JOBSHEET 9

Praktikum 1

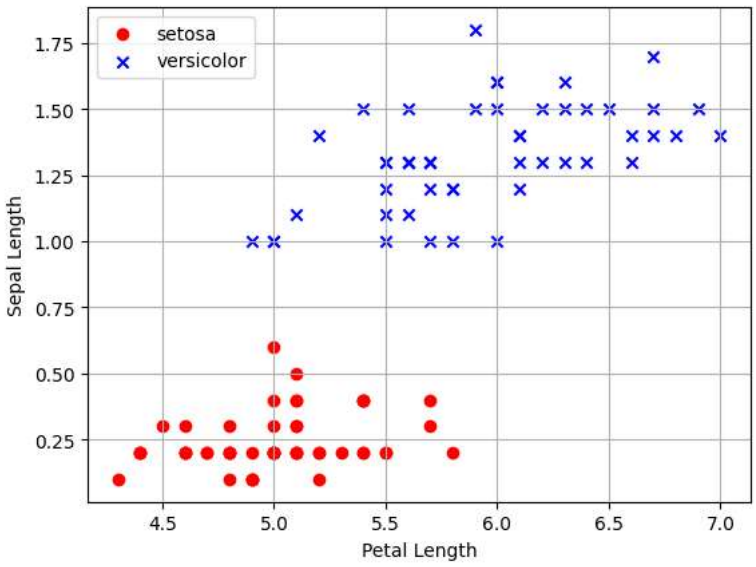
Klasifikasi Iris dengan Perceptron

Langkah 1 - Import Library

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

Langkah 2 - Load Data dan Visualisasi

```
df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data', header=None)
setosa = df[df[4] == 'Iris-setosa']
versicolor = df[df[4] == 'Iris-versicolor']
virginica = df[df[4] == 'Iris-virginica']

a, b = 0, 3
plt.scatter(setosa[a], setosa[b], color='red', marker='o', label='setosa')
plt.scatter(versicolor[a], versicolor[b], color='blue', marker='x', label='versicolor')

plt.xlabel('Petal Length')
plt.ylabel('Sepal Length')
plt.legend(loc='upper left')
plt.grid()
plt.show()
```



Langkah 3 - Membuat Kelas Perceptron

```
class Perceptron(object):
    def __init__(self, eta=0.01, n_iter=10):
        self.eta = eta
        self.n_iter = n_iter

    def fit(self, X, y):

        self.w_ = np.zeros(1 + X.shape[1])
        self.errors_ = []

        for _ in range(self.n_iter):
            errors = 0
            for xi, target in zip(X, y):
                update = self.eta * (target - self.predict(xi))
                self.w_[0] += update
                self.w_[1:] += update * xi
                errors += int(update != 0.0)
            self.errors_.append(errors)
        return self

    def net_input(self, X):
        return np.dot(X, self.w_[1:]) + self.w_[0]

    def predict(self, X):
        return np.where(self.net_input(X) >= 0.0, 1, -1)
```

Langkah 4 - Pilih Data dan Encoding Label

```
y = df.iloc[0:100, 4].values # pilih 100 data awal
y = np.where(y == 'Iris-setosa', -1, 1) # ganti coding label
X = df.iloc[0:100, [0, 3]].values # slice data latih
```

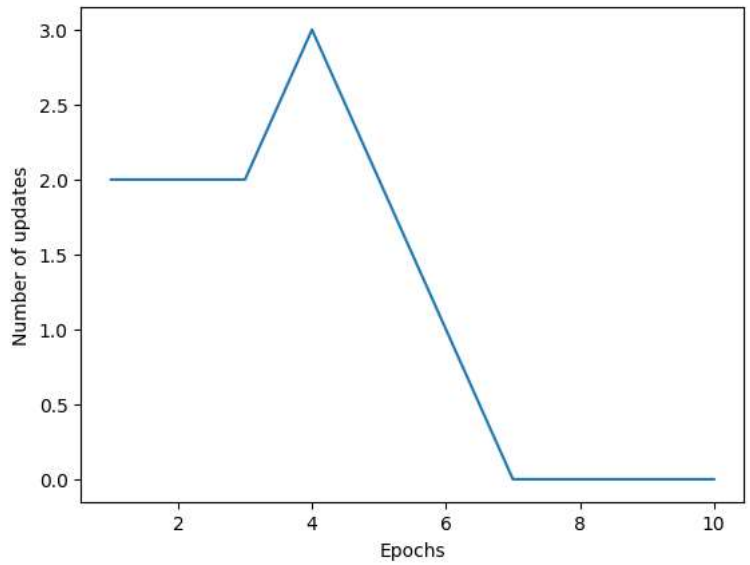
Langkah 5 - Fitting Model

```
ppn = Perceptron(eta=0.1, n_iter=10)
ppn.fit(X, y)
```

```
<__main__.Perceptron at 0x7e8dc1b32b30>
```

Langkah 6 - Visualisasi Nilai Error Per Epoch

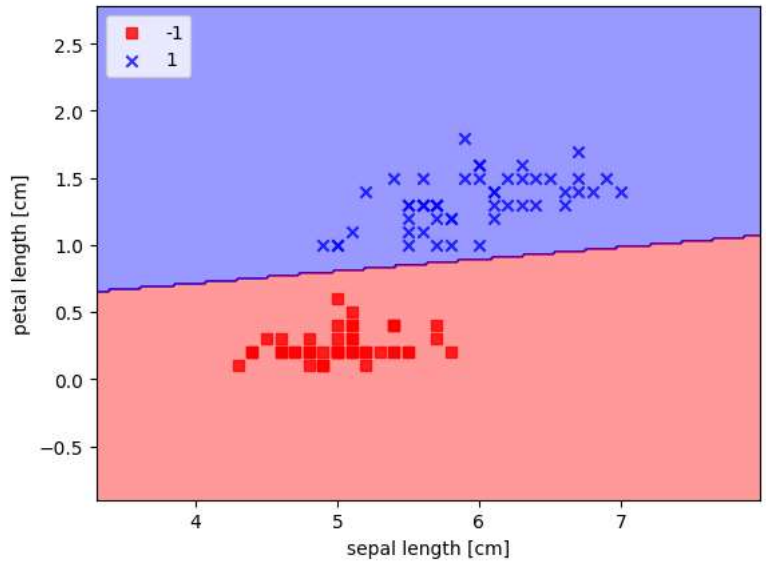
```
plt.plot(range(1, len(ppn.errors_)+1), ppn.errors_)
plt.xlabel('Epochs')
plt.ylabel('Number of updates')
plt.show()
```



Langkah 7 - Visualiasasi Decision Boundary

```
# buat fungsi untuk plot decision region

from matplotlib.colors import ListedColormap

def plot_decision_regions(X, y, classifier, resolution=0.02):
    # setup marker generator and color map
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('r', 'b', 'g', 'k', 'grey')
    cmap = ListedColormap(colors[:len(np.unique(y))])

    # plot the decision regions by creating a pair of grid arrays xx1 and xx2 via meshgrid function in Numpy
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution), np.arange(x2_min, x2_max, resolution))

    # use predict method to predict the class labels z of the grid points
    Z = classifier.predict(np.array([xx1.ravel(),xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)

    # draw the contour using matplotlib
    plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())

    # plot class samples
    for i, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y==cl, 0], y=X[y==cl, 1], alpha=0.8, c=cmap(i), marker=markers[i], label=cl)


plot_decision_regions(X, y, ppn)
plt.xlabel('sepal length [cm]')
plt.ylabel('petal length [cm]')
plt.legend(loc='upper left')
plt.show()
```

```
<ipython-input-58-324fb43e16bc>:27: UserWarning: *c* argument looks like a single numer
  plt.scatter(x=X[y==cl, 0], y=X[y==cl, 1], alpha=0.8, c=cmap(i), marker=markers[i], la
```



Praktikum 2

Klasifikasi Berita dengan Perceptron

Langkah 1 - Import Library

```
from sklearn.datasets import fetch_20newsgroups # download dataset
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import Perceptron
from sklearn.metrics import f1_score, classification_report
```

Langkah 2 - Pilih Label dan Split Data

```
categories = ['rec.sport.hockey', 'rec.sport.baseball', 'rec.autos']
newsgroups_train = fetch_20newsgroups(subset='train', categories=categories, remove=('headers', 'footers', 'quotes'))
newsgroups_test = fetch_20newsgroups(subset='test', categories=categories, remove=('headers', 'footers', 'quotes'))
```

Langkah 3 - Ekstrak Fitur dan Buat Model Perceptron

```
# Ekstrak Fitur
vectorizer = TfidfVectorizer()

# Fit fitur
X_train = vectorizer.fit_transform(newsgroups_train.data)
X_test = vectorizer.transform(newsgroups_test.data)

# Fit Model
clf = Perceptron(random_state=11)
clf.fit(X_train, newsgroups_train.target)

# Prediksi
predictions = clf.predict(X_test)
print(classification_report(newsgroups_test.target, predictions))
```

```
                   precision    recall  f1-score   support

               0       0.88      0.88      0.88       396
               1       0.82      0.83      0.83       397
               2       0.88      0.87      0.87       399

        accuracy                           0.86      1192
       macro avg       0.86      0.86      0.86      1192
    weighted avg       0.86      0.86      0.86      1192
```

Penjelasan

Dataset yang digunakan pada kode program diatas adalah 20newsgroup yang terdiri dari sekitar 20.000 dokumen. Scikit-learn bahkan menyediakan fungsi yang memberikan kemudahan untuk mengunduh dan membaca kumpulan dataset dengan menggunakan sklearn.datasets. pada kode program diatas Perceptron mampu melakukan klasifikasi multikelas; strategi yang digunakan adalah one-versus-all untuk melakukan pelatihan untuk setiap kelas dalam data training. Dokumen teks memerlukan ekstraksi fitur salah satunya adalah bobot tf-idf pada kodeprogram diatas digunakan tfidf-vectorizer.

Praktikum 3

Nilai Logika XOR dengan MLP

Langkah 1 - Import Library

```
from sklearn.neural_network import MLPClassifier
```

Langah 2 - Buat Data

```
y = [0, 1, 1, 0] # label
X = [[0, 0], [0, 1], [1, 0], [1, 1]] # data
```

Langkah 3 - Fit Model

```
# Fit model
clf = MLPClassifier(solver='lbfgs', activation='logistic', hidden_layer_sizes=(2,), max_iter=100, random_state=20)
clf.fit(X, y)
```

```
MLPClassifier
MLPClassifier(activation='logistic', hidden_layer_sizes=(2,), max_iter=100,
              random_state=20, solver='lbfgs')
```

Langkah 4 - Prediksi

```
pred = clf.predict(X)
print('Accuracy: %s' % clf.score(X, y))
for i,p in enumerate(pred[:10]):
    print('True: %s, Predicted: %s' % (y[i], p))
```

```
Accuracy: 1.0
True: 0, Predicted: 0
True: 1, Predicted: 1
True: 1, Predicted: 1
True: 0, Predicted: 0
```

Praktikum 4

Klasifikasi dengan ANN

Pra Pengolahan Data

Langkah 1 - Import Library

```
import numpy as np
import pandas as pd
import tensorflow as tf
```

Langkah 2 - Load Data

```
dataset = pd.read_csv('Churn_Modelling.csv')
X = dataset.iloc[:, 3:-1].values
y = dataset.iloc[:, -1].values
```

Cek data (X)

```
print(X)
```

```
[[619 'France' 'Female' ... 1 1 101348.88]
 [608 'Spain' 'Female' ... 0 1 112542.58]
 [502 'France' 'Female' ... 1 0 113931.57]
 ...
 [709 'France' 'Female' ... 0 1 42085.58]
 [772 'Germany' 'Male' ... 1 0 92888.52]
 [792 'France' 'Female' ... 1 0 38190.78]]
```

Langkah 3 - Encoding Data Kategorikal

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:, 2] = le.fit_transform(X[:, 2])
```

Cek data (X)

```
print(X)
```

```
[[619 'France' 0 ... 1 1 101348.88]
 [608 'Spain' 0 ... 0 1 112542.58]
 [502 'France' 0 ... 1 0 113931.57]
 ...
 [709 'France' 0 ... 0 1 42085.58]
 [772 'Germany' 1 ... 1 0 92888.52]
 [792 'France' 0 ... 1 0 38190.78]]
```

Langkah 4 - Encoding Kolom "Geography" dengan One Hot Encoder

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])], remainder='passthrough')
X = np.array(ct.fit_transform(X))
```

Cek data (X)

```
print(X)
```

```
[[1.0 0.0 0.0 ... 1 1 101348.88]
 [0.0 0.0 1.0 ... 0 1 112542.58]
 [1.0 0.0 0.0 ... 1 0 113931.57]
 ...
 [1.0 0.0 0.0 ... 0 1 42085.58]
 [0.0 1.0 0.0 ... 1 0 92888.52]
 [1.0 0.0 0.0 ... 1 0 38190.78]]
```

Langkah 5 - Split Data

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

Langkah 6 - Scaling Fitur

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Membuat Model ANN

Langkah 1 - Inisiasi Model ANN

```
ann = tf.keras.models.Sequential()
```

Langkah 2 - Membuat Input Layer dan Hidden Layer Pertama

```
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

Langkah 3 - Membuat Hidden Layer Kedua

```
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

Langkah 4 - Membuat Output Layer

```
ann.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

Training Model

Langkah 1 - Compile Model (Menyatukan Arsitektur) ANN

```
ann.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

Langkah 2 - Fitting Model

```
ann.fit(X_train, y_train, batch_size = 32, epochs = 100)
```

```
Epoch 55/100
250/250 [==============================] - 1s 4ms/step - loss: 0.3332 - accuracy: 0.8639
Epoch 56/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3331 - accuracy: 0.8643
Epoch 57/100
250/250 [==============================] - 1s 4ms/step - loss: 0.3330 - accuracy: 0.8639
Epoch 58/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3325 - accuracy: 0.8648
Epoch 59/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3328 - accuracy: 0.8645
Epoch 60/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3327 - accuracy: 0.8645
Epoch 61/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3322 - accuracy: 0.8637
Epoch 62/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3320 - accuracy: 0.8641
Epoch 63/100
250/250 [==============================] - 1s 5ms/step - loss: 0.3324 - accuracy: 0.8644
Epoch 64/100
250/250 [==============================] - 2s 6ms/step - loss: 0.3317 - accuracy: 0.8645
Epoch 65/100
250/250 [==============================] - 2s 6ms/step - loss: 0.3321 - accuracy: 0.8640
Epoch 66/100
250/250 [==============================] - 1s 5ms/step - loss: 0.3319 - accuracy: 0.8630
Epoch 67/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3320 - accuracy: 0.8649
Epoch 68/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3313 - accuracy: 0.8626
Epoch 69/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3318 - accuracy: 0.8648
Epoch 70/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3313 - accuracy: 0.8637
Epoch 71/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3312 - accuracy: 0.8644
Epoch 72/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3317 - accuracy: 0.8633
Epoch 73/100
250/250 [==============================] - 1s 5ms/step - loss: 0.3316 - accuracy: 0.8633
Epoch 74/100
250/250 [==============================] - 1s 4ms/step - loss: 0.3310 - accuracy: 0.8643
Epoch 75/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3314 - accuracy: 0.8646
Epoch 76/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3313 - accuracy: 0.8639
Epoch 77/100
250/250 [==============================] - 1s 4ms/step - loss: 0.3310 - accuracy: 0.8640
Epoch 78/100
250/250 [==============================] - 1s 5ms/step - loss: 0.3312 - accuracy: 0.8646
Epoch 79/100
250/250 [==============================] - 1s 5ms/step - loss: 0.3311 - accuracy: 0.8637
Epoch 80/100
250/250 [==============================] - 1s 5ms/step - loss: 0.3307 - accuracy: 0.8665
Epoch 81/100
250/250 [==============================] - 1s 5ms/step - loss: 0.3308 - accuracy: 0.8633
Epoch 82/100
250/250 [==============================] - 1s 4ms/step - loss: 0.3311 - accuracy: 0.8643
Epoch 83/100
250/250 [==============================] - 1s 4ms/step - loss: 0.3306 - accuracy: 0.8658
Epoch 84/100
```

Modelkan Data Baru dan Buat Prediksi

```
print(ann.predict(sc.transform([[1, 0, 0, 600, 1, 40, 3, 60000, 2, 1, 1, 50000]])) > 0.5)
```

```
1/1 [==============================] - 0s 344ms/step
[[False]]
```

Prediksi Dengan Data Testing

```
y_pred = ann.predict(X_test)
y_pred = (y_pred > 0.5)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

```
63/63 [==============================] - 0s 4ms/step
[[0 0]
 [0 1]
 [0 0]
 ...
 [0 0]
 [0 0]
 [0 0]]
```

Cek Akurasi dan Confusion Matrix

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[1496   99]
 [ 183  222]]
0.859
```