1 TUGAS PRAKTIKUM JOBSHEET 6

Nama : Elis Nurhidayati

NIM : 2241720035

Kelas : TI-3C

Link Google Colab :
https://colab.research.google.com/drive/1YFs3yQ_02ms2711bzf3dMWvC4vMwiRjO#scrollTo=yp66s498iQw9

1.1 Tugas 1

Terdapat dataset mushroom. Berdasarkan dataset yang tersebut, bandingkan peforma antara algoritma Decision Tree dan RandomForest. Gunakan tunning hyperparameter untuk mendapatkan parameter dan akurasi yang terbaik.

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[ ]:
```
#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
from sklearn.model_selection import GridSearchCV
```

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

df.head()
```

[ ]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	\
0	p	x	s	n	t	p	f	
1	e	x	s	y	t	a	f	
2	e	b	s	w	t	l	f	

```
3     p         x           y         w       t    p              f
4     e         x           s         g       f    n              f

  gill-spacing gill-size gill-color  … stalk-surface-below-ring  \
0            c         n          k  …                         s
1            c         b          k  …                         s
2            c         b          n  …                         s
3            c         n          n  …                         s
4            w         b          k  …                         s

  stalk-color-above-ring stalk-color-below-ring veil-type veil-color  \
0                      w                      w         p          w
1                      w                      w         p          w
2                      w                      w         p          w
3                      w                      w         p          w
4                      w                      w         p          w

  ring-number ring-type spore-print-color population habitat
0           o         p                 k          s       u
1           o         p                 n          n       g
2           o         p                 n          n       m
3           o         p                 k          s       u
4           o         e                 n          a       g

[5 rows x 23 columns]
```

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

[ ]:
```
class                     0
cap-shape                 0
cap-surface               0
cap-color                 0
bruises                   0
odor                      0
gill-attachment           0
gill-spacing              0
gill-size                 0
gill-color                0
stalk-shape               0
stalk-root                0
stalk-surface-above-ring  0
stalk-surface-below-ring  0
stalk-color-above-ring    0
stalk-color-below-ring    0
veil-type                 0
veil-color                0
```

```
ring-number               0
ring-type                 0
spore-print-color         0
population                0
habitat                   0
dtype: int64
```

[ ]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

# Load dataset
data = load_iris()
X = pd.DataFrame(data.data, columns=data.feature_names)  # Convert to DataFrame
y = data.target
```

[ ]:
```
from sklearn.model_selection import train_test_split
X = pd.get_dummies(X, drop_first=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[ ]:
```
# Training Decision Tree

from sklearn.model_selection import GridSearchCV
# Inisialisasi model Decision Tree
dt = DecisionTreeClassifier(random_state=42)

# Definisikan rentang hyperparameter yang akan diuji
param_grid_dt = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 10, 20, 30, 40, 50],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Gunakan GridSearchCV untuk mencari hyperparameter terbaik
grid_search_dt = GridSearchCV(dt, param_grid_dt, cv=5, n_jobs=-1)
grid_search_dt.fit(X_train, y_train)

# Cetak hyperparameter terbaik untuk Decision Tree
best_dt_params = grid_search_dt.best_params_
print("Hyperparameter terbaik untuk Decision Tree:", best_dt_params)

# Gunakan model Decision Tree dengan hyperparameter terbaik
best_dt_model = grid_search_dt.best_estimator_

# Evaluasi akurasi Decision Tree pada data pengujian
```

```
dt_accuracy = accuracy_score(y_test, best_dt_model.predict(X_test))
print("Akurasi Decision Tree:", dt_accuracy)
```

```
Hyperparameter terbaik untuk Decision Tree: {'criterion': 'entropy',
'max_depth': None, 'min_samples_leaf': 4, 'min_samples_split': 2}
Akurasi Decision Tree: 1.0
```

[ ]:
```
# Training RandomForest

# Inisialisasi model RandomForest
rf_model = RandomForestClassifier(random_state=42)

# Definisikan rentang hyperparameter yang akan diuji
param_grid_rf = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30, 40, 50],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'criterion': ['gini', 'entropy']
}

# Gunakan GridSearchCV untuk mencari hyperparameter terbaik
grid_search_rf = GridSearchCV(rf_model, param_grid_rf, cv=5, n_jobs=-1)
grid_search_rf.fit(X_train, y_train)

# Cetak hyperparameter terbaik untuk RandomForest
best_rf_params = grid_search_rf.best_params_
print("Hyperparameter terbaik untuk RandomForest:", best_rf_params)

# Gunakan model RandomForest dengan hyperparameter terbaik
best_rf_model = grid_search_rf.best_estimator_

# Evaluasi akurasi RandomForest pada data pengujian
rf_accuracy = accuracy_score(y_test, best_rf_model.predict(X_test))
print("Akurasi RandomForest:", rf_accuracy)
```

```
Hyperparameter terbaik untuk RandomForest: {'criterion': 'gini', 'max_depth':
None, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 200}
Akurasi RandomForest: 1.0
```

1.2 Tugas 2

Terdapat dataset mushroom. Berdasarkan dataset tersebut, bandingkan peforma antara algoritma Decision Tree dan AdaBoost. Gunakan tunning hyperparameter untuk mendapatkan parameter dan akurasi yang terbaik.

[ ]:
```
# Import library
import pandas as pd
```

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score
```

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

df.head()
```

[ ]:
```
  class cap-shape cap-surface cap-color bruises odor gill-attachment  \
0     p         x           s         n       t    p               f
1     e         x           s         y       t    a               f
2     e         b           s         w       t    l               f
3     p         x           y         w       t    p               f
4     e         x           s         g       f    n               f

  gill-spacing gill-size gill-color  … stalk-surface-below-ring  \
0            c         n          k  …                        s
1            c         b          k  …                        s
2            c         b          n  …                        s
3            c         n          n  …                        s
4            w         b          k  …                        s

  stalk-color-above-ring stalk-color-below-ring veil-type veil-color  \
0                      w                      w         p          w
1                      w                      w         p          w
2                      w                      w         p          w
3                      w                      w         p          w
4                      w                      w         p          w

  ring-number ring-type spore-print-color population habitat
0           o         p                 k          s       u
1           o         p                 n          n       g
2           o         p                 n          n       m
3           o         p                 k          s       u
4           o         e                 n          a       g

[5 rows x 23 columns]
```

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

[ ]:
```
class                     0
cap-shape                 0
cap-surface               0
```

```
cap-color                   0
bruises                     0
odor                        0
gill-attachment             0
gill-spacing                0
gill-size                   0
gill-color                  0
stalk-shape                 0
stalk-root                  0
stalk-surface-above-ring    0
stalk-surface-below-ring    0
stalk-color-above-ring      0
stalk-color-below-ring      0
veil-type                   0
veil-color                  0
ring-number                 0
ring-type                   0
spore-print-color           0
population                  0
habitat                     0
dtype: int64
```

[ ]:
```
# Preprocessing: Ubah label menjadi numerik
X = pd.get_dummies(X, drop_first=True)

# Bagi dataset menjadi data pelatihan dan pengujian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

1.2.1 1. Decision Tree

[ ]:
```
# Inisialisasi model Decision Tree
dt_model = DecisionTreeClassifier(random_state=42)

# Definisikan rentang hyperparameter yang akan diuji
param_grid_dt = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 10, 20, 30, 40, 50],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Gunakan GridSearchCV untuk mencari hyperparameter terbaik
grid_search_dt = GridSearchCV(dt_model, param_grid_dt, cv=5, n_jobs=-1)
grid_search_dt.fit(X_train, y_train)

# Cetak hyperparameter terbaik untuk Decision Tree
```

```
best_dt_params = grid_search_dt.best_params_
print("Hyperparameter terbaik untuk Decision Tree:", best_dt_params)

# Gunakan model Decision Tree dengan hyperparameter terbaik
best_dt_model = grid_search_dt.best_estimator_

# Evaluasi akurasi Decision Tree pada data pengujian
dt_accuracy = accuracy_score(y_test, best_dt_model.predict(X_test))
```

```
Hyperparameter terbaik untuk Decision Tree: {'criterion': 'entropy',
'max_depth': None, 'min_samples_leaf': 4, 'min_samples_split': 2}
```

1.2.2 2. Algoritma Ada Boost

[ ]:
```
# Inisialisasi model AdaBoost
adaboost_model =␣
 ↪AdaBoostClassifier(estimator=DecisionTreeClassifier(max_depth=2),␣
 ↪random_state=42, algorithm='SAMME')


# Definisikan rentang hyperparameter yang akan diuji
param_grid_adaboost = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 1.0]
}

# Gunakan GridSearchCV untuk mencari hyperparameter terbaik
grid_search_adaboost = GridSearchCV(adaboost_model, param_grid_adaboost, cv=5,␣
 ↪n_jobs=-1)
grid_search_adaboost.fit(X_train, y_train)

# Cetak hyperparameter terbaik untuk AdaBoost
best_adaboost_params = grid_search_adaboost.best_params_
print("Hyperparameter terbaik untuk AdaBoost:", best_adaboost_params)

# Gunakan model AdaBoost dengan hyperparameter terbaik
best_adaboost_model = grid_search_adaboost.best_estimator_

# Evaluasi akurasi AdaBoost pada data pengujian
adaboost_accuracy = accuracy_score(y_test, best_adaboost_model.predict(X_test))
```

```
Hyperparameter terbaik untuk AdaBoost: {'learning_rate': 0.01, 'n_estimators':
200}
```

[ ]:
```
print("Perbandingan akurasi dari Decision Tree dan AdaBoost")
print("==================================================")
print("Akurasi Decision Tree:", dt_accuracy)
```

```
print("Akurasi AdaBoost:", adaboost_accuracy)
```

```
Perbandingan akurasi dari Decision Tree dan AdaBoost
==================================================
Akurasi Decision Tree: 1.0
Akurasi AdaBoost: 1.0
```

1.2.3 Tugas 3

Dengan menggunakan dataset diabetes, buatlah ensemble voting dengan algoritma

1. Logistic Regression
2. SVM kernel polynomial
3. Decission Tree

Anda boleh melakukan eksplorasi dengan melakukan tunning hyperparameter

[ ]:
```
# Import library
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import accuracy_score
```

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

#dbt.head
print(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
```

[ ]:
```
# Cek kolom dengan nilai 0

feature_columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',␣
 ↪'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']
for column in feature_columns:
    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
```

[ ]:
```
# Impute 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])


# Split data training dan testing
```

```
# Pisahkan fitur dan label
X = dbt.drop('Outcome', axis=1)
y = dbt['Outcome']

# Bagi dataset menjadi data pelatihan dan pengujian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

1.2.4 1. Training dengan Logistic Regression

[ ]:
```
# Inisialisasi model Logistic Regression
logistic_model = LogisticRegression(max_iter=1000)

# Buat ensemble voting classifier hanya dengan Logistic Regression
ensemble_classifier = VotingClassifier(
    estimators=[
        ('logistic', logistic_model)
    ],
    voting='hard'  # 'hard' voting digunakan untuk klasifikasi
)

# Latih ensemble classifier pada data pelatihan
ensemble_classifier.fit(X_train, y_train)

# Memprediksi label pada data pengujian
y_pred = ensemble_classifier.predict(X_test)

# Hitung akurasi ensemble classifier pada data pengujian
accuracy = accuracy_score(y_test, y_pred)
print("Akurasi Ensemble Voting Classifier (Logistic Regression):", accuracy)
```

```
Akurasi Ensemble Voting Classifier (Logistic Regression): 0.7597402597402597
```

1.2.5 2. Training dengan SVM Kernel Polynomial

[ ]:
```
# Inisialisasi model SVM dengan kernel polynomial
svm_model = SVC(kernel='poly', degree=3)  # Kernel polynomial dengan derajat 3

# Latih model SVM pada data pelatihan
svm_model.fit(X_train, y_train)

# Memprediksi label pada data pengujian
y_pred = svm_model.predict(X_test)

# Hitung akurasi model SVM pada data pengujian
accuracy = accuracy_score(y_test, y_pred)
print("Akurasi SVM dengan Kernel Polynomial:", accuracy)
```

```
Akurasi SVM dengan Kernel Polynomial: 0.7597402597402597
```

1.2.6 3. Training dengan Decission Tree

```
[ ]: # Inisialisasi model Decision Tree
decision_tree_model = DecisionTreeClassifier(random_state=42)

# Latih model Decision Tree pada data pelatihan
decision_tree_model.fit(X_train, y_train)

# Memprediksi label pada data pengujian
y_pred = decision_tree_model.predict(X_test)

# Hitung akurasi model Decision Tree pada data pengujian
accuracy = accuracy_score(y_test, y_pred)
print("Akurasi Decision Tree:", accuracy)
```

```
Akurasi Decision Tree: 0.7012987012987013
```