Digit Recognizer Summary

Nicholas Bergeland

MSDS 422, Dr. Fulton

Summary:

This week for our assignment I built an algorithm trained to recognize images of numerical digits 1-10. The model I built tested at 99.175% recognition, which was good enough for 378 / 1806 (top 21%).

The model was built and trained with a combination of matplotlib, numpy, seaborne, tenorflow, SKlearn, and Keras. I first read in the testing and training datasets for the various models to work with. From this point, I began understanding the dataset with some basic EDA.

To perform the basic analysis, I used a graph within matplotlib to see the occurrence of different numbers. Is the dataset normally distributed? After observing the normality (shape) of the dataset, I got to work. The work involved some simple encoding, then creation of different layers of the CNN model.

The thing I was not prepared for was how time intensive the model was. Each different visualization model took 45-60 minutes to run through all of the sequences of testing. I found it fascinating to watch the model simulate through “epochs” of data and continually improve itself with no guidance provided by me.

The result was just as impressive. With no structure provided by me, the model was recognizing nearly every number it was presented with. One could argue the 99 + % success rate may be below a human, which would likely get all of the numbers. However, with no prior learning and in such a short period, it is truly shocking the success the model was able to develop given the timeframe.

Despite placing in the top 25% of this challenge, I will look to improve my model moving forward to next week. Having seen scores testing at 100% recognition, I will set my sights on this mark as I develop my models further.

Kaggle Score:

Unfortunately, despite the confusion matrix running over 90% accuracy it was not accepted by Kaggle. Therefore my score is the same as last week, but my work may be found below.

Graphical user interface, text, application

Description automatically generated

Appendix:

Out[2]:

In [1]:

**import**

**numpy**

**as**

**np**

**import**

**pandas**

**as**

**pd**

**import**

**matplotlib.pyplot**

**as**

**plt**

%

**matplotlib**

inline

**import**

**seaborn**

**as**

**sns**

**import**

**tensorflow**

**as**

**tf**

In [2]:

train

=

pd

.

read\_csv

(

"train2.csv"

)

test

=

pd

.

read\_csv

(

"test2.csv"

)

train

.

head

()

**label**

**pixel0**

**pixel1**

**pixel2**

**pixel3**

**pixel4**

**pixel5**

**pixel6**

**pixel7**

**pixel8**

**...**

**pixel774**

**pixel775**

1. 1 0 0 0 0 0 0 0 0 0 ... 0 0
2. 0 0 0 0 0 0 0 0 0 0 ... 0 0
3. 1 0 0 0 0 0 0 0 0 0 ... 0 0
4. 4 0 0 0 0 0 0 0 0 0 ... 0 0
5. 0 0 0 0 0 0 0 0 0 0 ... 0 0
6. rows × 785 columns

In [3]: y\_train = train['label'].astype('float32')

X\_train = train.drop(['label'], axis=1).astype('int32')

X\_test = test.astype('float32')

X\_train.shape, y\_train.shape, X\_test.shape

Out[3]: ((42000, 784), (42000,), (28000, 784))

In [4]:

sns

.

countplot

(

x

=

'label'

,

data

=

train

;

)

In [5]:

*# Data normalization*

X\_train

=

X\_train

/

255

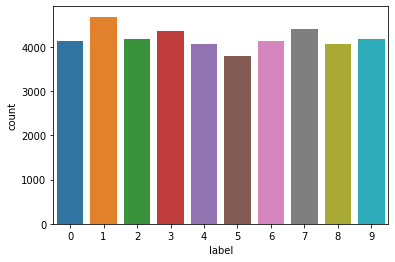
X\_test

=

X\_test

/

255



In [6]: X\_train = X\_train.values.reshape(-1,28,28,1)

X\_test = X\_test.values.reshape(-1,28,28,1)

X\_train.shape, X\_test.shape

Out[6]: ((42000, 28, 28, 1), (28000, 28, 28, 1))

In [7]: *# one-hot encoding* **from** **keras.utils.np\_utils** **import** to\_categorical y\_train = to\_categorical(y\_train, num\_classes = 10) y\_train.shape

Using TensorFlow backend.

Out[7]: (42000, 10)

In [8]:

print

(

train

[

'label'

]

.

head

())

y\_train

[

0

:

5

,:]

1. 1
2. 0
3. 1
4. 4
5. 0

Name: label, dtype: int64

Out[8]: array([[0., 1., 0., 0., 0., 0., 0., 0., 0., 0.], [1., 0., 0., 0., 0., 0., 0., 0., 0., 0.],

[0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],

[1., 0., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)

In [9]:

**from**

**sklearn.model\_selection**

**import**

train\_test\_split

X\_train

,

X\_cv

,

y\_train

,

y\_cv

=

train\_test\_split

(

X\_train

,

y\_train

,

test\_s

ize

=

0.1

,

random\_state

=

42

)

In [10]:

plt

.

imshow

(

X\_train

[

1

][:,:

,

0

])

plt

.

title

(

y\_train

[

1

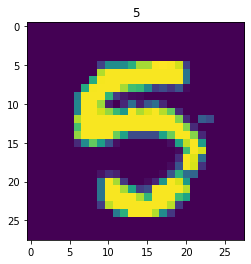
]

.

argmax

())

;



In [11]: **from** **keras.layers** **import** Input,InputLayer, Dense, Activation, ZeroPaddin g2D, BatchNormalization, Flatten, Conv2D

**from** **keras.layers** **import** AveragePooling2D, MaxPooling2D, Dropout **from** **keras.models** **import** Sequential,Model **from** **keras.optimizers** **import** SGD

**from** **keras.callbacks** **import** ModelCheckpoint,LearningRateScheduler **import** **keras**

**from** **keras** **import** backend **as** K

In [12]: *# Building a CNN model* input\_shape = (28,28,1) X\_input = Input(input\_shape)

# # layer 1

x = Conv2D(64,(3,3),strides=(1,1),name='layer\_conv1',padding='same')(X\_i nput)

x = BatchNormalization()(x) x = Activation('relu')(x)

x = MaxPooling2D((2,2),name='maxPool1')(x)

# # layer 2

x = Conv2D(32,(3,3),strides=(1,1),name='layer\_conv2',padding='same')(x) x = BatchNormalization()(x) x = Activation('relu')(x)

x = MaxPooling2D((2,2),name='maxPool2')(x)

# # layer 3

x = Conv2D(32,(3,3),strides=(1,1),name='conv3',padding='same')(x) x = BatchNormalization()(x) x = Activation('relu')(x)

x = MaxPooling2D((2,2), name='maxPool3')(x)

*# fc* x = Flatten()(x)

x = Dense(64,activation ='relu',name='fc0')(x) x = Dropout(0.25)(x)

x = Dense(32,activation ='relu',name='fc1')(x) x = Dropout(0.25)(x)

x = Dense(10,activation ='softmax',name='fc2')(x)

conv\_model = Model(inputs=X\_input, outputs=x, name='Predict') conv\_model.summary()

Model: "Predict"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param # ================================================================= input\_1 (InputLayer) (None, 28, 28, 1) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ layer\_conv1 (Conv2D) (None, 28, 28, 64) 640 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ batch\_normalization\_1 (Batch (None, 28, 28, 64) 256 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ activation\_1 (Activation) (None, 28, 28, 64) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ maxPool1 (MaxPooling2D) (None, 14, 14, 64) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ layer\_conv2 (Conv2D) (None, 14, 14, 32) 18464 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ batch\_normalization\_2 (Batch (None, 14, 14, 32) 128 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ activation\_2 (Activation) (None, 14, 14, 32) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ maxPool2 (MaxPooling2D) (None, 7, 7, 32) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ conv3 (Conv2D) (None, 7, 7, 32) 9248 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ batch\_normalization\_3 (Batch (None, 7, 7, 32) 128 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ activation\_3 (Activation) (None, 7, 7, 32) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ maxPool3 (MaxPooling2D) (None, 3, 3, 32) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ flatten\_1 (Flatten) (None, 288) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ fc0 (Dense) (None, 64) 18496 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ dropout\_1 (Dropout) (None, 64) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ fc1 (Dense) (None, 32) 2080 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ dropout\_2 (Dropout) (None, 32) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ fc2 (Dense) (None, 10) 330

=================================================================

Total params: 49,770

Trainable params: 49,514

Non-trainable params: 256

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In [13]: *# Adam optimizer* conv\_model.compile(optimizer='adam',loss='categorical\_crossentropy',metr ics=['accuracy'])

conv\_model.fit(X\_train, y\_train, epochs=10, batch\_size=100, validation\_d ata=(X\_cv,y\_cv))

Train on 37800 samples, validate on 4200 samples

Epoch 1/10

37800/37800 [==============================] - 144s 4ms/step - loss: 0.

4887 - accuracy: 0.8468 - val\_loss: 0.6940 - val\_accuracy: 0.8188

Epoch 2/10

37800/37800 [==============================] - 138s 4ms/step - loss: 0.

1349 - accuracy: 0.9610 - val\_loss: 0.0763 - val\_accuracy: 0.9776

Epoch 3/10

37800/37800 [==============================] - 138s 4ms/step - loss: 0.

0911 - accuracy: 0.9744 - val\_loss: 0.0541 - val\_accuracy: 0.9857

Epoch 4/10

37800/37800 [==============================] - 138s 4ms/step - loss: 0.

0760 - accuracy: 0.9791 - val\_loss: 0.0640 - val\_accuracy: 0.9840

Epoch 5/10

37800/37800 [==============================] - 138s 4ms/step - loss: 0.

0629 - accuracy: 0.9824 - val\_loss: 0.0658 - val\_accuracy: 0.9795

Epoch 6/10

37800/37800 [==============================] - 142s 4ms/step - loss: 0.

0566 - accuracy: 0.9845 - val\_loss: 0.0549 - val\_accuracy: 0.9838

Epoch 7/10

37800/37800 [==============================] - 139s 4ms/step - loss: 0.

0471 - accuracy: 0.9872 - val\_loss: 0.0378 - val\_accuracy: 0.9902

Epoch 8/10

37800/37800 [==============================] - 139s 4ms/step - loss: 0.

0460 - accuracy: 0.9872 - val\_loss: 0.0558 - val\_accuracy: 0.9864

Epoch 9/10

37800/37800 [==============================] - 139s 4ms/step - loss: 0.

0363 - accuracy: 0.9893 - val\_loss: 0.0427 - val\_accuracy: 0.9888

Epoch 10/10

37800/37800 [==============================] - 140s 4ms/step - loss: 0.

0366 - accuracy: 0.9899 - val\_loss: 0.0680 - val\_accuracy: 0.9850

Out[13]: <keras.callbacks.callbacks.History at 0x7f8bee1d2850>

In [14]: *# SGD optimizer* sgd = SGD(lr=0.0005, momentum=0.5, decay=0.0, nesterov=**False**) conv\_model.compile(optimizer=sgd,loss='categorical\_crossentropy',metrics

=['accuracy'])

conv\_model.fit(X\_train, y\_train, epochs=30, validation\_data=(X\_cv, y\_cv

))

Train on 37800 samples, validate on 4200 samples

Epoch 1/30

37800/37800 [==============================] - 173s 5ms/step - loss: 0.

0288 - accuracy: 0.9922 - val\_loss: 0.0345 - val\_accuracy: 0.9929

Epoch 2/30

37800/37800 [==============================] - 170s 5ms/step - loss: 0.

0237 - accuracy: 0.9937 - val\_loss: 0.0337 - val\_accuracy: 0.9929

Epoch 3/30

37800/37800 [==============================] - 172s 5ms/step - loss: 0.

0208 - accuracy: 0.9943 - val\_loss: 0.0327 - val\_accuracy: 0.9929

Epoch 4/30

37800/37800 [==============================] - 1599s 42ms/step - loss: 0.0188 - accuracy: 0.9949 - val\_loss: 0.0327 - val\_accuracy: 0.9938

Epoch 5/30

37800/37800 [==============================] - 1359s 36ms/step - loss: 0.0192 - accuracy: 0.9947 - val\_loss: 0.0330 - val\_accuracy: 0.9926

Epoch 6/30

37800/37800 [==============================] - 322s 9ms/step - loss: 0.

0178 - accuracy: 0.9951 - val\_loss: 0.0327 - val\_accuracy: 0.9929

Epoch 7/30

37800/37800 [==============================] - 1389s 37ms/step - loss: 0.0183 - accuracy: 0.9947 - val\_loss: 0.0323 - val\_accuracy: 0.9933

Epoch 8/30

37800/37800 [==============================] - 2006s 53ms/step - loss: 0.0186 - accuracy: 0.9948 - val\_loss: 0.0320 - val\_accuracy: 0.9933

Epoch 9/30

37800/37800 [==============================] - 1214s 32ms/step - loss: 0.0162 - accuracy: 0.9960 - val\_loss: 0.0321 - val\_accuracy: 0.9938

Epoch 10/30

37800/37800 [==============================] - 177s 5ms/step - loss: 0.

0172 - accuracy: 0.9952 - val\_loss: 0.0324 - val\_accuracy: 0.9938

Epoch 11/30

37800/37800 [==============================] - 174s 5ms/step - loss: 0.

0166 - accuracy: 0.9957 - val\_loss: 0.0329 - val\_accuracy: 0.9940

Epoch 12/30

37800/37800 [==============================] - 172s 5ms/step - loss: 0.

0170 - accuracy: 0.9952 - val\_loss: 0.0325 - val\_accuracy: 0.9940

Epoch 13/30

37800/37800 [==============================] - 568s 15ms/step - loss:

0.0169 - accuracy: 0.9953 - val\_loss: 0.0319 - val\_accuracy: 0.9943

Epoch 14/30

37800/37800 [==============================] - 180s 5ms/step - loss: 0.

0158 - accuracy: 0.9958 - val\_loss: 0.0321 - val\_accuracy: 0.9938

Epoch 15/30

37800/37800 [==============================] - 179s 5ms/step - loss: 0.

0156 - accuracy: 0.9959 - val\_loss: 0.0321 - val\_accuracy: 0.9938

Epoch 16/30

37800/37800 [==============================] - 179s 5ms/step - loss: 0.

0148 - accuracy: 0.9963 - val\_loss: 0.0324 - val\_accuracy: 0.9936

Epoch 17/30

37800/37800 [==============================] - 181s 5ms/step - loss: 0.

0163 - accuracy: 0.9953 - val\_loss: 0.0328 - val\_accuracy: 0.9938

Epoch 18/30

37800/37800 [==============================] - 180s 5ms/step - loss: 0.

0153 - accuracy: 0.9958 - val\_loss: 0.0328 - val\_accuracy: 0.9936

Epoch 19/30

37800/37800 [==============================] - 184s 5ms/step - loss: 0. 0159 - accuracy: 0.9958 - val\_loss: 0.0331 - val\_accuracy: 0.9938

Epoch 20/30

37800/37800 [==============================] - 181s 5ms/step - loss: 0.

0137 - accuracy: 0.9963 - val\_loss: 0.0329 - val\_accuracy: 0.9938

Epoch 21/30

37800/37800 [==============================] - 180s 5ms/step - loss: 0.

0153 - accuracy: 0.9959 - val\_loss: 0.0325 - val\_accuracy: 0.9936

Epoch 22/30

37800/37800 [==============================] - 180s 5ms/step - loss: 0.

0154 - accuracy: 0.9958 - val\_loss: 0.0325 - val\_accuracy: 0.9933

Epoch 23/30

37800/37800 [==============================] - 182s 5ms/step - loss: 0.

0151 - accuracy: 0.9957 - val\_loss: 0.0324 - val\_accuracy: 0.9936

Epoch 24/30

37800/37800 [==============================] - 182s 5ms/step - loss: 0.

0152 - accuracy: 0.9958 - val\_loss: 0.0328 - val\_accuracy: 0.9938

Epoch 25/30

37800/37800 [==============================] - 183s 5ms/step - loss: 0.

0134 - accuracy: 0.9966 - val\_loss: 0.0323 - val\_accuracy: 0.9940

Epoch 26/30

37800/37800 [==============================] - 182s 5ms/step - loss: 0.

0145 - accuracy: 0.9959 - val\_loss: 0.0323 - val\_accuracy: 0.9943

Epoch 27/30

37800/37800 [==============================] - 187s 5ms/step - loss: 0.

0130 - accuracy: 0.9965 - val\_loss: 0.0326 - val\_accuracy: 0.9938

Epoch 28/30

37800/37800 [==============================] - 182s 5ms/step - loss: 0.

0137 - accuracy: 0.9960 - val\_loss: 0.0324 - val\_accuracy: 0.9940

Epoch 29/30

37800/37800 [==============================] - 183s 5ms/step - loss: 0.

0139 - accuracy: 0.9961 - val\_loss: 0.0329 - val\_accuracy: 0.9936

Epoch 30/30

37800/37800 [==============================] - 184s 5ms/step - loss: 0.

0151 - accuracy: 0.9958 - val\_loss: 0.0326 - val\_accuracy: 0.9933

Out[14]: <keras.callbacks.callbacks.History at 0x7f8bf7b7c350>

In [16]: y\_pred = conv\_model.predict(X\_test) y\_pred = np.argmax(y\_pred,axis=1)

my\_submission = pd.DataFrame({'ImageId': list(range(1, len(y\_pred)+1)),

'Label': y\_pred}) my\_submission.to\_csv('dig\_submission.csv', index=**False**)

In [ ]:

*#confusion matrix*

In [4]:

**import** **numpy** **as** **np** *# linear algebra*

**import** **pandas** **as** **pd** *# data processing, CSV file I/O (e.g. pd.read\_csv)*

**import** **os**

In [5]:

df\_train = pd.read\_csv("train2.csv")

df\_test = pd.read\_csv("test2.csv")

In [6]:

df\_train.shape

Out[6]:

(42000, 785)

In [7]:

y = df\_train['label']

In [8]:

df = df\_train.drop(['label'], axis=1)

df.head()

Out[8]:

|  | **pixel0** | **pixel1** | **pixel2** | **pixel3** | **pixel4** | **pixel5** | **pixel6** | **pixel7** | **pixel8** | **pixel9** | **...** | **pixel774** | **pixel775** | **pixel776** | **pixel777** | **pixel778** | **pixel779** | **pixel780** | **pixel781** | **pixel782** | **pixel783** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

5 rows × 784 columns

In [9]:

*# To plot pretty figures*

%**matplotlib** inline

**import** **matplotlib**

**import** **matplotlib.pyplot** **as** **plt**

plt.rcParams['axes.labelsize'] = 14

plt.rcParams['xtick.labelsize'] = 12

plt.rcParams['ytick.labelsize'] = 12

In [10]:

digit\_to\_predict\_raw = np.array(df.iloc[5000,:])

digit\_to\_predict = np.array(digit\_to\_predict\_raw).reshape(28,28)

digit\_to\_predict.shape

Out[10]:

(28, 28)

In [11]:

plt.imshow(digit\_to\_predict,cmap = matplotlib.cm.binary)

plt.show()

A picture containing text

Description automatically generated

In [12]:

y[5000]

Out[12]:

8

In [13]:

**from** **sklearn.model\_selection** **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df, y, test\_size=0.30, random\_state=42)

In [14]:

y\_train\_8 = np.array(y\_train == 8)

y\_test\_8 = np.array(y\_test == 8)

In [16]:

**from** **sklearn.linear\_model** **import** SGDClassifier

sgd\_clf = SGDClassifier(max\_iter=5, random\_state=42)

sgd\_clf.fit(X\_train, y\_train\_8)

/Users/nicholasbergeland/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear\_model/stochastic\_gradient.py:561: ConvergenceWarning: Maximum number of iteration reached before convergence. Consider increasing max\_iter to improve the fit.

ConvergenceWarning)

Out[16]:

SGDClassifier(alpha=0.0001, average=False, class\_weight=None,

early\_stopping=False, epsilon=0.1, eta0=0.0, fit\_intercept=True,

l1\_ratio=0.15, learning\_rate='optimal', loss='hinge', max\_iter=5,

n\_iter\_no\_change=5, n\_jobs=None, penalty='l2', power\_t=0.5,

random\_state=42, shuffle=True, tol=0.001, validation\_fraction=0.1,

verbose=0, warm\_start=False)

In [17]:

pred = sgd\_clf.predict(X\_test)

pred

Out[17]:

array([ True, False, False, ..., False, False, True])

In [18]:

y\_test[:4]

Out[18]:

5457 8

38509 1

25536 9

31803 9

Name: label, dtype: int64

In [19]:

**from** **sklearn.model\_selection** **import** cross\_val\_score

cross\_val\_score(sgd\_clf, X\_train, y\_train\_8, cv=5, scoring="accuracy")

/Users/nicholasbergeland/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear\_model/stochastic\_gradient.py:561: ConvergenceWarning: Maximum number of iteration reached before convergence. Consider increasing max\_iter to improve the fit.

ConvergenceWarning)

/Users/nicholasbergeland/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear\_model/stochastic\_gradient.py:561: ConvergenceWarning: Maximum number of iteration reached before convergence. Consider increasing max\_iter to improve the fit.

ConvergenceWarning)

/Users/nicholasbergeland/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear\_model/stochastic\_gradient.py:561: ConvergenceWarning: Maximum number of iteration reached before convergence. Consider increasing max\_iter to improve the fit.

ConvergenceWarning)

/Users/nicholasbergeland/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear\_model/stochastic\_gradient.py:561: ConvergenceWarning: Maximum number of iteration reached before convergence. Consider increasing max\_iter to improve the fit.

ConvergenceWarning)

/Users/nicholasbergeland/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear\_model/stochastic\_gradient.py:561: ConvergenceWarning: Maximum number of iteration reached before convergence. Consider increasing max\_iter to improve the fit.

ConvergenceWarning)

Out[19]:

array([0.93691549, 0.93673469, 0.93843537, 0.89863946, 0.89743154])

In [20]:

**import** **collections**

collections.Counter(y\_train\_8)

Out[20]:

Counter({False: 26546, True: 2854})

In [21]:

num\_of\_8\_not\_occur = collections.Counter(y\_train\_8)[0]

print("The accuracy of model if we predict there are NO 8 present in the dataset :",

num\_of\_8\_not\_occur/len(y\_train\_8))

The accuracy of model if we predict there are NO 8 present in the dataset : 0.9029251700680272

In [22]:

*#confusion time!*

**from** **sklearn.model\_selection** **import** cross\_val\_predict

**from** **sklearn.metrics** **import** confusion\_matrix

y\_train\_8\_pred = cross\_val\_predict(sgd\_clf, X\_train, y\_train\_8, cv=3)

confusion\_matrix(y\_train\_8, y\_train\_8\_pred)

/Users/nicholasbergeland/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear\_model/stochastic\_gradient.py:561: ConvergenceWarning: Maximum number of iteration reached before convergence. Consider increasing max\_iter to improve the fit.

ConvergenceWarning)

/Users/nicholasbergeland/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear\_model/stochastic\_gradient.py:561: ConvergenceWarning: Maximum number of iteration reached before convergence. Consider increasing max\_iter to improve the fit.

ConvergenceWarning)

/Users/nicholasbergeland/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear\_model/stochastic\_gradient.py:561: ConvergenceWarning: Maximum number of iteration reached before convergence. Consider increasing max\_iter to improve the fit.

ConvergenceWarning)

Out[22]:

array([[25244, 1302],

[ 932, 1922]])

In [23]:

**from** **sklearn** **import** metrics

**def** get\_metrics(true\_labels, predicted\_labels):

print('Accuracy:', np.round(

metrics.accuracy\_score(true\_labels,

predicted\_labels),

4))

print('Precision:', np.round(

metrics.precision\_score(true\_labels,

predicted\_labels,

average='weighted'),

4))

print('Recall:', np.round(

metrics.recall\_score(true\_labels,

predicted\_labels,

average='weighted'),

4))

print('F1 Score:', np.round(

metrics.f1\_score(true\_labels,

predicted\_labels,

average='weighted'),

4))

In [24]:

get\_metrics(y\_train\_8, y\_train\_8\_pred)

Accuracy: 0.924

Precision: 0.9286

Recall: 0.924

F1 Score: 0.9261

In [25]:

**def** display\_confusion\_matrix(true\_labels, predicted\_labels, classes=[1,0]):

total\_classes = len(classes)

level\_labels = [total\_classes\*[0], list(range(total\_classes))]

cm = metrics.confusion\_matrix(y\_true=true\_labels, y\_pred=predicted\_labels,

labels=classes)

cm\_frame = pd.DataFrame(data=cm,

columns=pd.MultiIndex(levels=[['Predicted:'], classes],

labels=level\_labels),

index=pd.MultiIndex(levels=[['Actual:'], classes],

labels=level\_labels))

print(cm\_frame)

In [26]:

display\_confusion\_matrix(y\_train\_8, y\_train\_8\_pred)

Predicted:

1 0

Actual: 1 1922 932

0 1302 25244

/Users/nicholasbergeland/opt/anaconda3/lib/python3.7/site-packages/ipykernel\_launcher.py:10: FutureWarning: the 'labels' keyword is deprecated, use 'codes' instead

# Remove the CWD from sys.path while we load stuff.

/Users/nicholasbergeland/opt/anaconda3/lib/python3.7/site-packages/ipykernel\_launcher.py:12: FutureWarning: the 'labels' keyword is deprecated, use 'codes' instead

if sys.path[0] == '':

In [27]:

y\_scores = cross\_val\_predict(sgd\_clf, X\_train, y\_train\_8, cv=3,

method="decision\_function")

/Users/nicholasbergeland/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear\_model/stochastic\_gradient.py:561: ConvergenceWarning: Maximum number of iteration reached before convergence. Consider increasing max\_iter to improve the fit.

ConvergenceWarning)

/Users/nicholasbergeland/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear\_model/stochastic\_gradient.py:561: ConvergenceWarning: Maximum number of iteration reached before convergence. Consider increasing max\_iter to improve the fit.

ConvergenceWarning)

/Users/nicholasbergeland/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear\_model/stochastic\_gradient.py:561: ConvergenceWarning: Maximum number of iteration reached before convergence. Consider increasing max\_iter to improve the fit.

ConvergenceWarning)

In [28]:

**from** **sklearn.metrics** **import** precision\_recall\_curve

precisions, recalls, thresholds = precision\_recall\_curve(y\_train\_8, y\_scores)

In [29]:

**def** plot\_precision\_recall\_vs\_threshold(precisions, recalls, thresholds):

plt.plot(thresholds, precisions[:-1], "b--", label="Precision", linewidth=2)

plt.plot(thresholds, recalls[:-1], "g-", label="Recall", linewidth=2)

plt.xlabel("Threshold", fontsize=16)

plt.legend(loc="upper left", fontsize=16)

plt.ylim([0, 1])

plt.figure(figsize=(8, 4))

plot\_precision\_recall\_vs\_threshold(precisions, recalls, thresholds)

plt.xlim([-1700000, 1700000])

plt.show()

Diagram

Description automatically generated

In [30]:

y\_train\_prec\_90 = (y\_scores > 700000)

In [31]:

**from** **sklearn.metrics** **import** precision\_score, recall\_score

precision\_score(y\_train\_8, y\_train\_prec\_90)

Out[31]:

0.9741176470588235

In [32]:

recall\_score(y\_train\_8, y\_train\_prec\_90)

Out[32]:

0.14505956552207427

In [33]:

*#ROC*

**from** **sklearn.metrics** **import** roc\_auc\_score, roc\_curve

fpr, tpr, thresholds = roc\_curve(y\_train\_8, y\_scores)

In [34]:

**def** plot\_roc\_curve(fpr,tpr, label=**None**):

plt.plot(fpr, tpr, linewidth=2, label = label)

plt.plot([0,1], [0,1],'k--')

plt.axis([0,1,0,1])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plot\_roc\_curve(fpr,tpr)

print("The AUC score is :", roc\_auc\_score(y\_train\_8, y\_scores))

The AUC score is : 0.9098989016751398

Shape

Description automatically generated

In [35]:

**from** **sklearn.ensemble** **import** RandomForestClassifier

forest\_clf = RandomForestClassifier(n\_jobs=-1)

y\_forest\_pred = cross\_val\_predict(forest\_clf, X\_train, y\_train\_8, cv=3, method='predict\_proba')

y\_forest\_pred\_f = cross\_val\_predict(forest\_clf, X\_train, y\_train\_8, cv=3)

/Users/nicholasbergeland/opt/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of n\_estimators will change from 10 in version 0.20 to 100 in 0.22.

"10 in version 0.20 to 100 in 0.22.", FutureWarning)

/Users/nicholasbergeland/opt/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of n\_estimators will change from 10 in version 0.20 to 100 in 0.22.

"10 in version 0.20 to 100 in 0.22.", FutureWarning)

/Users/nicholasbergeland/opt/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of n\_estimators will change from 10 in version 0.20 to 100 in 0.22.

"10 in version 0.20 to 100 in 0.22.", FutureWarning)

/Users/nicholasbergeland/opt/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of n\_estimators will change from 10 in version 0.20 to 100 in 0.22.

"10 in version 0.20 to 100 in 0.22.", FutureWarning)

/Users/nicholasbergeland/opt/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of n\_estimators will change from 10 in version 0.20 to 100 in 0.22.

"10 in version 0.20 to 100 in 0.22.", FutureWarning)

/Users/nicholasbergeland/opt/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of n\_estimators will change from 10 in version 0.20 to 100 in 0.22.

"10 in version 0.20 to 100 in 0.22.", FutureWarning)

In [36]:

y\_scores\_forest= y\_forest\_pred[:,1]

fpr\_f, tpr\_f, thresholds\_f = roc\_curve(y\_train\_8, y\_scores\_forest)

In [37]:

plt.plot(fpr, tpr, "b:", label="SGD")

plot\_roc\_curve(fpr\_f, tpr\_f, "Random Forest")

plt.legend(loc="lower right")

plt.show()

print("Classification metrics for SGD :")

print("The AUC score for SGD is :", roc\_auc\_score(y\_train\_8, y\_scores))

get\_metrics(y\_train\_8, y\_train\_8\_pred)

print("**\n**Classification metrics for RandomForest :")

print("The AUC score is RandomForest is :", roc\_auc\_score(y\_train\_8, y\_scores\_forest))

get\_metrics(y\_train\_8, y\_forest\_pred\_f)

A picture containing graphical user interface

Description automatically generated

Classification metrics for SGD :

The AUC score for SGD is : 0.9098989016751398

Accuracy: 0.924

Precision: 0.9286

Recall: 0.924

F1 Score: 0.9261

Classification metrics for RandomForest :

The AUC score is RandomForest is : 0.9850807956106499

Accuracy: 0.9687

Precision: 0.9689

Recall: 0.9687

F1 Score: 0.9663

In [39]:

result = pd.DataFrame(df\_test)

result

Out[39]:

|  | **pixel0** | **pixel1** | **pixel2** | **pixel3** | **pixel4** | **pixel5** | **pixel6** | **pixel7** | **pixel8** | **pixel9** | **...** | **pixel774** | **pixel775** | **pixel776** | **pixel777** | **pixel778** | **pixel779** | **pixel780** | **pixel781** | **pixel782** | **pixel783** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 27995 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 27996 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 27997 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 27998 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 27999 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

28000 rows × 784 columns

In [40]:

result.to\_csv("confusion\_output.csv")

In [ ]:

Works Cited:

sriram2397. “Digit-Recognizer-Kaggle/digit\_recognizer.Ipynb at Master · SRIRAM2397/Digit-Recognizer-Kaggle.” GitHub. Accessed February 6, 2022. https://github.com/sriram2397/digit-recognizer-kaggle/blob/master/Digit\_

Recognizer.ipynb.