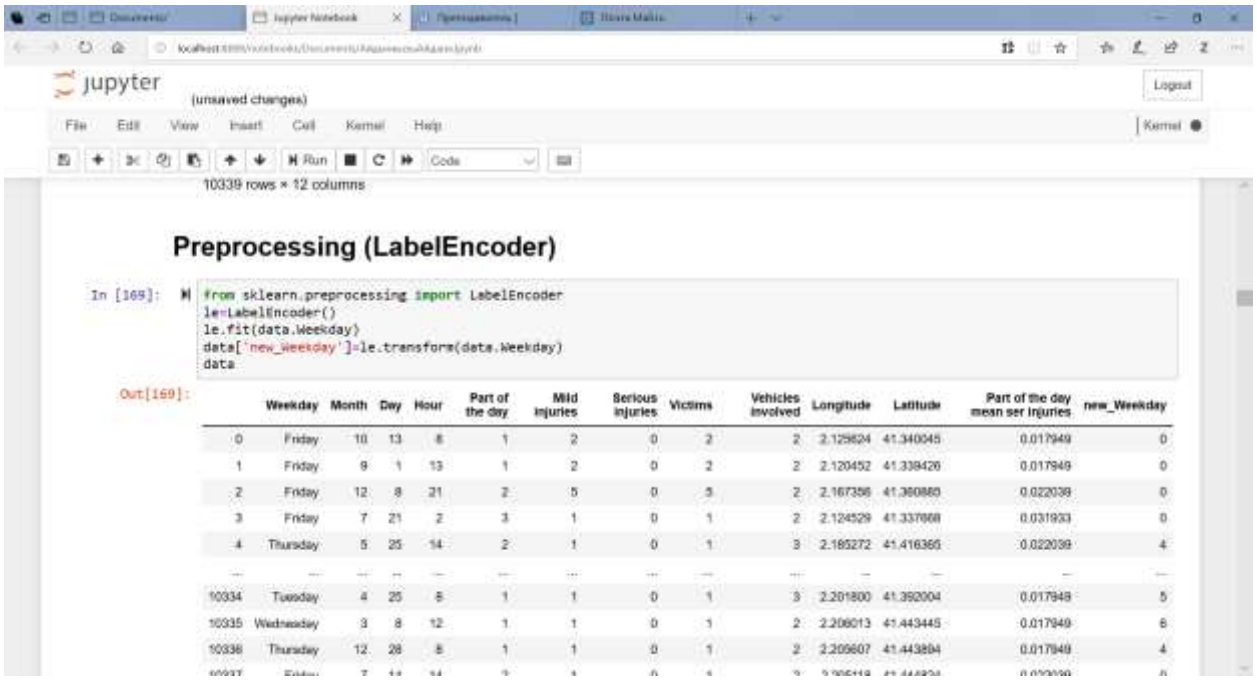
Лабораторное занятие 14. Обучение категориальным признакам. Label Encoding. One-Hot Encoding.

14-зертханалық сабақ. Категориялық белгілермен оқыту. Label Encoding. One-Hot Encoding..

Python және Anaconda да мысал. Сіз өз деректер қорына жасаныз!

Пример на Python и Anaconde. Ниже приведен пример. Каждый должен сделать для своего датасета.



jupyter (unsaved changes)

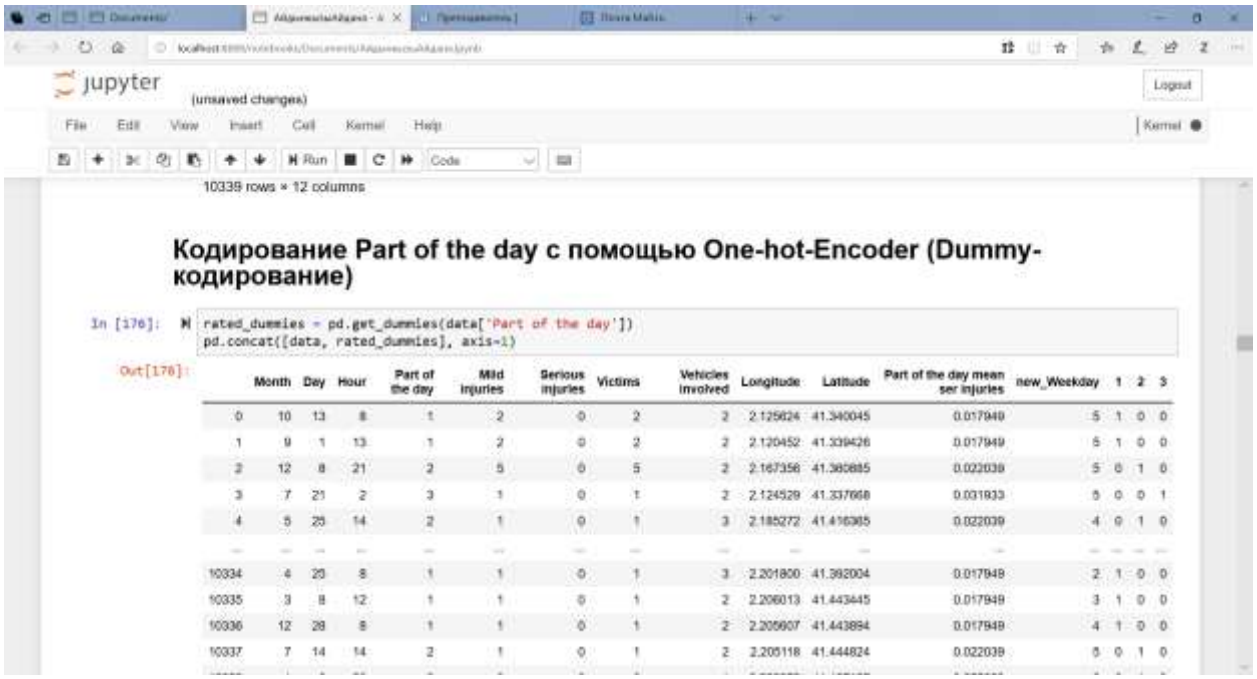
10339 rows × 12 columns

Preprocessing (LabelEncoder)

```
In [169]: from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
le.fit(data.Weekday)
data['new_Weekday']=le.transform(data.Weekday)
data
```

Out[169]:

	Weekday	Month	Day	Hour	Part of the day	Mild injuries	Serious injuries	Victims	Vehicles involved	Longitude	Latitude	Part of the day mean ser injuries	new_Weekday
0	Friday	10	13	8	1	2	0	2	2	2.125824	41.340045	0.017949	0
1	Friday	9	1	13	1	2	0	2	2	2.120452	41.339426	0.017949	0
2	Friday	12	8	21	2	5	0	5	2	2.167356	41.380885	0.022039	0
3	Friday	7	21	2	3	1	0	1	2	2.124529	41.337668	0.031933	0
4	Thursday	5	25	14	2	1	0	1	3	2.185272	41.416365	0.022039	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...
10334	Tuesday	4	25	8	1	1	0	1	3	2.201800	41.392004	0.017949	5
10335	Wednesday	3	8	12	1	1	0	1	2	2.206013	41.443445	0.017949	6
10336	Thursday	12	28	8	1	1	0	1	2	2.205607	41.443894	0.017949	4



jupyter (unsaved changes)

10339 rows × 12 columns

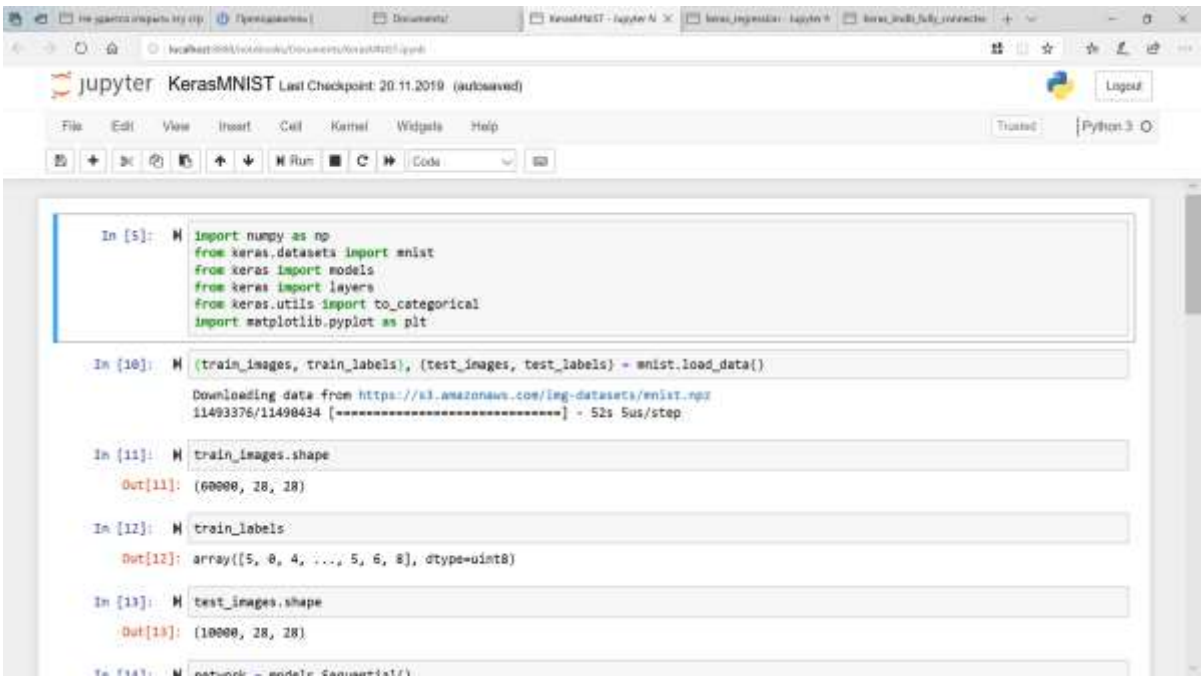
Кодирование Part of the day с помощью One-hot-Encoder (Dummy-кодирование)

```
In [176]: rated_dummies = pd.get_dummies(data['Part of the day'])
pd.concat([data, rated_dummies], axis=1)
```

Out[176]:

	Month	Day	Hour	Part of the day	Mild injuries	Serious injuries	Victims	Vehicles involved	Longitude	Latitude	Part of the day mean ser injuries	new_Weekday	1	2	3
0	10	13	8	1	2	0	2	2	2.125824	41.340045	0.017949	5	1	0	0
1	9	1	13	1	2	0	2	2	2.120452	41.339426	0.017949	5	1	0	0
2	12	8	21	2	5	0	5	2	2.167356	41.380885	0.022039	5	0	1	0
3	7	21	2	3	1	0	1	2	2.124529	41.337668	0.031933	5	0	0	1
4	5	25	14	2	1	0	1	3	2.185272	41.416365	0.022039	4	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10334	4	25	8	1	1	0	1	3	2.201800	41.392004	0.017949	2	1	0	0
10335	3	8	12	1	1	0	1	2	2.206013	41.443445	0.017949	3	1	0	0
10336	12	28	8	1	1	0	1	2	2.205607	41.443894	0.017949	4	1	0	0
10337	7	14	14	2	1	0	1	2	2.205118	41.444824	0.022039	5	0	1	0

Нейрондық желіге мысал. Пример на нейронную сеть!

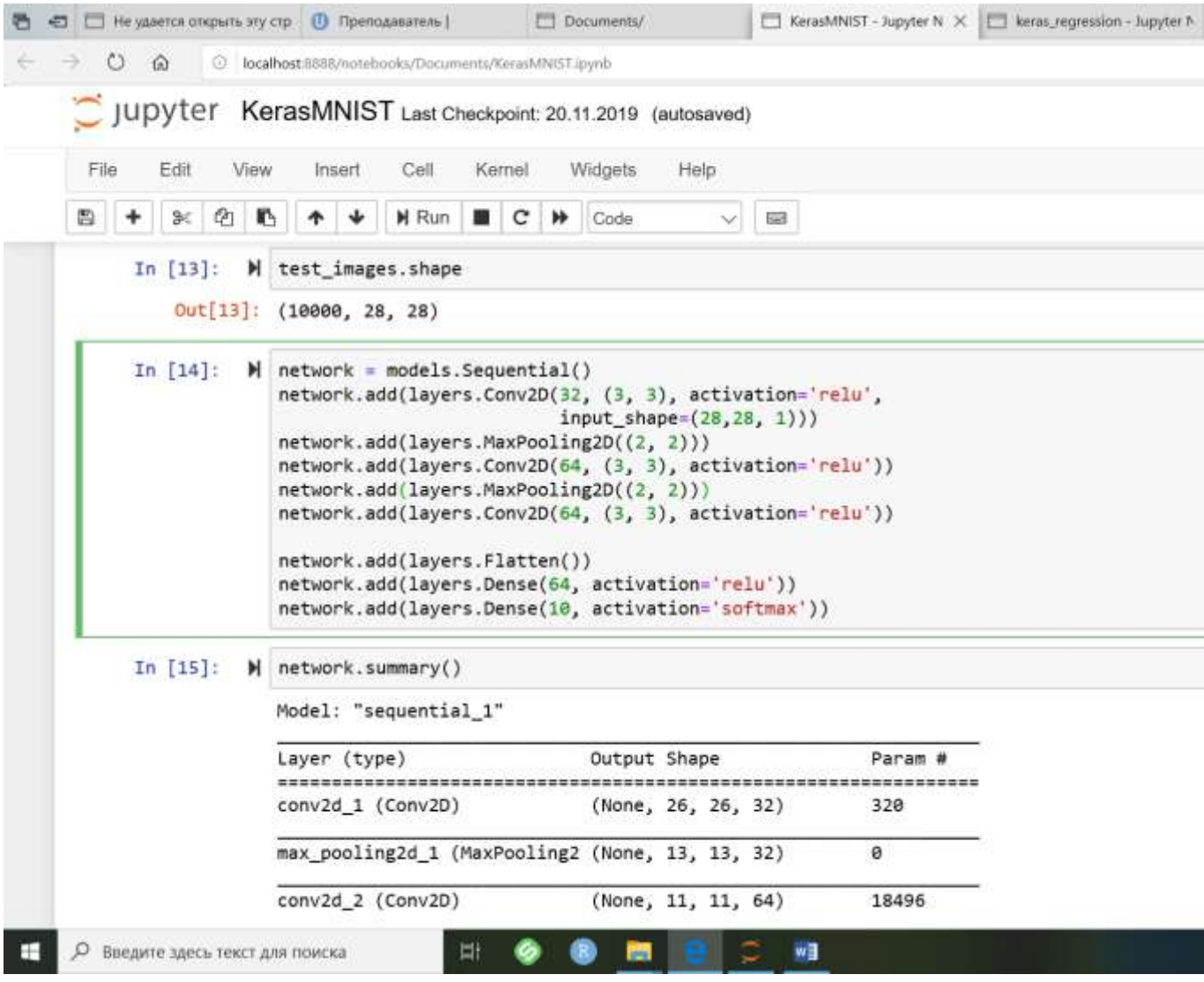


jupyter KerasMNIST Last Checkpoint: 20.11.2019 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
In [5]: import numpy as np
        from keras.datasets import mnist
        from keras import models
        from keras import layers
        from keras.utils import to_categorical
        import matplotlib.pyplot as plt

In [10]: (train_images, train_labels), (test_images, test_labels) = mnist.load_data()

Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 52s 5us/step

In [11]: train_images.shape
Out[11]: (60000, 28, 28)

In [12]: train_labels
Out[12]: array([5, 0, 4, ..., 5, 6, 8], dtype=uint8)

In [13]: test_images.shape
Out[13]: (10000, 28, 28)
```



Не удается открыть эту стр | Преподаватель | | Documents/ | KerasMNIST - Jupyter N | keras_regression - Jupyter N

localhost:8888/notebooks/Documents/KerasMNIST.ipynb

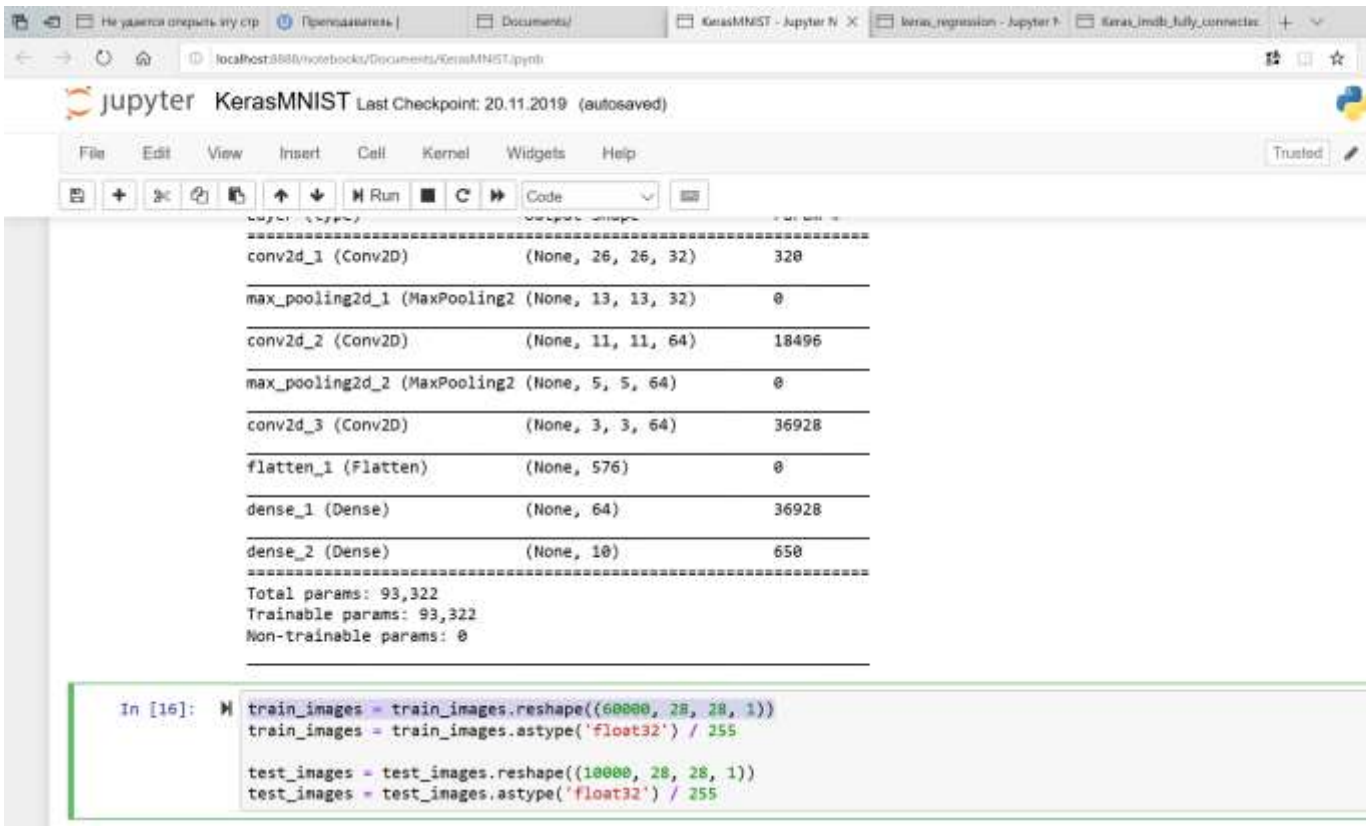
jupyter KerasMNIST Last Checkpoint: 20.11.2019 (autosaved)

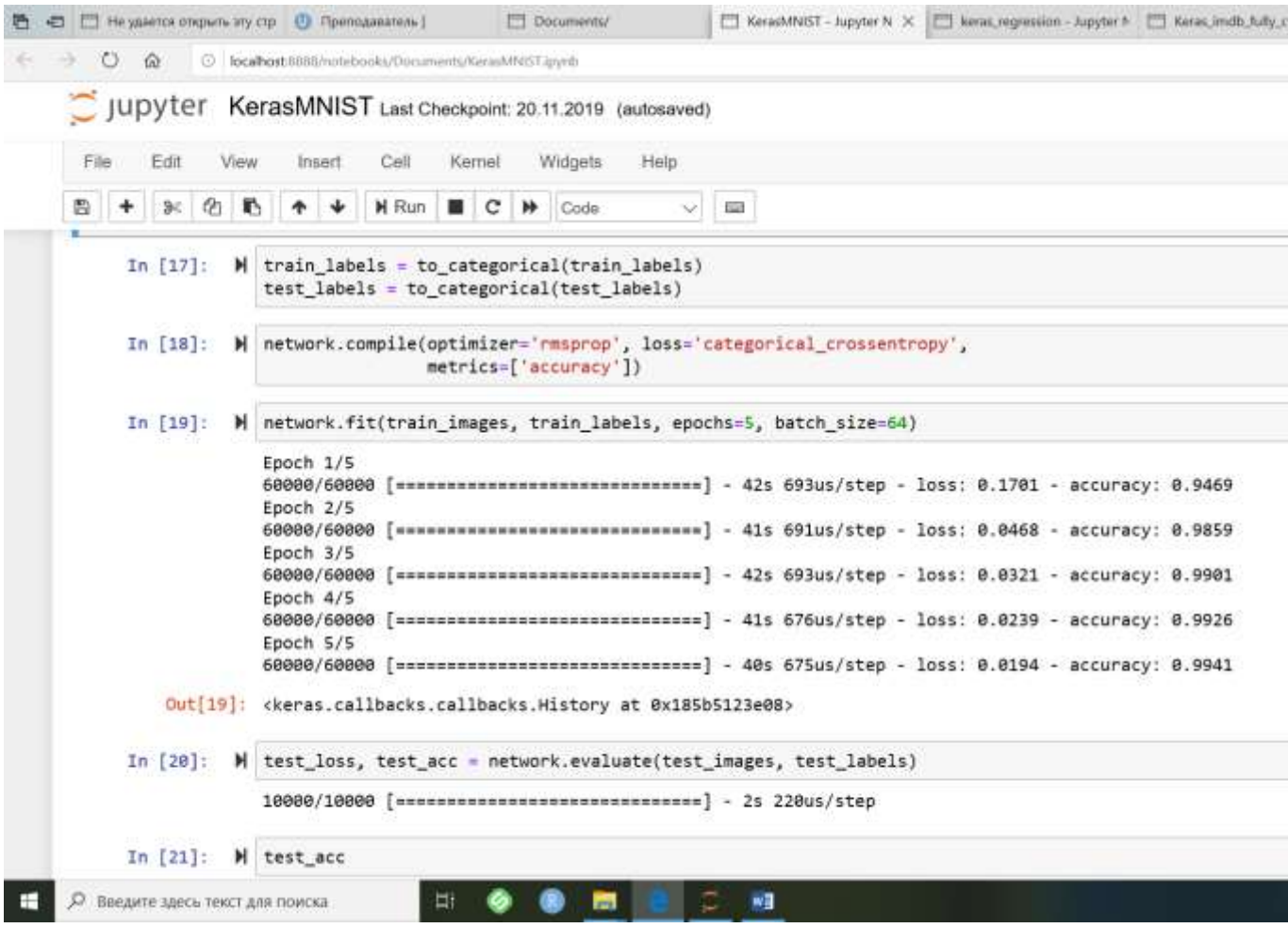
File Edit View Insert Cell Kernel Widgets Help

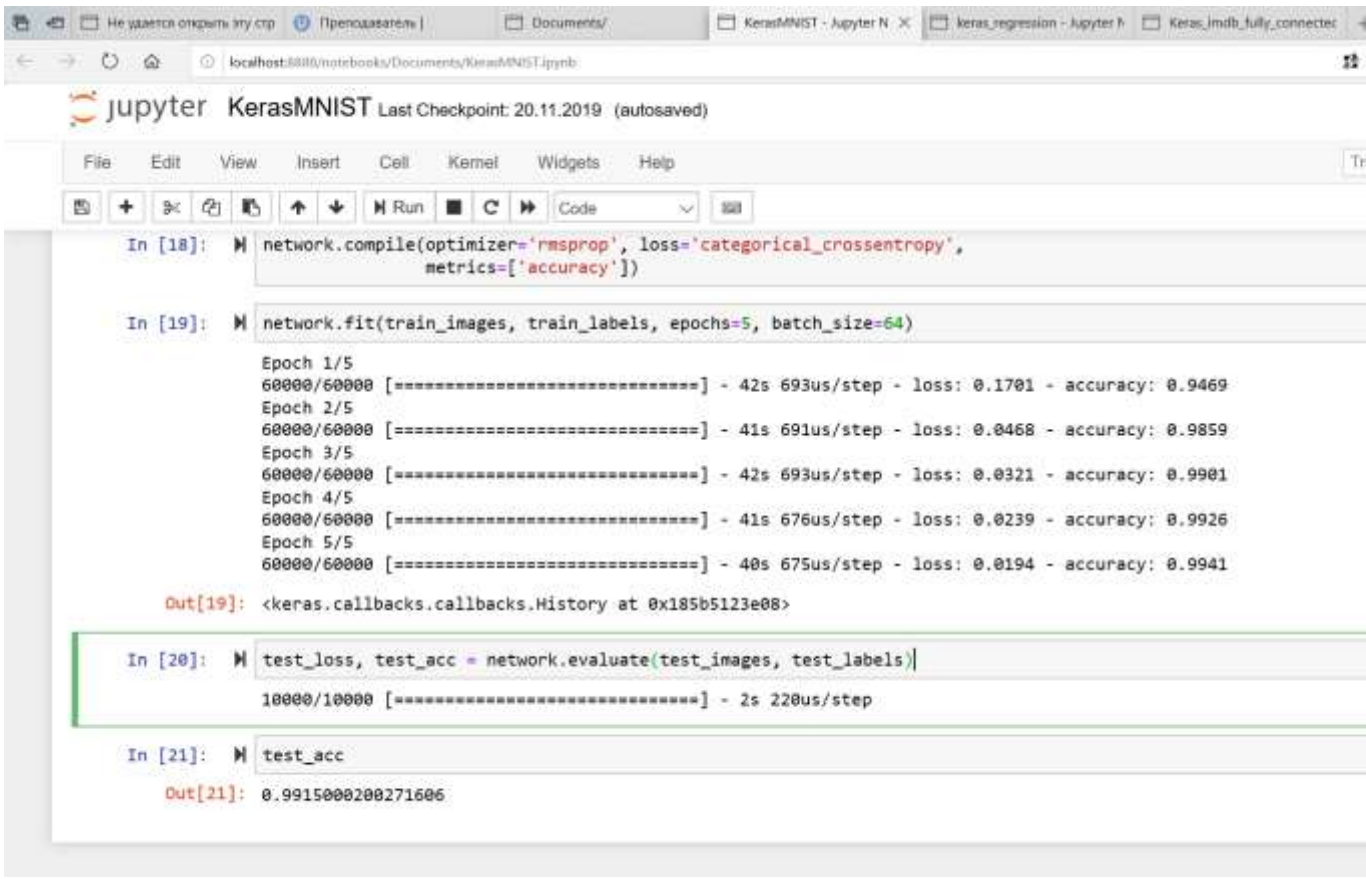
Run Code

```
In [13]: test_images.shape
Out[13]: (10000, 28, 28)

In [14]: network = models.Sequential()
         network.add(layers.Conv2D(32, (3, 3), activation='relu',
                                   input_shape=(28,28, 1)))
         network.add(layers.MaxPooling2D((2, 2)))
         network.add(layers.Conv2D(64, (3, 3), activation='relu'))
         network.add(layers.MaxPooling2D((2, 2)))
         network.add(layers.Conv2D(64, (3, 3), activation='relu'))

         network.add(layers.Flatten())
         network.add(layers.Dense(64, activation='relu'))
         network.add(layers.Dense(10, activation='softmax'))

In [15]: network.summary()
         Model: "sequential_1"
         _________________________________________________________________
         Layer (type)                 Output Shape              Param #
         =================================================================
         conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
         _________________________________________________________________
         max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32)        0
         _________________________________________________________________
         conv2d_2 (Conv2D)            (None, 11, 11, 64)        18496
```

Введите здесь текст для поиска


jupyter KerasMNIST Last Checkpoint: 20.11.2019 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run Code
conv2d_1 (Conv2D) (None, 26, 26, 32) 320
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32) 0
conv2d_2 (Conv2D) (None, 11, 11, 64) 18496
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 64) 0
conv2d_3 (Conv2D) (None, 3, 3, 64) 36928
flatten_1 (Flatten) (None, 576) 0
dense_1 (Dense) (None, 64) 36928
dense_2 (Dense) (None, 10) 650
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
In [16]:
train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype('float32') / 255
test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype('float32') / 255


jupyter KerasMNIST Last Checkpoint: 20.11.2019 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run Code
In [17]:
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
In [18]:
network.compile(optimizer='rmsprop', loss='categorical_crossentropy',
metrics=['accuracy'])
In [19]:
network.fit(train_images, train_labels, epochs=5, batch_size=64)
Epoch 1/5
60000/60000 [==========] - 42s 693us/step - loss: 0.1701 - accuracy: 0.9469
Epoch 2/5
60000/60000 [==========] - 41s 691us/step - loss: 0.0468 - accuracy: 0.9859
Epoch 3/5
60000/60000 [==========] - 42s 693us/step - loss: 0.0321 - accuracy: 0.9901
Epoch 4/5
60000/60000 [==========] - 41s 676us/step - loss: 0.0239 - accuracy: 0.9926
Epoch 5/5
60000/60000 [==========] - 40s 675us/step - loss: 0.0194 - accuracy: 0.9941
Out[19]: <keras.callbacks.callbacks.History at 0x185b5123e08>
In [20]:
test_loss, test_acc = network.evaluate(test_images, test_labels)
10000/10000 [==========] - 2s 220us/step
In [21]:
test_acc
Введите здесь текст для поиска


jupyter KerasMNIST Last Checkpoint: 20.11.2019 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run Code
In [18]:
network.compile(optimizer='rmsprop', loss='categorical_crossentropy',
metrics=['accuracy'])
In [19]:
network.fit(train_images, train_labels, epochs=5, batch_size=64)
Epoch 1/5
60000/60000 [==========] - 42s 693us/step - loss: 0.1701 - accuracy: 0.9469
Epoch 2/5
60000/60000 [==========] - 41s 691us/step - loss: 0.0468 - accuracy: 0.9859
Epoch 3/5
60000/60000 [==========] - 42s 693us/step - loss: 0.0321 - accuracy: 0.9901
Epoch 4/5
60000/60000 [==========] - 41s 676us/step - loss: 0.0239 - accuracy: 0.9926
Epoch 5/5
60000/60000 [==========] - 40s 675us/step - loss: 0.0194 - accuracy: 0.9941
Out[19]:
<keras.callbacks.callbacks.History at 0x185b5123e08>
In [20]:
test_loss, test_acc = network.evaluate(test_images, test_labels)
10000/10000 [==========] - 2s 220us/step
In [21]:
test_acc
Out[21]:
0.9915000200271606