться в рыночном пространстве и повысить рентабельность и прибыльность инвестиций. Тем не менее, из-за многомерного характера проблемы, масштаба системы и присущего ей изменения во времени, это была чрезвычайно трудная задача для людей, которую необходимо решить, даже с помощью обычных инструментов анализа данных. Тем не менее, с появлением последних достижений в области применения машинного обучения, эта область развивалась, чтобы использовать недетерминированные решения, чтобы «узнать», что происходит, чтобы делать более точные прогнозы.

В данном проекте я продемонстрирую простую модель прогнозирования цен на акции СБЕРБАНКА.

**Шаг 1: Выбор данных**

Одним из наиболее важных шагов в машинном обучении и прогнозном моделировании является сбор хороших данных, выполнение соответствующих этапов очистки и реализация ограничений.

В этом примере я буду использовать данные о ценах акций СБЕРБАНКА, которые были взяты с сайта finance.yahoo.com

Были взяты данные с 04.11.2016 по 04.11.2021

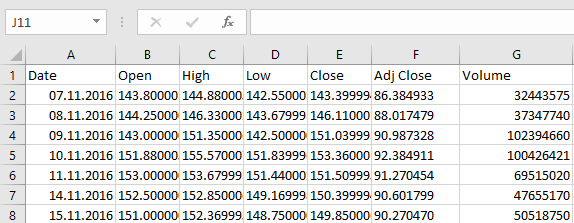


Рис.1 – Первичные данные в Excel

Можно заметить, что все поля являются числовыми значениями, кроме значения даты. Исправим это.

Значения, которые будут передаваться в модель, должны быть в формате, который легче всего понять.

Таким образом, выполним несколько шагов «предварительной обработки данных».

Вставим новый столбец после первого, назовем его «Date Value» и скопируем все даты из первого столбца в только что созданный.

Затем выберем все данные и измените тип с «Date» на «Текст». Результаты выглядят следующим образом:

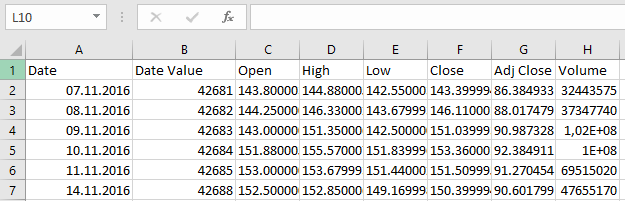


Рис.2 – Обработанные данные в Excel

Прежде чем начать, отметим ограничения.

Можно заметить, что единственными данными, которые предоставляются в эту модель, являются дата и цена. Есть много внешних факторов, которые влияют на цену за пределами исторической цены. В высоконадежных моделях могут использоваться внешние данные, например новости, время года, настроения в социальных сетях, погода, цена конкурентов, волатильность рынка, рыночные индексы и т. Д.

Эта модель очень проста, но со временем ее можно будет улучшить.

**Шаг 2: Выбор модели**

Итак, теперь, когда данные подготовлены, нужно выбрать модель. В этом случае было принято решение использовать нейронную сеть для выполнения функции регрессии.

Регрессия будет отображать числовое значение в непрерывном масштабе, применяя его к модели, которая может быть использована для классификационных усилий, которые дадут категорический результат. В этой ситуации пытаемся предсказать цену акции в любой данный день.

Для построения нашей модели будем использовать TensorFlow(упрощенный модуль под названием TFANN) . Для реализации будем использовать Google Colab

**Шаг 3: Построение модели**

Сначала нам нужно установить TFANN.

Для установки TFANN выполним следующую команду:

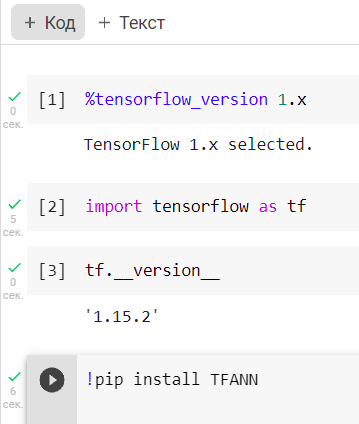


Рис.3 – Установка TFANN

Импортируем библиотеки:

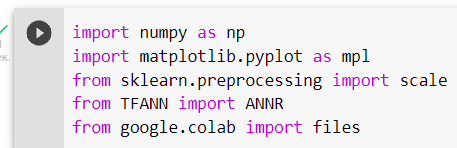


Рис.4 – Импорт библиотек

NumPy будет использоваться для наших матричных операций, Matplotlib для графиков, sykit-learn для обработки данных, TFANN для ML и файлы google.colab помогут загружать данные с компьютера в виртуальную среду.

Теперь импортируем данные, которые уже обработали. Для этого выполним следующую команду, которая предоставит нам окно для загрузки файла .csv

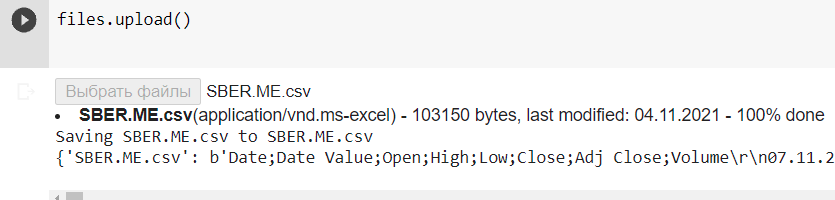


Рис.5 – Загрузка файла с исходными данными

Запишем код и выведем данные на график:

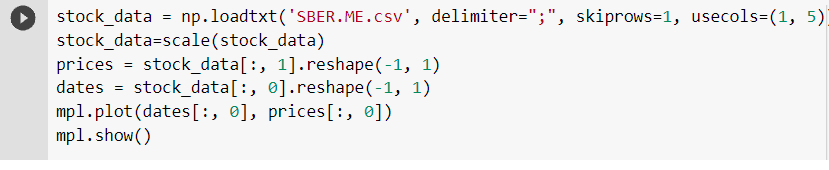


Рис.6 – Prices and Dates(plot)

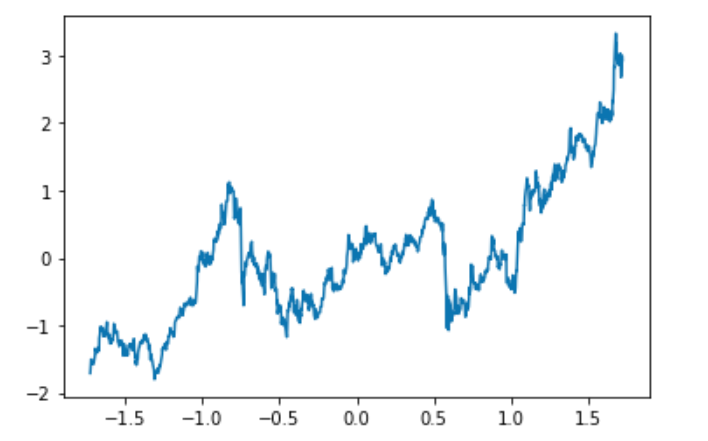


Рис.7 – График цен

Сократили объем данных, чтобы сделать процесс обучения более эффективным.

Теперь нужно построить модель. В этом случае мы будем использовать один входной и выходной нейрон (дата - ввод, цена выход) и будет иметь три скрытые слои по 75 нейронов каждый.

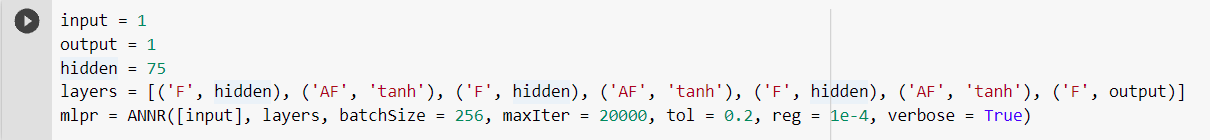
Каждый слой будет иметь функцию активации «tanh». 

Рис.8 – Объявление слоев

Модель готова к обучению.

**Шаг 4: Обучение модели**

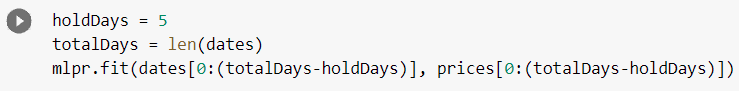


Рис.9 – Обучение

После завершения обучения мы можем выполнить следующие команды, чтобы увидеть результат.

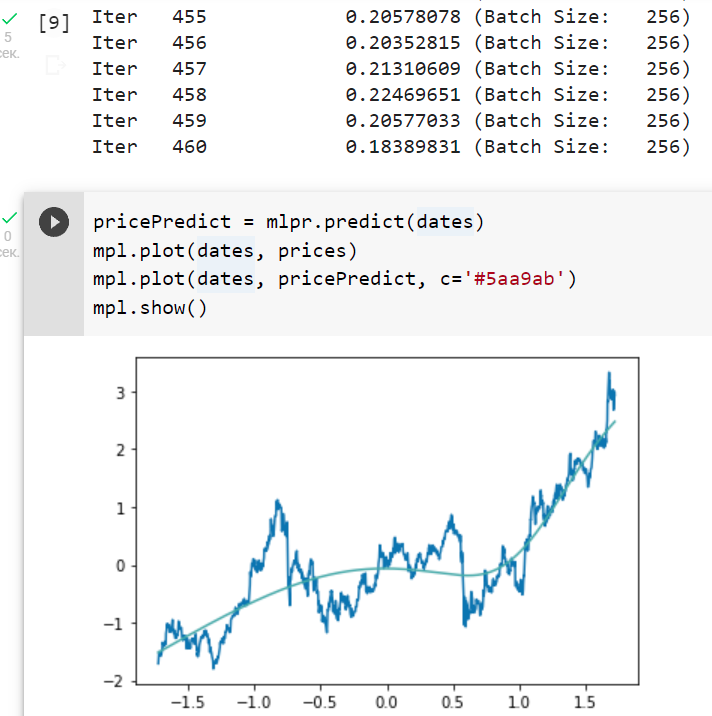


Рис.10 – Графический результат обучения модели

Получилось неплохо, но можно улучшить данную модель.

Существуют несколько способов улучшения модели. Для начала можно уменьшить терпимость к ошибкам.

Первое испытание, допуск ошибки был установлен как .2; тем не менее, нужно уменьшить его, например, до .1

Также обновим имя переменных, чтобы значения, которые уже были созданы – не изменились.

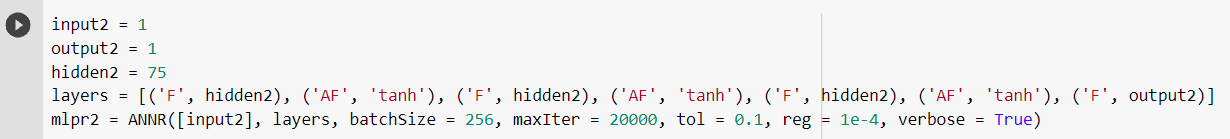


Рис.11 – Объявление слоев

Запустим модель еще раз с помощью следующих команд, и получим новые результаты:

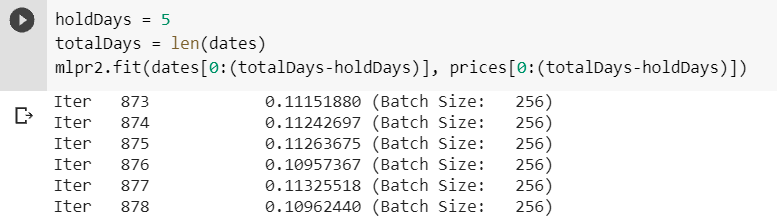


Рис.12 – Обучение

После завершения обучения мы можем выполнить следующие команды, чтобы увидеть результат.

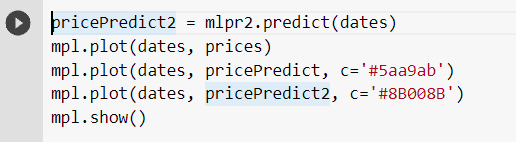


Рис.13 – Построение графика

Получим следующий график:

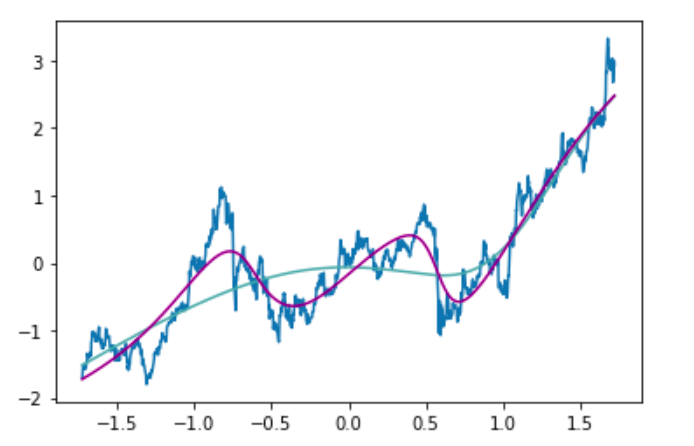


Рис.14 – Графический результат обучения модели с измененными параметрами

Выглядит намного лучше.

Можно попробовать установить ошибку на действительно маленькое число, но тогда максимальное количество итераций, которое используется, остановит выполнение до того, как оно достигнет желаемого уровня ошибки.

Увеличение максимального количества итераций не поможет в данной ситуации, так как проблема заключается в заданных параметрах модели.

Сама модель имеет ограничения, самая низкая достижимая ошибка для модели, которая была построена ограничена.

В этой ситуации не имеет значения, сколько еще итераций добавить, структура модели не даст лучших результатов, независимо от того, сколько итераций выполнено.

Модели имеют «гиперпараметры». Это параметры, которые управляют моделью, они определяют, как создается модель. Их изменение может дать лучшие (или, возможно, худшие) результаты.

Гиперпараметры, которые можно изменить для улучшения модели — это количество нейронов в каждом скрытом слое, количество скрытых слоев, функция активации и т. Д.

Попытаемся настроить эти гиперпараметры, чтобы добиться более низкой устойчивости к ошибкам, чем это было возможно ранее.

Самый простой способ сделать это - увеличить количество нейронов в скрытых слоях

Увеличим количество нейронов в каждом скрытом слое с 75 до 100 и установим допуск на .075:

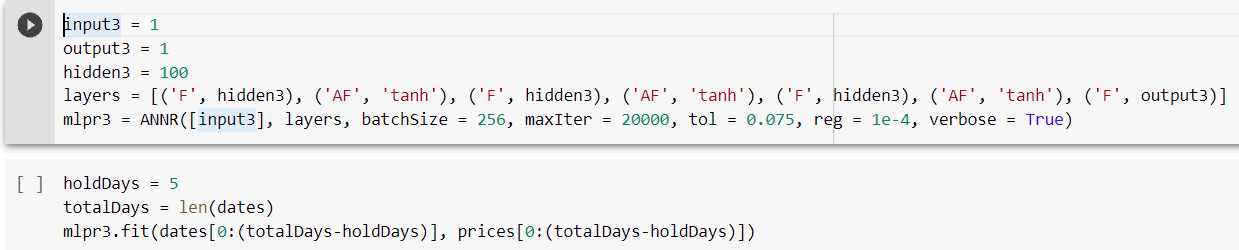


Рис.15 – Объявление слоев и обучение модели

Получим следующий график:

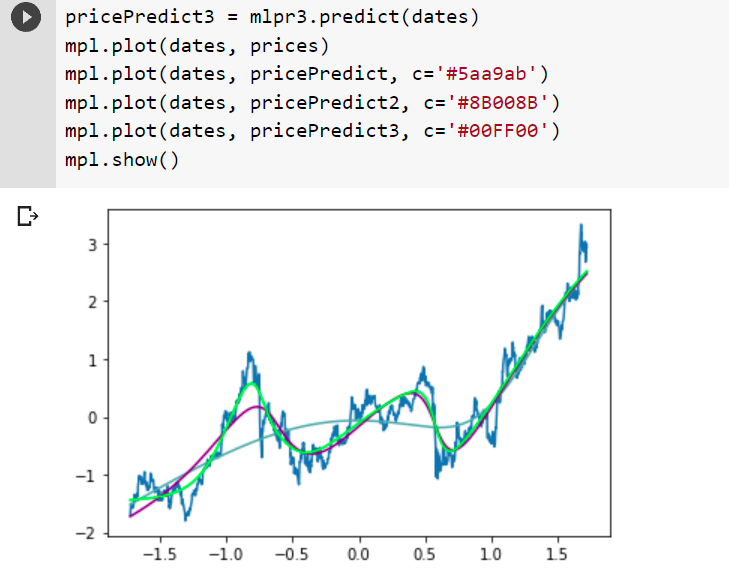
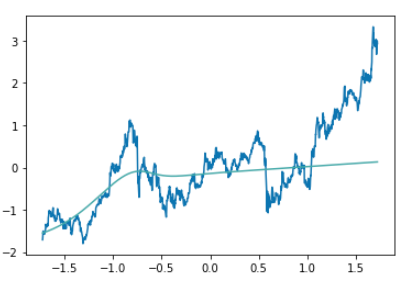
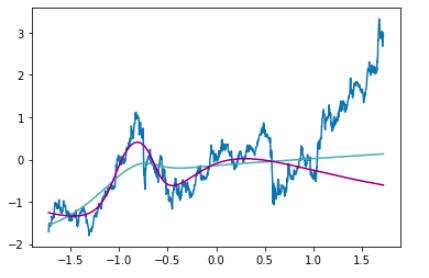


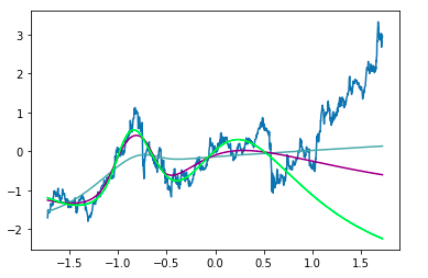
Рис.16 – Итоговая модель

Теперь давайте попробуем предсказать поведение цен. Обучающая выборка – 900 элементов из 1258.

Покажем поведение трех графиков:







Можно заметить, что созданная нейронная сеть предсказала спад цены, это отображает зеленая линия (нейронная сеть с подобранными наилучшими параметрами)

**Прогнозирование цены акций СБЕРБАНКА с помощью полиномиальной регрессии в Python**

Загружаем данные аналогичном образом, как было сделано ранее. И напишем код для создания полиномиальной регрессии с степенью функции 2.



Рис.17 – Код для создания полиномиальной регрессии

Полученный график имеет следующий вид:

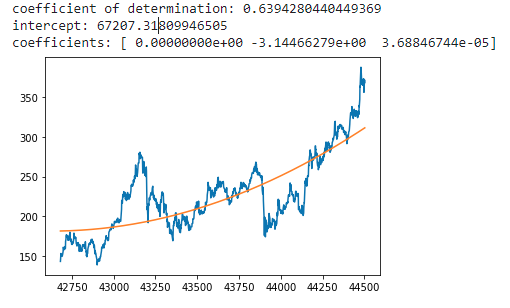


Рис.18 – График полиномиальной регрессии

Можно заметить, что коэффициент детерминации довольно маленький и сам по себе график не очень хорошо описывает данные.

Построим график с степенью 3:

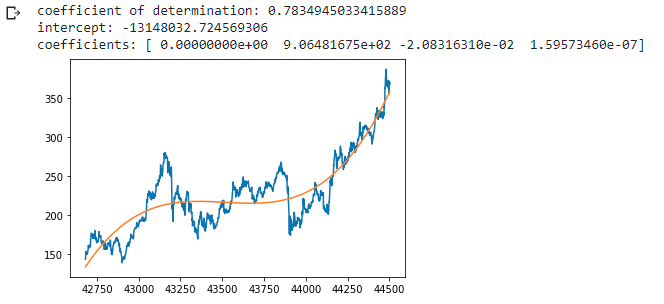


Рис.19 – График полиномиальной регрессии с степенью 3

Коэффициент детерминации увеличился, но прежнему график уступает по точности нейронной сети. Было выяснено, что при увеличении степени функции, точность модели не увеличивается.

Вывод

В ходе данной работы был написан код для прогнозирования цен акций компании Сбербанка двумя способами: нейронная сеть и полиномиальная регрессия.