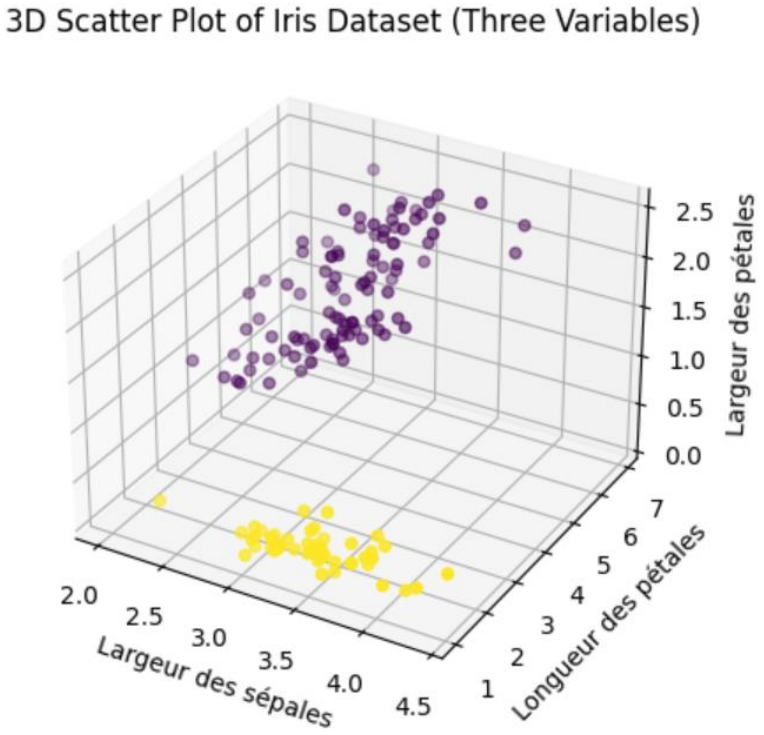
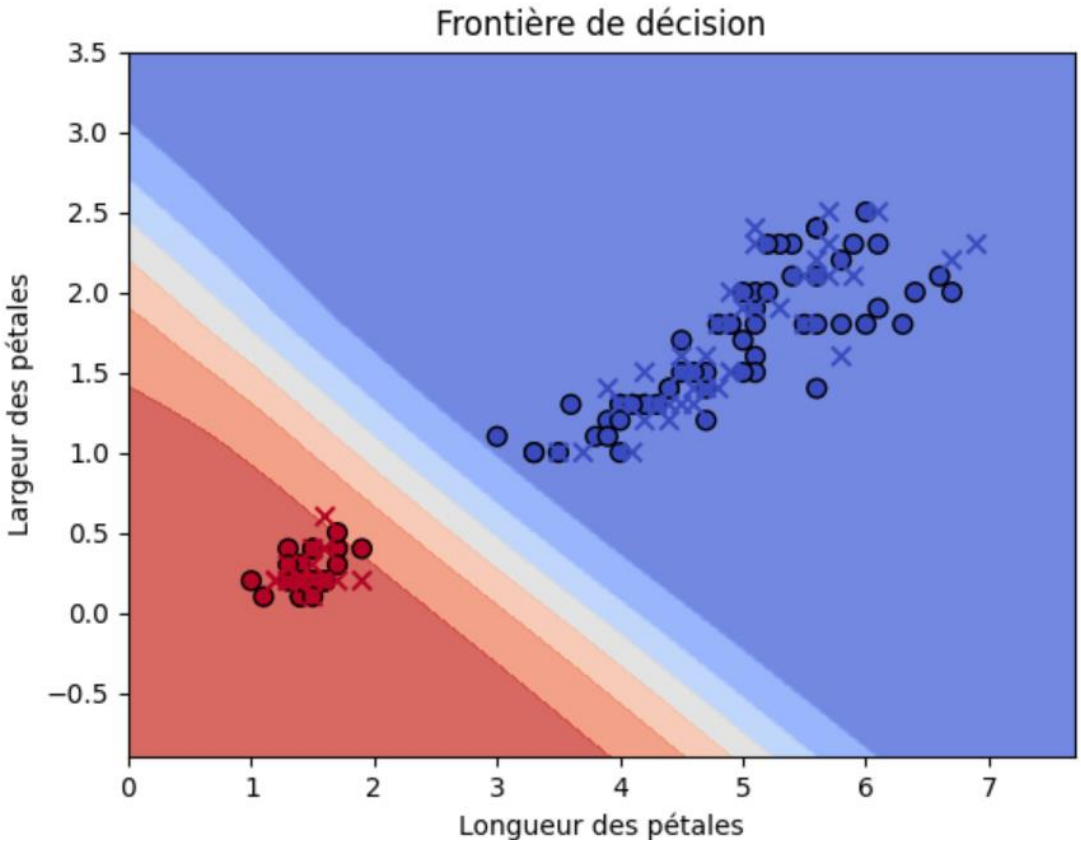
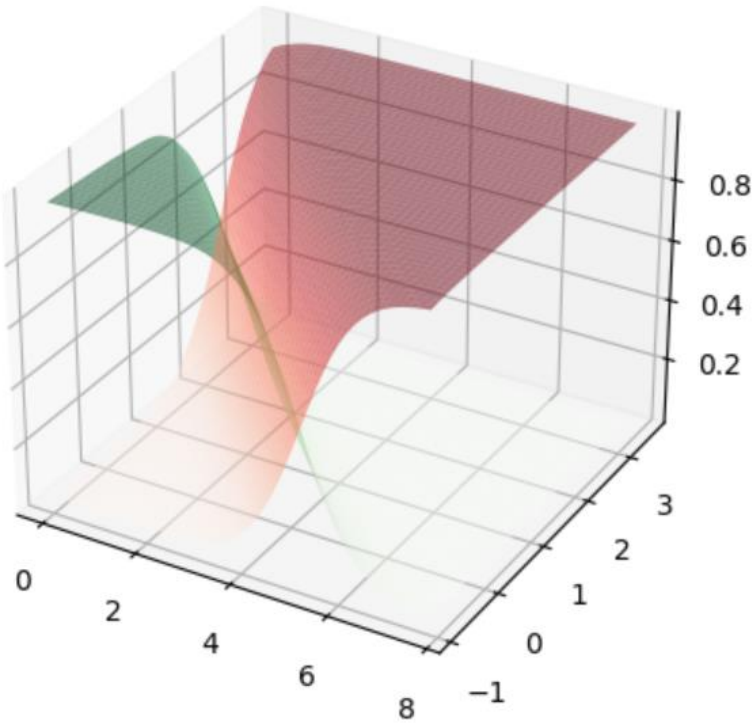
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import
load_iris
from sklearn.model_selection import train_test_split
from sklearn.neural_network
import MLPClassifier
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import
cm


donnees = load_iris()
data = donnees.data
target = donnees.target

l1c =
np.ones(50, dtype=int)
l2c = np.zeros(100, dtype=int)
target_binary =
np.concatenate((l1c, l2c))

np.random.seed(10)
X_train, X_test, y_train, y_test =
train_test_split(data[:, 2:4], target_binary, test_size=0.4)

# Exercice 1
fig =
plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(data[:, 1],
data[:, 2], data[:, 3], c=target_binary, cmap='viridis')
ax.set_xlabel('Largeur
des sépales')
ax.set_ylabel('Longueur des
pétales')
ax.set_zlabel('Largeur des pétales')
plt.title('3D Scatter
Plot of Iris Dataset (Three Variables)')
plt.show()

clf =
MLPClassifier(solver='sgd', learning_rate_init=0.01, max_iter=10000,
random_state=0)
clf.fit(X_train, y_train)

train_score = clf.score(X_train,
y_train)
test_score = clf.score(X_test, y_test)
print(f"Score en apprentissage :
{train_score}")
print(f"Score en test : {test_score}")

plt.figure()
nx, ny
= 200, 200
x_min, x_max = X_train[:, 0].min() - 1, X_train[:, 0].max() + 1
y_min, y_max =
X_train[:, 1].min() - 1, X_train[:, 1].max() + 1
xx, yy = np.meshgrid(np.linspace(x_min,
x_max, nx), np.linspace(y_min, y_max, ny))
Z = clf.predict_proba(np.c_[xx.ravel(),
yy.ravel()])[:, 1]
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.8,
cmap='coolwarm')
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train,
edgecolor='k', cmap='coolwarm', s=50)
plt.scatter(X_test[:, 0], X_test[:,
1], c=y_test, edgecolor='k', cmap='coolwarm', s=50,
marker='x')
plt.xlabel('Longueur des pétales')
plt.ylabel('Largeur
```

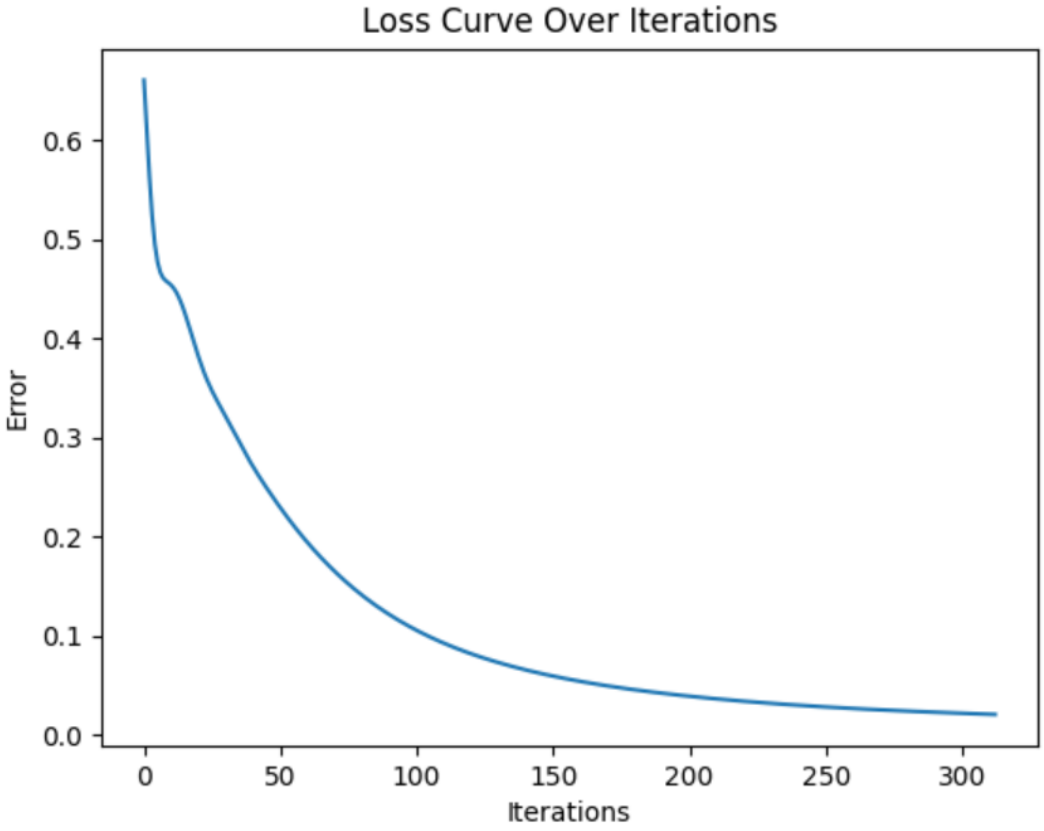
```
des pétales')
plt.title('Frontière de décision')
plt.show()

# Exercice
2
Z = clf.predict_proba(np.c_[xx.ravel(), yy.ravel()])
print("Somme des probabilités
par point :", np.allclose(Z.sum(axis=1), 1))

fig = plt.figure()
ax =
fig.add_subplot(111, projection='3d')
Z0 = Z[:, 0].reshape(xx.shape)
Z1 = Z[:,
1].reshape(xx.shape)
ax.plot_surface(xx, yy, Z0, alpha=0.5,
cmap='Reds')
ax.plot_surface(xx, yy, Z1, alpha=0.5,
cmap='Greens')
plt.show()

# Exercice 3
versicolor_column = 1

print("Column {} of Z corresponds to
Versicolor".format(versicolor_column))

plt.figure()
plt.xlabel('Iterations')

plt.ylabel('Error')
plt.plot(clf.loss_curve_)
plt.title('Loss Curve Over
Iterations')
plt.show()

# Exercice 4
activations = ['identity',
'logistic', 'tanh', 'relu']
for activation in activations:

clf = MLPClassifier(activation=activation, max_iter=1000, random_state=100)

clf.fit(X_train, y_train)
    train_score = clf.score(X_train, y_train)
    test_score =
clf.score(X_test, y_test)
    print(f"Fonction d'activation {activation}: Train
Score = {train_score}, Test Score = {test_score}")

# Exercice 5
hidden_sizes = [2, 10,
50, 100, 150, 200]
train_scores = []
test_scores = []
for size in hidden_sizes:
    clf =
MLPClassifier(hidden_layer_sizes=(size,), max_iter=3000, random_state=0)
    clf.fit(X_train,
y_train)
    train_scores.append(clf.score(X_train, y_train))

test_scores.append(clf.score(X_test, y_test))

plt.figure()
plt.plot(hidden_sizes,
train_scores, 'go-', label='Train Score')
plt.plot(hidden_sizes,
test_scores, 'r+-', label='Test Score')
plt.xlabel('Nombre de
neurones cachés')
plt.ylabel('Score')
plt.legend()
plt.show()
```

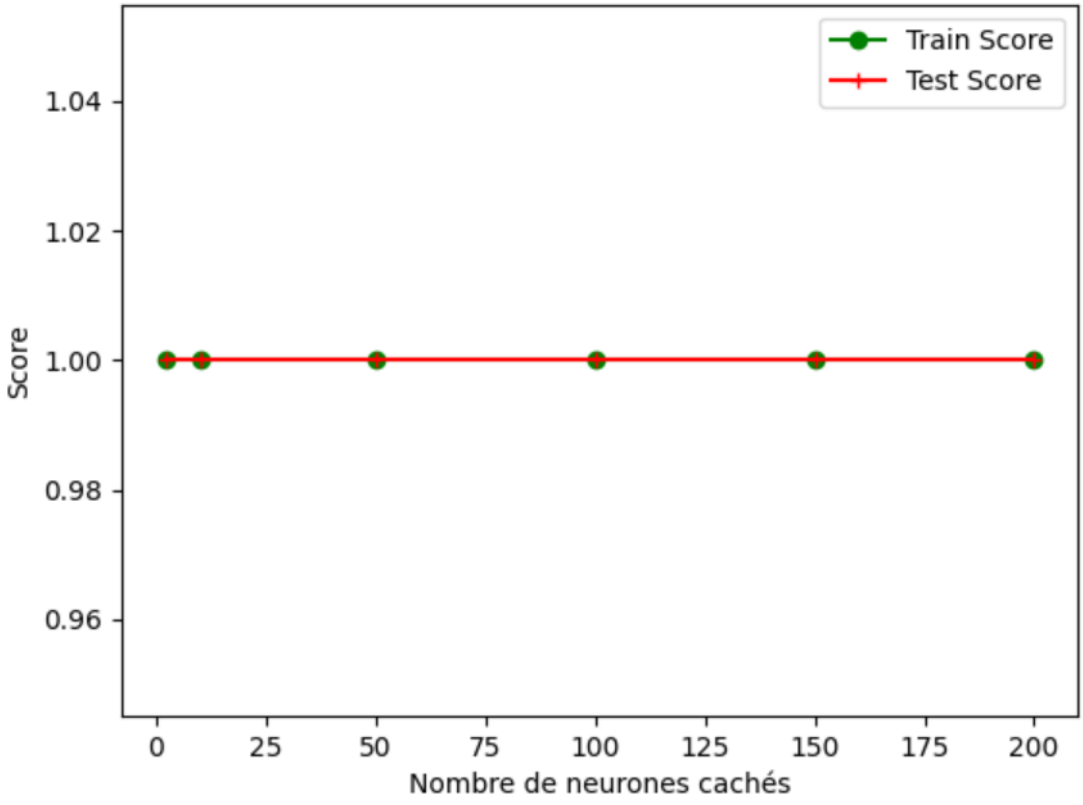
```
# Exercice
6
layer_counts = [2, 3, 4, 5]
train_scores = []
test_scores = []
for count in
layer_counts:
    clf = MLPClassifier(hidden_layer_sizes=(10,) * count, max_iter=3000,
random_state=0)
    clf.fit(X_train, y_train)
    train_scores.append(clf.score(X_train,
y_train))
    test_scores.append(clf.score(X_test,
y_test))

plt.figure()
plt.plot(layer_counts, train_scores, 'go-',
label='Train Score')
plt.plot(layer_counts, test_scores, 'r+-',
label='Test Score')
plt.xlabel('Nombre de couches
cachées')
plt.ylabel('Score')
plt.legend()
plt.show()

# Exercice
7
learning_rates = [0.001, 0.01, 0.1]
for lr in learning_rates:
    clf_lr =
MLPClassifier(solver='sgd', learning_rate_init=lr, max_iter=10000,
random_state=100)
    clf_lr.fit(X_train, y_train)
    train_score_lr = clf_lr.score(X_train,
y_train)
    test_score_lr = clf_lr.score(X_test, y_test)
    print(f"Taux
d'apprentissage {lr} : score en train = {train_score_lr}, score en test =
{test_score_lr}")

# Exercice 8
alphas = [0.0001, 0.001, 0.01]
for alpha in alphas:

  clf_alpha = MLPClassifier(alpha=alpha, max_iter=10000, random_state=100)

clf_alpha.fit(X_train, y_train)
    train_score_alpha = clf_alpha.score(X_train, y_train)

test_score_alpha = clf_alpha.score(X_test, y_test)
    print(f"Alpha {alpha} : score en
train = {train_score_alpha}, score en test = {test_score_alpha}")

# Analyse:
# Le
modèle MLPClassifier montre une bonne capacité de généralisation avec des scores de test
élevés.
# Les variations des paramètres (activation, nombre de neurones, couches, taux
d'apprentissage, alpha) influencent les performances du modèle.
```

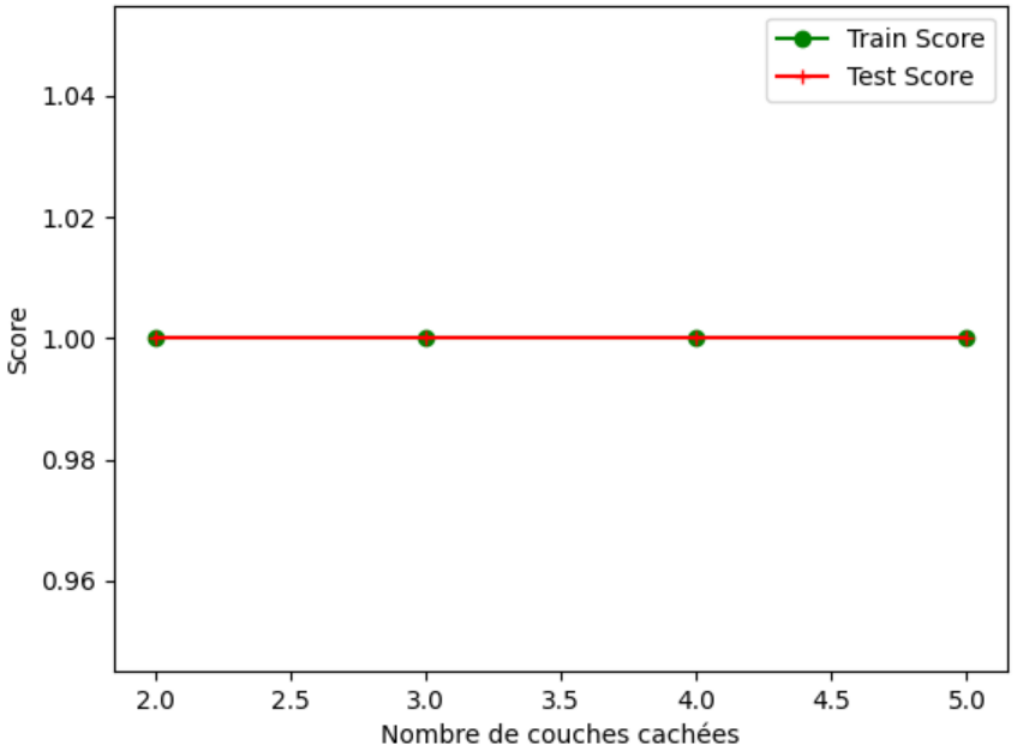

3D Scatter Plot of Iris Dataset (Three Variables)
Largeur des sépales
Longueur des pétales
Largeur des pétales


Frontière de décision
Largeur des pétales
Longueur des pétales


0.8
0.6
0.4
0.2
0
2
4
6
8
−1
0
1
2
3


Loss Curve Over Iterations
Error
Iterations
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
50
100
150
200
250
300


Train Score
Test Score
1.04
1.02
1.00
0.98
0.96
Score
0
25
50
75
100
125
150
175
200
Nombre de neurones cachés


Train Score
Test Score
1.04
1.02
1.00
0.98
0.96
Score
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Nombre de couches cachées

```
Score en apprentissage : 1.0
Score en test : 1.0
Somme des probabilités par point : True
Column 1 of Z corresponds to Versicolor
Fonction d'activation identity: Train Score = 1.0, Test Score = 1.0
Fonction d'activation logistic: Train Score = 1.0, Test Score = 1.0
Fonction d'activation tanh: Train Score = 1.0, Test Score = 1.0
Fonction d'activation relu: Train Score = 1.0, Test Score = 1.0
Taux d'apprentissage 0.001 : score en train = 1.0, score en test = 1.0
Taux d'apprentissage 0.01 : score en train = 1.0, score en test = 1.0
Taux d'apprentissage 0.1 : score en train = 1.0, score en test = 1.0
Alpha 0.0001 : score en train = 1.0, score en test = 1.0
Alpha 0.001 : score en train = 1.0, score en test = 1.0
Alpha 0.01 : score en train = 1.0, score en test = 1.0
```