МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российскийтехнологическийуниверситет»

РТУ МИРЭА

ИКБ направление «Киберразведка и противодействие угрозам с применением технологий искусственного интеллекта» 10.04.01

Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»

Отчет по практической работе № 4

по дисципилне

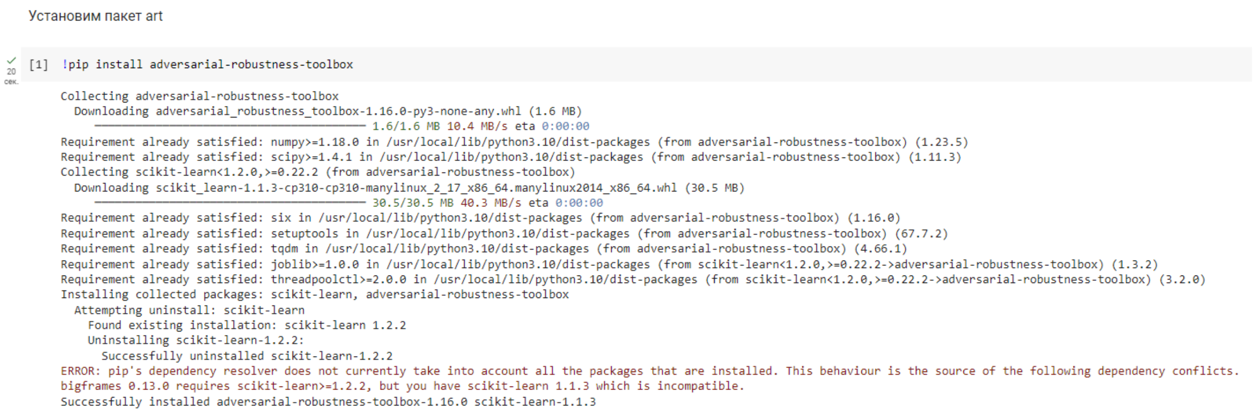
«Анализ защищенности систем искусственного интеллекта»

Группа:
ББМО-02-22
Выполнила:
Волкова Е.А.

Проверил:
Спирин А.А.

Москва 2023

Устанавливаем пакет ART adversarial-robustness-toolbox.

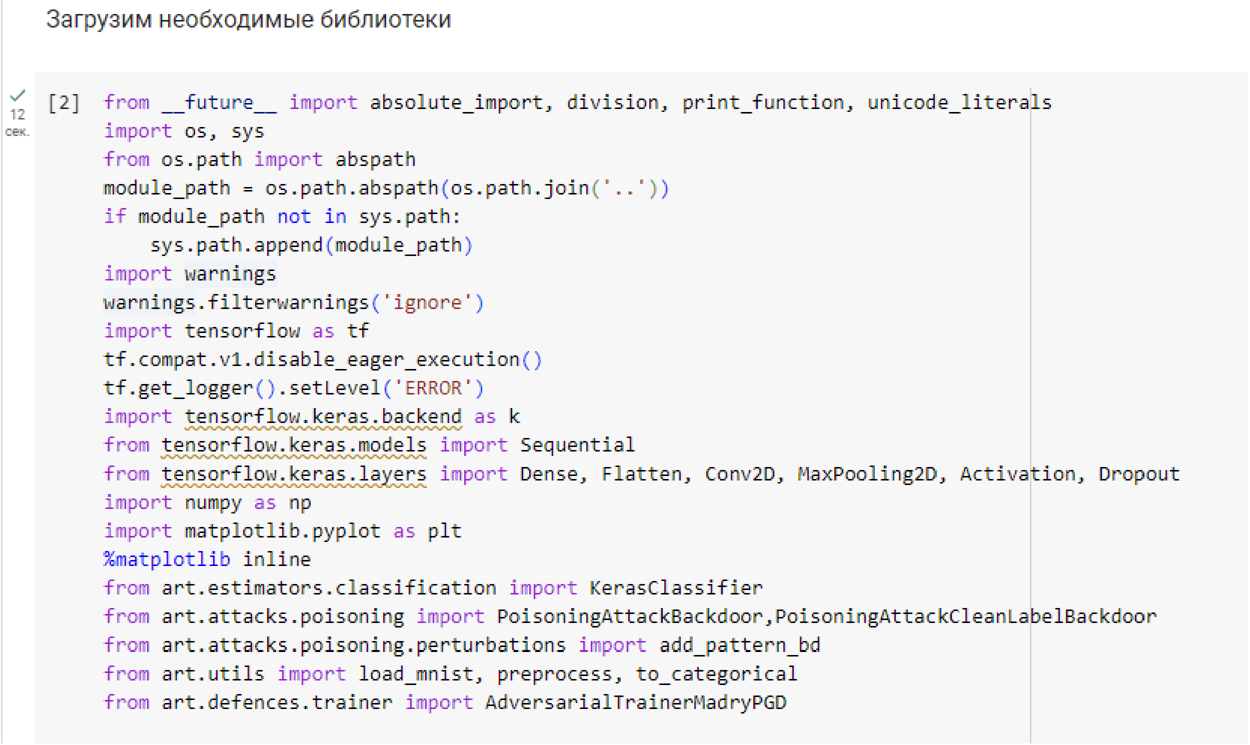


Установим пакет art

```
[1]  !pip install adversarial-robustness-toolbox

Collecting adversarial-robustness-toolbox
  Downloading adversarial_robustness_toolbox-1.16.0-py3-none-any.whl (1.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 10.4 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.18.0 in /usr/local/lib/python3.10/dist-packages (from adversarial-robustness-toolbox) (1.23.5)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from adversarial-robustness-toolbox) (1.11.3)
Collecting scikit-learn<1.2.0,>=0.22.2 (from adversarial-robustness-toolbox)
  Downloading scikit_learn-1.1.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (30.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 30.5/30.5 MB 40.3 MB/s eta 0:00:00
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from adversarial-robustness-toolbox) (1.16.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from adversarial-robustness-toolbox) (67.7.2)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from adversarial-robustness-toolbox) (4.66.1)
Requirement already satisfied: joblib>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn<1.2.0,>=0.22.2->adversarial-robustness-toolbox) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn<1.2.0,>=0.22.2->adversarial-robustness-toolbox) (3.2.0)
Installing collected packages: scikit-learn, adversarial-robustness-toolbox
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 1.2.2
    Uninstalling scikit-learn-1.2.2:
      Successfully uninstalled scikit-learn-1.2.2
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
bigframes 0.13.0 requires scikit-learn>=1.2.2, but you have scikit-learn 1.1.3 which is incompatible.
Successfully installed adversarial-robustness-toolbox-1.16.0 scikit-learn-1.1.3
```

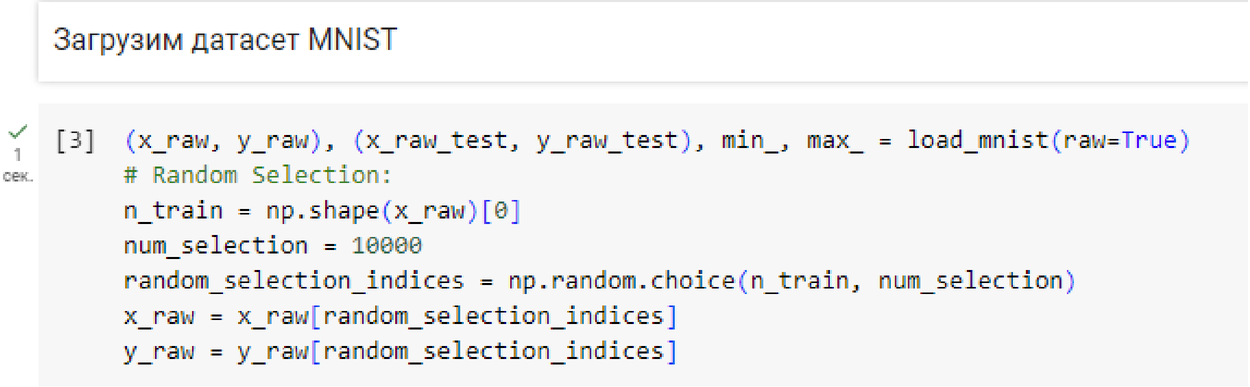
Загружаем необходимые библиотеки.



Загрузим необходимые библиотеки

```
[2]  from __future__ import absolute_import, division, print_function, unicode_literals
     import os, sys
     from os.path import abspath
     module_path = os.path.abspath(os.path.join('..'))
     if module_path not in sys.path:
         sys.path.append(module_path)
     import warnings
     warnings.filterwarnings('ignore')
     import tensorflow as tf
     tf.compat.v1.disable_eager_execution()
     tf.get_logger().setLevel('ERROR')
     import tensorflow.keras.backend as k
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Activation, Dropout
     import numpy as np
     import matplotlib.pyplot as plt
     %matplotlib inline
     from art.estimators.classification import KerasClassifier
     from art.attacks.poisoning import PoisoningAttackBackdoor,PoisoningAttackCleanLabelBackdoor
     from art.attacks.poisoning.perturbations import add_pattern_bd
     from art.utils import load_mnist, preprocess, to_categorical
     from art.defences.trainer import AdversarialTrainerMadryPGD
```

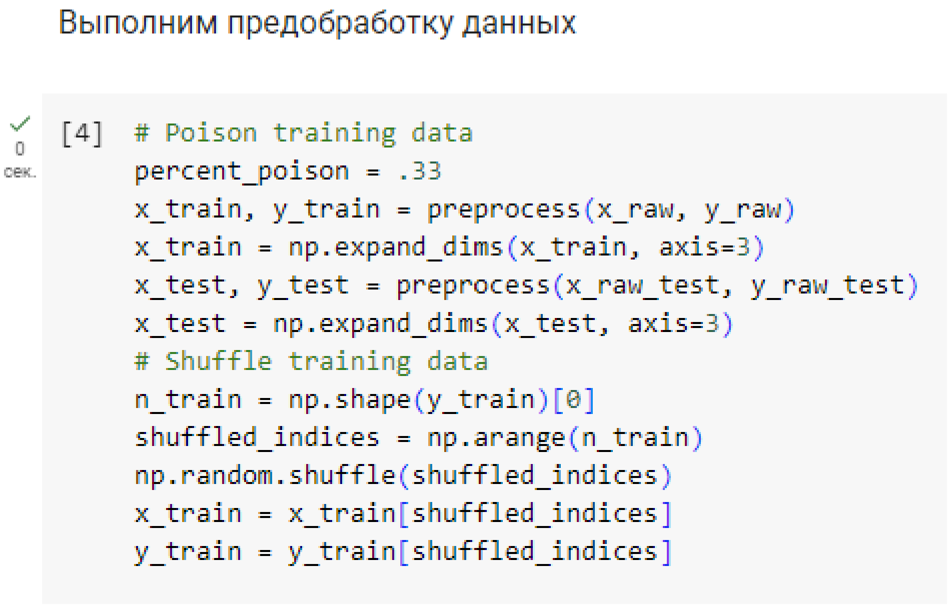
Загружаем датасет MNIST. Для ускорения обучения создается случайная выборка.



Загрузим датасет MNIST

```
[3]  (x_raw, y_raw), (x_raw_test, y_raw_test), min_, max_ = load_mnist(raw=True)
     # Random Selection:
     n_train = np.shape(x_raw)[0]
     num_selection = 10000
     random_selection_indices = np.random.choice(n_train, num_selection)
     x_raw = x_raw[random_selection_indices]
     y_raw = y_raw[random_selection_indices]
```

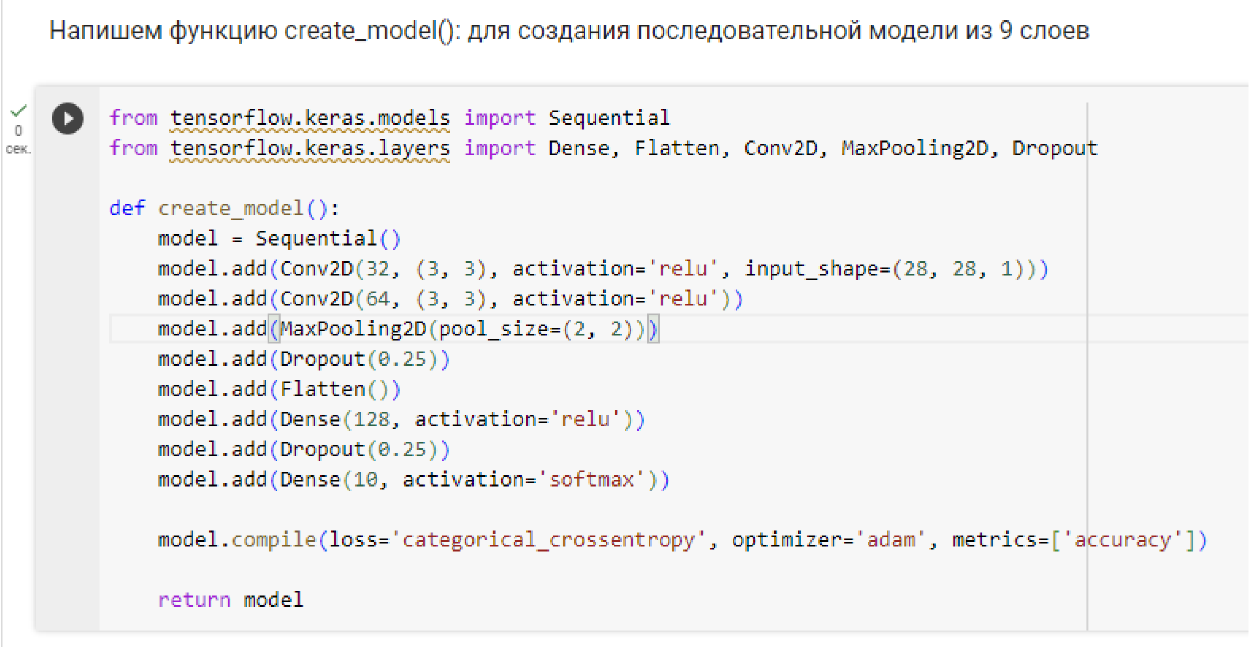
Выполняем предобработку данных.



Выполним предобработку данных

```
# Poison training data
percent_poison = .33
x_train, y_train = preprocess(x_raw, y_raw)
x_train = np.expand_dims(x_train, axis=3)
x_test, y_test = preprocess(x_raw_test, y_raw_test)
x_test = np.expand_dims(x_test, axis=3)
# Shuffle training data
n_train = np.shape(y_train)[0]
shuffled_indices = np.arange(n_train)
np.random.shuffle(shuffled_indices)
x_train = x_train[shuffled_indices]
y_train = y_train[shuffled_indices]
```

Создаем функцию create_model(): для создания последовательной модели из 9 слоев с заданными параметрами.



Напишем функцию create_model(): для создания последовательной модели из 9 слоев

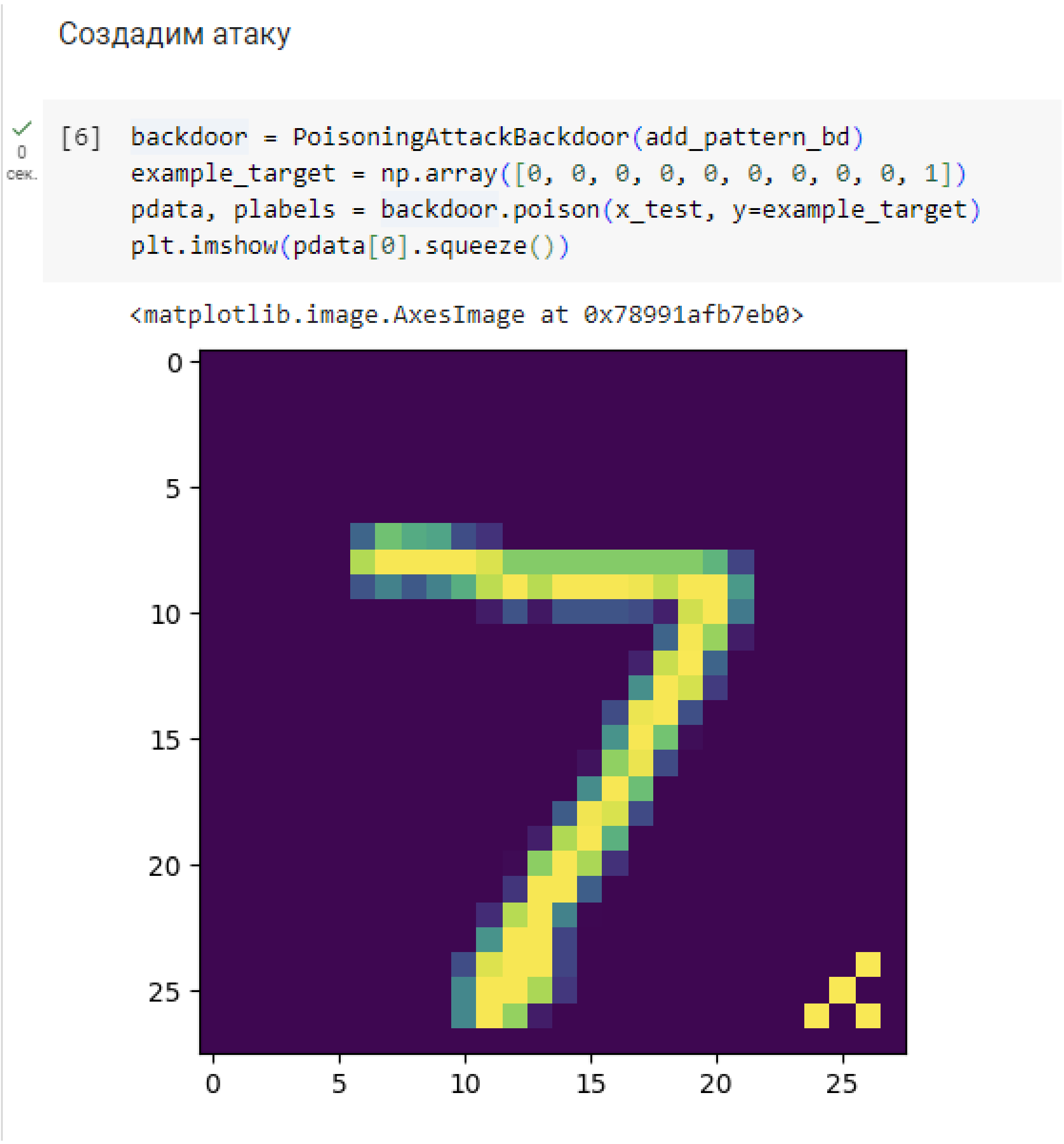
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout

def create_model():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.25))
    model.add(Dense(10, activation='softmax'))

    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

    return model
```

Создаем атаку для внедрения отравленных данных в тестовый набор данных и отобразим изображение.

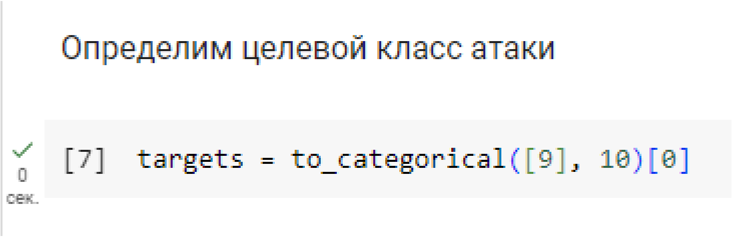


Создадим атаку

```
[6] backdoor = PoisoningAttackBackdoor(add_pattern_bd)
    example_target = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0, 1])
    pdata, plabels = backdoor.poison(x_test, y=example_target)
    plt.imshow(pdata[0].squeeze())
```

```
<matplotlib.image.AxesImage at 0x78991afb7eb0>
```

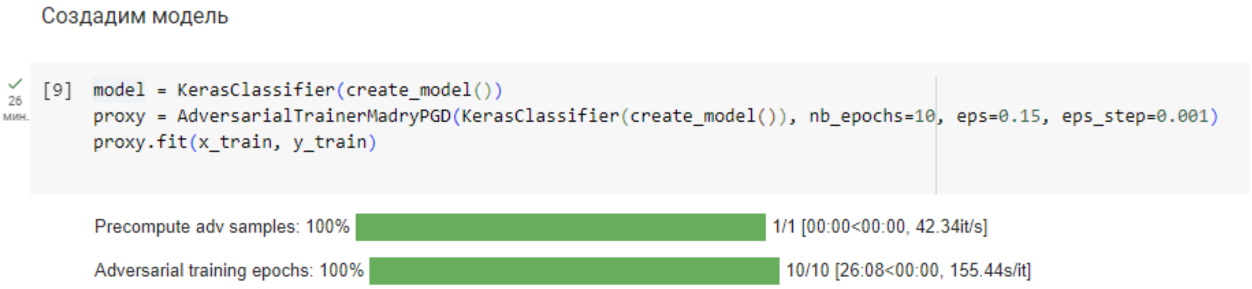
Создаем целевой класс атаки (9).



Определим целевой класс атаки

```
[7] targets = to_categorical([9], 10)[0]
```

Создаем модель.

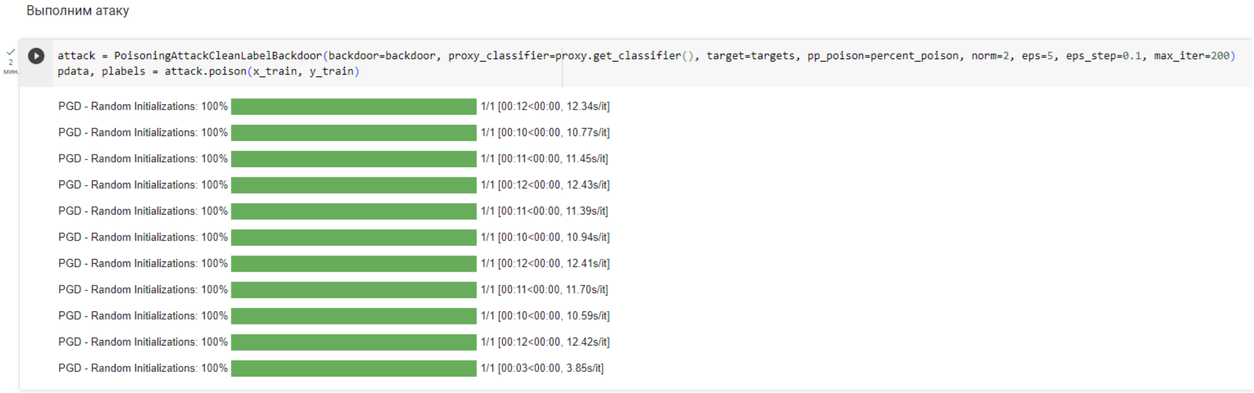


Создадим модель

```
[9] model = KerasClassifier(create_model())
    proxy = AdversarialTrainerMadryPGD(KerasClassifier(create_model()), nb_epochs=10, eps=0.15, eps_step=0.001)
    proxy.fit(x_train, y_train)
```

```
Precompute adv samples: 100% 1/1 [00:00<00:00, 42.34it/s]
Adversarial training epochs: 100% 10/10 [26:08<00:00, 155.44s/it]
```

Выполняем атаку.

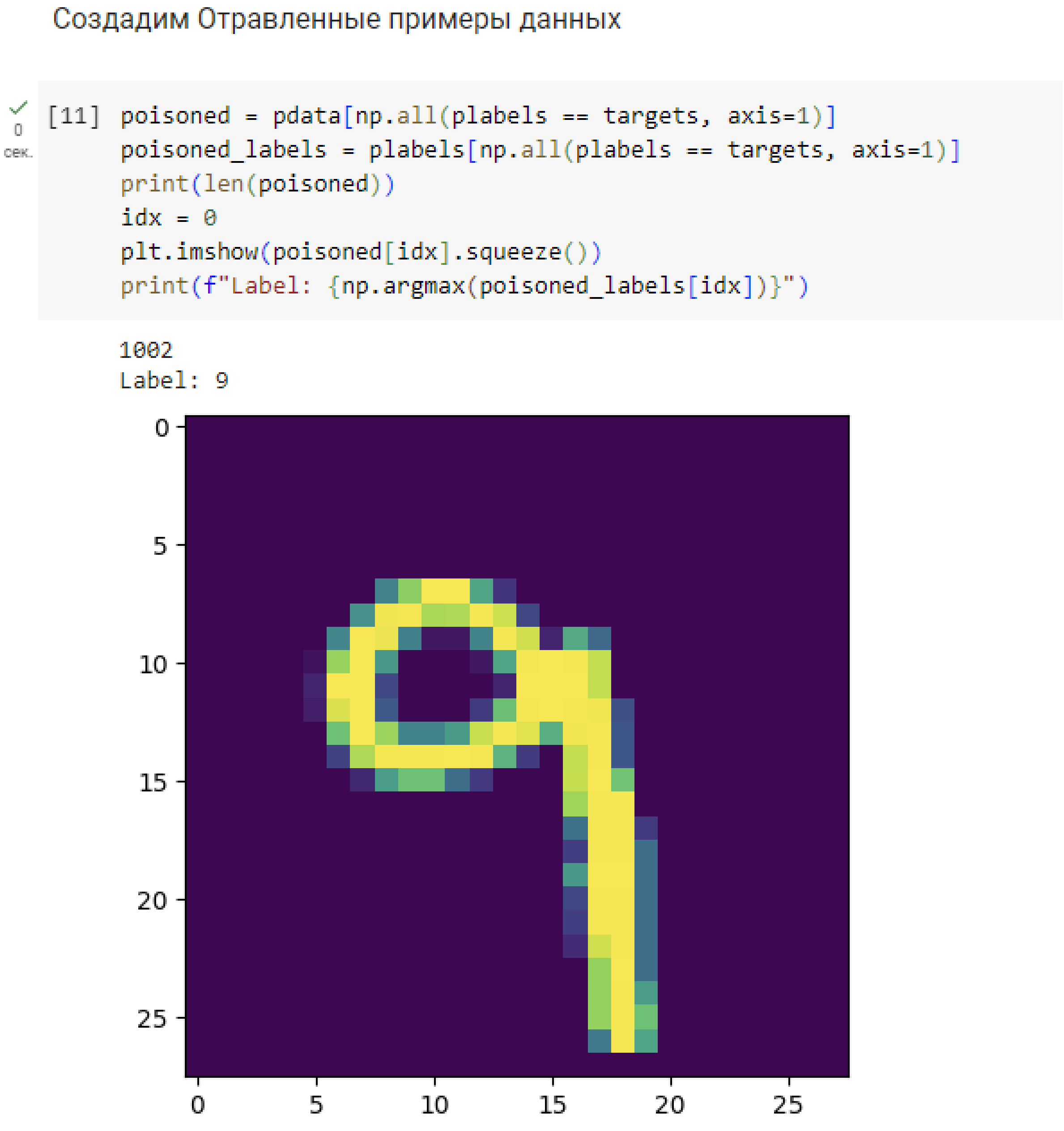


Выполним атаку

```
attack = PoisoningAttackCleanLabelBackdoor(backdoor=backdoor, proxy_classifier=proxy.get_classifier(), target=targets, pp_poison=percent_poison, norm=2, eps=5, eps_step=0.1, max_iter=200)
pdata, plabels = attack.poison(x_train, y_train)
```

```
PGD - Random Initializations: 100% 1/1 [00:12<00:00, 12.34s/it]
PGD - Random Initializations: 100% 1/1 [00:10<00:00, 10.77s/it]
PGD - Random Initializations: 100% 1/1 [00:11<00:00, 11.45s/it]
PGD - Random Initializations: 100% 1/1 [00:12<00:00, 12.43s/it]
PGD - Random Initializations: 100% 1/1 [00:11<00:00, 11.39s/it]
PGD - Random Initializations: 100% 1/1 [00:10<00:00, 10.94s/it]
PGD - Random Initializations: 100% 1/1 [00:12<00:00, 12.41s/it]
PGD - Random Initializations: 100% 1/1 [00:11<00:00, 11.70s/it]
PGD - Random Initializations: 100% 1/1 [00:10<00:00, 10.59s/it]
PGD - Random Initializations: 100% 1/1 [00:12<00:00, 12.42s/it]
PGD - Random Initializations: 100% 1/1 [00:03<00:00, 3.85s/it]
```

Создаем отравленные примеры данных.

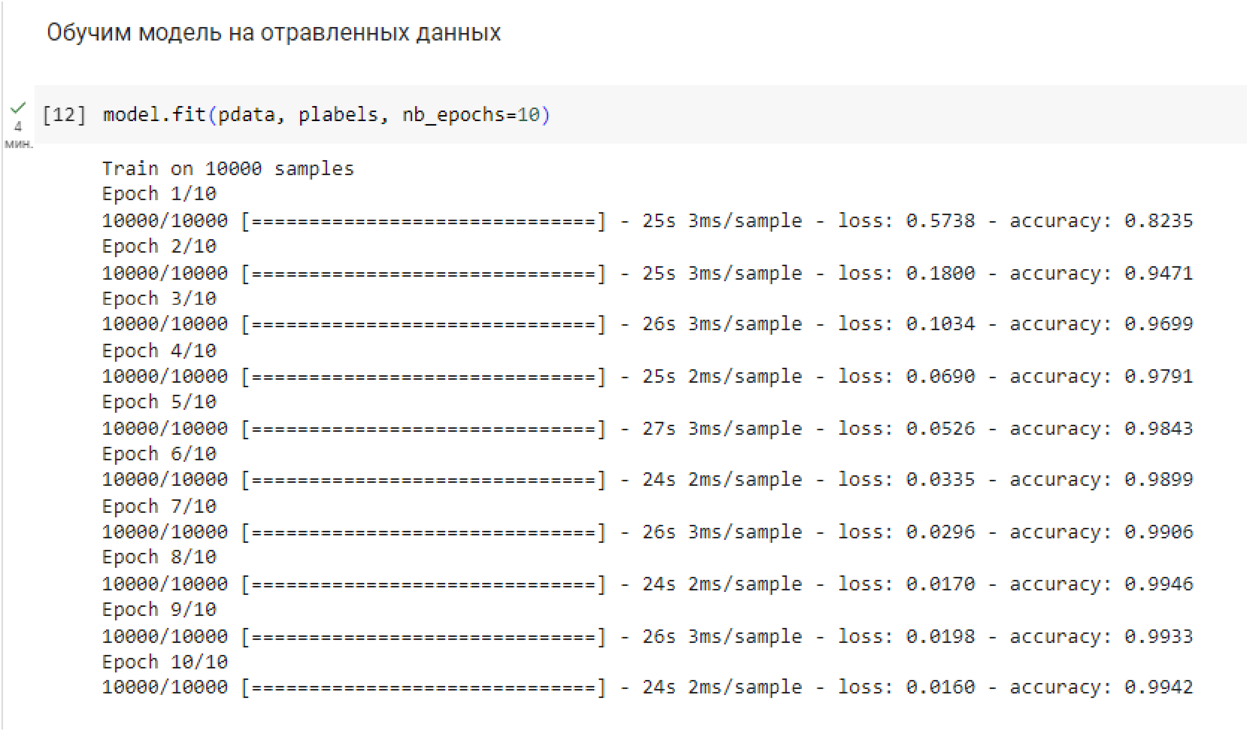


Создадим Отравленные примеры данных

```
[11] poisoned = pdata[np.all(plabels == targets, axis=1)]
     poisoned_labels = plabels[np.all(plabels == targets, axis=1)]
     print(len(poisoned))
     idx = 0
     plt.imshow(poisoned[idx].squeeze())
     print(f"Label: {np.argmax(poisoned_labels[idx])}")
```

```
1002
Label: 9
```

Обучаем модель на отравленных данных.

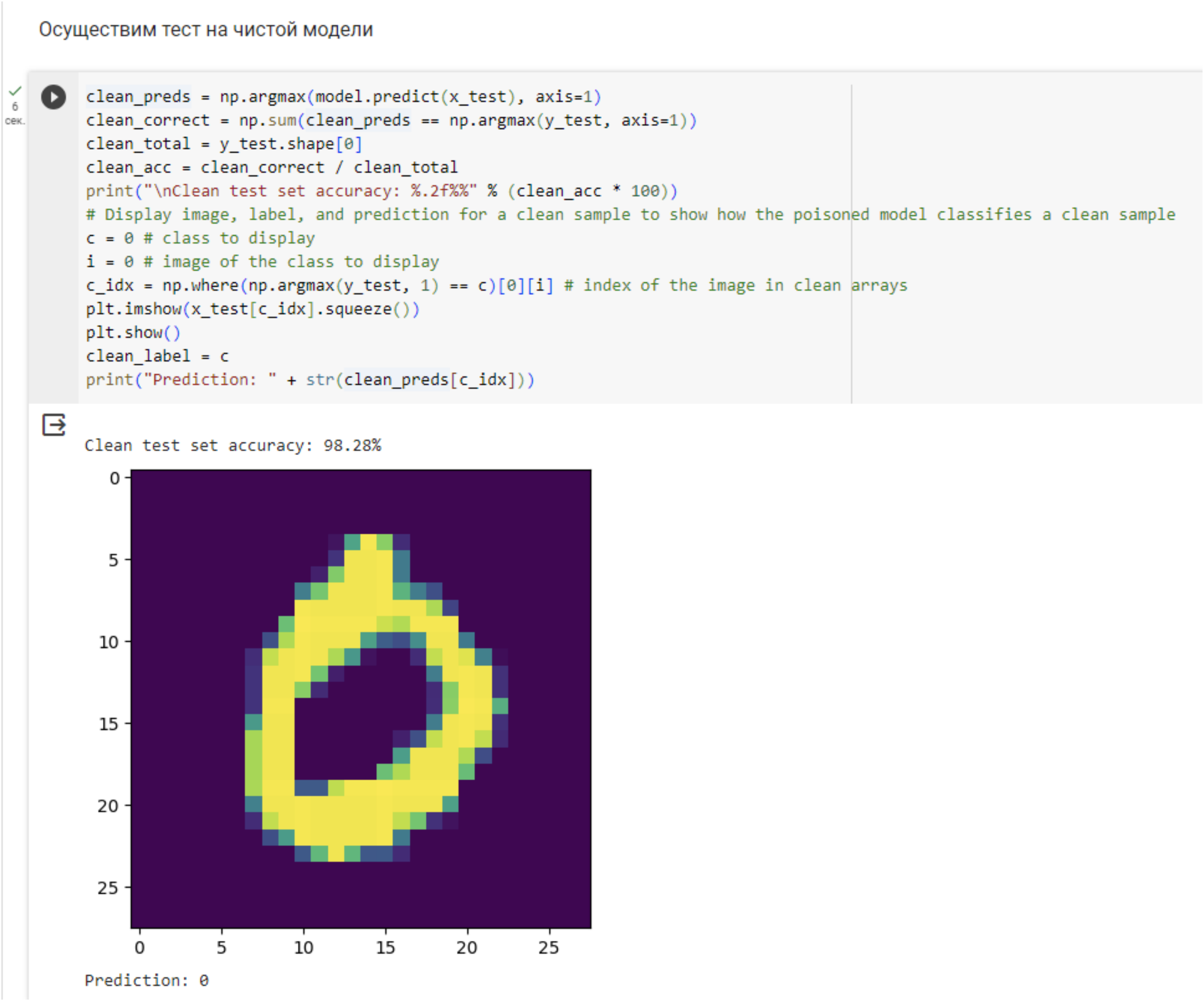


Обучим модель на отравленных данных

```
[12] model.fit(pdata, plabels, nb_epochs=10)
```

```
Train on 10000 samples
Epoch 1/10
10000/10000 [==============================] - 25s 3ms/sample - loss: 0.5738 - accuracy: 0.8235
Epoch 2/10
10000/10000 [==============================] - 25s 3ms/sample - loss: 0.1800 - accuracy: 0.9471
Epoch 3/10
10000/10000 [==============================] - 26s 3ms/sample - loss: 0.1034 - accuracy: 0.9699
Epoch 4/10
10000/10000 [==============================] - 25s 2ms/sample - loss: 0.0690 - accuracy: 0.9791
Epoch 5/10
10000/10000 [==============================] - 27s 3ms/sample - loss: 0.0526 - accuracy: 0.9843
Epoch 6/10
10000/10000 [==============================] - 24s 2ms/sample - loss: 0.0335 - accuracy: 0.9899
Epoch 7/10
10000/10000 [==============================] - 26s 3ms/sample - loss: 0.0296 - accuracy: 0.9906
Epoch 8/10
10000/10000 [==============================] - 24s 2ms/sample - loss: 0.0170 - accuracy: 0.9946
Epoch 9/10
10000/10000 [==============================] - 26s 3ms/sample - loss: 0.0198 - accuracy: 0.9933
Epoch 10/10
10000/10000 [==============================] - 24s 2ms/sample - loss: 0.0160 - accuracy: 0.9942
```

Осуществим тест на чистой модели.



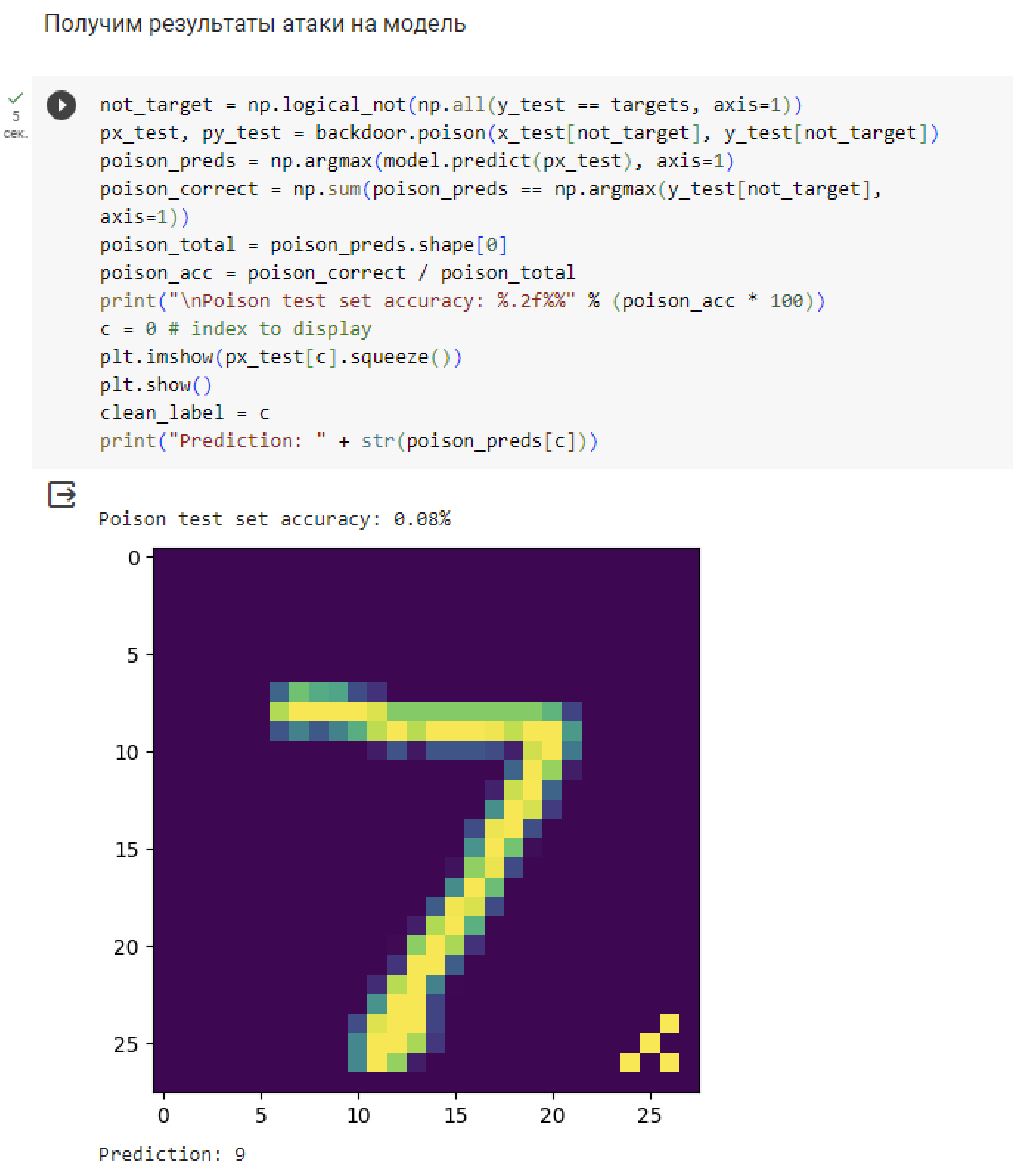
Осуществим тест на чистой модели

```
clean_preds = np.argmax(model.predict(x_test), axis=1)
clean_correct = np.sum(clean_preds == np.argmax(y_test, axis=1))
clean_total = y_test.shape[0]
clean_acc = clean_correct / clean_total
print("\nClean test set accuracy: %.2f%%" % (clean_acc * 100))
# Display image, label, and prediction for a clean sample to show how the poisoned model classifies a clean sample
c = 0 # class to display
i = 0 # image of the class to display
c_idx = np.where(np.argmax(y_test, 1) == c)[0][i] # index of the image in clean arrays
plt.imshow(x_test[c_idx].squeeze())
plt.show()
clean_label = c
print("Prediction: " + str(clean_preds[c_idx]))
```

```
Clean test set accuracy: 98.28%
```

```
Prediction: 0
```

Получим результаты атаки на модель. Увидим, что результат классификации искажен.



Получим результаты атаки на модель

```
not_target = np.logical_not(np.all(y_test == targets, axis=1))
px_test, py_test = backdoor.poison(x_test[not_target], y_test[not_target])
poison_preds = np.argmax(model.predict(px_test), axis=1)
poison_correct = np.sum(poison_preds == np.argmax(y_test[not_target],
axis=1))
poison_total = poison_preds.shape[0]
poison_acc = poison_correct / poison_total
print("\nPoison test set accuracy: %.2f%%" % (poison_acc * 100))
c = 0 # index to display
plt.imshow(px_test[c].squeeze())
plt.show()
clean_label = c
print("Prediction: " + str(poison_preds[c]))
```

```
Poison test set accuracy: 0.08%
```

```
Prediction: 9
```