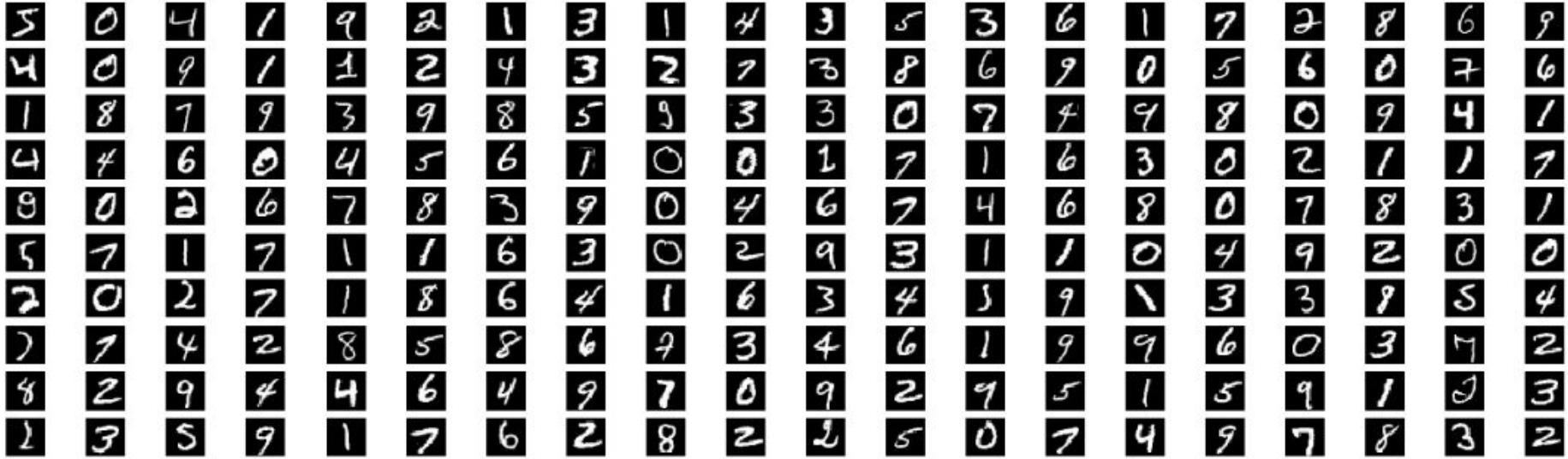
זיהוי ספרות בכתב ידי scikit-learn

גדי הרמן

מאגר ספרות בכתב יד - mnist_784

בשיעור זה נתרגל כתיבת קוד בשפת python המאמן מכונה לומדת לזיהוי מספרים מתוך מאגר תמונות. המאגר כולל 70000 תמונות מתויגות הנראות כך:



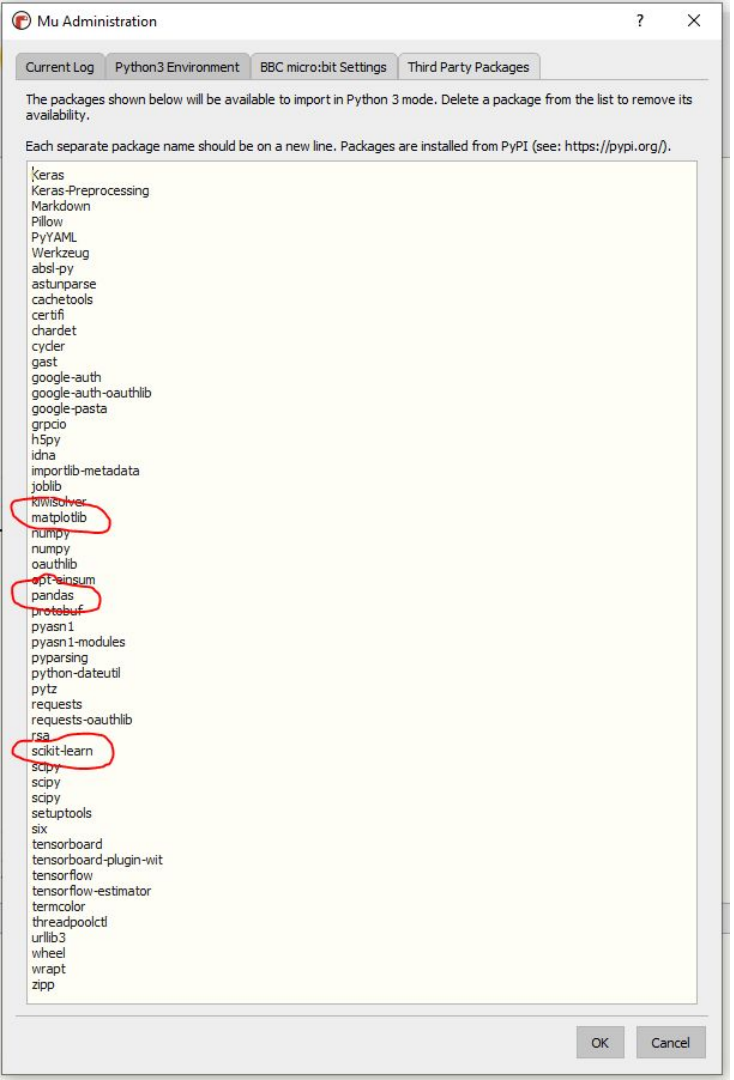
הספריה Scikit-learn

- 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.
- הספריה נכתבה במקור על ידי David Cournapeau כחלק מפרויקט של גוגל.
- ניתן לראות שימוש של הספריה כחלק ממערך המלצות השירים באפליקציה של Spotify כמו גם נעשה שימוש בספריה Scikit-learn באתר Booking.com כדי לספק לגולשים המלצות על מלונות ויעדים לטיול כמו גם איתור הונאות במערך ההזמנות של החברה.

Booking.com

Spotify®

התקנות




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

חזרה על אופן השימוש בספריה Scikit-learn

נתחיל בכתיבת קוד למכונה לומדת המבוססת על הספריה Scikit-learn כדי לממש שער XOR.

```
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='logistic',
                max_iter=100,
                hidden_layer_sizes=(2,),
                solver='lbfgs')
model.fit(inputs, expected_output)
print('predictions:', model.predict(inputs))
```

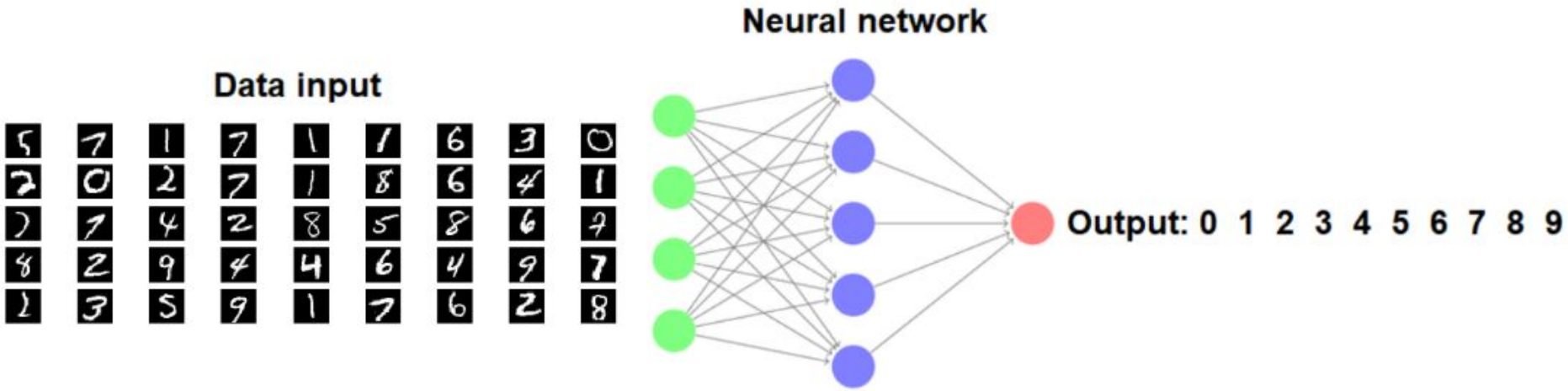
חזרה על אופן השימוש בספריה Scikit-learn

```
model = sklearn.neural_network.MLPClassifier(
                activation='logistic',
                max_iter=100,
                hidden_layer_sizes=(2,),
                solver='lbfgs')
```

- 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.

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

מערך הנתונים כולל 70000 תמונות

כל תמונה כוללת 784 פיקסלים (כלומר כל תמונה היא ברזולוציה של 28*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:

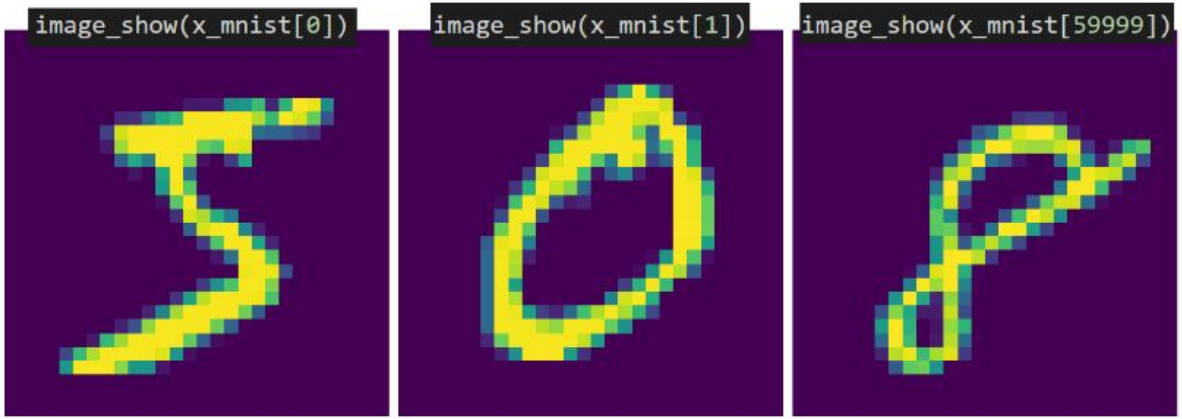
   https://github.com/amplab/datascience-sp14/raw/master/lab7/mldata/mnist-original.mat

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 שבספריה sklearn.datasets תדאג לכך. החיסרון הוא שצריך להמתין כחצי דקה בכל פעם שמפעילים את היישום.

ארגון נתוני האמון

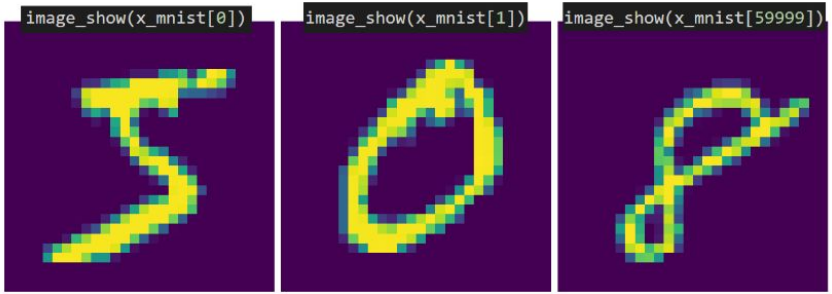
```
import sklearn.datasets

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.



ארגון נתוני האמון

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])
```

ארגון נתוני האמון

יש 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

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

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

ארגון נתוני האמון

יש 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

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

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

זהירות ! ערכים עלולים להתפוצץ.

ברשתות רדודות ועמוקות עלול להיווצר מצב שבו ערכי W יגדלו מאד או יקטנו מאד.

כדי למנוע ערכים גדולים מדי נבצע נרמול (המרה) של הקלט לערכים סביב המספר 1. כך המכפלות לא יהיו גדולות מדי.

ליישום הגישה נוסיף לקוד שלנו את ההוראה הבא

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

```
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]
```

המלצה לבניית רשת הנוירונים

```
mlp = MLPClassifier(
            hidden_layer_sizes=(50,),
            max_iter=20,
            solver='sgd',
            verbose=True,
            learning_rate_init=0.1)
```

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

תרגול 9:
זיהוי ספרות בכתב ידי
scikit-learn

שמירת מערך המשקלים.

```
model = sklearn.neural_network.MLPClassifier(
                activation='logistic',
                max_iter=100,
                hidden_layer_sizes=(3),
                solver='lbfgs')
model.fit(inputs, expected_output)
print('\npredictions:', model.predict(inputs))
print(model.n_iter_)
print('\nweight layer 0',model.coefs_[0])
print('\nbias vector corresponding to layer 0',model.intercepts_[0])
print('\nweight layer 1',model.coefs_[1])
print('\nbias vector corresponding to layer 1',model.intercepts_[1])
```

שמירת מערך המשקלים.

```
model = sklearn.neural_network.MLPClassifier(
            activation='logistic',
            max_iter=100,
            hidden_layer_sizes=(3),
            solver='lbfgs')
model.fit(inputs, expected_output)
print('\npredictions:', model.predict(inputs))
print(model.n_iter_)
print('\nweight layer 0',model.coefs_[0])
print('\nbias vector corresponding to layer 0',model.intercepts_[0])
print('\nweight layer 1',model.coefs_[1])
print('\nbias vector corresponding to layer 1',model.intercepts_[1])
```

```
weight layer 0 [[-6.32381756 -1.16055328  4.18132812]
 [-0.51685897 -0.36630015  0.15869511]
 [-0.07310384 -0.45925841 -0.60534783]]

bias vector corresponding to layer 0 [ 2.71536983 -2.08261883 -2.03237058]

weight layer 1 [[11.49217622]
 [-0.64307417]
 [-8.85569543]]

bias vector corresponding to layer 1 [-1.00915571]
```

הספריה pickle

ספריה הממירה מבני נתונים, object structure , לקובץ בינארי ולהיפך.

לדוגמה:

```
import pickle

example_dict = {1:"6",2:"2",3:"f"}

pickle_out = open("dict.pickle","wb")
pickle.dump(example_dict, pickle_out)
pickle_out.close()

pickle_in = open("dict.pickle","rb")
p_dict = pickle.load(pickle_in)

print(p_dict)
print(p_dict[3])
```

```
{1: '6', 2: '2', 3: 'f'}
f
```

שמירת מערך המשקלים.

```
import pickle
```

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

mlp.fit(train_data, train_lbl)


# save the model to disk
filename = 'finalized_model.sav'
pickle.dump(mlp, open(filename, 'wb'))
```

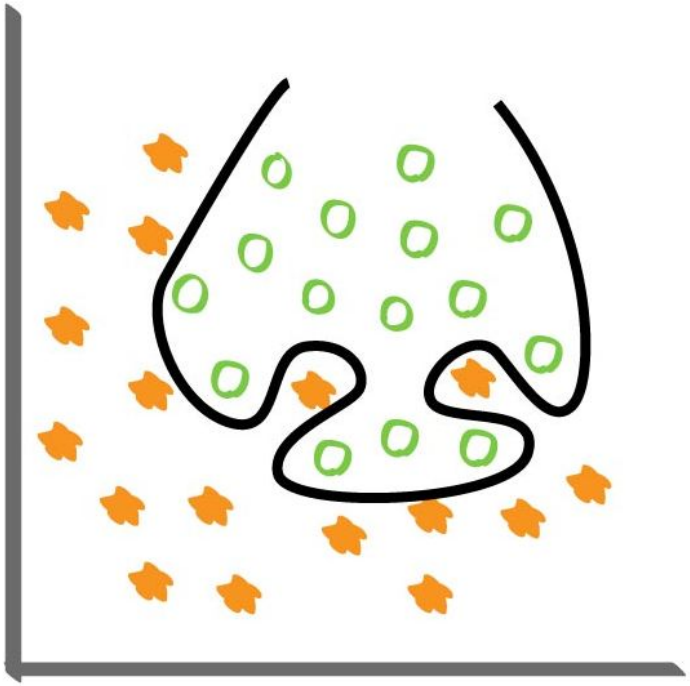
שמירת מערך המשקלים.

```
import pickle

# load the model from disk
filename = 'finalized_model.sav'
mlp2 = pickle.load(open(filename, 'rb'))

x = Image.open('4.png').resize( ( 28 , 28 ) ).convert( '1' )
x = np.array(x)
plt.axis("off")
plt.imshow(x.reshape((28,28))*255,cmap="gray", vmin=0, vmax=255)
plt.show()
lbl_pred = mlp2.predict(x.reshape((1,784)))
print ("\nPredict data: ",lbl_pred)
```

תרגול 9:
זיהוי ספרות בכתב ידי
scikit-learn
המשך

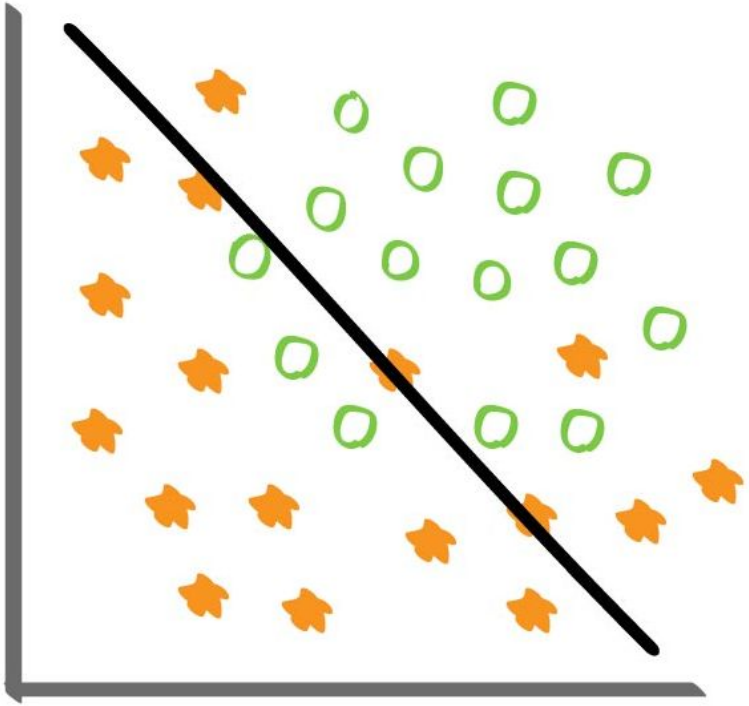
התאמת יתר (overfitting)

התאמת יתר (overfitting) היא בעיה בסטטיסטיקה ובלמידת מכונה שבה המודל מותאם יתר על המידה לאוסף מסוים של נתונים.

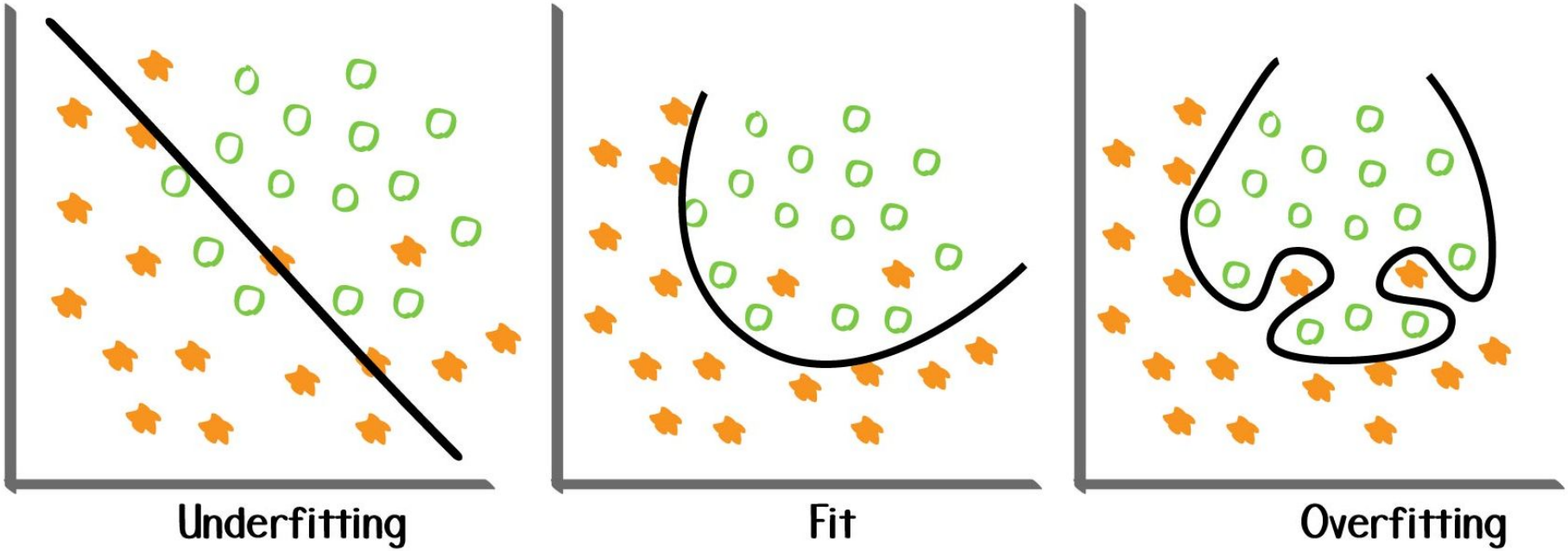


תת התאמה (Underfitting)

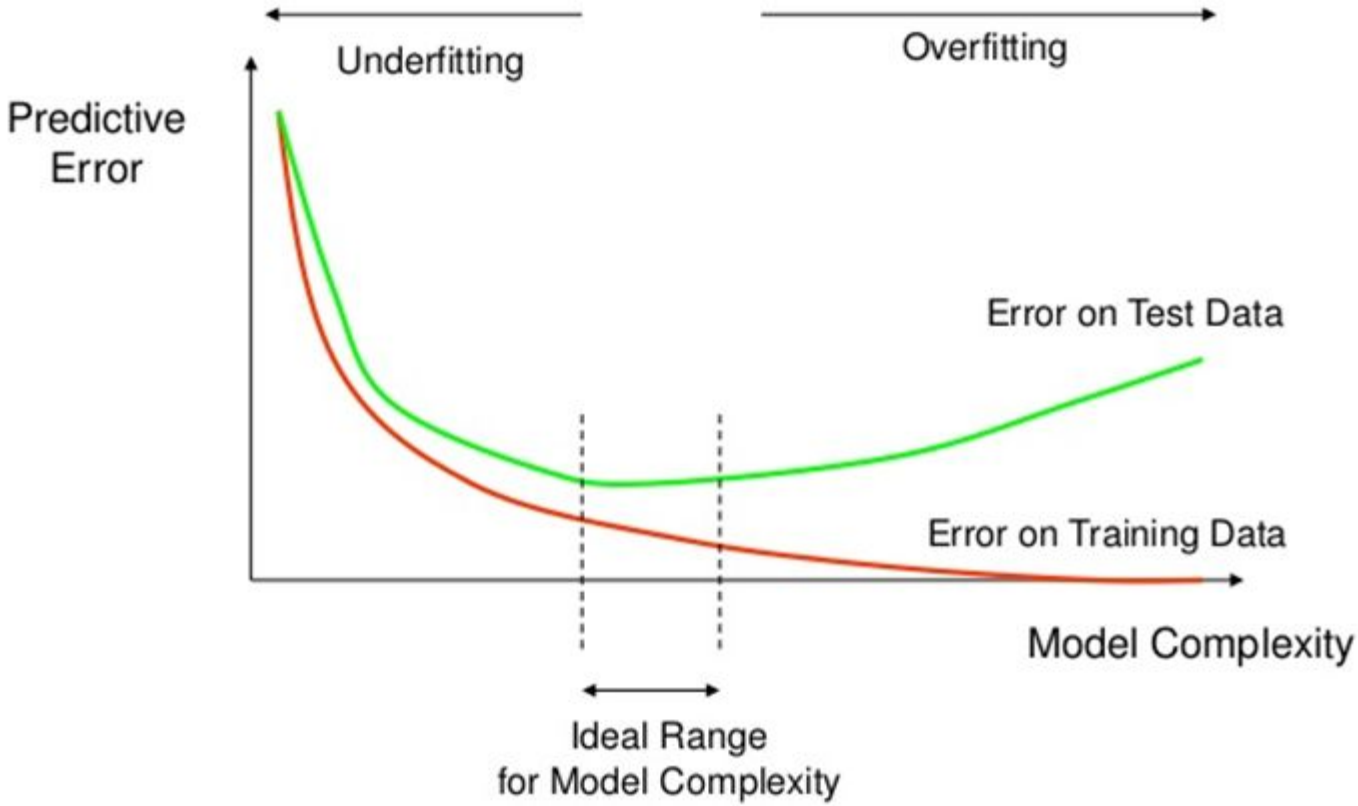
תת התאמה (Underfitting) מתרחשת כאשר המודל הסטטיסטי פשוט מדי מכדי לייצג כראוי את הנתונים



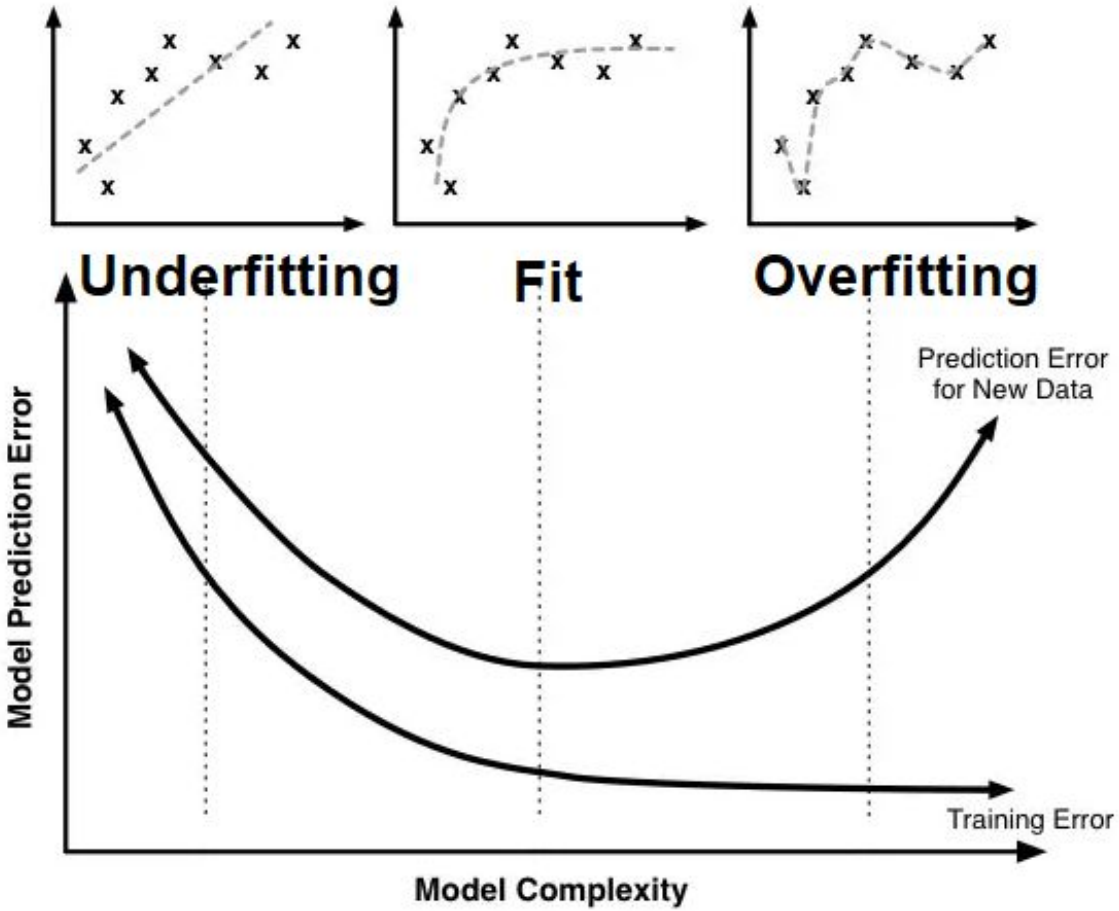
השוואה בין התאמת יתר (overfitting)
לבין תת התאמה (Underfitting)



התאמת יתר (overfitting) ותת התאמה (Underfitting) בגף

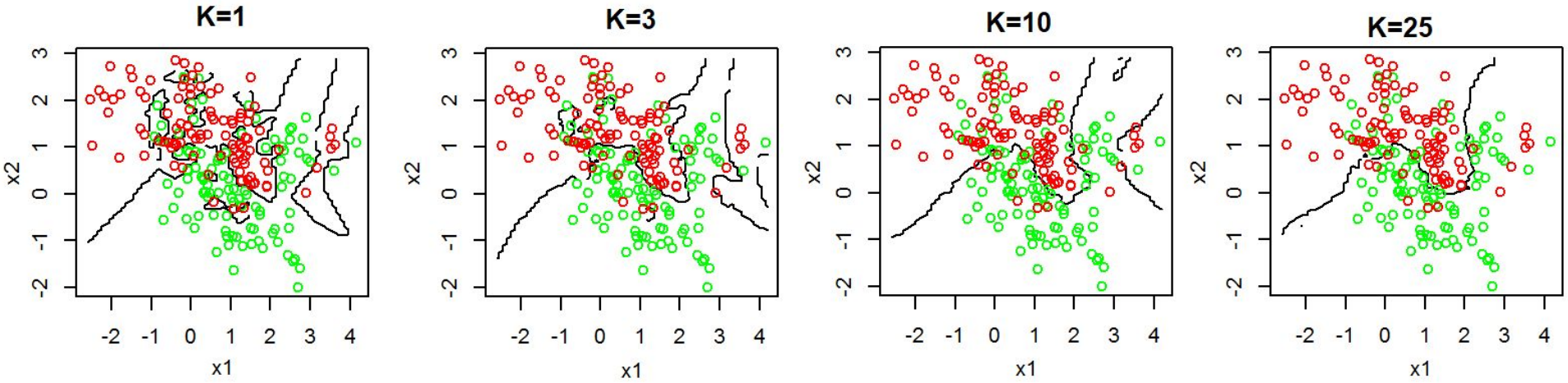


התאמת יתר (overfitting) ותת התאמה (Underfitting) בגף



אלגוריתם KNN

השוואה בין התאמת יתר (overfitting) לבין תת התאמה (Underfitting)



תרגול לבדיקת השפעת הפרמטר K
באלגוריתם KNN על accuracy_score

```
all_accuracy_score = []
for i in range(1,40):
    classifier = KNeighborsClassifier(n_neighbors=i)
    classifier.fit(train_data, train_lbl)
    lbl_pred = classifier.predict(test_data)
    score = accuracy_score(test_lbl, lbl_pred)
    all_accuracy_score.append(score)
```

בדיקת השפעת הפרמטר K
באלגוריתם KNN על
accuracy_score

פתרון

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from scipy.ndimage.filters import gaussian_filter1d
np.random.seed(0)
vir_iris_data = np.genfromtxt('iris_for_ML.csv', delimiter=',')
random_iris_data = np.random.permutation(vir_iris_data)
test_data = random_iris_data[0:50,:4]
train_data = random_iris_data[50:,:4]
test_lbl = np.ravel(random_iris_data[0:50,4:])
train_lbl = np.ravel(random_iris_data[50:,4:])
all_accuracy_score = []
for i in range(1,40):
    classifier = KNeighborsClassifier(n_neighbors=i)
    classifier.fit(train_data, train_lbl)
    lbl_pred = classifier.predict(test_data)
    score = accuracy_score(test_lbl, lbl_pred)
    all_accuracy_score.append(score)
plt.plot(np.arange(1, 40), gaussian_filter1d(all_accuracy_score, sigma=3))
plt.xlabel("K")
plt.ylabel("Accuracy Score")
plt.show()
```