

# יישומי מכונה לומדת בשפת Python תחת windows10

פעילות 13 - פיתוח מכונה לומדת לזיהוי ספרות בכתב יד תחת scikit-learn

מקורות:

[https://scikit-learn.org/stable/auto\\_examples/neural\\_networks/plot\\_mnist\\_filters.html#sphx-glr-auto-examples-neural-networks-plot-mnist-filters-py](https://scikit-learn.org/stable/auto_examples/neural_networks/plot_mnist_filters.html#sphx-glr-auto-examples-neural-networks-plot-mnist-filters-py)

<https://matplotlib.org/>

<https://scikit-learn.org/stable/install.html>

<https://playground.tensorflow.org/>

<https://www.openml.org/d/554>

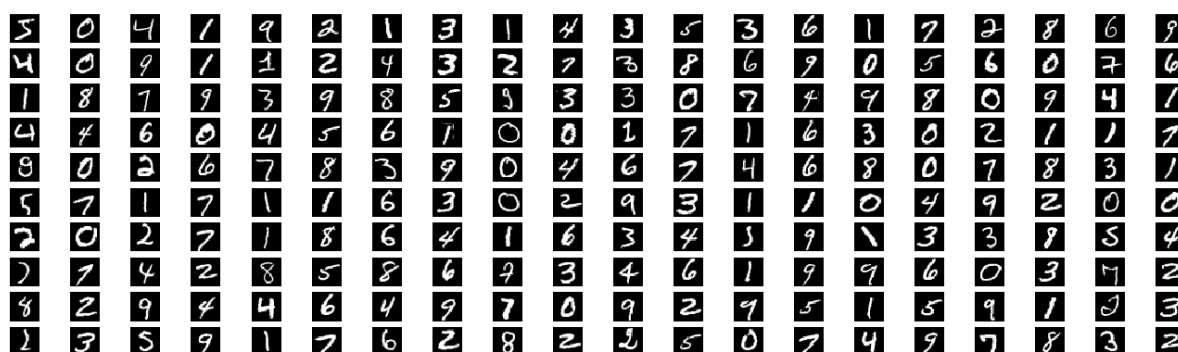
<https://towardsdatascience.com/image-classification-in-10-minutes-with-mnist-dataset-54c35b77a38d>

<https://machinelearningmastery.com/make-predictions-scikit-learn/>

<https://scikit-learn.org/stable/index.html>

<https://gogul09.github.io/software/image-classification-python>

בפעילות זה נתרגל כתיבת קוד בשפת python המאמן מכונה לומדת לזיהוי מספרים מתוך מאגר תמונות. בפעילות זה נעשה שימוש ב- 70000 תמונות מתיוגות הנראות כך:



כפי שסיכמנו את פרק 2 שבו למדנו את עקרונות התכנות של מכונה לומדת תוך כדי כתיבת קוד מהיסוד, או במילים אחרות כתיבת מכונה לומדת ללא שימוש בספריות python ייעודיות לך. החל מפרק 3 אנו עוברים לממש מכונות לומדות תוך שימוש בספריות מוכנות שנותנות לנו כלים נוחים, יעילים ומורכבים כאחד כדי לפתח מכונות למידה.

הספרייה הראשונה שנסתמש בה כדי ללמוד לפתח מכונה לומדת יהיה Scikit-learn שהיא אחת מהספריות ללמידת מכונה של Python. הספרייה נכתבה כך שתתמוך בספריות משנה שכבר הכרנו כמו Numpy. הספרייה ברובה נכתבה בשפת Python כאשר חלקים ממנה נכתבו בשפת C כדי לתת ביצועים גבוהים.

הספרייה נכתבה במקור על ידי David Courneau כחלק מפרויקט של גוגל. נכון להיום ניתן לראות שימוש של הספרייה כחלק ממערך המלצות השירים באפליקציה של Spotify כמו גם נעשה שימוש בספרייה

Scikit-learn באתר Booking.com כדי לספק לגולשים המלצות על מלונות ויעדים לטיול כמו גם איתור הונאות במערך ההזמנות של החברה.

## התקנות

התקנת הספרייה תתבצע על ידי ההוראה הבאה:

```
> python --version
> python -m pip install -U pip
> python -m pip install -U matplotlib
> python -m pip install -U pandas
> python -m pip install -U scikit-learn
> python -m pip install -U mglearn
```

## היכרות ראשונה על הספרייה Scikit-learn

נתחיל בכתיבת קוד למכונה לומדת המבוססת על הספרייה Scikit-learn כדי לממש שער XOR. כזכור בפעילות 10 כתבנו כבר קוד המממש שער XOR תוך שימוש ברשת של 5 פרספטרונים. באותה הפעילות התבססנו על 2 פרספטרונים במבוא שיקלטו את הרמות הלוגיות של שער XOR. בנוסף 2 פרסטרונים בשכבה הפנימית ועוד פרספטרון אחד במוצא כדי לספק לנו את מוצא הרשת. ננצל את הידע מפעילות 10 כדי לממש שער XOR תוך שימוש בספרייה ייעודית ללמידת מכונה.

להלן קוד התוכנית:

```
import numpy as np
import sklearn.neural_network

inputs = np.array([[0,0],[0,1],[1,0],[1,1]])
expected_output = np.array([[0],[1],[1],[0]])

model = sklearn.neural_network.MLPClassifier(activation='tanh', max_iter=10000,
hidden_layer_sizes=(3,), solver='lbfgs')
model.fit(inputs, expected_output)
print('predictions:', model.predict(inputs))
```

נקבל את הפלט הבא:

```
predictions: [0 1 1 0]
```

נבחן את השורה המגדירה את המבנה של המכונה:

```
model = sklearn.neural_network.MLPClassifier(activation='tanh', max_iter=10000,
hidden_layer_sizes=(3,), solver='lbfgs')
```

ההוראה מזמנת את הפעולה MLPClassifier של המחלקה neural\_network המהווה חלק מהספרייה sklearn. הפעולה מספקת לנו כלי הגדרת רשת נוירונים מלאכותית Multi-layer Perceptron. הפעולה מקבלת סדרה ארוכה של פרמטרים שאת חלקם שיבנו בדוגמה ואלה הם:

- פרמטר activation מגדיר את פונקציית התמסורת של פרספטרון. במקרה שלנו הגדרנו את הפונקציה כ- tanh.
- פרמטר max\_iter מגדיר מספר הפעמים שבה רשת הנוירונים תעבור על מערך נתוני האימון.
- פרמטר hidden\_layer\_sizes מגדיר את מערך השכבות הפנימיות של הרשת. במקרה שלנו הגדרנו שכבה בודדת אחת שבה 3 נוירונים.
- הפרמטר solver מגדיר האלגוריתם שבו רשת הנוירונים מבצעת חישובי משקלים. על פי המלצת המפתחים יש להשתמש במאפיין lbfgs כאשר עובדים עם מערך קטן של נתוני למידה.

לקבלת מפרט מלא של הפרמטרים:

[https://scikit-learn.org/stable/modules/generated/sklearn.neural\\_network.MLPClassifier.html](https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html)

לקבלת קוד המחלקה:

[https://github.com/scikit-learn/scikit-learn/blob/b194674c4/sklearn/neural\\_network/\\_multilayer\\_perceptron.py#L691](https://github.com/scikit-learn/scikit-learn/blob/b194674c4/sklearn/neural_network/_multilayer_perceptron.py#L691)

נבחן כעת את יעילות המחלקה עבור יכולת הסיווג של מערך הפרחים. נכתוב את הקוד הבא:

```
import numpy as np
from termcolor import colored
import sklearn.neural_network

vir_iris_data = np.genfromtxt('iris_for_ML.csv', delimiter=',')
random_iris_data = np.random.permutation(vir_iris_data)
test_data = random_iris_data[0:10,:4]
train_data = random_iris_data[10:,:4]
test_lbl = random_iris_data[0:10,4:]
train_lbl = random_iris_data[10:,:4:]

mlp = sklearn.neural_network.MLPClassifier(hidden_layer_sizes=(10), solver='sgd',
                                           learning_rate_init=0.01, max_iter=500)

mlp.fit(train_data, train_lbl)

for i in range(len(test_data)):
    tt=test_data[i].reshape( 1,(test_data[i].shape[0]))
    p = mlp.predict(tt)
```

```
print("test_lbl:" , colored(test_lbl[i], 'green') , "prediction:" , colored(p, 'green') )
```

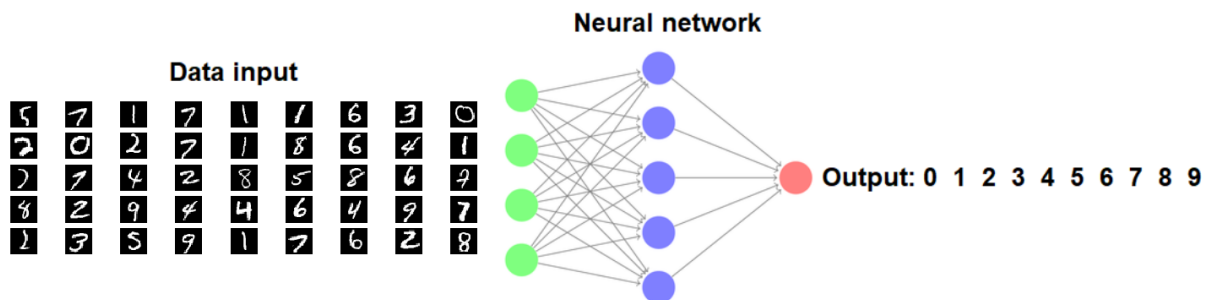
נקבל את הפלט הבא:

```
test_lbl: [3.] prediction: [3.]
test_lbl: [2.] prediction: [2.]
test_lbl: [3.] prediction: [3.]
test_lbl: [2.] prediction: [2.]
test_lbl: [2.] prediction: [2.]
test_lbl: [2.] prediction: [2.]
test_lbl: [3.] prediction: [3.]
test_lbl: [1.] prediction: [1.]
test_lbl: [3.] prediction: [3.]
test_lbl: [1.] prediction: [1.]
```

בתרגיל זה בנינו מכונה לומדת מבוססת על רשת נוירונים במבנה הבא: 4 נוירונים מבוא, 10 נוירונים בשכבה האמצעית ו-3 נוירונים במוצא. כיוול משקלים ברשת יתבצע על ידי אלגוריתם stochastic gradient descent with no batch-size או בקיצור sgd. כמו כן learning rate של 0.01 ו-500 סבבי למידה.

### מכונה לומדת לזיהוי ספרות בכתב יד

נתרגל כתיבת קוד בשפת python המאמן מכונה לומדת לזיהוי מספרים מתוך מאגר תמונות. בפעילות זה נעשה שימוש ב-70000 תמונות מתויגות.



### ארגון נתוני האמון

כפי שכבר למדנו ארגון מערך נתוני האימון הוא חלק מרכזי בכתיבת מכונה לומדת. בתרגיל זה נעשה שימוש במערך תמונות מוכן המתואר בקישור הבא:

<https://www.openml.org/d/554>

ניתן ללמוד מהאתר שמערך הנתונים כולל 70000 תמונות, כל תמונה כולל 784 פיקסלים (כלומר כל תמונה היא ברזולוציה של  $28 \times 28$ ) התמונות מתויגות ל-10 קטגוריות שונות (כלומר כל תמונה מתויגת לאחת הספרות 0 עד 9).

ניתן לעבוד עם מערך הנתונים ב-2 דרכים:

1. להוריד את קובץ התמונות ישירות למחשב האישי שלכם (קובץ בגודל 52.8M)

להלן קישור לקובץ מסד הנתונים:

<https://github.com/amplab/datascience-sp14/raw/master/lab7/mldata/mnist-original.mat>

2. להשתמש בספרייה מוכנה של sklearn כדי להוריד את הנתונים ישירות מהאינטרנט בכל פעם שמפעילים את התוכנה. בדרך זו לא צריך לוודא היכן קובץ הנתונים שמור במחשב והפעולה

fetch\_openml שבספרייה sklearn.datasets תדאג לכך. החיסרון בשיטה זו הוא שימש צורך להמתין כחצי דקה בכל פעם שמפעילים את היישום.

לשם כך יש לכתוב את הקוד הבא:

```
import sklearn.datasets
x, y = sklearn.datasets.fetch_openml('mnist_784', version=1, return_X_y=True)
```

הפעולה fetch\_openml תוריד את הנתונים ישירות מאתר <https://www.openml.org/d/554> לתוך 2 מערכים x ו-y מופרדים כבר ל- 60000 תמונות ב-x ו- 10000 תמונות ב-y.

בפעילות זו נעזר בפעולה fetch\_openml כדי להוריד למחשב את קובץ הנתונים ואז נשמור אותו במחשב כדי לחסוך את הזמן שייקח כאשר נפעיל מספר פעמים את התוכנית.

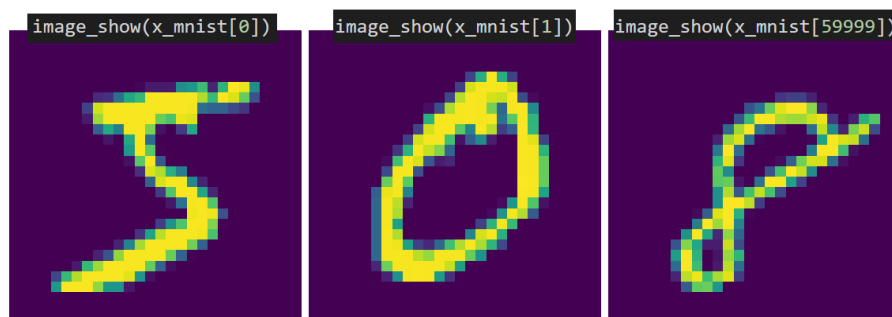
נדגים תוכנית שמורידה מהאינטרנט את מערך התמונות ומציגה ממנו 3 תמונות:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_openml

def image_show(arr):
    p = (np.reshape(arr, (28, 28))).astype(np.uint8)
    plt.axis('off')
    plt.imshow(p)
    plt.show()

print("downloading file...")
x_mnist, y_mnist = fetch_openml("mnist_784", version=1, return_X_y=True, data_home=".")
image_show(x_mnist[0])
image_show(x_mnist[1])
image_show(x_mnist[59999])
```

נקבל את הפלט הבא:



כדי לשמור את מערך הנתונים במחשב לאחר ההורדה מהאינטרנט נכתוב את הקוד הבא:

```
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path
from sklearn.datasets import fetch_openml

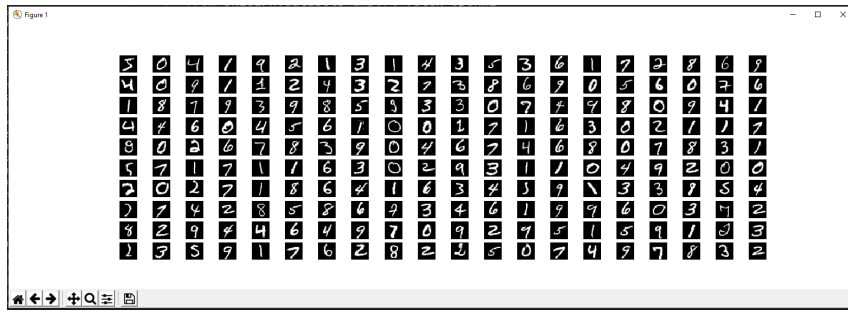
datapath = Path("mnist_784.npz")
if not(datapath.exists()):
    print("downloading file...")
    x_mnist, y_mnist = fetch_openml("mnist_784", version=1,
                                    return_X_y=True, data_home=".")
    x=np.array(x_mnist,dtype="u8")
    y=np.array(y_mnist,dtype="u8")
    np.savez(datapath,x=x,y=y)
    del x_mnist, y_mnist
print("Loading file...")
data = np.load(datapath)
print("\ntype: ", type(data))
print("\ntype: ", data.files)
x_mnist = data["x"]
y_mnist = data["y"]

plt.figure()
for i in range(200):
    plt.subplot(10,20,i+1)
    plt.axis("off")
    plt.imshow(x_mnist[i].reshape((28,28)),cmap="gray", vmin=0, vmax=255)
plt.show()
```

נקבל את הפלט הבא:

```
type:
<class 'numpy.lib.npyio.NpzFile'>
files:
['x', 'y']
```

ובהמשך הפלט נקבל דוגמה של 200 התמונות הראשונות מתוך מערך הנתונים x\_mnist



בקוד זה השתמשנו במחלקה Path מתוך הספרייה pathlib כדי לבדוק האם קיים במחשב המקומי קובץ מתוך הנתונים.

```
if not(datapath.exists()):
```

במידה והקובץ לא קיים נעזר במחלקה fetch\_openml כדי להוריד את הקובץ המכיל את התמונות מהאינטרנט

```
x_mnist, y_mnist = fetch_openml("mnist_784", version=1, return_X_y=True, data_home=".")
```

ואז לשמור את 2 המערכים x\_mnist הכולל את התמונות ו-y\_mnist הכולל את תיוג התמונות לקובץ בשם mnist\_784.npz

```
np.savez(datapath,x=x,y=y)
```

הפעולה savez שומרת מערכי נתונים מסוג numpy.ndarray לקובץ בפורמט ייחודי בשם npz למידע בנושא:

<https://kite.com/python/docs/numpy.lib.npyio.NpzFile>

אחזור מערכי הנתונים מהקובץ מתבצע על ידי ההוראות:

```
data = np.load(datapath)
x_mnist = data["x"]
y_mnist = data["y"]
```

השלב האחרון בארגון נתוני האימון יהיה לחלק את התמונות באופן הבא:

train\_data - מערך הכולל 60000 תמונות לצורך שלב האימון

train\_lbl - מערך הכולל 60000 מספרים בין 0 ל-9 המייצגים את המספרים שתמונות, לצורך שלב האימון

test\_data - מערך הכולל 10000 תמונות לצורך שלב בדיקת המכונה

test\_lbl - מערך הכולל 10000 מספרים בין 0 ל-9 המייצגים את המספרים שתמונות, לצורך בדיקת המכונה

כמו כן ראינו שכל תמונה בנוייה ממערך של 28\*28 פיקסלים, כאשר כל פיקסל שמור כמספר בין 0 ל-255. בפעילות זו נהפוך את התמונה לתמונה בניארית כלומר תמונה שבה כל פיקסל יקבל 2 ערכים בלבד, לבן או

שחור. נעשה את המהלך על ידי כך שנחלק כל אחד מהמספרים בקובץ התמונות ב- 255 באופן זה. תהליך זה מכונה נרמול נתונים או טרנספורמציה של נתונים והוא חלק משלב הוצאת feature במכונות לומדות בכלל ובתכנות רשתות ניורונים בפרט.

להלן קטע הקוד המסיים את שלב הכנת הנתונים:

```
x_mnist = x_mnist / 255
train_data, train_lbl = x_mnist[:60000], y_mnist[:60000]
test_data, test_lbl = x_mnist[60000:70000], y_mnist[60000:70000]
```

השלב האחרון בתרגיל שלנו יהיה ליצור את רשת הניורונים כרשת הכוללת 784 ניורונים בשכבת המבוא, 50 ניורונים בשכבה פנימית ו- 10 ניורונים בשכבת המוצא.

```
mlp = MLPClassifier(hidden_layer_sizes=(50,), max_iter=20, alpha=1e-4,
                    solver='sgd', verbose=10, tol=1e-4, random_state=1,
                    learning_rate_init=.1)
```

נאמן את המכונה על ידי הפעולה fit ונספק לה את מערך נתוני האימון:

```
mlp.fit(train_data, train_lbl)
```

לבסוף נבדוק את איכות המערכת ב- 2 דרכים:

הדרך הראשונה על ידי הפעולה score שבדוקת את כמות השגיאות ביחס לכמות ההצלחות ומחזירה לנו מספר בין 0 ל- 1

```
print("Training set score: %f" % mlp.score(train_data, train_lbl))
print("Test set score: %f" % mlp.score(test_data, test_lbl))
```

נקבל את הפלט הבא:

```
Training set score: 0.996883
Test set score: 0.972300
```

ניתן לראות שהמכונה הצליחה לסווג בהצלחה 99.67% ממערך נתוני האימון. ו- 97.23% מכלל נתוני הבדיקה. במילים אחרות המערכת טובה יותר בזיהוי תמונות שכבר ראתה מאשר בזיהוי תמונות שלא פגשה בהם בעבר. כמובן ניקח את הנתונים בפרופורציה כי המכונה הצליחה לסווג 9,723 תמונות שמעולם לא פגשה ונכשלה בסיווג 277 תמונות.

כמובן שאימון נוסף של המכונה יגרור שיפור בתוצאות. מספיק שנגדיל את מספר מחזורי האימון על ידי המאפיין max\_iter מ- 20 ל- 50 max\_iter נקבל את התוצאות הבאות:

```
Training set score: 1.000000
Test set score: 0.973100
```

כלומר 100% הצלחה בזיהוי מערך תמונות האימון ו- 97.31% בזיהוי תמונות ממערך הבדיקה (כלומר שיפור קטנה ביכולת הזיהוי של התמונות ממערך הבדיקה).



דרך נוספת לבחון את יכולת המכונה יהיה על ידי הקוד הבא:

```
print("\nTest data: ",test_lbl[:30])
newp = mlp.predict(test_data[:30])
print ("\nPredict data: ",newp)
```

נקבל את הפלט הבא:

```
Test data:      [7 2 1 0 4 1 4 9 5 9 0 6 9 0 1 5 9 7 3 4 9 6 6 5 4 0 7 4 0 1]
Predict data:  [7 2 1 0 4 1 4 9 5 9 0 6 9 0 1 5 9 7 3 4 9 6 6 5 4 0 7 4 0 1]
```

גם כאן נקבל התאמה בין נתוני הבדיקה לנתוני הסיווג של המכונה.

להלן הקוד המלא של התרגיל לאימון מכונה לומדת לזיהוי ספרות הכתובות בכתב יד:

```
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path
from sklearn.datasets import fetch_openml
from sklearn.neural_network import MLPClassifier

datapath = Path("mnist_784.npz")
if not(datapath.exists()):
    print("downloading file...")
    x_mnist, y_mnist = fetch_openml("mnist_784", version=1,
                                    return_X_y=True, data_home=".")
    x=np.array(x_mnist,dtype="u8")
    y=np.array(y_mnist,dtype="u8")
    np.savez(datapath,x=x,y=y)
    del x_mnist, y_mnist

print("Loading file...")
data = np.load(datapath)
print("\ntype: ", type(data))
print("\ntype: ", data.files)
x_mnist = data["x"]
x_mnist = x_mnist / 255
y_mnist = data["y"]
```

```

train_data, train_lbl = x_mnist[:60000], y_mnist[:60000]
test_data, test_lbl = x_mnist[60000:70000], y_mnist[60000:70000]

plt.figure()
for i in range(200):
    plt.subplot(10,20,i+1)
    plt.axis("off")
    plt.imshow(train_data[i].reshape((28,28))*255,cmap="gray", vmin=0, vmax=255)
plt.show()

mlp = MLPClassifier(hidden_layer_sizes=(50,), max_iter=20, alpha=1e-4,
                    solver='sgd', verbose=10, tol=1e-4, random_state=1,
                    learning_rate_init=.1)

mlp.fit(train_data, train_lbl)
print("Training set score: %f" % mlp.score(train_data, train_lbl))
print("Test set score: %f" % mlp.score(test_data, test_lbl))

print("\nTest data: ",test_lbl[:30])
newp = mlp.predict(test_data[:30])
print ("\nPredict data: ",newp)

```

## תרגיל

הוספת רעש לתמונות

<https://debuggercafe.com/adding-noise-to-image-data-for-deep-learning-data-augmentation/>

## תרגיל:

נחזור לתרגיל ים/ישיבה שבו סיווגנו תמונות נוף ים ותמונות נוף יבשה ונממש את הנכונה תוך שימוש בספרייה `sklearn.neural_network`.

בתרגיל ניקח 20 התמונות המתוייגות חלקם ים וחלקם יבשה ונלמד את המחשב לבצע את הסיווג. בהמשך נציג למחשב 6 תמונות חדשות ונבדוק את יכולת המכונה.

```

from PIL import Image
import numpy as np
from sklearn.neural_network import MLPClassifier

#-----start get data from images-----
sea_colors = list()
for i in range(10):
    img = Image.open("data/sea" + str(i) + ".jpg")
    img.load()
    data = np.array(img, dtype=np.uint8)
    sea_colors.append([data[:, :, color].sum() / data[:, :, color].size for color in range(3)])

land_colors = list()
for i in range(10):
    img = Image.open("data/land" + str(i) + ".jpg")
    img.load()
    data = np.array(img, dtype=np.uint8)
    land_colors.append([data[:, :, color].sum() / data[:, :, color].size for color in range(3)])

test_colors = list()
for i in range(6):
    img = Image.open("test/test" + str(i) + ".jpg")
    img.load()
    data = np.array(img, dtype=np.uint8)
    test_colors.append([data[:, :, color].sum() / data[:, :, color].size for color in range(3)])
#-----end get data from images-----

#-----Preparing the data for machine learned-----
sea_array = np.array(sea_colors)
land_array = np.array(land_colors)
test_array = np.array(test_colors)

```

```

x1 = sea_array[:,1]
y1 = sea_array[:,2]
x2 = land_array[:,1]
y2 = land_array[:,2]
xtest = test_array[:,1]
ytest = test_array[:,2]

x = list()
y = list()

x = np.append(x1,x2)
y = np.append(y1,y2)

data = np.stack((x, y), axis=-1)
test_data = np.stack((xtest, ytest), axis=-1)

lbl=[0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1]
#-----End Preparing the data for machine learned-----

mlp = MLPClassifier(hidden_layer_sizes=(5,),max_iter=1000,
                    solver='sgd', verbose=10, random_state=1)

mlp.fit(data, lbl)
print("Training set score: %f" % mlp.score(data, lbl))
print("Test set score: %f" % mlp.score(test_data, lbl))

newp = mlp.predict(test_data)
print (newp)

```

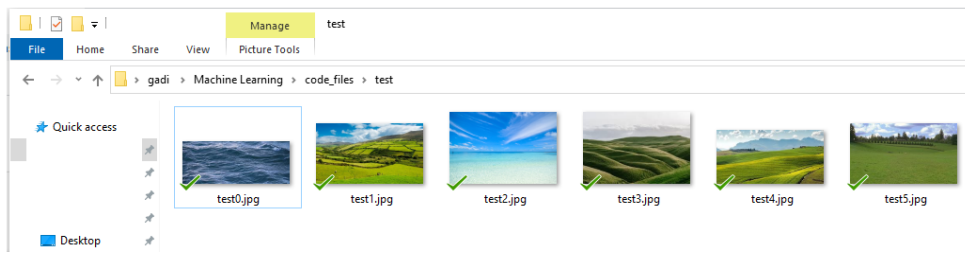
נקבל את הפלט הבא (כאשר 0 זה ים ו-1 זה יבשה):

```

Training set score: 1.000000
Test set score: 1.000000
[0 1 0 1 1 1]

```

נבדוק את התוצאות מול מערך תמונות הבדיקה:



קיבלנו התאמה מלאה בין פלט המכונה לבין סדר התמונות המסופק למכונה.

## תנאי השימוש

תנאי השימוש במסמך זה הם לפי הסטנדרט הבא:

You are free:

to Share – to copy, distribute and transmit the material  
to Remix – to adapt the material

Under the following conditions:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

NonCommercial — You may not use the material for commercial purposes.

ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.