**Valute\_Prediction models**

Навигация

[**Информация по работе, начало работы** 2](#_Toc179461606)

[**Описание моделей** 2](#_Toc179461607)

[*Линейная параметрическая авторегрессия* 2](#_Toc179461608)

[*Нелинейная параметрическая авторегрессия* 2](#_Toc179461609)

[*Нейронная сеть* 2](#_Toc179461610)

# **Информация по работе, начало работы**

**Модель представляет собой рабочий файл. Внутри которого находится три основных файла .py:**

* *MAIN – рабочий файл по вызову моделей. Содержит в себе примеры вызова моделей и описания в виде комментариев для лучшей навигации.*
* *Get\_data – файл, выполняющий функцию поставщика данных. Использует API московской биржи, чтобы получить и сохранить данные в виде dollar\_data / euro\_data. За период, указанный в этих функциях. Поэтому рекомендуется однократный запуск файла Get\_data (с корректировкой конечной даты), в случае необходимости получения более актуальных данных. ТАКЖЕ ТАМ НАХОДЯТСЯ ФУНКЦИИ ПО Z-НОРМАЛИЗАЦИИ*
* *Models.py – файл, содержащий в себе классы моделей по аналитике. Содержит в себе три модели: модель линейной регрессии с циклом, модель нелинейной регрессии с циклом, модель регрессии на основе нейронной сети*

# **Описание моделей**

## *Линейная параметрическая авторегрессия*

Линейная параметрическая авторегрессия представляет собой простейшую авторегрессию, заключенную в цикл, где на каждом шагу высчитывается MSE и изменяется количество лагов. Таким образом, очень быстро можно получить оптимальную модель без большого количества тестов по оценке временного ряда.

Параметры модели:

 **data**:

Входной временной ряд, представленный в виде вектор-столбца. Это основная информация, на основе которой модель будет обучаться и делать предсказания.

 **train\_size**:

Опциональный параметр, определяющий долю данных, которые будут использоваться для тренировки модели. По умолчанию 90% данных идут на тренировку, а оставшиеся 10% — на тестирование.

 **max\_iters**:

Максимальное количество итераций для поиска оптимального числа лагов (авторегрессивных шагов). По умолчанию — 50. Этот параметр задает, сколько раз модель будет пытаться найти оптимальные лаги до остановки.

 **mse\_threshold**:

Порог минимального значения среднеквадратичной ошибки (MSE), при котором модель останавливает дальнейшие попытки улучшить результаты. По умолчанию установлен на 1e-4. Если достигнута эта ошибка, поиск прекращается.

 **no\_improvement\_threshold**:

Порог, указывающий, сколько раз подряд модель может не улучшать MSE, прежде чем прекратить попытки улучшения. Если число итераций без улучшений превышает это значение (по умолчанию 50), процесс поиска лагов останавливается.

## *Нелинейная параметрическая авторегрессия*

Полиномиальная регрессия. Является базовой обучающейся моделью, которая в отличие от линейной параметрической авторегрессии, считается более качественной, в случае описывания нелинейных зависимостей. Также имеет цикл внутри, который изменяет размер полинома и высчитывает MSE. Что также помогает получить оптимальную модель для данного временного ряда без проведения серьезных, долгих тестов

Параметры модели:

 **data**:

* Входной временной ряд, представленный как вектор-столбец. Модель использует эти данные для обучения и предсказаний.

 **max\_degree**:

* Максимальная степень полинома для подбора модели. По умолчанию установлена на 20. Чем выше степень полинома, тем сложнее модель, которая может лучше подстраиваться под данные, но при этом риск переобучения возрастает.

 **train\_size**:

* Опциональный параметр, определяющий долю данных, которая будет использоваться для обучения модели. По умолчанию 90% данных используется для тренировки.

 **max\_iters**:

* Максимальное количество итераций для поиска оптимальной степени полинома. По умолчанию — 50. Этот параметр задает, сколько раз модель может увеличивать степень полинома в процессе поиска.

 **mse\_threshold**:

* Порог для минимального значения среднеквадратичной ошибки (MSE). Если модель достигает этого значения, поиск степени полинома останавливается. По умолчанию установлено на 1e-4.

 **no\_improvement\_threshold**:

* Порог количества итераций без улучшений. Если MSE не улучшается на протяжении заданного количества шагов (по умолчанию 20), поиск степени останавливается.

## *Нейронная сеть*

Нейронная сеть представляет собой пример рекурентной нейронной сети (RNN), подвида LSTM. Эта нейронная сеть обучается на основе определенного количества данных и затем выдает predict, который и является основной для MSE.

Однако по причине сложности модели, рекомендуется использовать с дополнительными коррелирующими данными, которые бы были интерпритируемы в рамках данной модели

Параметры модели:

1. **data (основной временной ряд)**:
   * Тип: numpy.ndarray
   * Описание: Это основной временной ряд, на основании которого модель будет обучаться. Он представляет собой одномерный массив значений (например, котировки валюты по времени).
   * Пример: временной ряд цен акций, где каждое значение — это цена в конкретный момент времени.
2. **additional\_data (дополнительные временные ряды)** (опциональный параметр):
   * Тип: numpy.ndarray, либо None
   * Описание: Дополнительные временные ряды (например, коррелированные данные), которые могут улучшить модель. Это может быть набор признаков, влияющих на основной временной ряд (например, объём торгов, другие валютные пары). Если не передан, используется только основной временной ряд.
   * Пример: если основной ряд — это курс доллара, то дополнительный может быть курс евро или объём торгов.
3. **sequence\_length**:
   * Тип: int
   * Описание: Количество значений, используемых для формирования одного входа (последовательности) для модели. Это то, сколько предыдущих значений модель будет использовать для предсказания следующего значения.
   * Пример: если sequence\_length = 30, то модель будет использовать последние 30 значений временного ряда для предсказания следующего значения.
4. **train\_size**:
   * Тип: float
   * Описание: Доля данных, которые будут использованы для обучения модели. Остальные данные будут отведены на тестирование. Если train\_size = 0.8, то 80% данных пойдут на обучение, а 20% — на тестирование.
   * Пример: train\_size=0.8 означает, что из 1000 наблюдений 800 пойдут на тренировку, а 200 — на тест.
5. **units**:
   * Тип: int
   * Описание: Количество нейронов в каждом слое LSTM. Этот параметр задаёт размерность скрытых состояний для каждого слоя LSTM.
   * Пример: units=50 означает, что каждый слой LSTM будет содержать 50 нейронов.
6. **dropout\_rate**:
   * Тип: float
   * Описание: Вероятность зануления выхода нейронов в слое Dropout для предотвращения переобучения. Значение должно быть от 0 до 1.
   * Пример: dropout\_rate=0.2 означает, что 20% нейронов будут случайно "выключаться" во время обучения.
7. **epochs**:
   * Тип: int
   * Описание: Количество полных проходов по всему тренировочному набору данных во время обучения модели.
   * Пример: epochs=100 означает, что модель будет обновляться 100 раз для всего набора данных.
8. **batch\_size**:
   * Тип: int
   * Описание: Количество примеров, которые модель будет обрабатывать за один шаг перед обновлением весов. Чем больше batch\_size, тем быстрее обучение, но тем больше требуется памяти.
   * Пример: batch\_size=32 означает, что модель будет обновляться после обработки 32 примеров.
9. **steps**:
   * Тип: int
   * Описание: Количество шагов предсказания. Это указывает, сколько будущих значений будет предсказано моделью после обучения.
   * Пример: steps=3 означает, что модель предскажет 3 будущих значения на основе последних известных данных.