1 JOBSHEET 6 - ENSEMBLE LEARNING

Nama : **Elis Nurhidayati**

NIM : **2241720035**

Kelas : **TI-3C**

Link Google Colab:
https://colab.research.google.com/drive/11VjLBjwg-Qj4-3lNMbnThEQMKpMzLQ3y?usp=sharing

2 Praktikum 1

2.1 Bagging

Bagging dengan RandomForest Pada kasus ini kita akan menggunakan salah satu metode bagging yaitu RandomForest untuk mengklasifikasikan jenis tumor. Dalam latihan ini Anda akan melakukan training dengan data Wisconsin Breast Cancer Dataset dari UCI machine learning repository. Latihan ini akan melakukan prediksi memprediksi apakah tumor ganas atau jinak.

Kita akan membandingkan performa dari algoritma Decision Tree dan RandomForest pada kasus ini.

2.1.1 Import Library

[ ]:
```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # import DT
from sklearn.ensemble import RandomForestClassifier # import RandomForest
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
```

2.1.2 Persiapan Data

[ ]:
```
# Load data
df = pd.read_csv('/content/drive/MyDrive/Elis-3C/SMT 5/ML/Jobsheet 6/dataset/
  ↪wbc.csv')

df.head()
```

[ ]:
```
         id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0    842302         M        17.99         10.38          122.80     1001.0
1    842517         M        20.57         17.77          132.90     1326.0
2  84300903         M        19.69         21.25          130.00     1203.0
3  84348301         M        11.42         20.38           77.58      386.1
4  84358402         M        20.29         14.34          135.10     1297.0

   smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0          0.11840           0.27760          0.3001              0.14710
1          0.08474           0.07864          0.0869              0.07017
2          0.10960           0.15990          0.1974              0.12790
3          0.14250           0.28390          0.2414              0.10520
4          0.10030           0.13280          0.1980              0.10430

   …  texture_worst  perimeter_worst  area_worst  smoothness_worst  \
0  …          17.33           184.60      2019.0            0.1622
1  …          23.41           158.80      1956.0            0.1238
2  …          25.53           152.50      1709.0            0.1444
3  …          26.50            98.87       567.7            0.2098
4  …          16.67           152.20      1575.0            0.1374

   compactness_worst  concavity_worst  concave points_worst  symmetry_worst  \
0             0.6656           0.7119                0.2654          0.4601
1             0.1866           0.2416                0.1860          0.2750
2             0.4245           0.4504                0.2430          0.3613
3             0.8663           0.6869                0.2575          0.6638
4             0.2050           0.4000                0.1625          0.2364

   fractal_dimension_worst  Unnamed: 32
0                  0.11890          NaN
1                  0.08902          NaN
2                  0.08758          NaN
3                  0.17300          NaN
4                  0.07678          NaN

[5 rows x 33 columns]
```

[ ]:
```
# Cek kolom null
df.isnull().sum()
```

[ ]:
```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
```

```
compactness_mean             0
concavity_mean               0
concave points_mean          0
symmetry_mean                0
fractal_dimension_mean       0
radius_se                    0
texture_se                   0
perimeter_se                 0
area_se                      0
smoothness_se                0
compactness_se               0
concavity_se                 0
concave points_se            0
symmetry_se                  0
fractal_dimension_se         0
radius_worst                 0
texture_worst                0
perimeter_worst              0
area_worst                   0
smoothness_worst             0
compactness_worst            0
concavity_worst              0
concave points_worst         0
symmetry_worst               0
fractal_dimension_worst      0
Unnamed: 32                569
dtype: int64
```

[ ]:
```
# Seleksi fitur

# Slice dataframe mulai dari kolom 'radius_mean' sampai␣
↪'fractal_dimension_worst'
X = df.iloc[:,3:-1]
y = df['diagnosis']
y = y.map({'M':1, 'B':0}) # Encode label

# Cek jumlah fitur dan instance
X.shape
```

[ ]:
```
(569, 29)
```

2.1.3 Split data training dan testing

[ ]:
```
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=1)
```

2.1.4 Traning Decision Tree

[ ]:
```
# Secara default, DecisionTreeClassifier dari scikit-learn akan menggunakan␣
 ↪nilai "Gini" untuk kriteria
# Terdapat beberapa "hyperparamater" yang dapat digunakan. Silahka baca␣
 ↪dokumentasi
# Pada kasus ini kita akan menggunakan parameter default
dt = DecisionTreeClassifier()

# Sesuaikan dt ke set training
dt.fit(X_train, y_train)

# Memprediksi label set test
y_pred_dt = dt.predict(X_test)

#  menghitung set accuracy
acc_dt = accuracy_score(y_test, y_pred_dt)
print("Test set accuracy: {:.2f}".format(acc_dt))
print(f"Test set accuracy: {acc_dt}")
```

```
Test set accuracy: 0.96
Test set accuracy: 0.9649122807017544
```

2.1.5 Training RandomForest

[ ]:
```
# Pada kasus kali ini kita akan menggunakan estimator pada RandomForest
# Untuk detail parameter (hyperparameter) silahkan cek dokumentasi

rf = RandomForestClassifier(n_estimators=10, random_state=1)

# Sesuaikan dt ke set training
rf.fit(X_train, y_train)

# Memprediksi label set test
y_pred_rf = rf.predict(X_test)

#  menghitung set accuracy
acc_rf = accuracy_score(y_test, y_pred_rf)
print("Test set accuracy: {:.2f}".format(acc_rf))
print(f"Test set accuracy: {acc_rf}")
```

```
Test set accuracy: 0.96
```

```
Test set accuracy: 0.956140350877193
```

3 Praktikum 2

3.1 Boosting

Boosting dengan AdaBoost Pada kasus ini kita akan menggunakan salah satu metode boosting yaitu AdaBoost untuk mengklasifikasikan jenis bunga Iris. Dalam latihan ini kita akan menggunakan dataset Iris yang sangat lazim digunakan. Latihan ini akan melakukan prediksi memprediksi 3 jenis bunga Iris yaitu, Iris Setosa, Iris Versicolor, dan Iris Virginica berdasarkan panjang dan lebar sepal dan petal.

Kita akan membandingkan performa dari algoritma Decision Tree dan AdaBoost pada kasus ini.

3.1.1 Import Library

[ ]:
```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # import DT
from sklearn.ensemble import AdaBoostClassifier # import AdaBoost
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder # Kebutuhan encoding label
from sklearn.ensemble import AdaBoostClassifier # import AdaBoost
```

3.1.2 Persiapan Data

[ ]:
```
# Load data
df = pd.read_csv('/content/drive/MyDrive/Elis-3C/SMT 5/ML/Jobsheet 6/dataset/
  ↪iris.csv')

df.head()
```

[ ]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

[ ]:
```
# Cek kolom null
df.isnull().sum()
```

[ ]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
```

```
Species          0
dtype: int64
```

[ ]:
```
# Seleksi fitur
X = df.iloc[:,2:-1]
y = df['Species']

# encode label
ec = LabelEncoder()
y = ec.fit_transform(y)

# Cek jumlah fitur dan instance
print(X.shape)

# Cek label
print(y)
```

```
(150, 3)
[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 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

3.1.3 Split data training dan testing

[ ]:
```
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=1)
```

3.1.4 Training Decision Tree

[ ]:
```
# Secara default, DecisionTreeClassifier dari scikit-learn akan menggunakan␣
 ↪nilai "Gini" untuk kriteria
# Terdapat beberapa "hyperparamater" yang dapat digunakan. Silahka baca␣
 ↪dokumentasi
# Pada kasus ini kita akan menggunakan parameter default
dt = DecisionTreeClassifier()

# Sesuaikan dt ke set training
dt.fit(X_train, y_train)

# Memprediksi label set test
y_pred_dt = dt.predict(X_test)

#  menghitung set accuracy
```

```
acc_dt = accuracy_score(y_test, y_pred_dt)
print("Test set accuracy: {:.2f}".format(acc_dt))
print(f"Test set accuracy: {acc_dt}")
```

```
Test set accuracy: 0.97
Test set accuracy: 0.9666666666666667
```

3.1.5 Training AdaBoost

[ ]:
```
# Pada kasus kali ini kita akan menggunakan estimator pada AdaBoost
# Untuk detail parameter (hyperparameter) silahkan cek dokumentasi

ada = AdaBoostClassifier(algorithm='SAMME')

# Sesuaikan dt ke set training
ada.fit(X_train, y_train)

# Memprediksi label set test
y_pred_ada = ada.predict(X_test)

#  menghitung set accuracy
acc_ada = accuracy_score(y_test, y_pred_ada)
print("Test set accuracy: {:.2f}".format(acc_ada))
print(f"Test set accuracy: {acc_ada}")
```

```
Test set accuracy: 0.97
Test set accuracy: 0.9666666666666667
```

4 Praktikum 3

4.1 Stackking

Lengkapi bagian berikut dengan data sesuai tugas, dan tentukan perbedaan nilai akurasi antara Random Forest, Adaboost, dan Stacking

[ ]:
```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier, StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier

# Load dataset (contoh dengan dataset diabetes)
dbt = pd.read_csv('/content/drive/MyDrive/Elis-3C/SMT 5/ML/Jobsheet 6/dataset/
 ↪diabetes.csv')

# Pisahkan fitur dan target
```

```
feature_columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',␣
 ↪'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']
X = dbt[feature_columns]  # Fitur
y = dbt['Outcome']  # Target

layer_one_estimators = [
                        ('rf_1', RandomForestClassifier(n_estimators=10,␣
 ↪random_state=42)),
                        ('knn_1', KNeighborsClassifier(n_neighbors=5))
                       ]
layer_two_estimators = [
                        ('dt_2', DecisionTreeClassifier()),
                        ('rf_2', RandomForestClassifier(n_estimators=50,␣
 ↪random_state=42)),
                       ]
layer_two = StackingClassifier(estimators=layer_two_estimators,␣
 ↪final_estimator=LogisticRegression())


clf = StackingClassifier(estimators=layer_one_estimators,␣
 ↪final_estimator=layer_two)

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,␣
 ↪random_state=42)
clf.fit(X_train, y_train).score(X_test, y_test)
```

[ ]: 0.7239583333333334

5 Praktikum 4

5.1 Voting

5.1.1 Stacking dengan Voting

Pada kasus ini kita akan menggunakan salah satu metode stacking yaitu voting untuk mengklasifikasikan pasien penderita diabetes dengan beberapa ciri. Pasien akan di klasifikasikan menjadi pasien menderita diabetes (1) dan tidak menderita diabetes (0). Pertama-tama, kita akan menggunakan beberapa algoritma klasifikasi secara terpisah, yaitu Naive Bayes, SVM Linier, dan SVM RBF. Setelah itu, kita akan menggabungkan performa dari 3 algoritma tersebut dengan menggunakan metode ensemble voting.

5.1.2 Import Library

[ ]:
```
import numpy as np
import pandas as pd
from sklearn.naive_bayes import GaussianNB # import Naive Bayes model Gaussian␣
 ↪(asumsi data terdistribusi normal)
```

```
from sklearn.svm import SVC # import SVM classifier
from sklearn.ensemble import VotingClassifier # import model Voting
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
```

5.1.3 Persiapan Data

[ ]:
```
# Load Data

dbt = pd.read_csv('/content/drive/MyDrive/Elis-3C/SMT 5/ML/Jobsheet 6/dataset/
 ↪diabetes.csv')

dbt.head()
```

[ ]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	

	DiabetesPedigreeFunction	Age	Outcome
0	0.627	50	1
1	0.351	31	0
2	0.672	32	1
3	0.167	21	0
4	2.288	33	1

[ ]:
```
# Cek nama kolom
dbt.columns
```

[ ]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

[ ]:
```
# Cek kolom null
dbt.isnull().sum()
```

[ ]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
```

```
dtype: int64
```

[ ]:
```
# Nilai 0 Tidak Masuk Akal: Parameter seperti 'Glucose', 'BloodPressure', dan
 ↪'Insulin' seharusnya tidak bernilai 0, karena setiap orang pasti memiliki
 ↪nilai tersebut.
# Nilai 0 akan diganti dengan nilai sintetis melalui proses imputasi.
# Nilai yang akan digunakan untuk imputasi adalah nilai mean.

# Cek kolom neng nilai 0
feature_columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
 ↪'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']
for column in feature_columns:
    print("--------------------------------------------------")
    print(f"{column} ==> Missing zeros : {len(dbt.loc[dbt[column] == 0])}")
```

```
--------------------------------------------------
Pregnancies ==> Missing zeros : 111
--------------------------------------------------
Glucose ==> Missing zeros : 5
--------------------------------------------------
BloodPressure ==> Missing zeros : 35
--------------------------------------------------
SkinThickness ==> Missing zeros : 227
--------------------------------------------------
Insulin ==> Missing zeros : 374
--------------------------------------------------
BMI ==> Missing zeros : 11
--------------------------------------------------
DiabetesPedigreeFunction ==> Missing zeros : 0
--------------------------------------------------
Age ==> Missing zeros : 0
```

[ ]:
```
# Imput nilai 0 dengan mean
from sklearn.impute import SimpleImputer

fill_values = SimpleImputer(missing_values=0, strategy="mean", copy=False)

dbt[feature_columns] = fill_values.fit_transform(dbt[feature_columns])
```

5.1.4 Split data training dan testing

[ ]:
```
X = dbt[feature_columns]
y = dbt.Outcome

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
 ↪random_state=42)
```

5.1.5 Training dengan GaussianNB

5.1.6 Standarisasi Fitur

[ ]:
```
# Karena asumsi Gaussian NB adalah data terdistribusi secara normal,
# maka kita perlu melakukan standarisasi

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

# Standarisasi pada fitur di X_train dan X_test
X_train_std = sc.fit_transform(X_train)
X_test_std = sc.transform(X_test)
```

5.1.7 Training dan Evaluasi

[ ]:
```
# Buat obyek GaussianNB
gnb_std = GaussianNB()

# Fit dengan data yang telah di standarisasi
gnb_std.fit(X_train_std, y_train)

# Prediksi dengan data test
y_pred_gnb = gnb_std.predict(X_test_std)

# Evaluasi akurasi testing data
acc_gnb = accuracy_score(y_test, y_pred_gnb)

# Print hasil evaluasi
print("Test set accuracy: {:.2f}".format(acc_gnb))
print(f"Test set accuracy: {acc_gnb}")
```

```
Test set accuracy: 0.74
Test set accuracy: 0.7359307359307359
```

5.1.8 Training dengan SVM Linier

[ ]:
```
# Model SVM linier tanpa tunnning hyperparameter
svm_lin = SVC(kernel='linear')

# Fit ke model
svm_lin.fit(X_train_std, y_train)

# Prediksi
y_pred_svm_lin = svm_lin.predict(X_test_std)

# Evaluasi akurasi testing data
```

```
acc_svm_lin = accuracy_score(y_test, y_pred_svm_lin)

# Print hasil evaluasi
print("Test set accuracy: {:.2f}".format(acc_svm_lin))
print(f"Test set accuracy: {acc_svm_lin}")
```

```
Test set accuracy: 0.74
Test set accuracy: 0.7402597402597403
```

5.1.9 Training dengan SVM RBF

[ ]:
```
# Model SVM RBF tanpa tunnning hyperparameter
svm_rbf = SVC(kernel='rbf')

# Fit ke model
svm_rbf.fit(X_train_std, y_train)

# Prediksi
y_pred_svm_rbf = svm_rbf.predict(X_test_std)

# Evaluasi akurasi testing data
acc_svm_rbf = accuracy_score(y_test, y_pred_svm_rbf)

# Print hasil evaluasi
print("Test set accuracy: {:.2f}".format(acc_svm_rbf))
print(f"Test set accuracy: {acc_svm_rbf}")
```

```
Test set accuracy: 0.72
Test set accuracy: 0.7229437229437229
```

5.1.10 Training dengan Voting

[ ]:
```
# Definisikan algoritma yang akan digunakan untuk voting

clf1 = GaussianNB()
clf2 = SVC(kernel='linear')
clf3 = SVC(kernel='rbf', probability=True)

# model hard voting
voting = VotingClassifier(estimators=[('GaussianNB', clf1), ('SVM-LIN', clf2),␣
 ↪('SVM-RBF', clf3)], voting='hard')

# Fit model
voting.fit(X_train_std, y_train)

# Prediksi
y_pred_vt1 = voting.predict(X_test_std)
```

```python
# Evaluasi akurasi testing data
acc_vt1 = accuracy_score(y_test, y_pred_vt1)

# Print hasil evaluasi
print('Voting Hard')
print("Test set accuracy: {:.2f}".format(acc_vt1))
print(f"Test set accuracy: {acc_vt1}")
```

```
Voting Hard
Test set accuracy: 0.74
Test set accuracy: 0.7402597402597403
```