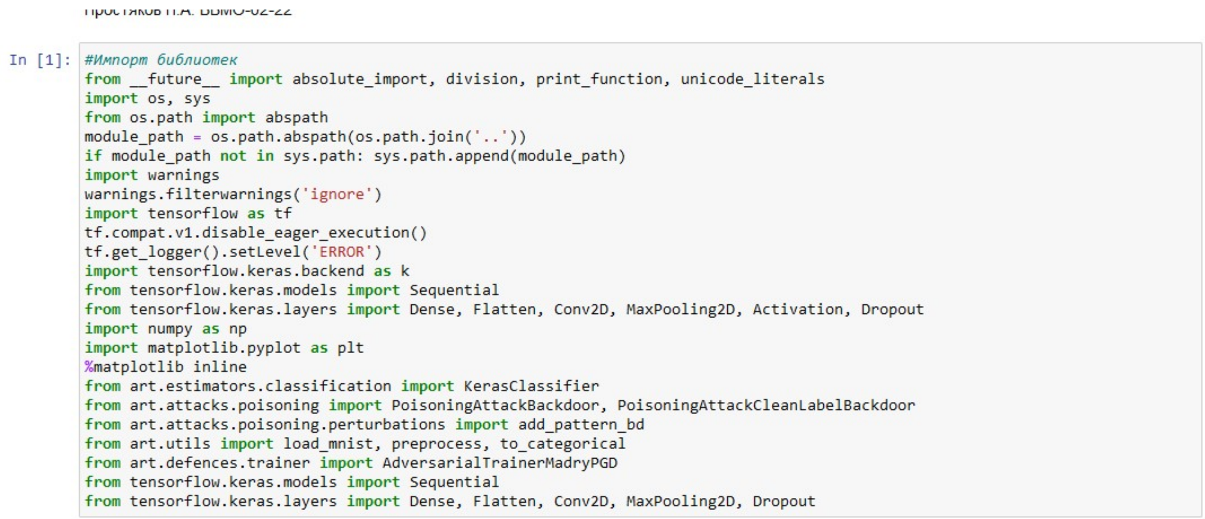
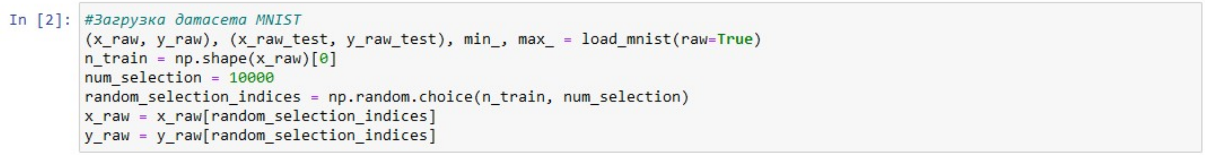
Практическая работа 4 Простяков Н.А. ББМО-02-22

Импорт библиотек и загрузка датасета



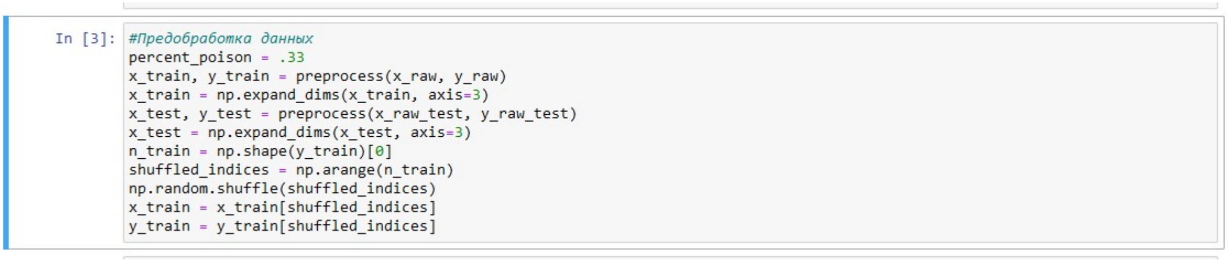
Простяков Н.А. ББМО-02-22

In [1]:
```
#Импорт библиотек
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
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout
```



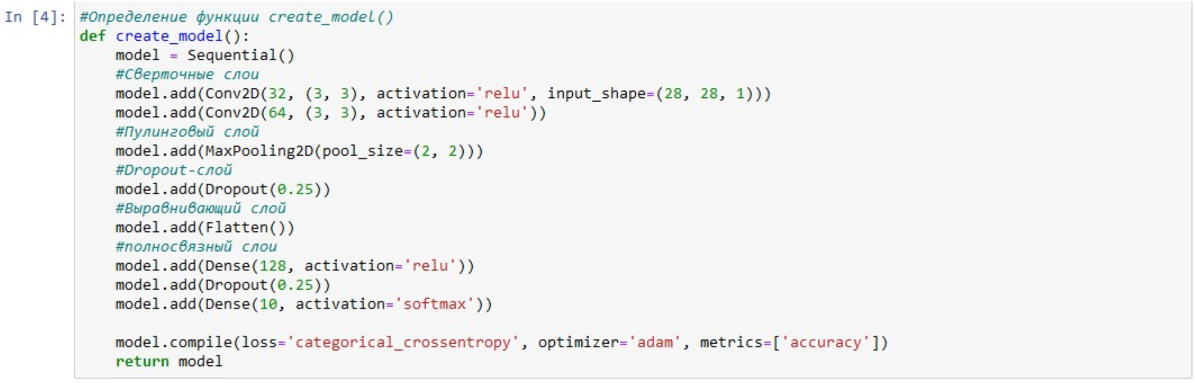
In [2]:
```
#Загрузка датасета MNIST
(x_raw, y_raw), (x_raw_test, y_raw_test), min_, max_ = load_mnist(raw=True)
n_train = np.shape(x_raw)[0]
num_selection = 10000
random_selection_indices = np.random.choice(n_train, num_selection)
x_raw = x_raw[random_selection_indices]
y_raw = y_raw[random_selection_indices]
```

Разбиение данных на выборки



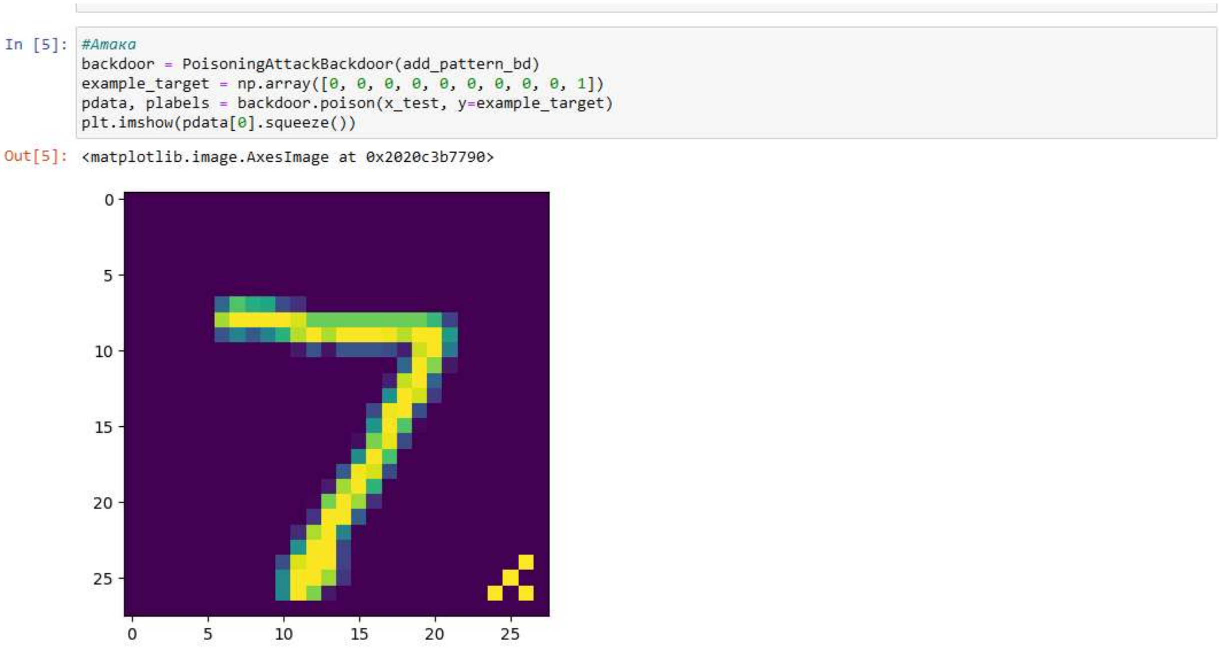
In [3]:
```
#Предобработка данных
percent_poison = .33
x_train, y_train = preprocess(x_raw, y_raw)
x_train = np.expand_dims(x_train, axis=3)
x_test, y_test = preprocess(x_raw_test, y_raw_test)
x_test = np.expand_dims(x_test, axis=3)
n_train = np.shape(y_train)[0]
shuffled_indices = np.arange(n_train)
np.random.shuffle(shuffled_indices)
x_train = x_train[shuffled_indices]
y_train = y_train[shuffled_indices]
```

Определение функции create_model для создания модели



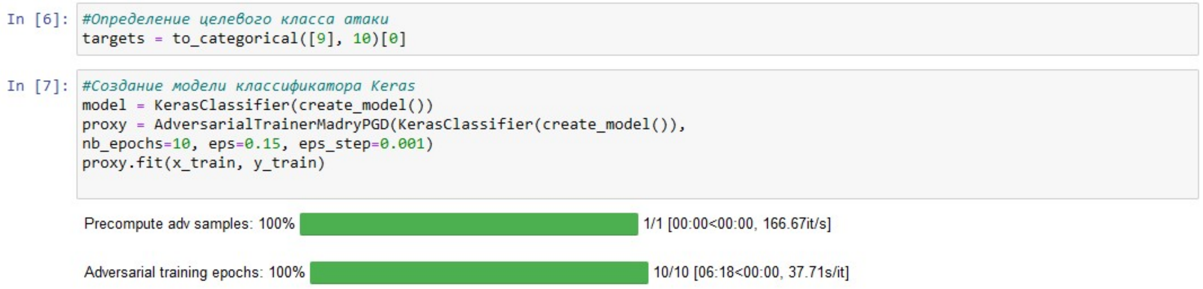
In [4]:
```
#Определение функции create_model()
def create_model():
    model = Sequential()
    #Сверточные слои
    model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    #Пулинговый слой
    model.add(MaxPooling2D(pool_size=(2, 2)))
    #Dropout-слой
    model.add(Dropout(0.25))
    #Выравнивающий слой
    model.add(Flatten())
    #полносвязный слои
    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
```

Реализация атаки: приведение примера атаки и отравление данных

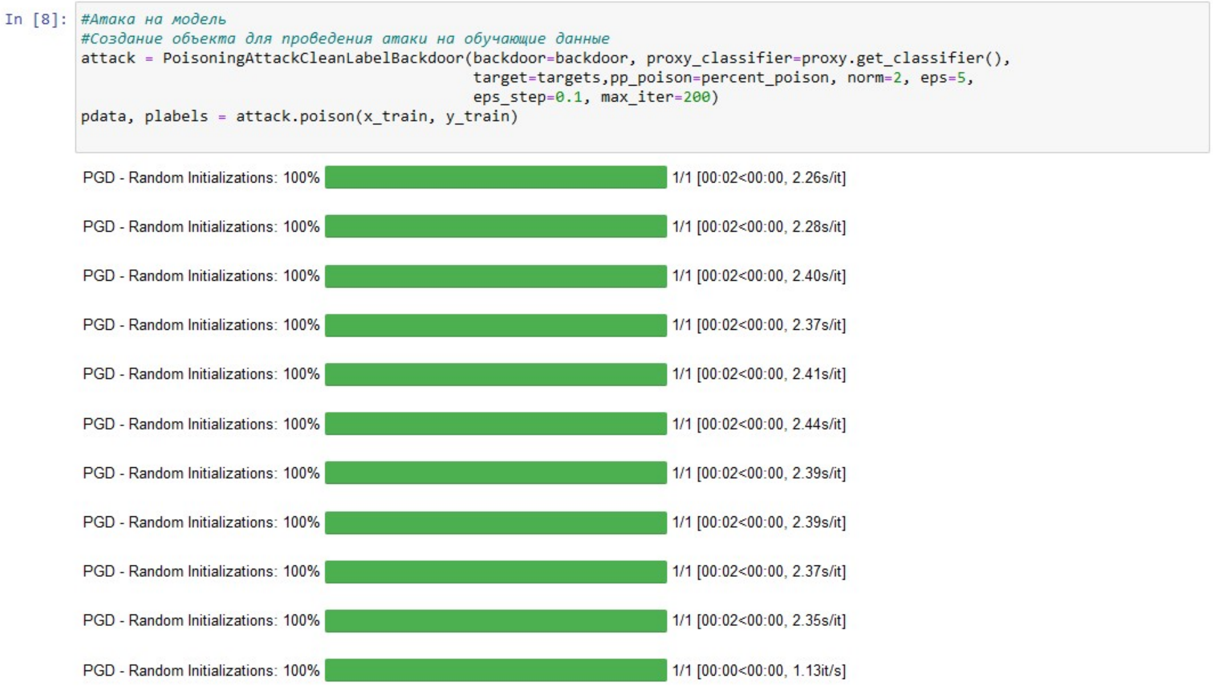


```
In [5]: #Атака
        backdoor = PoisoningAttackBackdoor(add_pattern_bd)
        example_target = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0, 1])
        pdata, plabels = backdoor.poison(x_test, y=example_target)
        plt.imshow(pdata[0].squeeze())

Out[5]: <matplotlib.image.AxesImage at 0x2020c3b7790>
```

Создание модели и ее обучение с учетом проведенной атаки

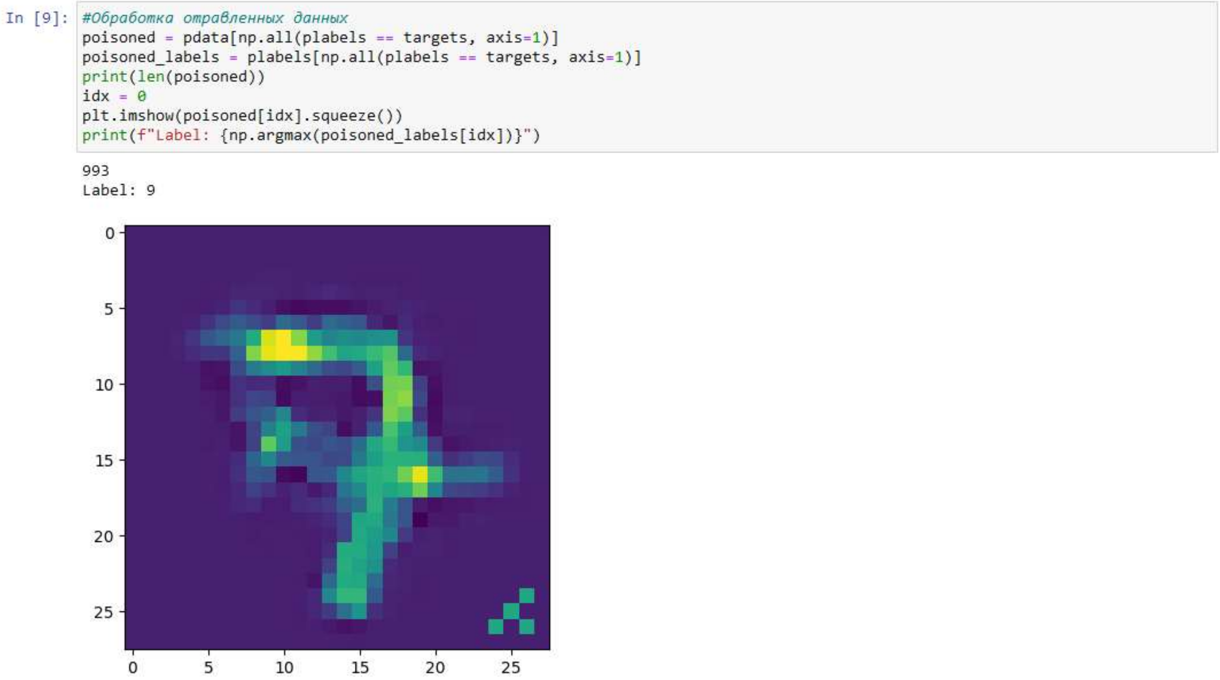


```
In [6]: #Определение целевого класса атаки
        targets = to_categorical([9], 10)[0]

In [7]: #Создание модели классификатора Keras
        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, 166.67it/s]

        Adversarial training epochs: 100%  10/10 [06:18<00:00, 37.71s/it]
```



```
In [8]: #Атака на модель
        #Создание объекта для проведения атаки на обучающие данные
        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 [00:02<00:00, 2.26s/it]

        PGD - Random Initializations: 100%  1/1 [00:02<00:00, 2.28s/it]

        PGD - Random Initializations: 100%  1/1 [00:02<00:00, 2.40s/it]

        PGD - Random Initializations: 100%  1/1 [00:02<00:00, 2.37s/it]

        PGD - Random Initializations: 100%  1/1 [00:02<00:00, 2.41s/it]

        PGD - Random Initializations: 100%  1/1 [00:02<00:00, 2.44s/it]

        PGD - Random Initializations: 100%  1/1 [00:02<00:00, 2.39s/it]

        PGD - Random Initializations: 100%  1/1 [00:02<00:00, 2.39s/it]

        PGD - Random Initializations: 100%  1/1 [00:02<00:00, 2.37s/it]

        PGD - Random Initializations: 100%  1/1 [00:02<00:00, 2.35s/it]

        PGD - Random Initializations: 100%  1/1 [00:00<00:00, 1.13it/s]
```

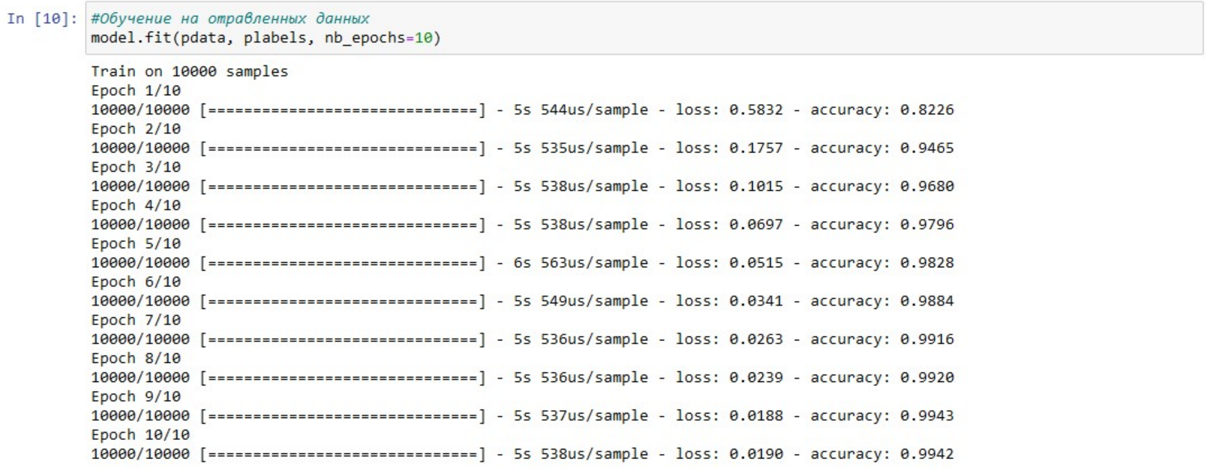
Обработка отравленных данных после проведения атаки



```
In [9]: #Обработка отравленных данных
        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])}")
```

```
993
Label: 9
```

Обучение модели на отравленных данных



```
In [10]: #Обучение на отравленных данных
         model.fit(pdata, plabels, nb_epochs=10)
```

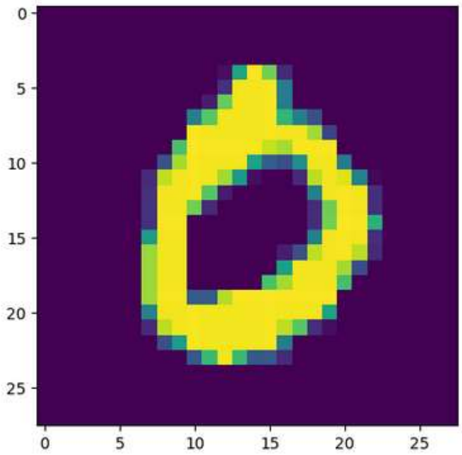
```
Train on 10000 samples
Epoch 1/10
10000/10000 [==============================] - 5s 544us/sample - loss: 0.5832 - accuracy: 0.8226
Epoch 2/10
10000/10000 [==============================] - 5s 535us/sample - loss: 0.1757 - accuracy: 0.9465
Epoch 3/10
10000/10000 [==============================] - 5s 538us/sample - loss: 0.1015 - accuracy: 0.9680
Epoch 4/10
10000/10000 [==============================] - 5s 538us/sample - loss: 0.0697 - accuracy: 0.9796
Epoch 5/10
10000/10000 [==============================] - 6s 563us/sample - loss: 0.0515 - accuracy: 0.9828
Epoch 6/10
10000/10000 [==============================] - 5s 549us/sample - loss: 0.0341 - accuracy: 0.9884
Epoch 7/10
10000/10000 [==============================] - 5s 536us/sample - loss: 0.0263 - accuracy: 0.9916
Epoch 8/10
10000/10000 [==============================] - 5s 536us/sample - loss: 0.0239 - accuracy: 0.9920
Epoch 9/10
10000/10000 [==============================] - 5s 537us/sample - loss: 0.0188 - accuracy: 0.9943
Epoch 10/10
10000/10000 [==============================] - 5s 538us/sample - loss: 0.0190 - accuracy: 0.9942
```

Проверка на чистой модели

In [11]:

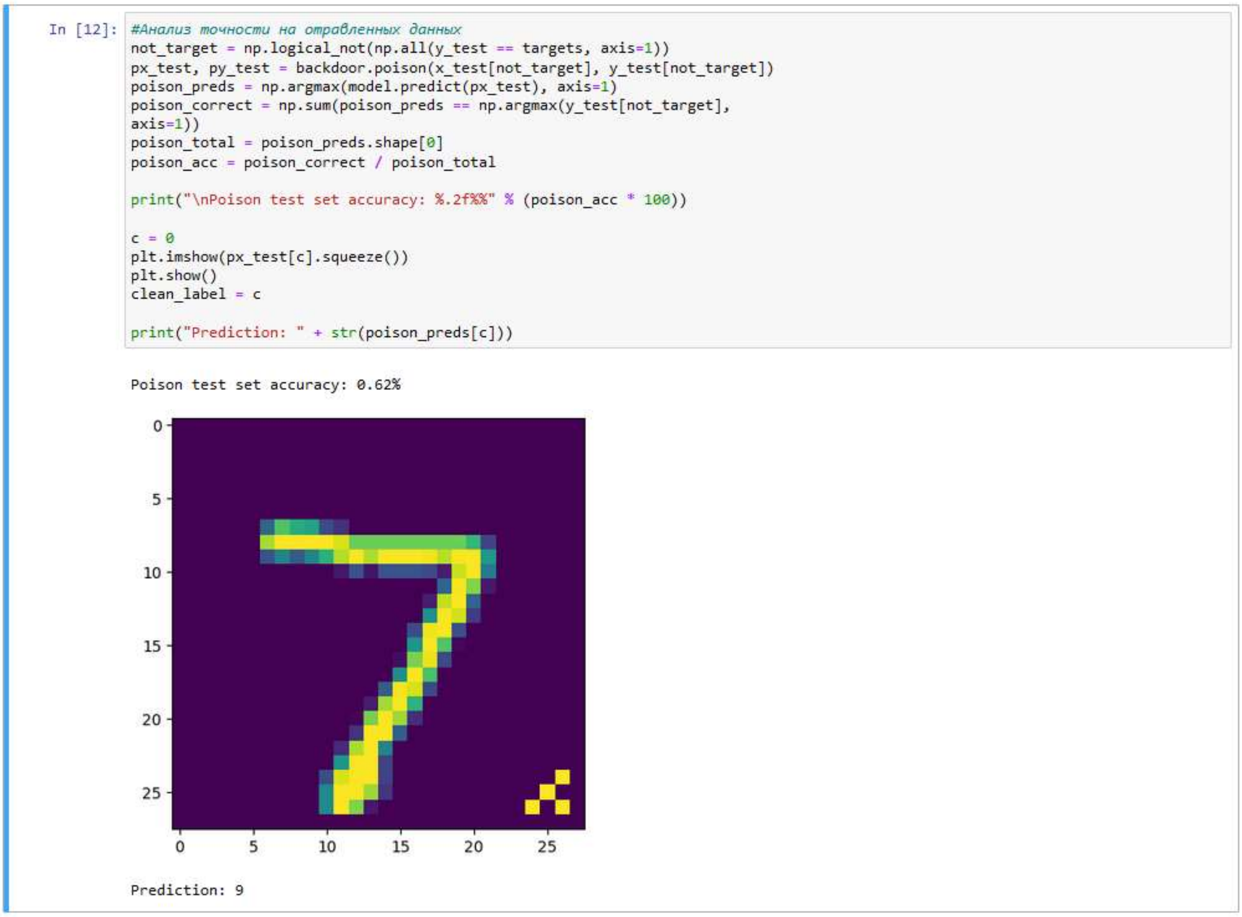
```
#Проверка на чистой модели
clean_preds = np.argmax(model.predict(x_test), axis=1)
#Корректное число предсказанных классов
clean_correct = np.sum(clean_preds == np.argmax(y_test, axis=1))
clean_total = y_test.shape[0]
clean_acc = clean_correct / clean_total
print("\nClean test set accuracy: %.2f%%" % (clean_acc * 100))
c = 0
i = 0
c_idx = np.where(np.argmax(y_test, 1) == c)[0][i] #
plt.imshow(x_test[c_idx].squeeze())
plt.show()
clean_label = c
print("Prediction: " + str(clean_preds[c_idx]))
```

```
Clean test set accuracy: 98.21%
```



```
Prediction: 0
```

Проверка на отравленных данных



In [12]:

```
#Анализ точности на отравленных данных
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
plt.imshow(px_test[c].squeeze())
plt.show()
clean_label = c

print("Prediction: " + str(poison_preds[c]))
```

```
Poison test set accuracy: 0.62%
```

```
Prediction: 9
```