



УНИВЕРСИТЕТ
искусственного
интеллекта

Фреймворк **TERRA_AI**

Документация





Фреймворк TERRA_AI

Документация

1. Установка фреймворка

Для работы с фреймворком необходимо его установить:

```
!pip -q install terra_ai
```

Для корректной работы функций фреймворка необходимо загрузить модуль `терра_ии`:

```
from terra_ai import терра_ии
```

2. Загрузка датасетов

Загрузить датасет можно с помощью функции `загрузить_базу`. Общий вид функции:

```
терра_ии.загрузить_базу(база, справка)
```

Параметры функции:

- **база** – обязательный параметр. Содержит название датасета, который необходимо скачать. Может принимать следующие значения:



Загрузка датасетов

- АВТО-2 - база изображений автомобилей двух марок: Феррари, Мерседес
- АВТО-3 - база изображений автомобилей двух марок: Феррари, Мерседес, Рено
- САМОЛЕТЫ - база изображений самолетов и их сегментированных изображений
- СИМПТОМЫ ЗАБОЛЕВАНИЙ – текстовая база описаний симптомов заболеваний (10 категорий: 'Аппендицит', 'Гастрит', 'Гепатит', 'Дуоденит', 'Колит', 'Панкреатит', 'Холецистит', 'Эзофагит', 'Энтерит', 'Язва')
- КВАРТИРЫ – текстовая база для предсказания стоимости квартир по их параметрам
- ТРЕЙДИНГ - база прогнозирования стоимости акций трех вариантов: полиметаллы, Газпром и Яндекс
- УМНЫЙ ДОМ - база аудио-файлов для распознавания голосовых команд
- ГУБЫ – база изображений для сегментации губ
- МОЛОЧНАЯ ПРОДУКЦИЯ – база изображений бутылок молока трёх видов: 'Parmalat', 'Кубанский_молочник', 'Кубанская_буренка'
- АВТОБУСЫ – база изображений пассажиров, входящих в автобус и выходящих из него
- ОБНАРУЖЕНИЕ ВОЗГОРАНИЙ – база изображений со следующими категориями: 'Есть огонь', 'Нет огня'
- ТРЕКЕР – база изображений для трекинга пассажиров
- МАЙОНЕЗ – база изображений майонез трех видов: 'Ряба', 'Махеев', 'ЕЖК'
- TESLA – текстовая база отзывов об автомобиле Тесла
- ВАКАНСИИ – текстовая база вакансий для подбора кандидата на должность менеджера по продажам

Загрузка датасетов

- **справка** – необязательный параметр. По умолчанию справочная информация не выводится. Если указать значение «Показать», то после загрузки будет выведена справочная информация о скачанном датасете.

Функция возвращает: None

Например:

```
терра_ии.загрузить_базу('АВТО-2')  
терра_ии.загрузить_базу('АВТО-3', справка = 'Показать')  
подбора кандидата на должность менеджера по продажам
```

3. Визуализация примеров обучающих данных

Для визуализации примеров при работе с базами для распознавания изображений используется функция **показать_примеры**:

```
терра_ии.показать_примеры(путь)
```

Параметры:

- **путь** – путь к базе

Функция возвращает: None

Например:

```
терра_ии.показать_примеры(путь='/автомобили' )
```

Для распознавания изображений могут быть использованы следующие базы: АВТО-2 (путь = «/автомобили»), АВТО-3 (путь = «/автомобили»), МОЛОЧНАЯ ПРОДУКЦИЯ (путь = «/Молочная_продукция»), АВТОБУСЫ (путь = «/Автобусы»), ОБНАРУЖЕНИЕ ВОЗГОРАНИЙ (путь = «/Обнаружение_возгораний»), МАЙОНЕЗ (путь = «/Майонез»).

Для визуализации примеров при работе с базами для сегментации изображений используется функция **показать_примеры**:

```
терра_ии.показать_примеры(параметры)
```

Например:

```
терра_ии.показать_примеры(оригиналы = изображения,  
сегментированные_изображения  
=сегментированные_изображения)
```

Визуализация примеров обучающих данных

Параметры:

- **оригиналы** – изображения, используемые для сегментации
- **сегментированные_изображения** – маски, используемые для сегментации

Функция возвращает: None

Для сегментации изображений могут быть использованы следующие базы: САМОЛЕТЫ, ГУБЫ.

Для визуализации примеров при работе с классификацией текстовых данных используется функция **показать_примеры**:

```
терра_ии.показать_примеры(база)
```

Например:

```
терра_ии.показать_примеры(база = 'симптомы')
```

Параметры:

- база – название базы

Функция возвращает: None

Для классификации текстовых данных могут быть использованы следующие базы: СИМПТОМЫ ЗАБОЛЕВАНИЙ (база = «симптомы»), ВАКАНСИИ (база = «Вакансии»), TESLA (база = «TESLA»).

Для вывода примеров при работе с аудио-файлами используется функция **показать_примеры_голосовых_команд**:

```
терра_ии.показать_примеры_голосовых_команд()
```

Параметры: None

Функция возвращает: None

Для вывода примеров аудио-файлов могут быть использованы следующие базы: УМНЫЙ ДОМ

. Для вывода примеров при работе с данными для трейдинга используются функции показать_примеры_для_трейдинга, **показать_примеры_для_интервала**:

Визуализация примеров обучающих данных

- Для визуализации данных

```
терра_ии.показать_примеры_для_трейдинга (данные)
```

Параметры:

- **данные** – табличные данные, содержащие информацию об изменении акций

Функция возвращает: None

- Для визуализации данных на заданном интервале

```
терра_ии.показать_примеры_для_интервала (данные, а, б)
```

Параметры:

- **данные** – табличные данные, содержащие информацию об изменении акций
- **а** – начало интервала
- **б** – конец интервала

Функция возвращает: None

Для вывода примеров при работе с данными для трейдинга могут быть использованы следующие базы: ТРЕЙДИНГ.

Для вывода примеров информации о квартирах используется функция **показать_пример_квартиры**:

```
терра_ии.показать_пример_квартиры (количество)
```

Например:

```
терра_ии.показать_пример_квартиры (количество = 3)
```

Параметры:

- **количество** – количество выводимых примеров

Функция возвращает: None

Для вывода примеров информации о квартирах могут быть использованы следующие базы: КВАРТИРЫ.

Визуализация примеров обучающих данных

Для вывода примеров изображений для трекинга пассажиров используется функция **показать_примеры**:

```
терра_ии.показать_примеры (параметры)
```

Например:

```
терра_ии.показать_примеры(база = 'Трекер',  
тип_изображений = 'Один человек')
```

Параметры:

- **база** – название базы
- **тип_изображений** – параметр, отвечающий за вывод одного и того же человека и различных людей (принимает значения: 'Один человек', 'Разные люди')

Функция возвращает: None

Для вывода примеров изображений для трекинга пассажиров могут быть использованы следующие базы: ТРЕКИНГ.

4. Предобработка данных и создание выборок для обучения нейронной сети

Создание обучающих и тестовых наборов для тренировки нейронной сети

Для создания выборок при работе с базами для распознавания изображений используется функция создать выборки:

```
терра_ии.создать_выборки (параметры)
```

Например:

```
(обучающая_выборка, метки_обучающей_выборки),  
(тестовая_выборка, метки_тестовой_выборки) =  
терра_ии.создать_выборки(путь='Молочная_продукция',  
размер = (96, 53))
```

Предобработка данных и создание выборок

Параметры:

- путь – путь к базе
- размер – размер изображений

Функция возвращает:

- (**обучающая_выборка, метки_обучающей_выборки**) – обучающая выборка, состоящая из изображений заданного размера и меток деления изображений на классы
- (**тестовая_выборка, метки_тестовой_выборки**) – тестовая выборка, состоящая из изображений заданного размера и меток деления изображений на классы

Для создания выборок при работе с базами для сегментации изображений используется функция **показать_примеры**:

terra_ии.показать_примеры (параметры)

Например:

```
(обучающая_выборка, размеченная_обучающая_выборка),  
(тестовая_выборка, размеченная_тестовая_выборка) =  
терра_ии.показать_примеры(оригиналы = изображения,  
сегментированные_изображения  
=сегментированные_изображения)
```

Параметры:

- **оригиналы** – изображения, используемые для сегментации
- **сегментированные_изображения** – маски, используемые для сегментации

Функция возвращает:

- (**обучающая_выборка, размеченная_обучающая_выборка**) – обучающая выборка, состоящая из изображений заданного размера и изображений того же размера, содержащих маски.
- (**тестовая_выборка, размеченная_тестовая_выборка**) – тестовая выборка, состоящая из изображений заданного размера и изображений того же размера, содержащих маски.

Предобработка данных и создание выборок

Для создания выборок оценки отзывов об автомобиле TESLA используется функция **создать_текстовые_выборки**:

```
терра_ии.создать_текстовые_выборки(параметры)
```

Например:

```
# Задаем параметры для формирования выборок
количество_анализируемых_слов = '10000'
размер_окна = '50'
шаг = '10'
# Создаем выборки для обучения нейронной сети
(обучающая_выборка, метки_обучающей_выборки),
(тестовая_выборка, метки_тестовой_выборки) =
терра_ии.создать_текстовые_выборки(
    количество_анализируемых_слов,
    размер_окна,
    шаг,
    путь_к_базе = '/content/Отзывы/')
```

Параметры:

- **количество_анализируемых_слов** – длина отзыва для анализа
- **размер_окна** - составляющие части фразы фиксированной длины
- **шаг** – число, показывающее на сколько слов происходит смещение при формировании выборок
- **путь_к_базе** – путь к базе

Функция возвращает:

(обучающая_выборка, метки_обучающей_выборки) – обучающая выборка, состоящая из текстовых данных, поделенных на положительные и отрицательные отзывы

(тестовая_выборка, метки_тестовой_выборки) – тестовая выборка, состоящая из текстовых данных, поделенных на положительные и отрицательные отзывы

Для создания выборок подбора кандидатов на вакансию используется функция **создать_выборки_вакансии**:

```
терра_ии.создать_выборки_вакансии()
```

Предобработка данных и создание выборок

Например:

```
(обучающая_выборка, метки_обучающей_выборки),  
(тестовая_выборка, метки_тестовой_выборки) =  
терра_ии.создать_выборки_вакансии()
```

Параметры: None

Функция возвращает:

- **(обучающая_выборка, метки_обучающей_выборки)** – обучающая выборка, состоящая из текстовых данных, содержащих информацию о вакансиях и соответствии им кандидатов
- **(тестовая_выборка, метки_тестовой_выборки)** – тестовая выборка, состоящая из текстовых данных, содержащих информацию о вакансиях и соответствии им кандидатов

Для создания выборок для определения симптомов заболеваний используется функция **создать_текстовые_выборки**:

```
терра_ии.создать_текстовые_выборки(параметры)
```

Например:

```
# Задаем параметры для формирования выборок  
количество_анализируемых_слов = '10000'  
размер_окна = '50'  
шаг = '10'  
# Создаем выборки для обучения нейронной сети  
(обучающая_выборка, метки_обучающей_выборки),  
(тестовая_выборка, метки_тестовой_выборки) =  
терра_ии.создать_текстовые_выборки(  
    количество_анализируемых_слов,  
    размер_окна,  
    шаг)
```

Параметры:

- **количество_анализируемых_слов** – часть текста для определения симптомов
- **размер_окна** - составляющие части текста фиксированной длины
- **шаг** – число, показывающее на сколько слов происходит смещение при формировании выборок

Предобработка данных и создание выборок

Функция возвращает:

- **(обучающая_выборка, метки_обучающей_выборки)** – обучающая выборка, состоящая из текстовых данных, классифицированных по заболеваниям
- **(тестовая_выборка, метки_тестовой_выборки)** – тестовая выборка, состоящая из текстовых данных, классифицированных по заболеваниям

Для вывода примеров при работе с аудио-файлами используется функция **создать_выборки_голосовых_команд**:

```
терра_ии.создать_выборки_голосовых_команд(параметры)
```

Например:

```
длина = 0.5
шаг = 0.04
(обучающая_выборка, метки_обучающей_выборки),
(тестовая_выборка, метки_тестовой_выборки) =
терра_ии.создать_выборки_голосовых_команд(длина, шаг)
```

Параметры:

- **длина** – длина аудио-отрезка при формировании выборки
- **шаг** – шаг при формировании выборки

Функция возвращает:

- **(обучающая_выборка, метки_обучающей_выборки)** – обучающая выборка, состоящая из аудио-файлов, классифицированных по командам для умного дома
- **(тестовая_выборка, метки_тестовой_выборки)** – тестовая выборка, состоящая из аудио-файлов, классифицированных по командам для умного дома

Для вывода примеров при работе с данными для трекинга пассажиров используется функция **создать_выборки_трекер**:

```
терра_ии.создать_выборки_трекер(параметры)
```

Предобработка данных и создание выборок

Например:

```
(обучающая_выборка, метки_обучающей_выборки),  
(тестовая_выборка, метки_тестовой_выборки) =  
терра_ии.создать_выборки_трекер(путь='/Трекер',  
размер = (64, 64))
```

Параметры:

- **путь** – путь к базе
- **размер** – размер изображений

Функция возвращает:

- **(обучающая_выборка, метки_обучающей_выборки)** – изображения заданного размера, классифицированные по идентичности людей
- **(тестовая_выборка, метки_тестовой_выборки)** – изображения заданного размера, классифицированные по идентичности людей

Для вывода примеров информации о квартирах используется функция **создать_выборки_квартир**:

```
терра_ии.создать_выборки_квартир()
```

Например:

```
(обучающая_выборка, метки_обучающей_выборки),  
(тестовая_выборка, метки_тестовой_выборки), инструменты =  
терра_ии.создать_выборки_квартир()
```

Параметры: None

Функция возвращает:

- **(обучающая_выборка, метки_обучающей_выборки)** – текстовые данные, содержащие информацию о квартире и её стоимости
- **(тестовая_выборка, метки_тестовой_выборки)** – текстовые данные, содержащие информацию о квартире и её стоимости
- **инструменты** – нормированные данные

Для вывода примеров изображений для трекинга пассажиров используется функция **показать_примеры**:

```
терра_ии.показать_примеры(параметры)
```

Предобработка данных и создание выборок

Например:

```
терра_ии.показать_примеры(база = 'Трекер',  
                           тип_изображений = 'Один человек')
```

Параметры:

- **база** – название базы
- **тип_изображений** – параметр, отвечающий за вывод одного и того же человека и различных людей (принимает значения: 'Один человек', 'Разные люди')

Функция возвращает: None

Для вывода примеров изображений для трекинга пассажиров могут быть использованы следующие базы: ТРЕКИНГ.

5. Задачи

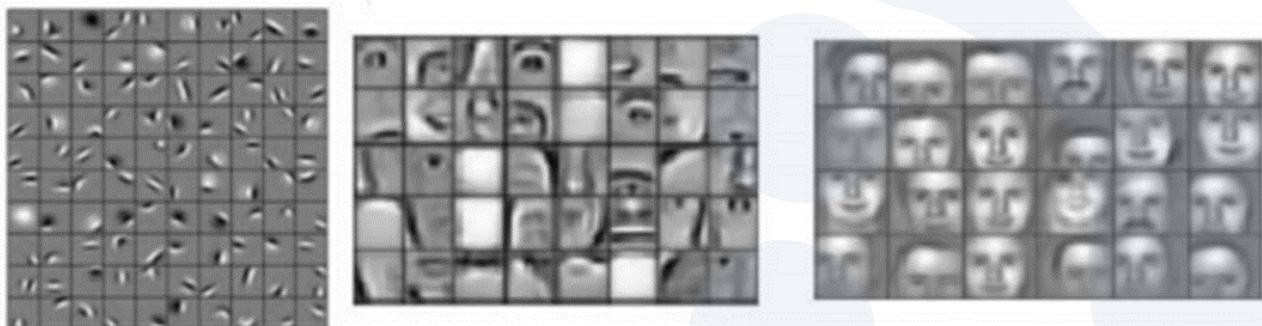
Задача классификации

Задача классификации изображений состоит из следующих шагов:

- Генерация значимых признаков изображения.
- Классификация изображения на основе его признаков.

Сверточные сети выявляют в поступающих картинках признаки, которыми характеризуется рисунок. Это может быть линия, точки, область или несколько пикселей. Что дает возможность изучить рисунок как бы под лупой, причем несколько раз: по количеству нейронов в сверточном слое. Это дает возможность классифицировать рисунки по классам.

Вот что происходит внутри сверточной сети:



Задачи

Первый слой ищет геометрические фигуры, далее следует другой слой, который уменьшает размерность рисунка. Второй слой из фигур предыдущего ищет отдельные элементы лица, а потом опять уменьшение размерности. Третий слой уже из элементов лиц составляет лицо.

Частным случаем задачи классификации изображений является трекинг объектов.

Трекинг объектов в реальном времени является достаточно сложной задачей. Трекинг с высокой точностью позволит решать следующие задачи:

- контроль доступа и движения пассажиров
- отслеживание оставленного багажа
- обнаружение конфликтов и т.д.

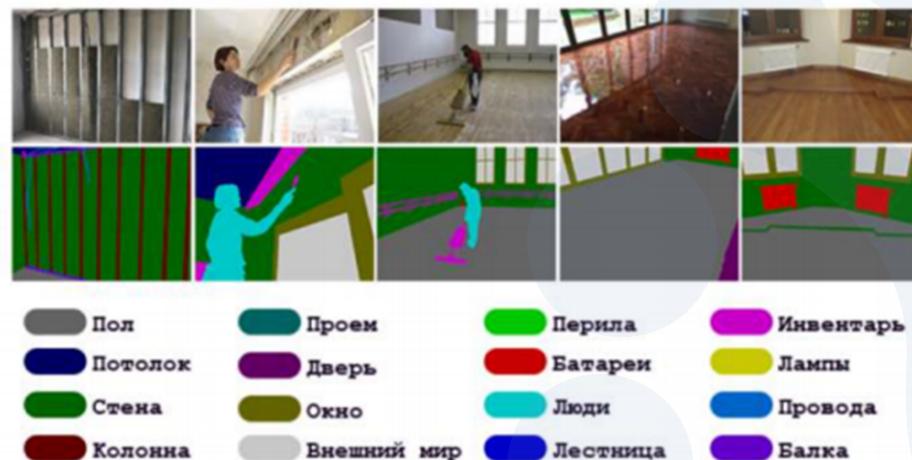
Для задачи трекинга пассажиров могут быть использованы следующие базы: ТРЕКЕР.

Для задачи классификации могут быть использованы следующие базы: АВТО-2, АВТО-3, МОЛОЧНАЯ ПРОДУКЦИЯ, АВТОБУСЫ, ОБНАРУЖЕНИЕ ВОЗГОРАНИЙ, МАЙОНЕЗ.

Обозначение во фреймворке: 'классификация изображений'.

Задача сегментации изображений

Сегментация – это выделение объектов в исходных данных. Это выражается в подсвечивании объектов на изображении. Другими словами можно сказать, что сегментация изображений – это попиксельная классификация, где каждому пикселю объекта присваивается определенный класс:



Задачи

Изображения стройки сегментированы на 16 классов: пол, потолок, стена, колонна, проем, дверь, окно, внешний мир, перила, батареи, люди, лестница, инвентарь, лампы, провода, балка. Каждому классу выделен свой цвет.

Сегментация применяется во многих областях: в медицине (первичная обработка цифровых снимков, например, рентгеновских), на производстве (определение дефектов детали по фотографии), в машиностроении (машинное зрение для автопилотов).

Для задачи сегментации могут быть использованы следующие базы: САМОЛЕТЫ, ГУБЫ.

Обозначение во фреймворке: 'сегментация изображений'.

Задача классификации текста

Задача классификации текста – отнести текст к одному или другому классу в зависимости от содержания текста.

Например:

- выяснить, кто написал этот отрывок произведения;
- отличить положительный комментарий к товару в интернет-магазине от отрицательного;
- распознать болезнь по симптоматике.

Основная проблема при работе с текстами заключается в невозможности передать необработанные текстовые данные на вход нейронной сети, так как нейронная сеть работает только с числовыми данными. Для того чтобы была возможность работать с текстами, необходимо перевести текст в числа, если быть точнее, в набор чисел.

Для задачи классификации текста могут быть использованы следующие базы: СИМПТОМЫ ЗАБОЛЕВАНИЙ, TESLA, ВАКАНСИИ.

Обозначение во фреймворке: для баз СИМПТОМЫ ЗАБОЛЕВАНИЙ и TESLA параметр задача отсутствует, для базы ВАКАНСИИ – 'классификация вакансий'.

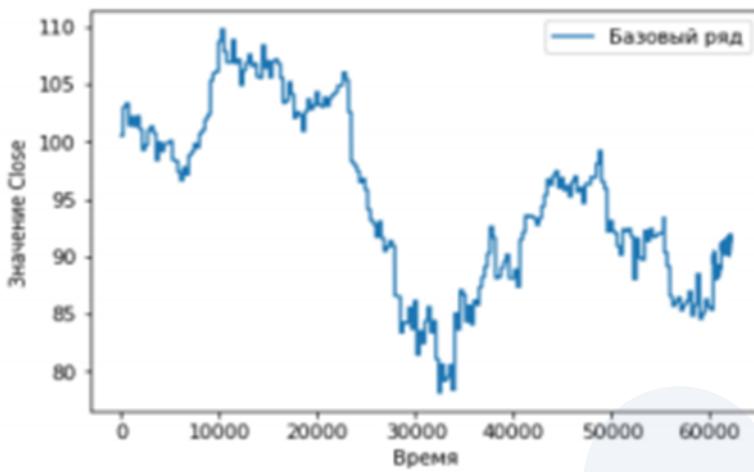
Задачи

Задача прогнозирования временных рядов

Временной ряд — собранный в разные моменты времени статистический материал о значении каких-либо параметров (в простейшем случае, одного) исследуемого процесса. Каждая единица статистического материала называется измерением или отсчётом, также допустимо называть её уровнем на указанный момент времени. Во временном ряде для каждого отсчёта должно быть указано время измерения или номер измерения по порядку. Временной ряд существенно отличается от простой выборки данных, так как при анализе учитывается взаимосвязь измерений и времени, а не только статистическое разнообразие и статистические характеристики выборки. Временными рядами являются:

- показания датчиков приборов и машин;
- статистические данные, собранные в разное время;
- финансовые данные;
- прогнозы погоды;
- аудио.

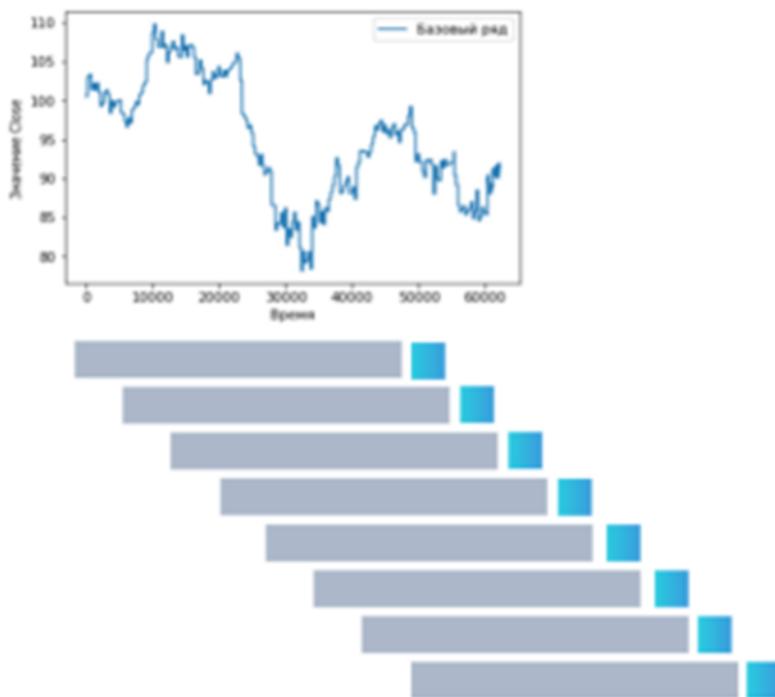
Пример временного ряда:



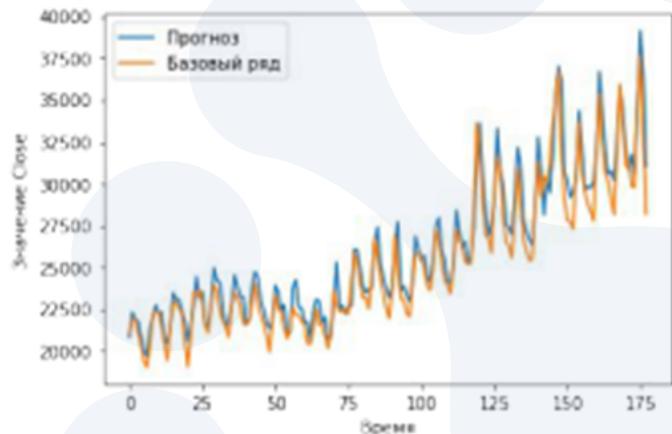
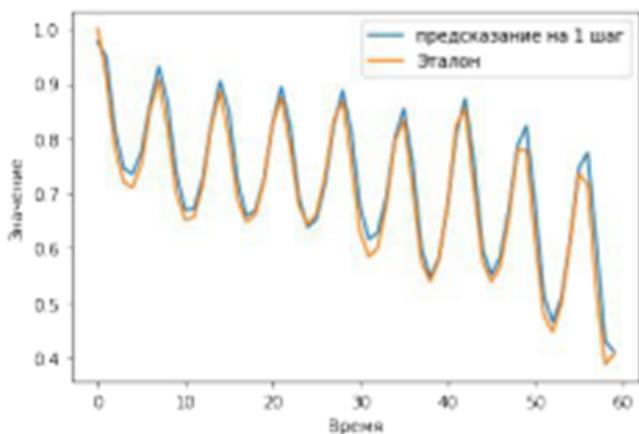
К созданию выборки для прогнозирования временных рядов применяется особый подход. Так как временной ряд на каждом своем отрезке уникален, то в качестве обучающей выборки берется отрезок фиксированной длины, а правильным ответом к нему – следующее за этим отрезком значение. Затем берется отрезок такой же длины, но сдвинутый на один шаг, и также в качестве правильного ответа – следующее за отрезком значение. В итоге получается довольно плотный набор данных, где правильные ответы к предыдущему шагу содержатся в выборке следующего шага.

Задачи

Графически это выглядит так:



Указанный выше подход по созданию выборок для обработки временных рядов с помощью нейронных сетей является причиной одного интересного эффекта. При прогнозировании временных рядов нейронные сети могут «хитрить» и выдавать в качестве прогноза текущего шага ответ из предыдущего шага. Данный эффект можно увидеть при расчете корреляции предсказанного ряда с реальным временным рядом. Проявится эффект в виде «горбика» в начале графика корреляции. Также при выводе на график предсказанного ряда и реального временного ряда будет отчетливо заметна полная идентичность графиков, но со сдвигом в один шаг.



Задачи

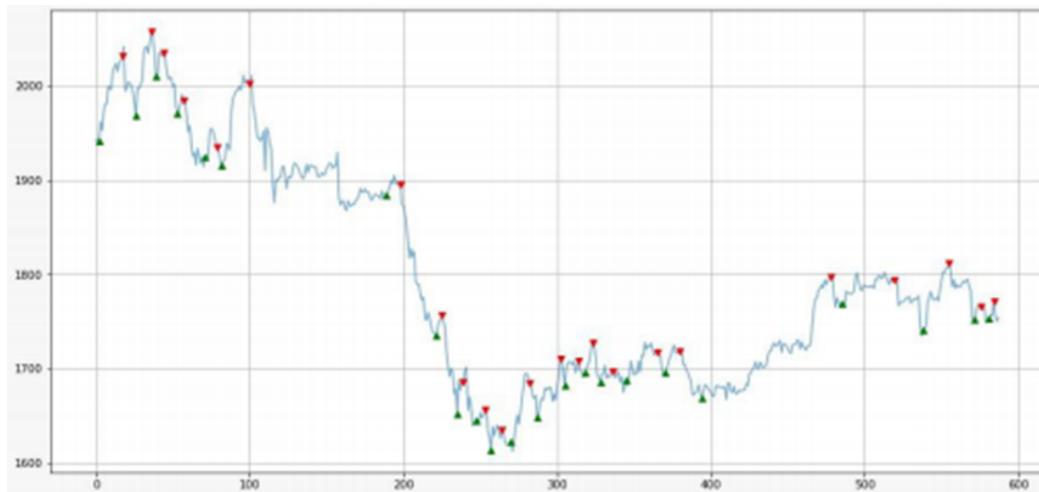
Задача трейдинга

Отдельным кластером задач, решаемых с помощью прогнозирования временных рядов, являются задачи трейдинга.

Вообще в трейдинге существует несколько способов анализа графика изменения цены:

- семантический анализ на основе новостей или публикаций в соцсетях;
- анализ на основе паттернов (фигур и свечей);
- анализ на основе предсказания тренда;
- анализ по техническим индикаторам.

В качестве примера выберем анализ на основе предсказания тренда.



Зелеными треугольниками показаны изменения тренда в сторону роста цены, красные треугольники показывают изменение тренда в сторону спада цены. На основе данного прогноза трейдер принимает решение о покупке или же продаже активов.

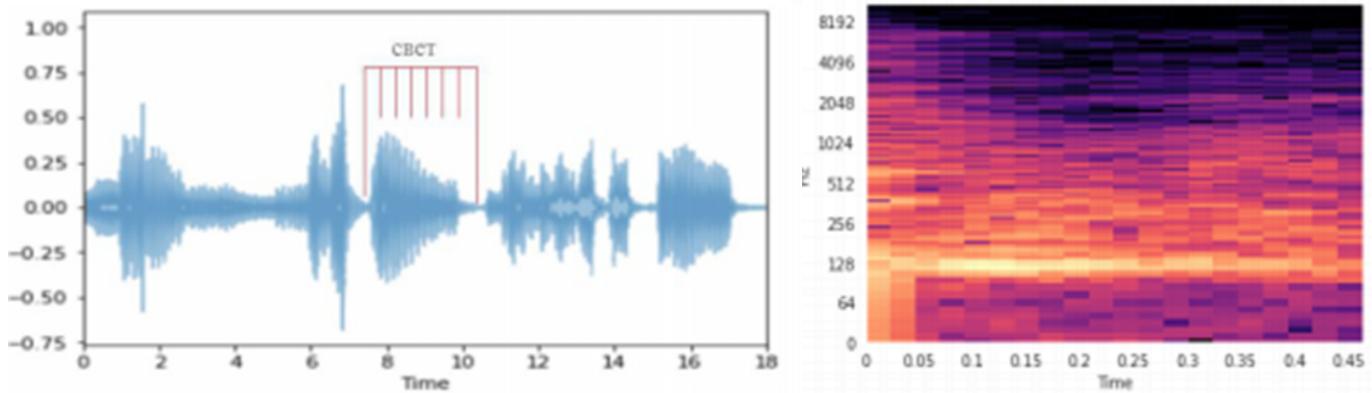
Для задачи трейдинга могут быть использованы следующие базы:
ТРЕЙДИНГ.

Обозначение во фреймворке: – 'временной ряд'.

Задача распознавания аудиоданных

Любые аудиоданные являются частным случаем временных рядов. Также аудиоданные могут быть представлены в виде амплитудно-частотного спектра и визуализированы в виде графической информации.

Задачи



Для задачи распознавания аудиоданных могут быть использованы следующие базы: УМНЫЙ ДОМ.

Обозначение во фреймворке: – 'аудио'.

Задача предсказания цены квартиры

Задача предсказания цены квартиры заключается в определении стоимости квартиры по внешним признакам: количество комнат, площадь квартиры, метро/ЖД станция, от станции, дом, балкон, санузел, примечание. Все эти данные оцифровываются и передаются в сеть, и на этом основании предсказывается цена. Чем больше таких параметров, тем выше точность предсказаний.

Для задачи предсказания цен на квартиры могут быть использованы следующие базы: КВАРТИРЫ.

Обозначение во фреймворке: – параметр задача отсутствует.

6. Слои

Названия слоёв необходимо писать русскими буквами, не латиницей. Ошибка в названии слоя приводит к ошибке при создании модели.

Также не все слои могут совмещаться между собой. Например, Выравнивающий нельзя ставить перед Сверточным, так как не подойдут размерности данных, а перед Объединением лучше закончить каждую ветвь сети Выравнивающим слоем, т. к. будет проще совместить размеры слоёв. Использование слишком большого числа МаксПуллинг слоёв способно привести к уменьшению картины признаков до нуля, что также вызовет ошибку.

Слои

Для создания полносвязных и составных сетей используются следующие слои:

Свёрточный слой.

В свёрточном слое указывается число ядер (нейронов, фильтров), размер ядра - количество связей нейронов или размер обрабатываемой области, приводится ли размер выходной матрицы к исходной, функция активации. Количество нейронов может быть любым положительным числом. Размер ядра может быть также любым, но не больше размера самой картинки (желательно не более 9, например, (2,2) или (3,3), или (3,4)).



Для последней доступной версии фреймворка после свёрточных слоёв можем указать количество нейронов (другими словами, фильтров), размер ядра, паддинг (коррекцию выходного размера слоя - same, valid), доступные функции активации:

```
слои = 'Свёрточный2D-32-(3, 3)-same-relu\
        Свёрточный2D-32-(3, 3)-same-relu\
        Свёрточный2D-32-(3, 3)-same-relu\
        Дропаут-0.2\
        Свёрточный2D-32-(3, 3)-same-relu\
        Выравнивающий\
        Полносвязный-16-relu\
        Полносвязный-2-softmax'
```

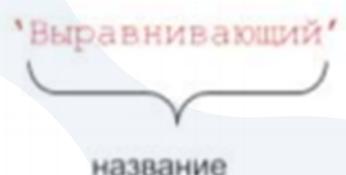
Полносвязный слой.

Количество нейронов может быть любым положительным числом.



Выравнивающий слой.

Слой предназначен для перехода от многомерных слоев (свёрточный) к одномерным (полносвязный).



Свёрточный слой после выравнивающего использовать нельзя!

Слой

Слой МаксПуллинг.

Сжимает картинку, оставляя максимальные значения в пикселях (размер сжатия зависит от размеров пулинга). Размер пулинга может быть любой, например, если использовать 'МаксПуллинг2D-(2,2)', то он уменьшит размер картинки в 2 раза (если картинка была 28x28 пикселей, то после этого слоя станет 14x14).



Слой Дропаут.

Дропаут - 0.4 - это слой, отключающий часть нейронов. 0.4 - это процент нейронов - 40%, которые будут отключены.



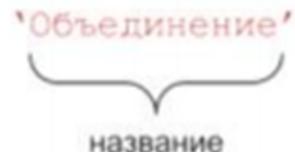
Слой пакетной нормализации.

Применение слоя пакетной нормализации может дать прирост точности сети.



Слой объединения.

Слой объединения нескольких предыдущих слоев.



Слой Эмбеддинг.

Слой вложения. Указываем размер плотного (dense) embedding-а - выходной размер, в нашем случае, 100, количество анализируемых слов - это размер словаря самых часто встречающихся в тексте слов, размер окна - размер скользящего окна при взятии данных для обучающей выборки.

Например:

```
'Эмбеддинг-100-' + количество_анализируемых_слов + '-' +  
размер_окна + '\
```

Создание нейронной сети

7. Создание нейронной сети

Для создания простой полносвязной нейронной сети используется функция создать сеть:

```
нейронка_4 = терра_ии.создать_сеть (слои,  
входной_размер, задача)
```

Параметры функции:

- **слои** - список слоев создаваемой нейронной сети (см. раздел «Слои»)
- **входной_размер** - размер входных данных (можно получить с помощью функции **терра_ии.получить_входной_размер()**:

```
входной_размер =  
терра_ии.получить_входной_размер (обучающая_выборка)
```

- **задача** - необязательный параметр (используется):

Например:

```
# создание слоев для нейронной сети  
слои = 'Сверточный2D-32-3\  
Сверточный2D-32-3\  
Сверточный2D-32-3\  
Выравнивающий\  
Полносвязный-16\  
Полносвязный-2'  
  
# получение входного размера данных  
входной_размер =  
терра_ии.получить_входной_размер (обучающая_выборка)  
# создание сети  
нейронка_1 = терра_ии.создать_сеть (  
    слои = слои,  
    входной_размер = входной_размер,  
    задача=' ' классификация изображений'  
)
```

Для создания составной нейронной сети используется функция **создать_составную_сеть**:

```
терра_ии.создать_составную_сеть (параметры)
```

Создание нейронной сети

Например:

```
ветвь1 = 'Эмбеддинг-100-' + количество_анализируемых_слов + '-' +  
размер_окна + ' Выравнивающий Полносвязный-200'  
ветвь2 = 'Полносвязный-4096 Полносвязный-256 Полносвязный-64'  
ветвь3 = 'Полносвязный-100'
```

```
нейронка = терра_ии.создать_составную_сеть(  
обучающая_выборка, метки_обучающей_выборки,  
ветвь1, ветвь2, ветвь3)
```

Для создания нейронной сети для оценки стоимости квартир используется функция **создать_составную_сеть_квартиры**:

```
терра_ии.создать_составную_сеть_квартиры(параметры)
```

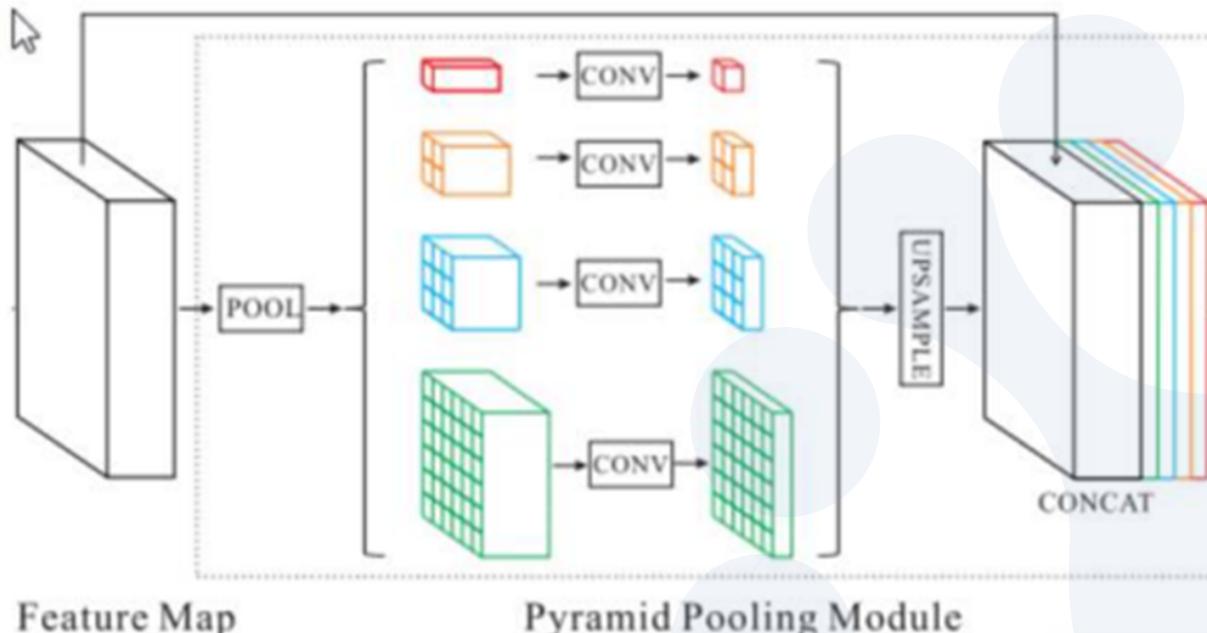
Например:

```
ветвь1 = 'Полносвязный-100-relu Полносвязный-50-linear'  
ветвь2 = 'Полносвязный-500-linear'  
финальная_часть = 'Полносвязный-1000-relu Полносвязный-500-relu  
Полносвязный-100-relu Полносвязный-1-linear'  
нейронка = терра_ии.создать_составную_сеть_квартиры(  
обучающая_выборка, ветвь1, ветвь2, финальная_часть)
```

Для создания сети архитектуры PSP используется функция `создать_PSP`:

```
терра_ии.создать_PSP(параметры)
```

Архитектура PSP-модели:



Создание нейронной сети

Например:

```
# Указываем слои создаваемой модели
стартовый_блок = 'Сврточный2D-32-3\
Сврточный2D-32-3'

блок_PSP = 'Сврточный2D-32-3'

финальный_блок = 'Сврточный2D-32-3\
Сврточный2D-32-3\
Сврточный2D-2-3'

# Получаем входной размер данных
входной_размер =
терра_ии.получить_входной_размер(обучающая_выборка)

# Создаем модель
нейронка_5 = терра_ии.создать_PSP(
    стартовый_блок = стартовый_блок,
    блок_PSP = блок_PSP,
    финальный_блок = финальный_блок,
    входной_размер = входной_размер,
    количество_блоков = 2
)
```

Параметры:

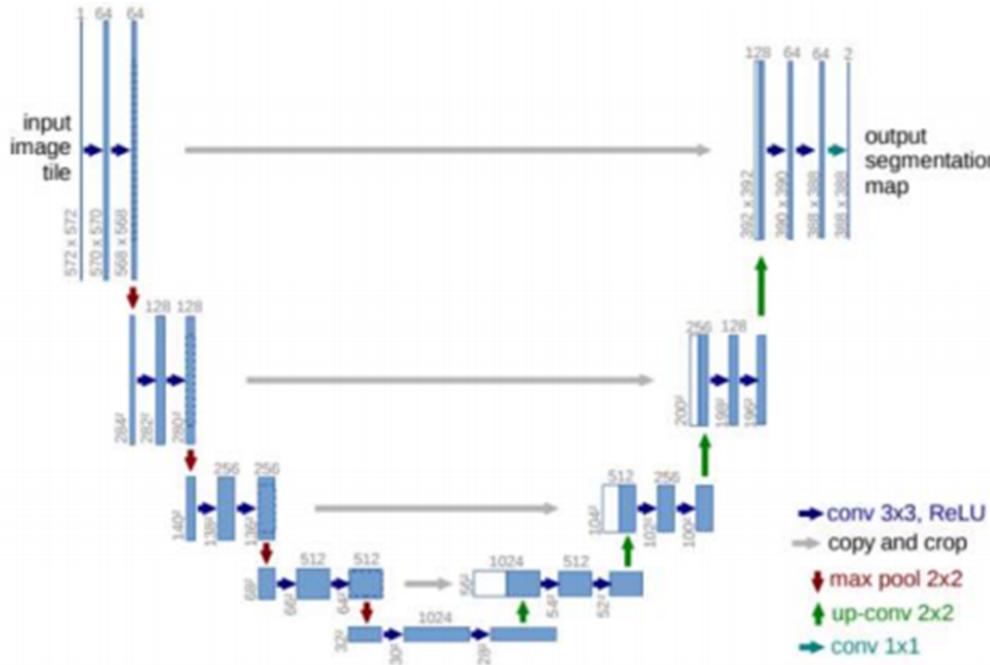
- **стартовый_блок** – стартовый блок слоёв
- **блок_PSP** - блок слоев PSP
- **финальный_блок** – финальный блок слоёв
- **количество_блоков** - количество блоков слоев (см. архитектуру PSPnet)
- **входной_размер** - размер входных данных (можно получить с помощью функции терра_ии.получить_входной_размер())

Для создания сети архитектуры UNET используется функция **создать_UNET**:

```
терра_ии.создать_UNET(параметры)
```

Создание нейронной сети

Архитектура UNET-модели:



Например.

```
# Указываем слои для создаваемой модели
блок_вниз_1 = 'Сверточный2D-32-3 Сверточный2D-32-3'
блок_вниз_2 = 'Сверточный2D-64-3 Сверточный2D-64-3'
блок_вниз_3 = 'Сверточный2D-128-3 Сверточный2D-128-3'

блок_внизу = 'Сверточный2D-256-3 Сверточный2D-256-3'

блок_вверх_1 = 'Объединение Сверточный2D-128-3 Сверточный2D-128-3'
блок_вверх_2 = 'Объединение Сверточный2D-64-3 Сверточный2D-64-3'
блок_вверх_3 = 'Объединение Сверточный2D-32-3 Сверточный2D-32-3'

# Получаем размер входных данных
входной_размер =
терра_ии.получить_входной_размер(обучающая_выборка)
# Создаем модель
нейронка_7 = терра_ии.создать_UNET(
    блоки_вниз = [блок_вниз_1, блок_вниз_2, блок_вниз_3],
    блок_внизу = блок_внизу,
    блоки_вверх = [блок_вверх_1, блок_вверх_2, блок_вверх_3],
    входной_размер = входной_размер
)
```

Создание нейронной сети

Параметры функции:

- **блок_вниз** - последовательность слоев блока уменьшения UNET
- **блок_вверх** - последовательность слоев блока увеличения UNET
- **блок_внизу** – последовательность слоев нижнего блока UNET
- **фильтры_вверх** - фильтры для верхнего блока
- **начальное_значение** - стартовое количество фильтров в первом блоке
- **входной размер** - размер входных данных (можно получить с помощью функции **терра_ии.получить_входной_размер()**)

8. Схема нейронной сети

Для вывода схемы нейронной сети используется функция **схема_модели**:

`терра_ии.схема_модели(нейронка)`

9. Обучение нейронной сети

Обучение нейронной сети происходит с помощью функции **обучение_модели**:

```
терра_ии.обучение_модели(  
    нейронка,  
    обучающая_выборка, метки_обучающей_выборки,  
    тестовая_выборка, метки_тестовой_выборки,  
    размер_пакета,  
    количество_эпох,  
    количество_запусков)
```

Параметры функции:

- **нейронка** - нейронная сеть для обучения
- **обучающая_выборка** - набор обучающих данных
- **метки_обучающей_выборки** - правильные ответы для обучающего набора
- **тестовая_выборка** - набор тестовых данных
- **метки_тестовой_выборки** - правильные ответы для тестового набора

Обучение нейронной сети

размер_пакета - размер пакета (количество картинок, на котором обучается нейронная сеть за один проход)

- количество_эпох - количество шагов обучения
- количество_запусков – количество запусков обучения

Примеры:

Для обучения сети на примерах АВТО-2:

```
эксперимент_1 = терра_ии.обучение_модели(
    нейронка_1,
    обучающая_выборка, метки_обучающей_выборки,
    тестовая_выборка, метки_тестовой_выборки,
    размер_пакета = 64,
    количество_эпох = 20,
    количество_запусков = 10)
```

Для обучения сети предсказания цены квартиры:

```
эксперимент_1 = терра_ии.обучение_модели_квартиры(
    нейронка,
    обучающая_выборка, метки_обучающей_выборки,
    тестовая_выборка, метки_тестовой_выборки,
    размер_пакета = 256,
    количество_эпох = 20,
    инструменты = инструменты)
```

10. Проверка работы нейронной сети

Для проверки качества работы сети для классификации изображений используется функция тест_модели_классификации:

```
терра_ии.тест_модели_классификации(параметры)
```

Например:

```
терра_ии.тест_модели_классификации(
    нейронка_1,
    тестовый_набор = тестовая_выборка,
    правильные_ответы = метки_тестовой_выборки,
    классы = ['Входящий пассажир', 'Выходящий пассажир'],
    количество = 3)
```

Проверка работы нейронной сети

Параметры функции:

- **нейронка** - проверяемая нейронка
- **тестовый_набор, правильные_ответы** - данные и метки тестовой выборки
- **классы** - используемые классы, например: классы = ['Феррари', 'Мерседес']
- **количество** - количество тестовых примеров

Или

```
терра_ии.тест_модели_на_своем_изображении(нейронка,  
размер_изображения, классы)
```

Параметры функции:

- **нейронка** - проверяемая нейронка
- **размер_изображения** - размер входных данных
- **классы** - используемые классы, например: классы = ['Феррари', 'Мерседес']

Данную функцию можно использовать для загрузки своего изображения (подходит только для задачи классификации изображений)

Для проверки качества работы сети для классификации изображений используется функция **тест_модели_сегментации**:

```
терра_ии.тест_модели_сегментации(параметры)
```

Например:

```
терра_ии.тест_модели_сегментации(  
нейронка_4,  
тестовые_изображения = тестовая_выборка)
```

Параметры функции:

- **нейронка** - проверяемая нейронка
- **тестовые_изображения** - набор изображений для проверки

Проверка работы нейронной сети

Для проверки качества работы сети для классификации симптомов заболеваний используется функция **тест_модели_симптомы**:

терра_ии.тест_модели_симптомы(параметры)

Например:

```
симптомы = 'общую слабость; повышение температуры  
тела; систематические приступы тошноты; регулярную рвоту с  
примесями желчи.'
```

```
терра_ии.тест_модели_симптомы(нейронка, симптомы,  
классы=['Колит', 'Гепатит', 'Гастрит', 'Холицестит', 'Дуоденит',  
'Энтерит', 'Язва', 'Эзофагит', 'Аппендицит', 'Панкреатит'])
```

Параметры функции:

- **нейронка** - проверяемая нейронка
- **симптомы** - строка с текстом симптомов, заключенным в кавычки

Пример:

```
симптомы = '''боль и тяжесть в правом подреберье, лопатке, боку;  
отсутствие аппетита, рвота, тошнота'''
```

- **классы** - список классов, подаваемых при обучении
- **размер окна** - размер окна индексов слов
- **шаг** - шаг смещения по данным выборки

Для проверки качества работы сети для классификации текстов писателей используется функция **тест_модели_писатели**:

терра_ии.тест_модели_писатели(параметры)

Параметры функции:

- **нейронка** - проверяемая нейронка
- **размер окна** - размер окна индексов слов
- **шаг** - шаг смещения по данным выборки
- **классы** - список классов - писатели, подаваемых при обучении

Для проверки качества работы сети для определения цены на квартиру используется функция **тест_модели_квартиры**:

терра_ии.тест_модели_квартиры(параметры)

Проверка работы нейронной сети

- **нейронка** - проверяемая нейронка
- **инструменты** - инструменты, загруженные из базы квартир
- **метро, до_станции, способ_передвижения, этаж, всего_этажей, тип_балкона, тип_санузла, площадь, описание** - параметры, необходимые для определения стоимости квартиры (их задаём вручную в ноутбуке)

Например:

```
терра_ии.тест_модели_квартиры(  
    нейронка, инструменты, метро, до_станции,  
    способ_передвижения, этаж,  
    всего_этажей, тип_балкона, тип_санузла, площадь, описание)
```

Для проверки качества работы сети для трейдинга используется функция **тест_модели_торговли**:

```
терра_ии.тест_модели_торговли(параметры)
```

Параметры функции:

- **нейронка** - проверяемая нейронка
- **выборки, данные, период предсказания, количество анализируемых дней** – определяются по названию

Для проверки качества работы сети для распознавания голосовых команд используется функция **тест_модели_голосовых_команд**:

```
терра_ии.тест_модели_голосовых_команд(параметры)
```

- **нейронка** - проверяемая нейронка
- **порог** - определяет чувствительность к команде
- **длина** - длина массива данных аудио при формировании выборки
- **шаг** - шаг при формировании выборки

Например:

```
терра_ии.тест_модели_голосовых_команд(нейронка_1, порог=0.95,  
длина=длина, шаг=шаг)
```

Для проверки качества работы сети подбора вакансий используется функция **тест_модели_вакансии**:

Проверка работы нейронной сети

```
терра_ии.тест_модели_вакансии(параметры)
```

Параметры функции:

- **нейронка** - проверяемая нейронка
- **тестовая_выборка, метки_тестовой_выборки** – набор данных для проверки

Например:

```
терра_ии.тест_модели_вакансии(нейронка,  
тестовая_выборка, метки_тестовой_выборки)
```

Для проверки качества работы сети по классификации отзывов об автомобиле TESLA используется функция **тест_модели_отзывы**:

```
терра_ии.тест_модели_отзывы(параметры)
```

Параметры функции:

- **нейронка** - проверяемая нейронка
- **размер окна** - размер окна индексов слов
- **шаг** - шаг смещения по данным выборки
- **отзыв** – анализируемый отзыв
- **классы** - список классов - писатели, подаваемых при обучении

Например:

```
отзыв = 'Машина скоростная. По трассе идет отлично'  
терра_ии.тест_модели_отзывы(нейронка_1, размер_окна, шаг,  
отзыв, классы = [ 'Негативный отзыв', 'Позитивный отзыв' ] )
```

Рекомендации по устранению ошибок при использовании

11. Рекомендации по устранению ошибок при использовании

Внимание! Сбои в работе фреймворка возможны также по причине неполадок Google Colaboratory. Если возникает непонятная ошибка, вначале нажимаем Среда выполнения, перезапустить среду выполнения, затем запускаем все ячейки программного кода с самого начала. Если не помогло, тогда смотрим указанные здесь возможные причины сбоев.

Google Colaboratory предоставляет ограниченное количество памяти. В случае ее нехватки для решения отдельной задачи рекомендуется создать отдельный ноутбук под эту задачу и решать её там. Это позволит провести больше тестов. Если память снова закончилась, перезапустить все ячейки, то есть выполнить их последовательно с самого начала. Если хотите оставить первый результат обучения, не перезапускайте эту ячейку, а создайте новую и запустите обучение в новой ячейке. Так вы сможете визуально сравнить результаты обучения новой и предыдущей модели.

При серьезных сбоях запуска рекомендуется работать в другом браузере, перезапустить компьютер.

Теперь рассмотрим, как устранять ошибки, возникающие при работе с фреймворком:

1. При появлении ошибок в работе функций фреймворка проверьте, что вы установили наиболее подходящую его версию (см. рекомендации при установке, п.1.). Если все остальные рекомендации выполнены, а сбои остались, можно менять версию на другую из перечисленных.
2. Если у вас модель сети Свёрточная, то данные для неё необходимо готовить, как для Свёрточной, а если Полносвязная - как для Полносвязной, т. к. размерности данных для этих сетей различны, и если перепутать, возникает ошибка при обучении модели.
3. При написании названий тренировочных и тестовых данных эти названия должны быть одинаковы как при создании выборок, так и при обработке данных и обучении модели.

Рекомендации по устранению ошибок при использовании

4. Ошибки при создании модели часто связаны с ошибками в названии слоёв (п.9).
5. Другой распространённой ошибкой бывает то, что при создании модели указывают одно название модели сети, например, нейронка_1, а при обучении другое, например, нейронка.
6. Чтобы улучшить точность при обучении, необходимо пробовать варианты и подбирать подходящую модель.
7. При проверке работы чат-бота задаваемый вопрос должен содержать те слова, на которых чат-бот обучался или которые он выводит в качестве ответов в том же падеже. Использование других слов может привести к ошибке.
8. При выходе из чат-бота слово Выход нужно вводить с заглавной буквы, как в этом примере.
9. Если не достигается требуемая точность сети, можно упростить сеть: уменьшить количество слоёв в ней. Если же это не помогает, стоит проверить, выполняются ли следующие простые правила:
 1. После 1-3 слоёв Сверточный2D или Сверточный1D ставится МаксПуллинг.
 2. Между слоями, а иногда и перед ними, ставятся слои Нормализация и/или Дропаут. При этом лучше при каждой установке такого слоя между другими слоями проверять, не ухудшает ли это обучение.
10. Также лучше обратить внимание на значение отступов. Фреймворк использует отступы по образцу, принятому в стандарте языка Python. Отступы дают возможность интерпретатору программы понять, к каким логическим блокам относятся данные строки программного кода. Обратите внимание, что не все примеры кода в данной документации уместились в одну строку и поэтому были перенесены на следующую, и не везде получилось сделать это по правилам. Перенос выполняется после запятой (либо перед скобкой без запятой), и последующие строки берутся с отступом до конца записи команды, как, например, здесь:

Рекомендации по устранению ошибок при использовании

```
терра_ии.тест_модели_торговли(  
    нейронка_1,  
    тестовая_выборка, метки_тестовой_выборки,  
    данные,  
    период_предсказания,  
    количество_анализируемых_дней  
)
```

Если не уверены, лучше запишите команду в одну строку, убедитесь, что она правильно выполняется, а затем переносите строки и запускайте код, чтобы убедиться, что перенос выполнен правильно.