Data Mining

Lab 03

Introduction to Classification Techniques

Department of Computer Science
Kristianstad University
Course: DT584C
Master's in computer science

Student: Hazrat Ali <react.dev.se@gmail.com>
Teacher: Dawit Mengistu

Date: 8 January 2020

Objective

In this lab, you shall

1. Learn how to apply various classification techniques
2. Evaluate the performance of your implementation

- Note: You may implement your own classification code from scratch or build on available tools.

Task1:

- The following table consists of training data from an employee database. The data have been generalized. For example, “31 . . . 35” for age represents the age range of 31 to 35. For a given row entry, count represents the number of data tuples having the values for department, status, age, and salary given in that row.

Department	status	age	salary	count
sales	senior	31..35	46K..50K	30
sales	junior	26..30	26K..30K	40
sales	junior	31..35	31K..35K	40
systems	junior	21..25	46K..50K	20
systems	senior	31..35	66K..70K	5
systems	junior	26..30	46K..50K	3
systems	senior	41..45	66K..70K	3
marketing	senior	36..40	46K..50K	10
marketing	junior	31..35	41K..45K	4
secretary	senior	46..50	36K..40K	4
secretary	junior	26..30	26K..30K	6

Let status be the class-label attribute.

A. Implement a Decision Tree classification solution
B. Repeat the same problem using Naïve Bayesian classifier
C. Evaluate the performance of the two algorithms

   Repeat the above problem using ANN:

D. Design a multilayer feed-forward neural network for the given data. Label the nodes in the input and output layers.
E. Using the multilayer feed-forward neural network obtained in (a), show the weight values after one iteration of the backpropagation algorithm, given the training instance "(sales, senior, 31..35, 46K..50K)". Indicate your initial weight values and biases and the learning rate used.

Task2:

- The MNIST database of handwritten digits, available at http://yann.lecun.com/exdb/mnist/, has a training set of 60,000 examples, of which it includes a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

  Read the information provided on this site about the content of the dataset.

- Implement the K-nearest neighbor (and/or any other) classification algorithm to recognize handwritten digits.
- Repeat the classification using neural network implementation
- Repeat the classification using SVM
- Discuss the following:
  - Validation method used -
  - Selection of training samples
  - Accuracy of your results.

Introduction

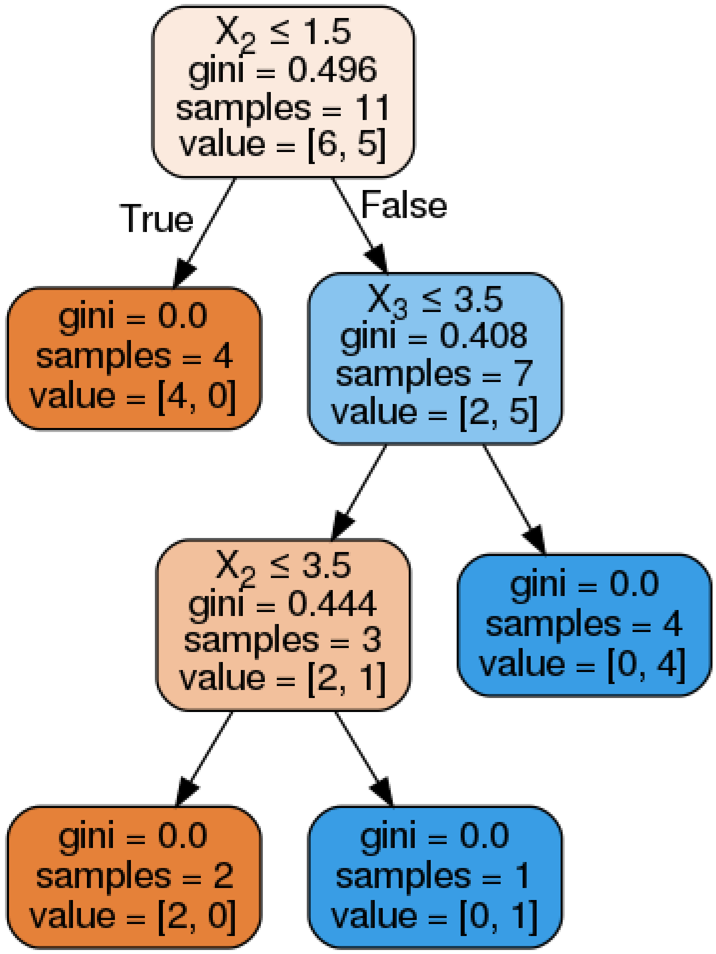
Classification is a supervised learning method in data mining. In supervised learning or classification the labels (or classes) are known in advance and the task is to classify the new data based on the model learned from training data.

It is a two step process in which case we first construct a model and then use the model for estimation. Accuracy of the model is the rate of correctly classified test samples in percentage. The test set is independent or unknown to the model otherwise it is called overfitting.

There are several classification algorithms for example decision tree, Naive Bayesian, Random forest, and neural network to name a few.

Task 1

A. In this task I implemented a decision tree solution for the given dataset. I also need to preprocess the data first and convert it into numerical form using Python libraries. I divided the dataset into target and input to the model. Then I tested by giving a value to the model and testing the result, with a score of 1.



B. In this task I implemented solution for Naive Bayesian classification. I preprocessed data to numeric form. I divided the data to 80% training and 20% testing and I got 0.66 score.

Task 2

A. <u>Convolutional neural network classification</u>

In this task I implemented a convolutional neural network (CNN) using tensorflow python library. CNN is usually used for natural language processing and image processing tasks.

There are three layers in CNN given below.

- **Convolutional Layers:**

  This is the first layer in CNN and here we extract the features of the image

- **Pooling Layer:**

  We insert pooling layer after each convolutional layer to reduce the spatial size of the images.

- **Fully connected layer:**

  Here each node is connected to each other to determine the relationship of each parameter on the resulting label/class.

I have used the following steps / functions in the code to do this task:

1. **Load Data**

   Here we load data from mnist dataset into the dataframe.

2. **preprocess data**

   In the method we preprocess data, e.g to change it to 4 dimensional data

3. **normalize_data**

   In this step we normalize data by changing the RGB code to 255 limit, and to apply float.

4. **create_model**

   Create the model and layers here.

5. **train_model**

   Here we train the model with x_train, y_train sets of data. The training data contains **60,000** of samples, while the testing data contains **10,000** samples.

6. **predict_image**

   We can predict/validate any image by providing image_index of x_train (with max 10 000)

```
# 7 - predict an image by index ( < 10 000)
predict_image(52) # specify image index in x_test
```

```
The predected image is :  5
```

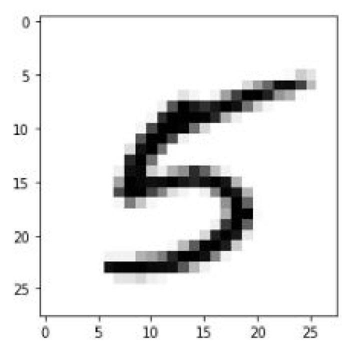


Figure : an image (5) at index is predicted as 5

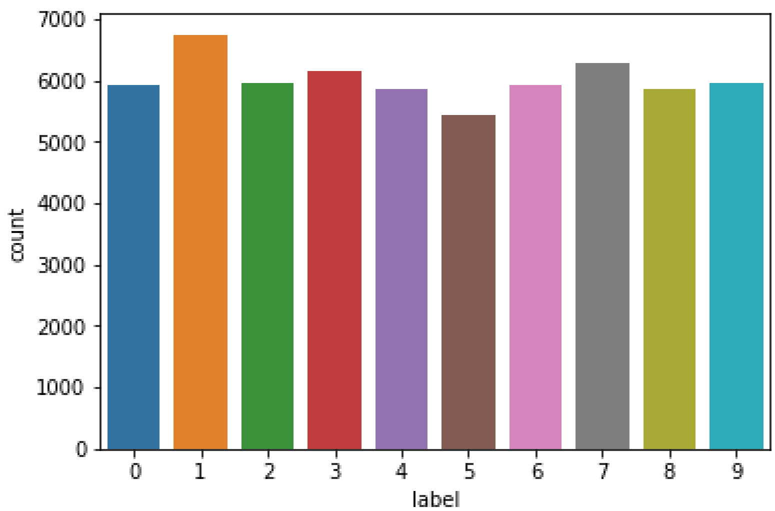
Evaluation

We used the builtin method of model called evaluate and got about 99.38% success rate.

B. Support vector machine SVM

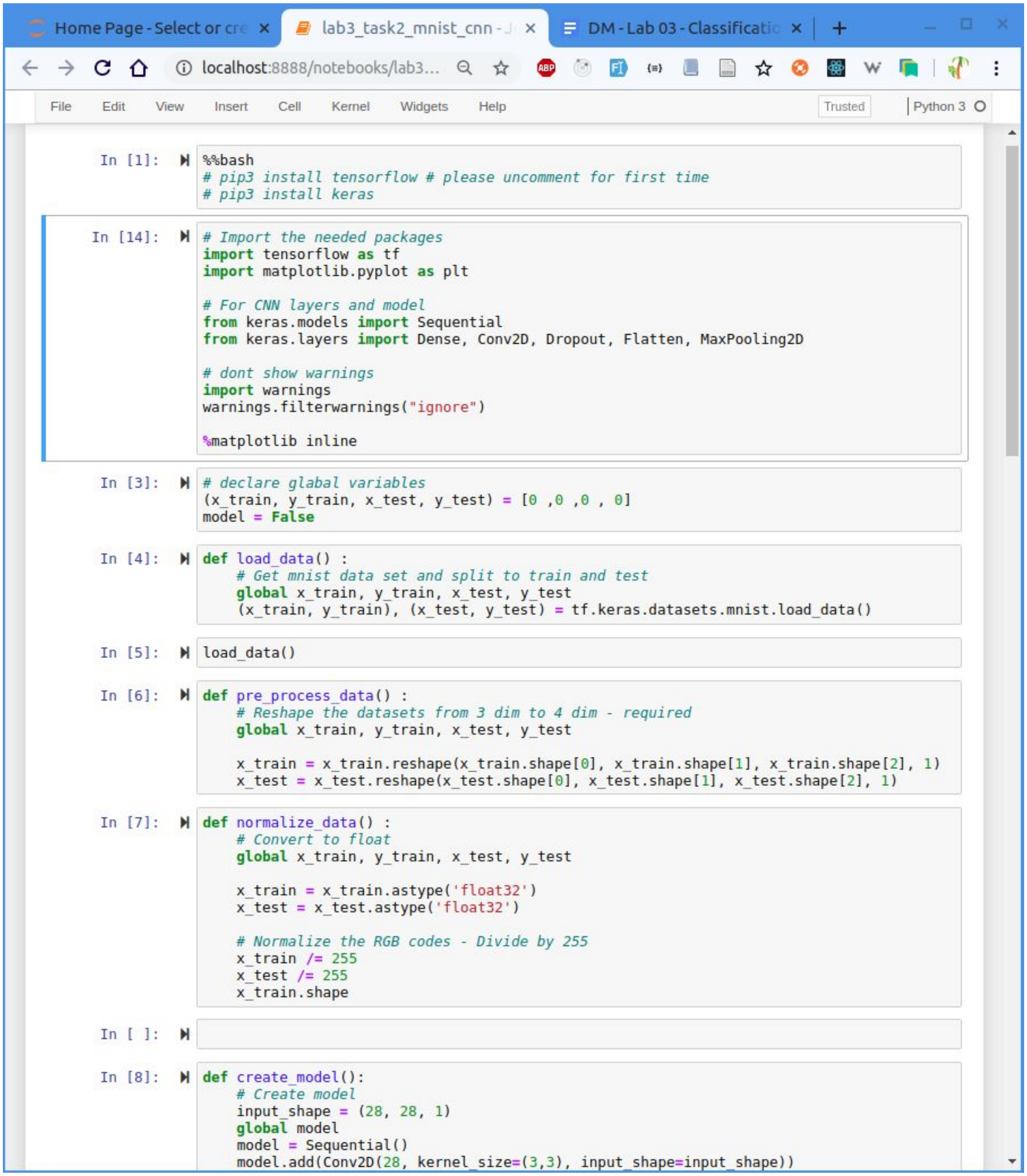
In this task we implemented SVM with accuracy of 94.04 % as shown in Appendix 2.

The frequency distribution of figures from 0 to 9 is given below:

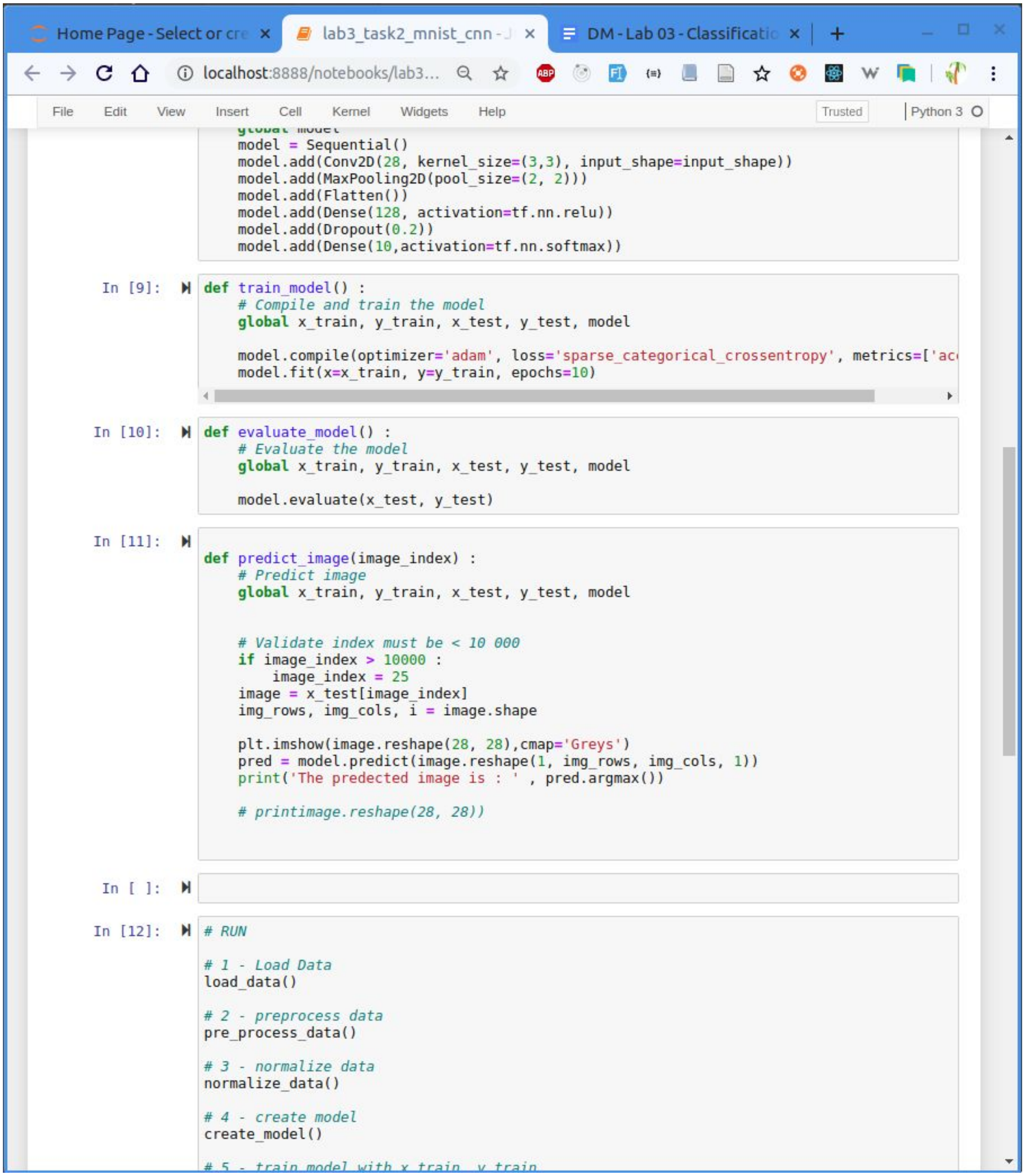


APPENDIX 1 - Source code and results screenshots

Convolutional network



```
In [1]:
%%bash
# pip3 install tensorflow # please uncomment for first time
# pip3 install keras

In [14]:
# Import the needed packages
import tensorflow as tf
import matplotlib.pyplot as plt

# For CNN layers and model
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D

# dont show warnings
import warnings
warnings.filterwarnings("ignore")

%matplotlib inline

In [3]:
# declare glabal variables
(x_train, y_train, x_test, y_test) = [0 ,0 ,0 , 0]
model = False

In [4]:
def load_data() :
    # Get mnist data set and split to train and test
    global x_train, y_train, x_test, y_test
    (x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

In [5]:
load_data()

In [6]:
def pre_process_data() :
    # Reshape the datasets from 3 dim to 4 dim - required
    global x_train, y_train, x_test, y_test

    x_train = x_train.reshape(x_train.shape[0], x_train.shape[1], x_train.shape[2], 1)
    x_test = x_test.reshape(x_test.shape[0], x_test.shape[1], x_test.shape[2], 1)

In [7]:
def normalize_data() :
    # Convert to float
    global x_train, y_train, x_test, y_test

    x_train = x_train.astype('float32')
    x_test = x_test.astype('float32')

    # Normalize the RGB codes - Divide by 255
    x_train /= 255
    x_test /= 255
    x_train.shape

In [ ]:

In [8]:
def create_model():
    # Create model
    input_shape = (28, 28, 1)
    global model
    model = Sequential()
    model.add(Conv2D(28, kernel_size=(3,3), input_shape=input_shape))
```


Home Page - Select or cre
lab3_task2_mnist_cnn - J
DM - Lab 03 - Classificatio
localhost:8888/notebooks/lab3...
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
model = Sequential()
model.add(Conv2D(28, kernel_size=(3,3), input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation=tf.nn.relu))
model.add(Dropout(0.2))
model.add(Dense(10,activation=tf.nn.softmax))
In [9]:
def train_model() :
Compile and train the model
global x_train, y_train, x_test, y_test, model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['ac
model.fit(x=x_train, y=y_train, epochs=10)
In [10]:
def evaluate_model() :
Evaluate the model
global x_train, y_train, x_test, y_test, model
model.evaluate(x_test, y_test)
In [11]:
def predict_image(image_index) :
Predict image
global x_train, y_train, x_test, y_test, model
Validate index must be < 10 000
if image_index > 10000 :
image_index = 25
image = x_test[image_index]
img_rows, img_cols, i = image.shape
plt.imshow(image.reshape(28, 28),cmap='Greys')
pred = model.predict(image.reshape(1, img_rows, img_cols, 1))
print('The predected image is : ' , pred.argmax())
# printimage.reshape(28, 28))
In [ ]:
In [12]:
# RUN
# 1 - Load Data
load_data()
# 2 - preprocess data
pre_process_data()
# 3 - normalize data
normalize_data()
# 4 - create model
create_model()
# 5 - train model with x_train, y_train

Home Page - Select or cre × | lab3_task2_mnist_cnn - J × | DM - Lab 03 - Classificatio ×

localhost:8888/notebooks/lab3...

File Edit View Insert Cell Kernel Widgets Help | Trusted | Python 3

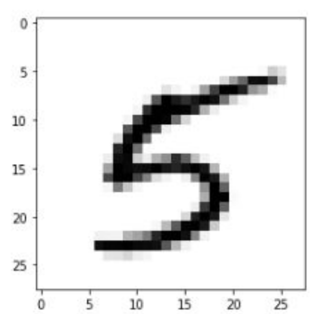
```
# 4 - create model
create_model()

# 5 - train model with x_train, y_train
train_model()

# 6 - evaluate model
evaluate_model()
```

```
Epoch 1/10
60000/60000 [==============================] - 22s 373us/step - loss: 0.2114 - accurac
y: 0.9356
Epoch 2/10
60000/60000 [==============================] - 23s 375us/step - loss: 0.0848 - accurac
y: 0.9739
Epoch 5/10
60000/60000 [==============================] - 22s 372us/step - loss: 0.0360 - accurac
y: 0.9883
Epoch 6/10
60000/60000 [==============================] - 23s 382us/step - loss: 0.0294 - accurac
y: 0.9903
Epoch 7/10
60000/60000 [==============================] - 23s 379us/step - loss: 0.0256 - accurac
y: 0.9909
Epoch 8/10
60000/60000 [==============================] - 23s 380us/step - loss: 0.0220 - accurac
y: 0.9926
Epoch 9/10
60000/60000 [==============================] - 23s 388us/step - loss: 0.0174 - accurac
y: 0.9939
Epoch 10/10
60000/60000 [==============================] - 22s 366us/step - loss: 0.0186 - accurac
y: 0.9938
10000/10000 [==============================] - 1s 103us/step
```

In [15]:

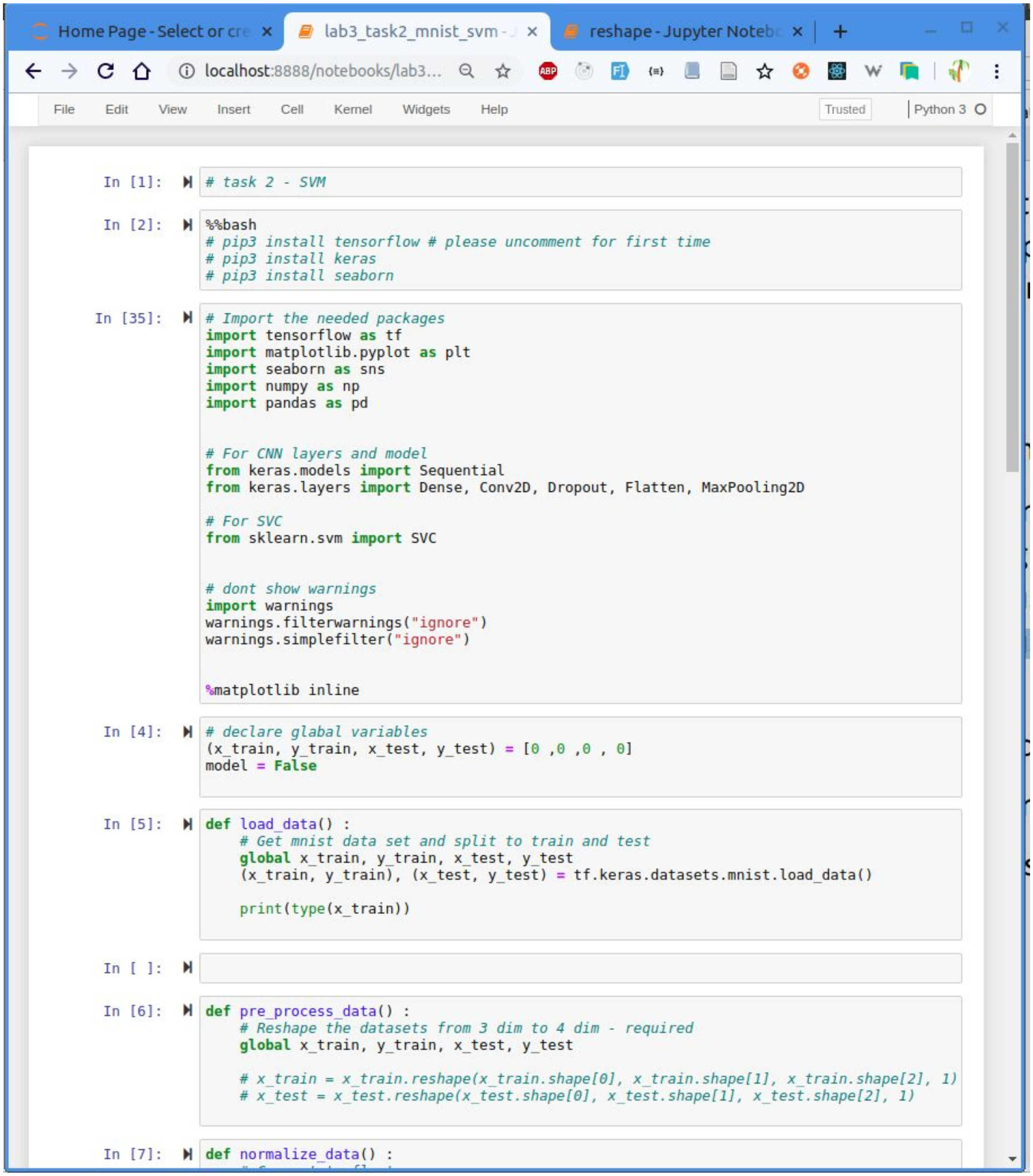
```
# 7 - predict an image by index ( < 10 000)
predict_image(52) # specify image index in x_test
```

```
The predected image is :  5
```



In [ ]:

APPENDIX 2 - SVM



In [1]:
```
# task 2 - SVM
```

In [2]:
```
%%bash
# pip3 install tensorflow # please uncomment for first time
# pip3 install keras
# pip3 install seaborn
```

In [35]:
```
# Import the needed packages
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd


# For CNN layers and model
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D

# For SVC
from sklearn.svm import SVC


# dont show warnings
import warnings
warnings.filterwarnings("ignore")
warnings.simplefilter("ignore")


%matplotlib inline
```

In [4]:
```
# declare glabal variables
(x_train, y_train, x_test, y_test) = [0 ,0 ,0 , 0]
model = False
```

In [5]:
```
def load_data() :
    # Get mnist data set and split to train and test
    global x_train, y_train, x_test, y_test
    (x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

    print(type(x_train))
```

In [ ]:

In [6]:
```
def pre_process_data() :
    # Reshape the datasets from 3 dim to 4 dim - required
    global x_train, y_train, x_test, y_test

    # x_train = x_train.reshape(x_train.shape[0], x_train.shape[1], x_train.shape[2], 1)
    # x_test = x_test.reshape(x_test.shape[0], x_test.shape[1], x_test.shape[2], 1)
```

In [7]:
```
def normalize_data() :
```

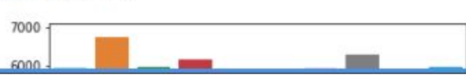
```python
In [7]:  def normalize_data() :
             # Convert to float
             global x_train, y_train, x_test, y_test

             x_train = x_train.astype('float32')
             x_test = x_test.astype('float32')

             # Normalize the RGB codes - Divide by 255
             x_train /= 255
             x_test /= 255
             x_train.shape
```

```python
In [ ]:
```

```python
In [12]: def visualaize_train_data() :
             global y_train
             x = y_train.reshape(-1,1)
             df = pd.DataFrame.from_records(x)
             df.columns = [ 'label']
             # print(df['label'].value_counts()) # Frequency distribution
             sns.countplot(df['label'])
```

```python
In [13]: # RUN

         # 1 - Load Data
         load_data()

         # 2 - Preprocess data
         pre_process_data()

         # 3 - Normalize data
         normalize_data()

         # 4 - Visualize data
         visualaize_train_data()


         # 4 - create model
         # create_model()

         # 5 - train model with x_train, y_train
         # train_model()

         # 6 - evaluate model
         # evaluate_model()


         print(x_train.shape)
```

```
<class 'numpy.ndarray'>
(60000, 28, 28)
```

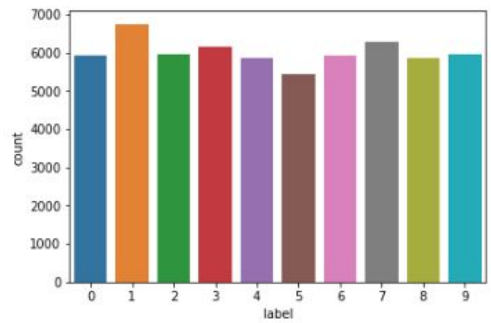


Home Page - Select or cre × | lab3_task2_mnist_svm - J × | reshape - Jupyter Notebo × | +

localhost:8888/notebooks/lab3...

File Edit View Insert Cell Kernel Widgets Help | Trusted | Python 3

```
<class 'numpy.ndarray'>
(60000, 28, 28)
```



In [21]:
```
# linear model

model_linear = SVC(kernel='linear')
model_linear.fit(x_train.reshape(60000, 784), y_train)

# predict using x_test
y_pred = model_linear.predict(x_test.reshape(10000, 784))
```

In [25]:
```
x_test.reshape(10000, 784)
```

Out[25]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

In [34]:
```
from sklearn import metrics
from sklearn.metrics import confusion_matrix
# accuracy
accuracy = metrics.accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy:", accuracy, 'Percentage:' , accuracy * 100)

# cm
print('Confusion matrix:')
print(metrics.confusion_matrix(y_true=y_test, y_pred=y_pred))
```

```
Accuracy: 0.9404 Percentage: 94.04
Confusion matrix:
[[ 957    0    4    1    1    6    9    1    0    1]
 [    0 1122    3    2    0    1    2    1    4    0]
 [    8    6  967   11    3    3    7    8   17    2]
 [    4    3   16  947    1   16    0    9   12    2]
 [    1    1   10    1  942    2    4    2    3   16]
 [   10    4    3   36    6  803   13    1   14    2]
 [    9    2   13    1    5   16  910    1    1    0]
 [    1    8   21   10    8    1    0  957    3   19]
 [    8    4    6   25    7   26    6    7  877    8]
 [    7    7    2   11   33    4    0   18    5  922]]
```

APPENDIX 3 - Demo links

- TASK 1 - a : Decision tree

  https://github.com/iloveyii/data-mining-lab3/blob/master/lab3_task1_a.ipynb

- TASK 1 - b : Classification

  https://github.com/iloveyii/data-mining-lab3/blob/master/lab3_task1_b.ipynb

- TASK 2 - Convolutional network

  https://github.com/iloveyii/data-mining-lab3/blob/master/lab3_task2_mnist_cnn.ipynb

- TASK 2 - SVM

  https://github.com/iloveyii/data-mining-lab3/blob/master/lab3_task2_mnist_svm.ipynb