

# Создание веб-приложений для демонстрации моделей машинного обучения



# Краткий план лекции

- Библиотеки (фреймворки) для демонстрации моделей машинного обучения:
  - voilà
  - gradio
  - streamlit
  - panel
  - dash (наиболее старый и стабильный)
- Сравнение фреймворков:
  - [Gradio vs Streamlit vs Dash vs Flask](#)
  - [Streamlit vs. Dash vs. Shiny vs. Voila vs. Flask vs. Jupyter](#)

# Фреймворки для демонстрации моделей машинного обучения

- Основное отличие от традиционных серверных фреймворков, таких как Django или Flask:
  - Традиционные фреймворки ориентированы на работу с базами данных, ввод данных, формирование отчетов.
  - Фреймворки для демонстрации моделей машинного обучения позволяют быстро создать пользовательский интерфейс для моделей машинного обучения.
  - Фреймворки для демонстрации моделей машинного обучения используют принцип дашбордов (приборных панелей).

# Фреймворк Voilà

- Voilà превращает Jupyter-ноутбуки в автономные веб-приложения, используя сервер Tornado.
- Каждый пользователь, подключающийся к приложению Voilà tornado, получает отдельное ядро Jupyter, которое может выполнять обратные вызовы при изменениях в интерактивных виджетах Jupyter-ноутбуков.
- По умолчанию Voilà выполняется с параметром `strike_source=True`, который удаляет входные ячейки из отображаемого ноутбука. Используя параметр `strike_source=False` можно показывать входные ячейки.
- Галерея примеров.

# Фреймворк gradio

- Основные особенности фреймворка:
  - Gradio-приложение является веб-приложением и запускается в браузере. Режим «горячего обновления» НЕ поддерживается - необходимо останавливать и перезапускать приложение.
  - Вместо реактивности:
    1. Создаются функции обработки данных;
    2. К входам и выходам функций присоединяются компоненты;
    3. Компоненты явно разделяются на входные и выходные, приложение содержит две панели – для входных и выходных компонентов.
    4. Функции и компоненты оборачиваются в основной компонент, который называется «интерфейс».
    5. Удобная функциональность для создания копий экрана, выводится время работы функций.
- Базовый пример приложения – [https://www.gradio.app/getting\\_started](https://www.gradio.app/getting_started)
- Работа с моделями машинного обучения – <https://www.gradio.app/guides>
- Описание основных функций (API) – <https://www.gradio.app/docs>
- Примеры находятся в репозитории – [https://github.com/ugapanyuk/ml\\_gradio\\_example](https://github.com/ugapanyuk/ml_gradio_example)



# Фреймворк gradio (example\_1)

## Метод ближайших соседей

КОЛИЧЕСТВО СОСЕДЕЙ

КОЛИЧЕСТВО ФОЛДОВ

### ОЦЕНКИ ПО ФОЛДАМ (LABEL)

4

4	99%
5	99%
2	96%
8	96%
6	94%
0	93%
7	92%
1	91%
3	88%

### ОЦЕНКИ ПО ФОЛДАМ (KEYVALUES)


Property	Value
0	0.9348066298342541
1	0.9060773480662984
2	0.9646408839779006
3	0.8751381215469614
4	0.994475138121547
5	0.9867403314917127
6	0.9403314917127071
7	0.915929203539823
8	0.9579646017699115

### УСРЕДНЕННОЕ ЗНАЧЕНИЕ ACCURACY

Latency: 106.30s

CLEAR

SUBMIT



FLAG

# Фреймворк gradio (example\_2)

## Модели машинного обучения

ВЫБЕРИТЕ МОДЕЛИ

☒ LogR

☒ KNN\_5

☒ SVC

☐ Tree

☒ RF

☒ GB

ROC AUC

RF



Latency: 41.85s

CLEAR

SUBMIT



FLAG

# Фреймворк streamlit (введение)

- Установка
  - Как пакет Python с помощью pip, но при этом устанавливается исполняемый файл streamlit (streamlit.exe для windows) и запуск проектов производится с его помощью.
- Примеры приложений
  - Стандартные примеры – запускаются командой *streamlit hello*
  - Примеры на сайте – <https://streamlit.io/gallery>
- Основные особенности фреймворка:
  - Streamlit-приложение является веб-приложением и запускается в браузере. Поддерживается режим «горячего обновления» - можно внести изменения в код на Python, сохранить изменения, выбрать пункт меню «rerun» и изменения автоматически применятся.
  - Реактивность – изменения компонентов автоматически передаются всем связанным компонентам. Это упрощает разработку, но может приводить к задержкам при работе приложения.
  - Входные и выходные компоненты располагаются в общем потоке и могут перемешиваться. Для удобства расположения компонентов предусмотрена боковая панель.



# Фреймворк streamlit (документация)

- Страница документации – <https://docs.streamlit.io/>
- Тьюториалы по созданию приложений (TUTORIALS)
  - Пример простого приложения - <https://docs.streamlit.io/library/get-started/create-an-app>
  - Пример приложения, использующего модели машинного обучения - <https://towardsdatascience.com/building-machine-learning-apps-with-streamlit-667cef3ff509>
- Основные концепции создания streamlit-приложения - <https://docs.streamlit.io/library/get-started/main-concepts>
- Описание основных функций (API) - <https://docs.streamlit.io/library/api-reference>
- Ускорение работы приложений и кэширование – <https://docs.streamlit.io/library/advanced-features/caching>
- Конфигурирование приложений - <https://docs.streamlit.io/library/advanced-features/configuration>  
Расширение фреймворка за счет разрабатываемых компонентов - <https://docs.streamlit.io/library/components/create>

# Фреймворк streamlit (примеры)

- Примеры находятся в репозитории - [https://github.com/ugapanyuk/ml\\_streamlit\\_example\\_22](https://github.com/ugapanyuk/ml_streamlit_example_22)
- Запуск примеров:
  - «streamlit run сценарий.py»

# Фреймворк streamlit (example\_1)

## Вывод данных и графиков

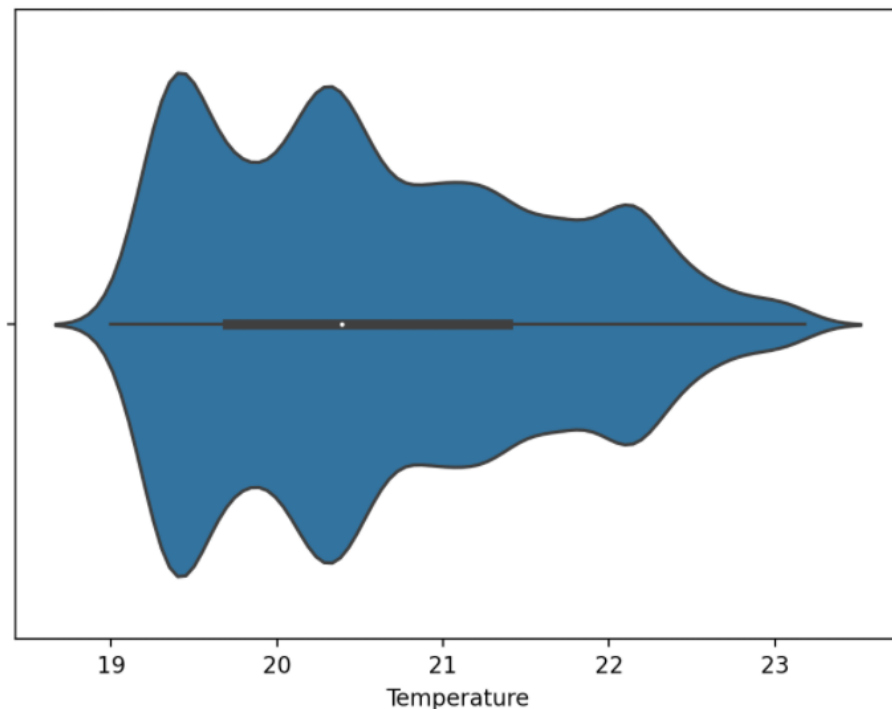
Данные загружены!

### Первые 5 значений

	date	Temperature	Humidity	Light	CO2	HumidityRat
1	2015-02-04 17:51:00	23.1800	27.2720	426	721.2500	0.00
2	2015-02-04 17:51:59	23.1500	27.2675	429.5000	714	0.00
3	2015-02-04 17:53:00	23.1500	27.2450	426	713.5000	0.00
4	2015-02-04 17:54:00	23.1500	27.2000	426	708.2500	0.00
5	2015-02-04 17:55:00	23.1000	27.2000	426	704.5000	0.00

☐ Показать все данные

### Скрипичные диаграммы для числовых колонок



# Фреймворк streamlit (example\_2)

## Обучение модели ближайших соседей

Данные загружены!

☐ Описание метода

☐ Показать корреляционную матрицу

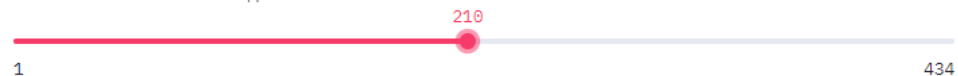
Количество фолдов:



Количество строк в наборе данных - 500

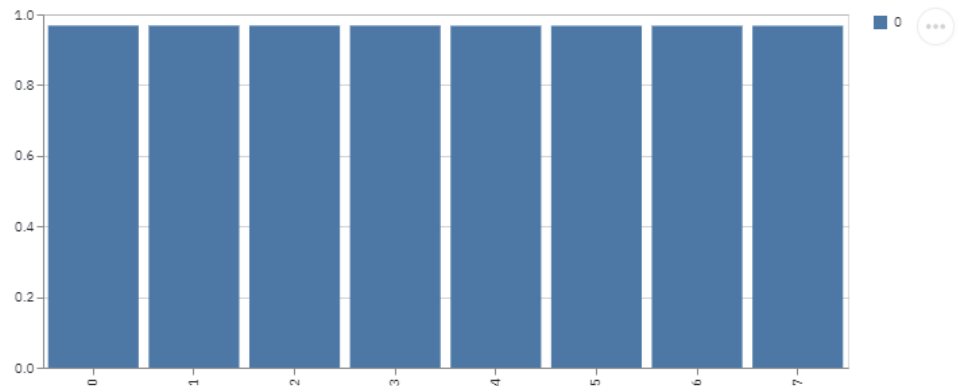
Максимальное допустимое количество ближайших соседей с учетом выбранного количества фолдов - 434

Количество ближайших соседей:



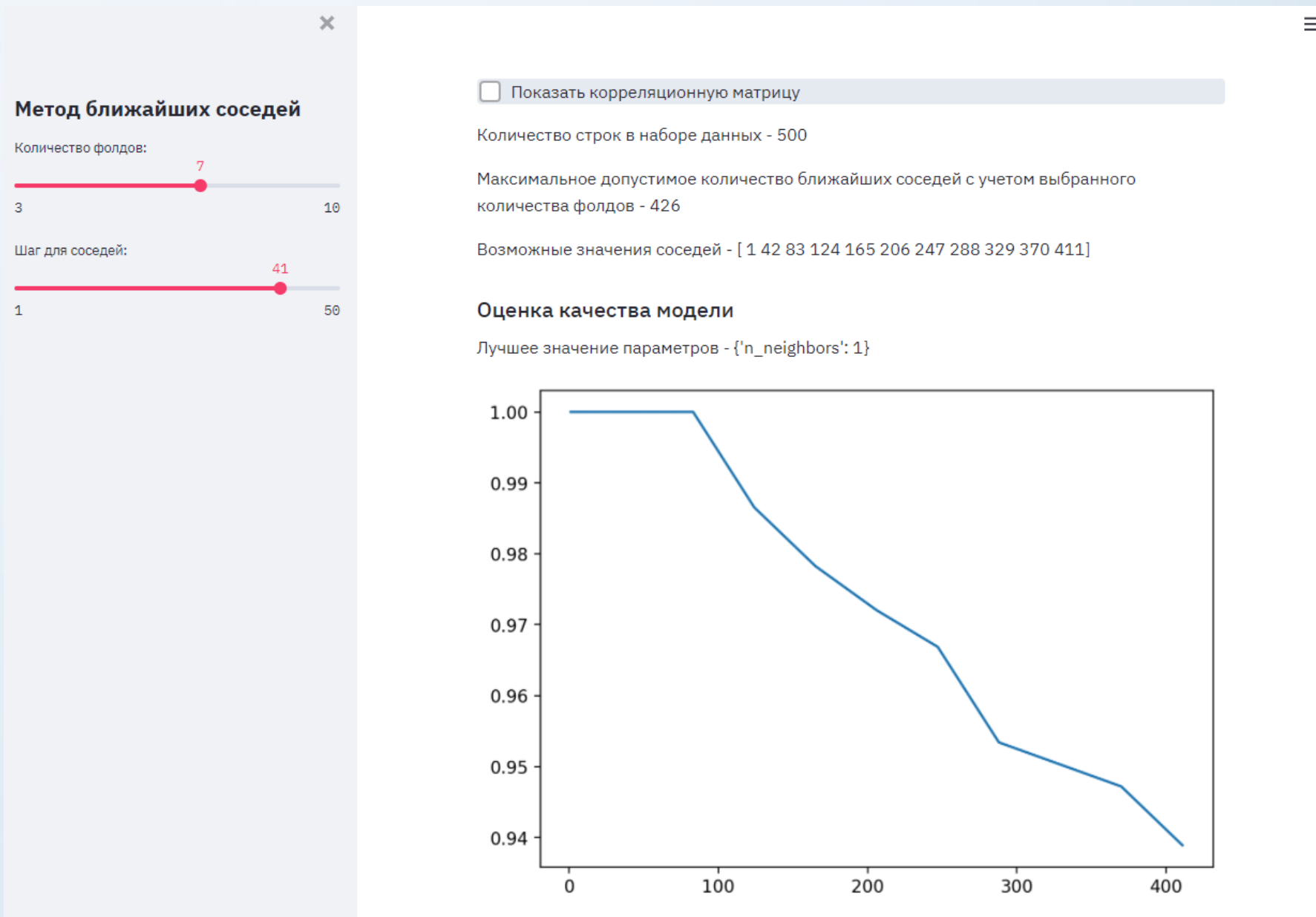
## Оценка качества модели

Значения ассигасу для отдельных фолдов



Усредненное значение ассигасу по всем фолдам - 0.9679979518689197

# Фреймворк streamlit (example\_3)



# Фреймворк streamlit (example\_4)





# Фреймворк [panel](#)

- Может работать как в режиме ноутбука, так и в режиме автономного веб-приложения.
- [Репозиторий](#) (с описанием совместимых библиотек).
- [Документация](#).
- [Галерея примеров](#).
- [Статья с описанием фреймворка](#) – отмечается что фреймворк является довольно сложным в освоении.
- Является частью семейства библиотек [holoviz](#).

# Фреймворк dash (введение)

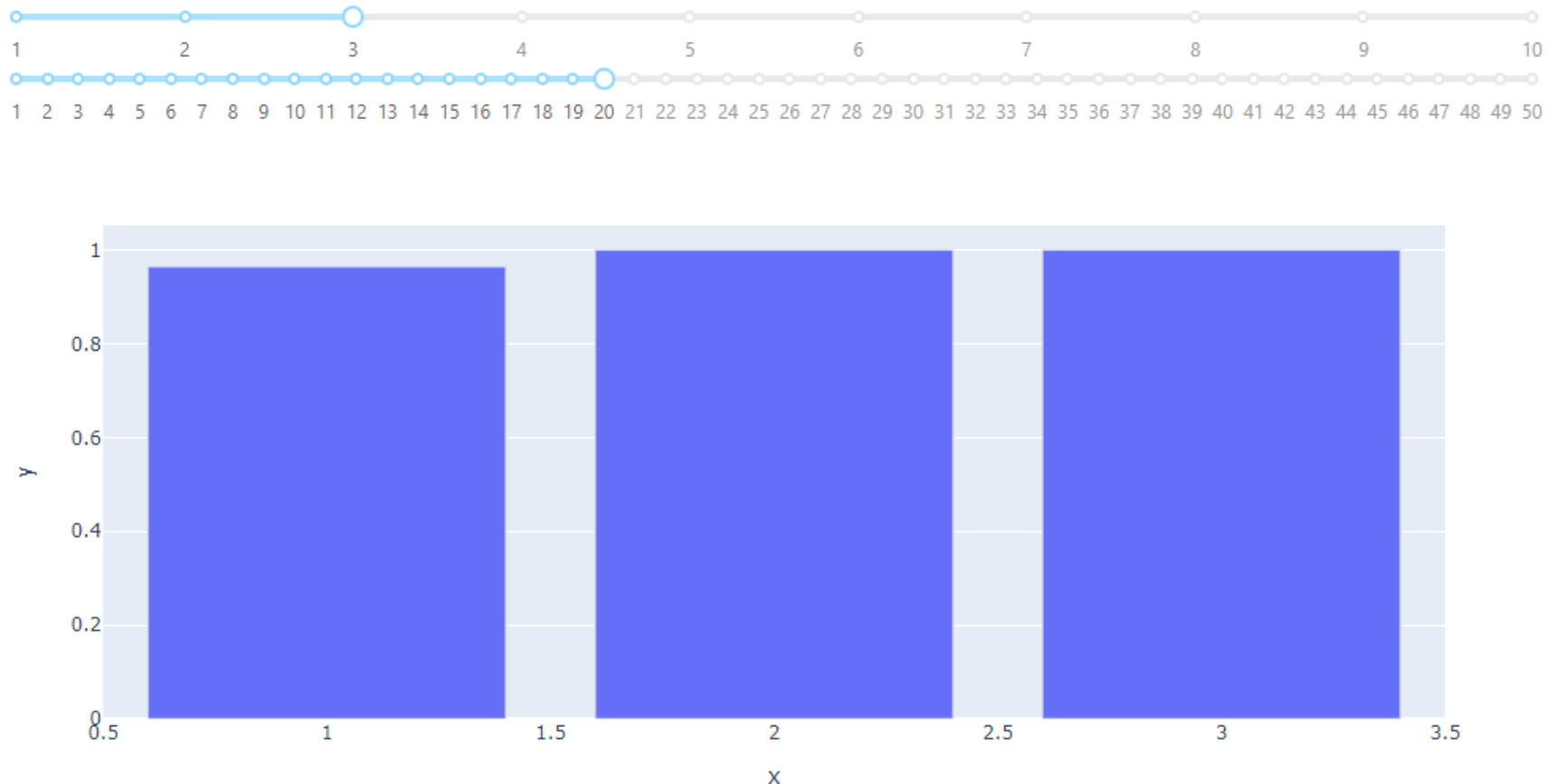
- Основные особенности фреймворка:
  - Dash является единственным из рассматриваемых фреймворков, который является достаточно зрелым для разработки промышленных решений.
  - Dash-приложение является веб-приложением и запускается в браузере. Поддерживается режим «горячего обновления» - можно внести изменения в код на Python, сохранить изменения, обновить окно браузера, и изменения автоматически применятся.
  - Реактивность – изменения компонентов автоматически передаются связанным компонентам.
  - Приложение на dash состоит из двух основных элементов:
    - Layout – определяет расположение компонентов. В отличие от динамического добавления компонентов в streamlit, dash ориентирован на статическое расположение компонентов.
    - Callbacks – определяют связь между компонентами. Реактивность реализуется только в необходимых случаях.
  - Может выводиться граф вызова callback'ов.

# Фреймворк dash (документация)

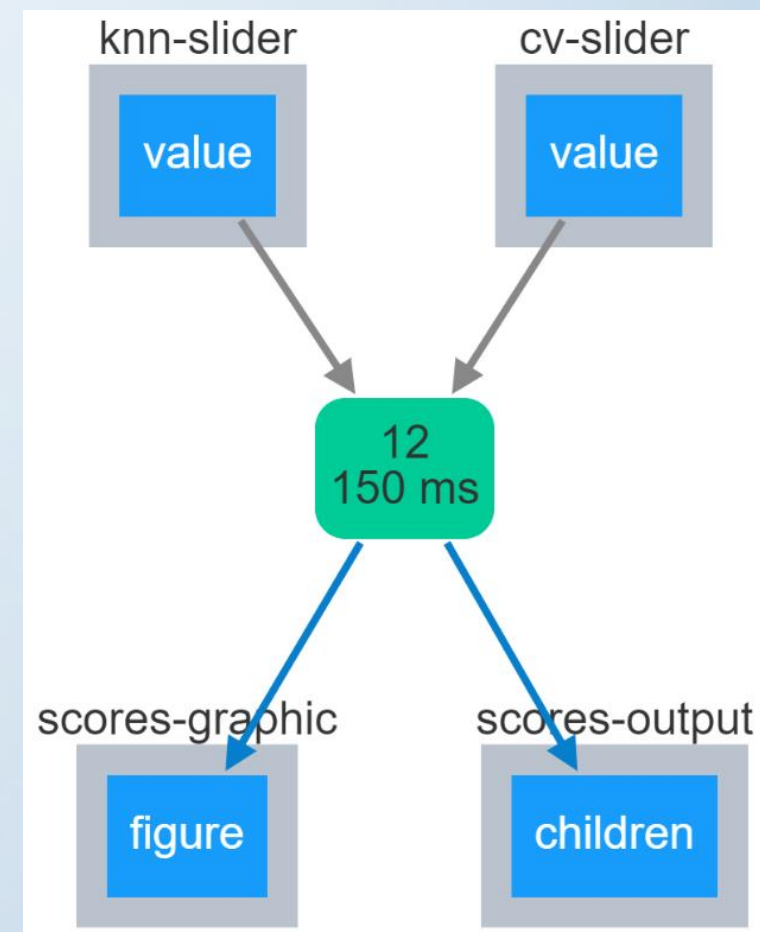
- Базовый тьюториал – установка и создание простого приложения.
- *Dash* содержит очень развитую документацию и большое количество компонентов.
- Разделы документации:
  - Dash Callbacks – разработка callback'ов.
  - Open Source Component Libraries – библиотеки компонентов.
  - Creating Your Own Components – создание собственных компонентов.
  - Beyond the Basics – дополнительные разделы, увеличение производительности приложения.
- Примеры находятся в репозитории – [https://github.com/ugapanyuk/ml\\_dash\\_example](https://github.com/ugapanyuk/ml_dash_example)

# Фреймворк dash (example\_1)

## Метод ближайших соседей



Усредненное значение ассигуру по всем фолдам - 0.9880239520958084



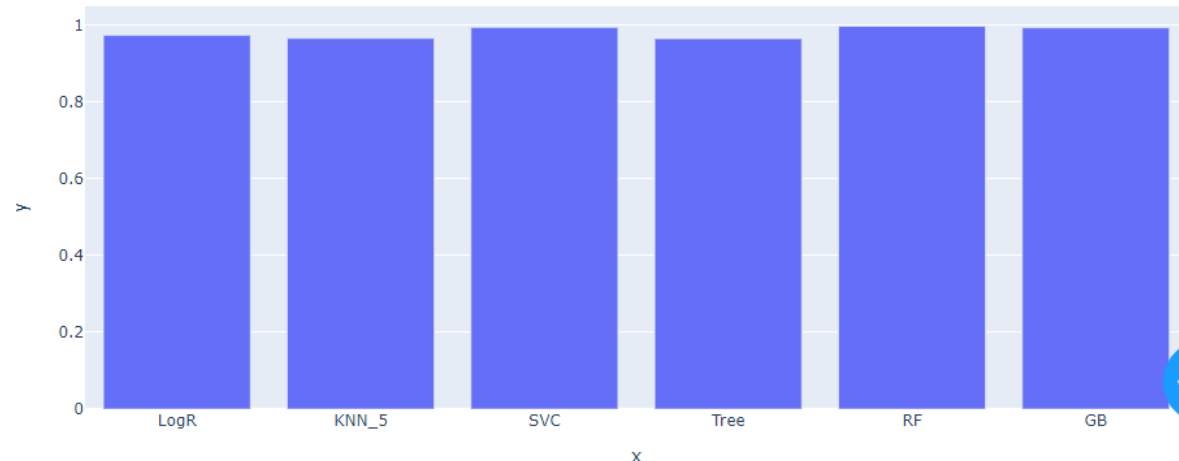
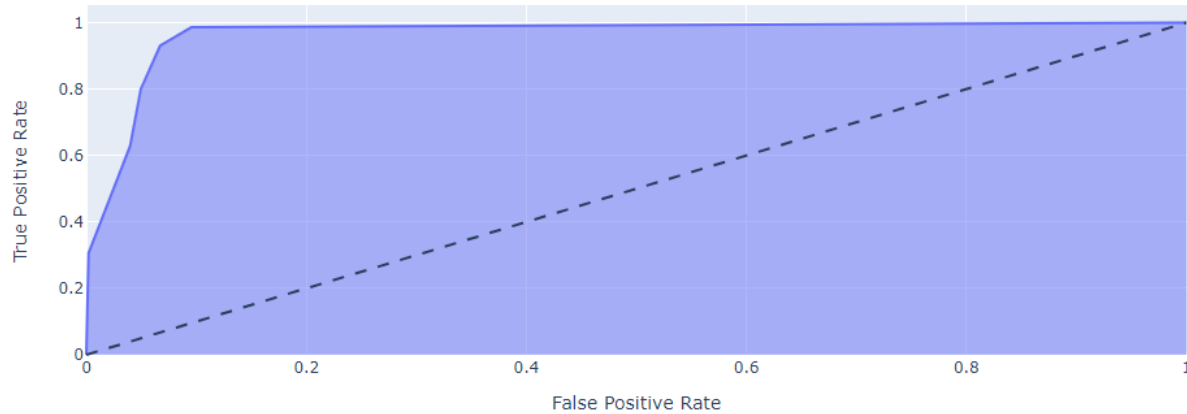
Граф вызова callback'ов

# Фреймворк dash (example\_2)

## Модели машинного обучения

KNN\_5

ROC Curve (AUC=0.9655)



model-name

value

7

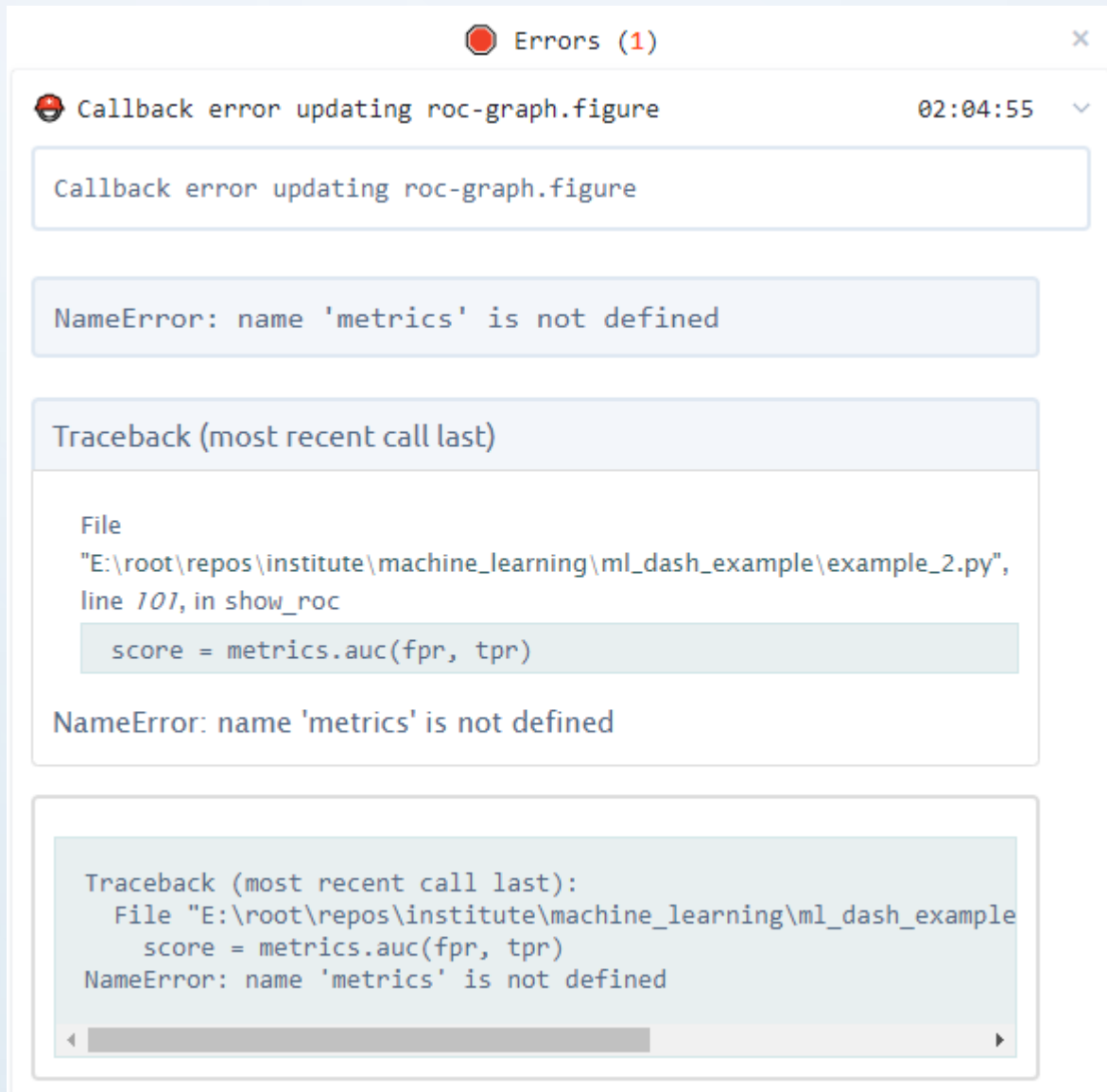
100 ms

roc-graph

figure

Граф вызова callback'ов

# Фреймворк dash (example\_2)



The screenshot shows the Dash web interface with an error message displayed. The error message is titled "Errors (1)" and contains the following text:

Callback error updating roc-graph.figure 02:04:55

Callback error updating roc-graph.figure

NameError: name 'metrics' is not defined

Traceback (most recent call last)

File  
"E:\root\repos\institute\machine\_learning\ml\_dash\_example\example\_2.py",  
line 101, in show\_roc  
score = metrics.auc(fpr, tpr)

NameError: name 'metrics' is not defined

Traceback (most recent call last):  
File "E:\root\repos\institute\machine\_learning\ml\_dash\_example  
score = metrics.auc(fpr, tpr)  
NameError: name 'metrics' is not defined

Детальные сообщения  
об ошибках выдаются  
в браузере



# Выводы

- Фреймворки `voilà`, `panel`, `streamlit` и `gradio` в большей степени предназначены для быстрого создания визуальных интерфейсов на основе моделей машинного обучения.
- Фреймворк `dash` является промышленным решением с большим количеством компонентов.
- Трудоемкость разработки приложения для `dash`, как правило, выше чем для `voilà`, `panel`, `streamlit` и `gradio`.