**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ**

**Отчет к лабораторной работе №9**

**Дисциплины «Классификатор на основе Dense-слоев.»**

**Выполнили студенты группы А-13-22**

***Кокляева Мария и Сафонова София***

**Преподаватель: *Бартеньев О.В.***

**Москва 2024 г.**

1. ***Описание MNIST***

MNIST - объёмная база данных образцов рукописного написания цифр (белым цветом, на черном фоне).Размер каждой картинки 28\*28 = пикселей.

**Информация о наборе данных в указанном формате:**

*Количество экземпляров: 70 000 изображений*

*Количество атрибутов: 784 (28x28 пикселей)*

*Цель: Столбец представляет цифру (0-9), соответствующую рукописному изображению*

*Пиксель 1-784: Значение каждого пикселя (0-255) представляет интенсивность оттенков серого соответствующего пикселя на изображении.*

*Набор данных разделен на два основных подмножества:*

*Обучающий набор: Состоит из 60 000 изображений вместе с их метками, обычно используемых для обучения моделей машинного обучения.*

*Набор тестов: Содержит 10 000 изображений с соответствующими им метками, используемых для оценки производительности обученных моделей.*

# ***2.Примеры изображений из ОМ и ПМ***

Для вывода моделей из проверочных и обучающих файлов написан следующий код:

def plot\_images(images, labels, num\_images=5):

plt.figure(figsize=(10, 2)) # Размер фигуры увеличен для более удобного отображения

for i in range(1, num\_images+1):

plt.subplot(1,5,i)

plt.imshow(images[i-1].reshape(img\_rows, img\_cols), cmap='gray')

plt.title(labels[i-1])

plt.axis('off')

plt.show()

# Вывод первых 5 изображений из обучающего набора

print("OM")

for i in range (5):

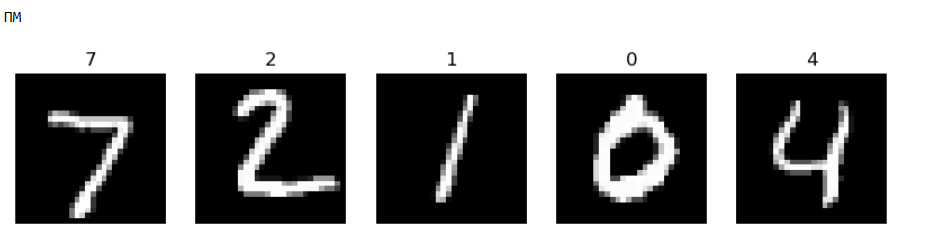
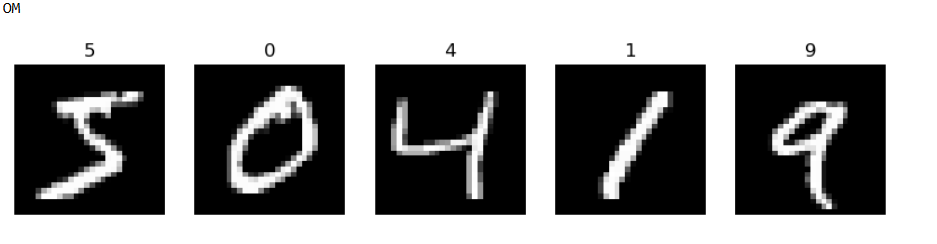
plot\_images(x\_trn, y\_trn)

print("ПМ")

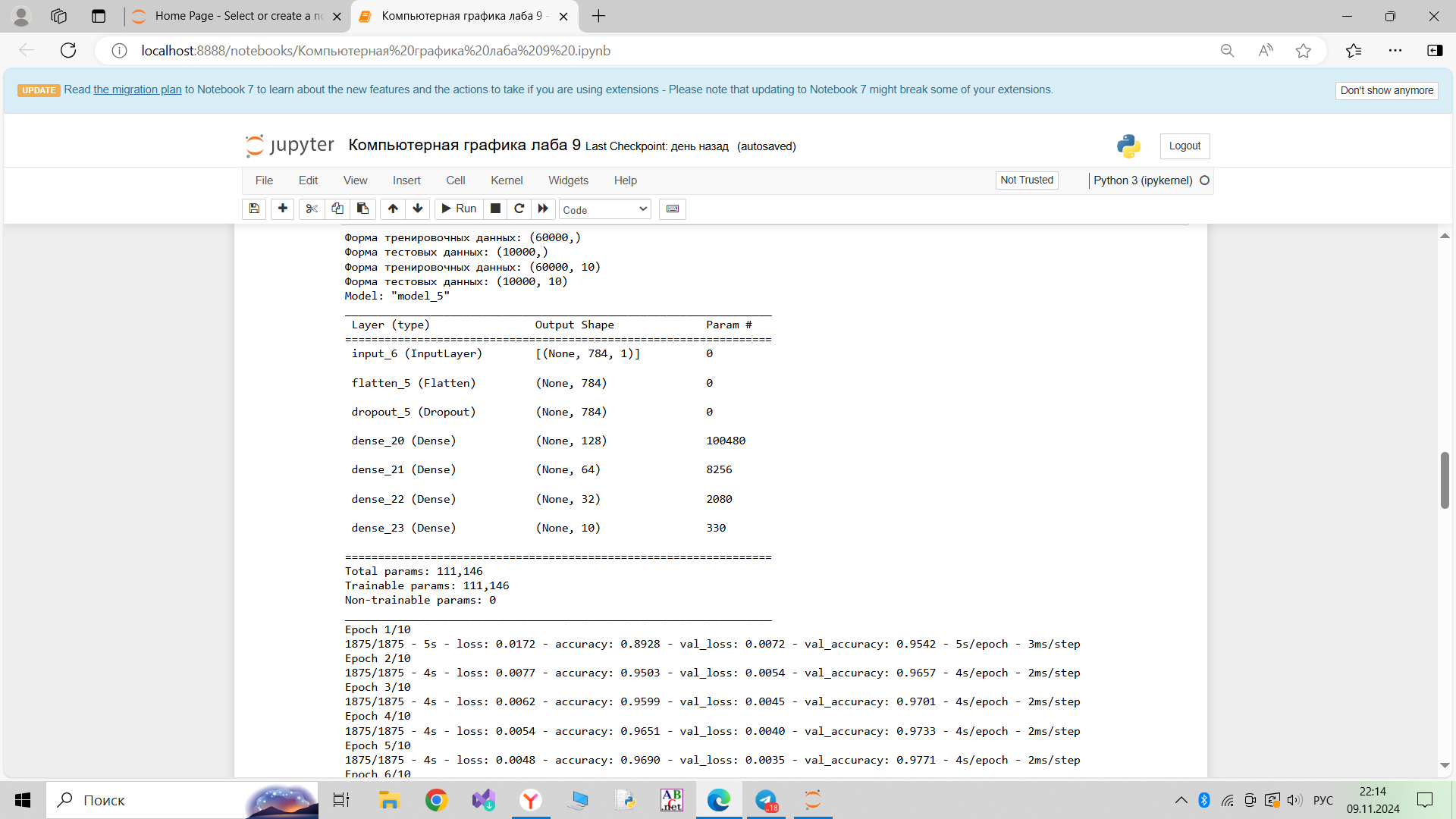
# Вывод первых 5 изображений из обучающего набора

for i in range (5):

plot\_images(x\_tst, y\_tst)



# ***3. model.summary()***



# ***4.  Графики истории обучения.***

Графики обучения лучшего по точности на оценочной выборке варианта нейронной сети.

Для вывода истории точности был написан следующий код:

*# Вывод истории обучения*

*def one\_plot(n, y\_lb, loss\_acc, val\_loss\_acc):*

*plt.subplot(1, 2, n)*

*if n == 1:*

*lb, lb2 = 'loss', 'val\_loss'*

*yMin = 0*

*yMax = 1.05 \* max(max(loss\_acc), max(val\_loss\_acc))*

*else:*

*lb, lb2 = 'acc', 'val\_acc'*

*yMin = min(min(loss\_acc), min(val\_loss\_acc))*

*yMax = 1.0*

*plt.plot(loss\_acc, color = 'r', label = lb, linestyle = '--')*

*plt.plot(val\_loss\_acc, color = 'g', label = lb2)*

*plt.ylabel(y\_lb)*

*plt.xlabel('Эпоха')*

*plt.ylim([0.95 \* yMin, yMax])*

*plt.legend()*

*# История обучения; вывод графиков обучения*

*history = h.history*

*plt.figure(figsize = (9, 4))*

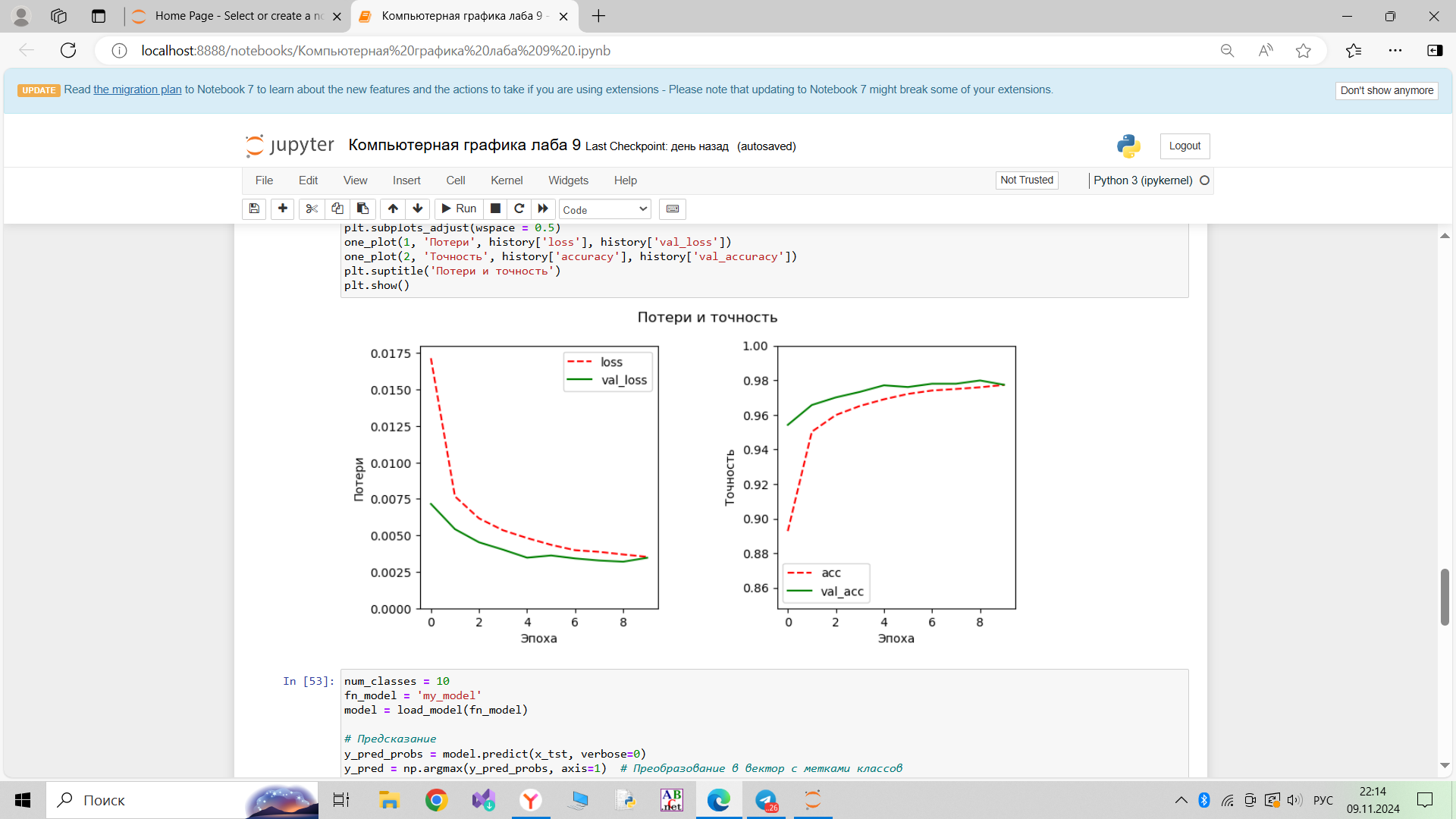
*plt.subplots\_adjust(wspace = 0.5)*

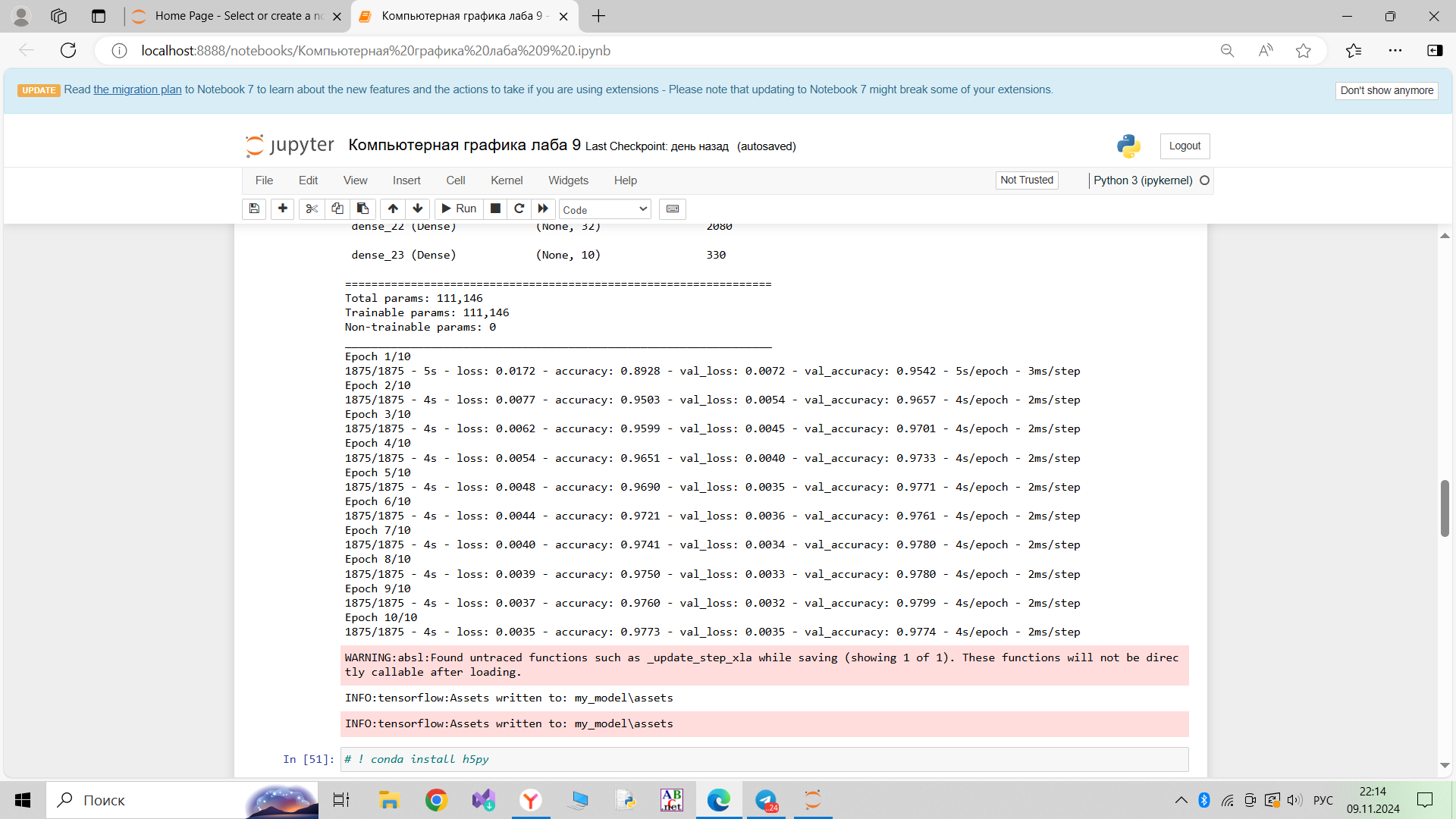
*one\_plot(1, 'Потери', history['loss'], history['val\_loss'])*

*one\_plot(2, 'Точность', history['accuracy'], history['val\_accuracy'])*

*plt.suptitle('Потери и точность')*

*plt.show()*





# ***5.*** ***Результат classification\_report.***

***В данном «куске» кода, происходит следующее:***

***1.Создание модели***

***2.Обучение на тестовых примерах***

***3.Проверка***

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import classification\_report

*linear\_regressor = LinearRegression()*

*linear\_regressor.fit(x\_trn, y\_trn)*

*y\_true = y\_tst*

*y\_pred = linear\_regressor.predict(x\_tst).astype('int32')*

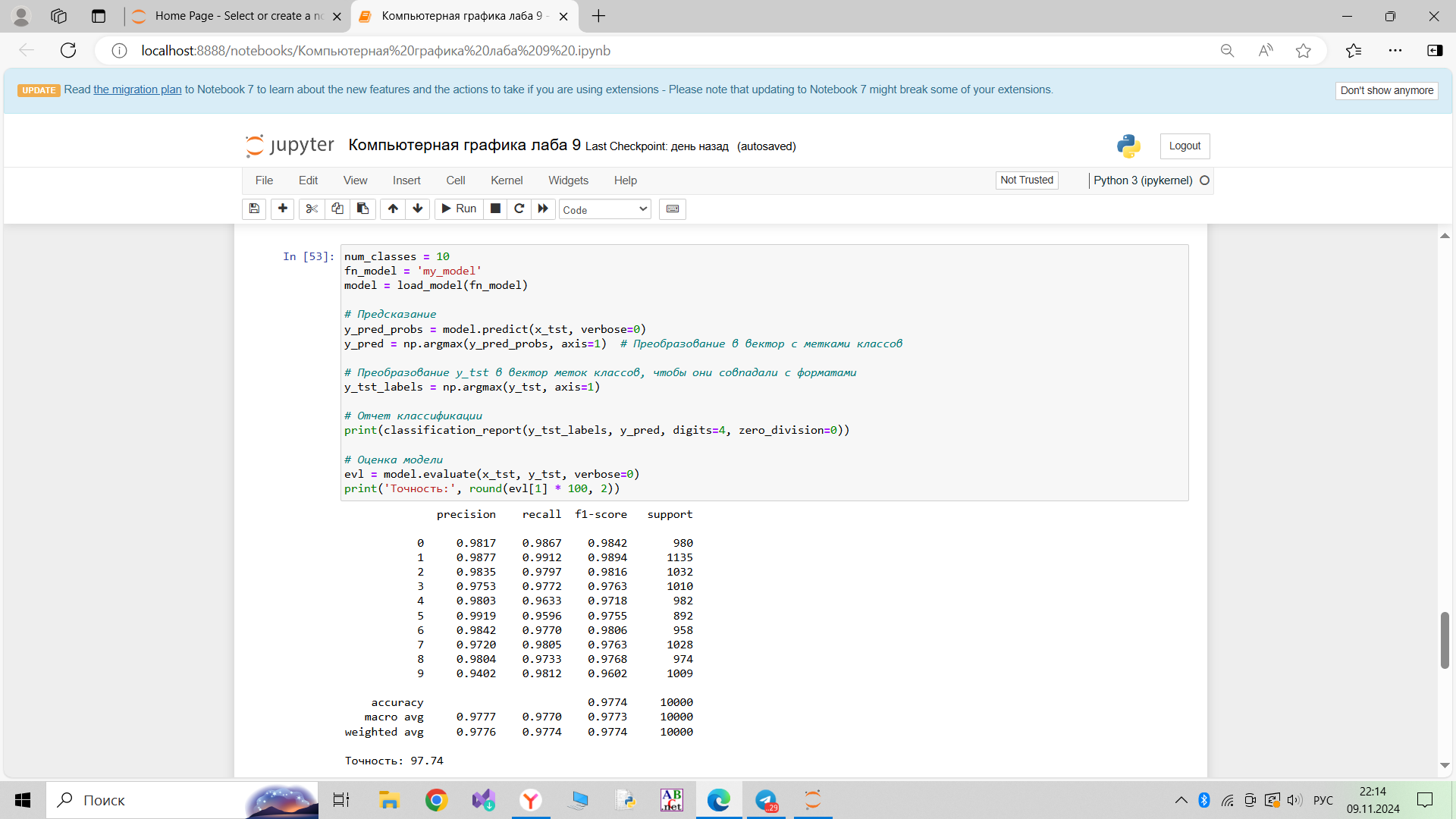
*unique\_classes = np.unique(y\_tst)*

*print(y\_pred)*

*target\_names = [f'class {i}' for i in unique\_classes]*

*print(classification\_report(y\_tst, y\_pred, target\_names=target\_names, labels=unique\_classes))*

***Результат:***



# ***6. Свой код вычисления precision, recall и F1.***

*def precision\_recall\_f1(y\_true, y\_pred, average='macro'):*

*# Найти уникальные классы*

*unique\_classes = np.unique(np.concatenate((y\_true, y\_pred)))*

*precisions = []*

*recalls = []*

*f1\_scores = []*

*for cls in unique\_classes:*

*# True Positives*

*TP = np.sum((y\_pred == cls) & (y\_true == cls))*

*# False Positives*

*FP = np.sum((y\_pred == cls) & (y\_true != cls))*

*# False Negatives*

*FN = np.sum((y\_pred != cls) & (y\_true == cls))*

*precision = TP / (TP + FP) if (TP + FP) > 0 else 0*

*recall = TP / (TP + FN) if (TP + FN) > 0 else 0*

*f1 = (2 \* precision \* recall) / (precision + recall) if (precision + recall) > 0 else 0*

*precisions.append(precision)*

*recalls.append(recall)*

*f1\_scores.append(f1)*

*if average == 'macro':*

*# Макроусреднение*

*precision = np.mean(precisions)*

*recall = np.mean(recalls)*

*f1 = np.mean(f1\_scores)*

*elif average == 'micro':*

*# Микроусреднение, суммируя все TP, FP и FN для всех классов*

*TP = np.sum([(y\_pred == cls) & (y\_true == cls) for cls in unique\_classes])*

*FP = np.sum([(y\_pred == cls) & (y\_true != cls) for cls in unique\_classes])*

*FN = np.sum([(y\_pred != cls) & (y\_true == cls) for cls in unique\_classes])*

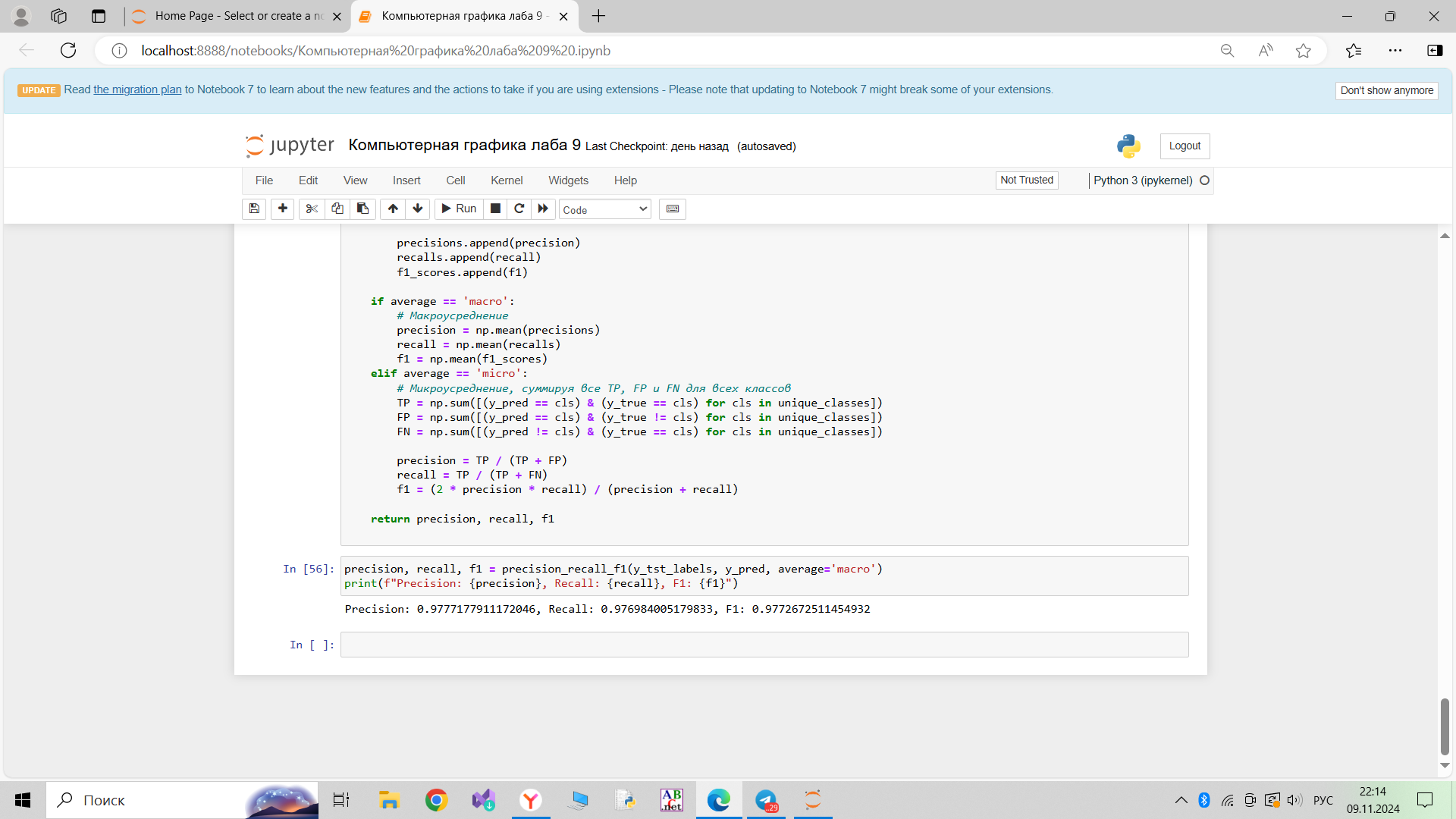
*precision = TP / (TP + FP)*

*recall = TP / (TP + FN)*

*f1 = (2 \* precision \* recall) / (precision + recall)*

*return precision, recall, f1*

**Результат:**

****