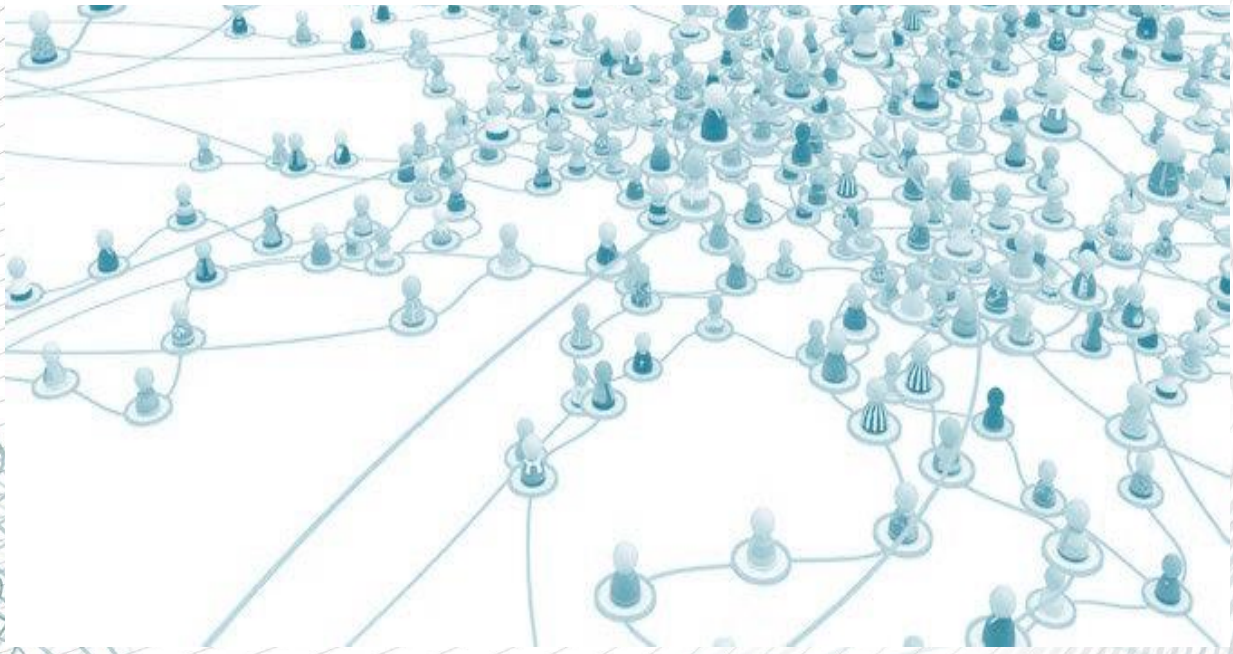
MODUL PRAKTIKUM

SD3107-Pembelajaran Mesin



Program Studi Sains Data
Fakultas Sains
Institut Teknologi Sumatera

2024

MODUL 8

Bootstrap Aggregating (Bagging)

A. Konsep Dasar

Bootstrap Aggregating (Bagging) adalah teknik ensemble yang menggabungkan hasil dari beberapa model yang dilatih secara independen untuk meningkatkan akurasi prediksi dan mengurangi variansi model. Prinsip dasar bagging adalah menghasilkan beberapa model dari data pelatihan yang berbeda dengan cara *bootstrap sampling*, yaitu mengambil sampel acak dengan pengulangan dari dataset asli. Setiap model dilatih pada subset data yang berbeda, dan hasil akhirnya digabungkan, baik melalui *voting mayoritas* (untuk klasifikasi) atau rata-rata (untuk regresi). *Bagging* sangat efektif untuk model yang memiliki kecenderungan overfitting, karena dengan melatih beberapa model pada data yang berbeda, hasil akhirnya cenderung lebih stabil dan dapat menghasilkan prediksi yang lebih baik.

BaggingRegressor dan *BaggingClassifier* adalah dua implementasi dari algoritma ensemble Bagging di Scikit-learn. Meskipun keduanya menggunakan prinsip yang sama, ada beberapa perbedaan mendasar terkait dengan tujuan dan metode penggunaannya:

1. *BaggingRegressor*

   Digunakan untuk tugas regresi, yaitu memprediksi nilai kontinu (seperti harga, suhu, atau skor). Dengan nilai prediksi untuk setiap input tes x adalah:

$$\hat{y} = \frac{1}{M} \sum_{m=1}^{M} G_m(x)$$

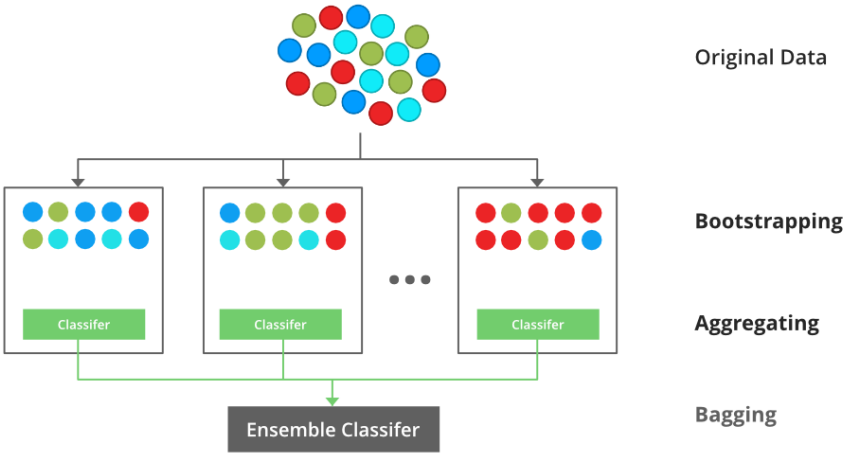
Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

Gambar 1. Contoh penerapan teknik bootsrap

   Base estimator pada *BaggingRegressor* yang dapat digunakan yaitu algoritma regresi seperti *Linear Regression,* dan *Decision Tree Regressor*. Metrik yang digunakan untuk evaluasi yang sesuai untuk regresi, seperti *Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE)* dan R^2 *Score.*

2. *BaggingClassifier*

   Dapat digunakan untuk masalah klasifikasi, di mana tujuan utamanya adalah memprediksi kelas atau kategori. *BaggingClassifier* efektif dalam meningkatkan akurasi prediksi, terutama ketika model dasar rentan terhadap overfitting, seperti pada *decision trees* atau *Naive Bayes*. Metrik yang digunakan untuk evaluasi adalah yang cocok untuk klasifikasi, seperti *Accuracy, Precision, Recall, F1-Score, dan ROC-AUC Score.*



Gambar 2. Ilustrasi konsep bagging untuk klasifikasi (sumber)

Salah satu kelemahan utama *bagging* adalah kebutuhan komputasi yang tinggi karena melibatkan pelatihan beberapa model dasar, terutama pada dataset besar. Meski efektif mengurangi variansi, *Bagging* tidak mengurangi bias, sehingga jika model dasar *underfitting*, hasilnya tetap bias. Pada dataset sederhana atau model dasar yang kuat, *Bagging* sering kali tidak memberikan peningkatan signifikan.

B. Tujuan Praktikum

I. Tujuan Instruksional Umum

Praktikum bertujuan untuk menerapkan teori Bootstrap Aggregating (Bagging) untuk regresi dan klasifikasi data dalam pembelajaran mesin.

II. Tujuan Instruksional Khusus

1. Mahasiswa mampu menguasai konsep dasar Bootstrap Aggregating (Bagging)
2. Mahasiswa mampu menyelesaikan studi kasus regresi dan klasifikasi data menggunakan metode Bootstrap Aggregating (Bagging).
3. Mahasiswa mampu menganalisis hasil dari studi kasus regresi dan klasifikasi data menggunakan metode Bootstrap Aggregating (Bagging).

C. Dataset dan bahasa pemrograman Python

C.1 Dataset dalam praktikum

Dalam praktikum bagian D1 dan D2 akan menggunakan dataset berupa data x dan y pada tabel berikut.

X	Y(D1)	Y(D2)
1	2.5	0
2	3.7	1
3	4.8	0
4	5.9	1
5	6.2	0
6	7.4	1
7	8.5	0
8	9.6	1
9	10.7	0
10	11.8	1
11	12.9	0
12	14.0	1

Data X dan Y(D1) akan digunakan untuk bagging regresi, Kemudian data X dan Y(D2) akan digunakan untuk bagging klasifikasi.

Pada D3, dataset yang digunakan yaitu dataset "Cleaned Dataset" di Kaggle berisi data yang telah diproses, dengan kolom- kolom yang relevan dan nilai yang hilang telah ditangani. Dataset ini dapat digunakan untuk analisis data atau pemodelan machine learning.

C.2 Bahasa Pemrograman Python

Pada praktikum kali ini, kita akan menggunakan IDE Python Service dari Google Colab, maka dari itu Anda perlu menyiapkan akun google pribadi untuk mengakses servis ini di akun google masing-masing.

Library yang akan digunakan diantaranya numpy, pandas, dan sklearn.

1) NumPy: digunakan untuk operasi matematika berbasis array dan matriks.
2) Pandas : untuk manipulasi, analisis, dan pembersihan data berbasis struktuk DataFrame dan Series dengan berbagai fungsi efisien untuk membaca, mengolah, dan menyimpan data.

3) Scikit-learn (sklearn): library untuk implementasi berbagai algoritma machine learning seperti klasifikasi, regresi, clustering, dan reduksi dimensi.

   Dalam praktikum :

   from sklearn.ensemble import BaggingRegressor, BaggingClassifier

Praktikum pembelajaran mesin pada modul ini dapat maksimal anda praktikan dengan pemahaman bidang atau mata kuliah terkait seperti algoritma pemrograman, struktur data, dan data mining (data preprocessing).

Dataset dan Source code akan diberikan saat praktikum berlangsung dan dapat anda unduh pada folder berikut Modul 8 PM 2024 .

D. Implementasi Bagging untuk regresi dan klasifikasi dengan skLearn

D.1 Bagging untuk regresi

1) Import Library

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import BaggingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

NumPy untuk manipulasi data, **train_test_split** untuk membagi dataset, **BaggingRegressor** untuk membuat model ensemble dengan **Linear Regression** sebagai *estimator*, dan **mean_squared_error** untuk mengevaluasi akurasi model regresi menggunakan metrik MSE.

2) Import Dataset

```
X = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])  # Satu fitur
# Reshape X agar menjadi array 2D
X = X.reshape(-1, 1)
Y = np.array([2.5, 3.7, 4.8, 5.9, 6.2, 7.4, 8.5, 9.6, 10.7, 11.8, 12.9, 14.0])
```

Dataset dengan X sebagai fitur dan Y merupakan data target berurutan untuk regresi.

3) Split data

```
# Split data untuk regresi
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=42)
```

Split data menjadi empat yaitu X train danY train untuk pelatihan, dan data uji dengan X test dab Y test.

4) Proses *bagging* dengan Regresi Linier

```
# Bagging Regressor dengan Linear Regression sebagai base estimator

be = LinearRegression()
model = BaggingRegressor(be, n_estimators=10, random_state=42)
model.fit(X_train, Y_train)
```

membuat model regresi dengan menggunakan **LinearRegression** sebagai *estimator* dasar dalam BaggingRegressor. **n_estimators=10** berarti model akan menggunakan 10 estimator (model dasar) dalam ensemble, dan random_state=42 memastikan hasil yang dapat direproduksi. Kemudian, **model.fit(X_train, Y_train)** melatih model dengan data pelatihan X_train (fitur) dan Y_train (target). Model ini akan menghasilkan prediksi regresi dengan menggabungkan hasil dari 10 model *Linear Regression* yang dilatih secara paralel.

5) Lakukan prediksi dan evaluasi

```
# Prediksi dan evaluasi
Y_pred= model.predict(X_test)
mse = mean_squared_error(Y_test, Y_pred)
print("MSE for Bagging Regressor (Linear Regression):", mse)
```

```
MSE for Bagging Regressor (Linear Regression): 0.10325684374812753
```

melakukan prediksi dan evaluasi model dengan **model.predict(X_test)** menghasilkan prediksi Y_pred untuk data uji X_test. Selanjutnya, **mean_squared_error(Y_test, Y_pred)** menghitung Mean Squared Error (MSE) antara prediksi Y_pred dan nilai aktual Y_test.

6) Menampilkan data pada setiap model

```
print("Data tiap model bootsrap:")
for i, samples in enumerate(model.estimators_samples_, start=1):
    print(f"Data model {i} = {X_train[samples].flatten()}")
```

```
Data tiap model bootsrap:
Data model 1 = [ 3  4  4  7  2  3  7 12  8]
Data model 2 = [4 2 4 2 5 2 9 2 7]
Data model 3 = [8 9 4 4 7 5 7 3 5]
Data model 4 = [ 9  5  4 12  6  2  8  3  5]
Data model 5 = [8 8 2 2 7 2 4 7 8]
Data model 6 = [5 3 6 3 4 2 6 2 8]
Data model 7 = [2 3 6 5 3 2 6 5 5]
Data model 8 = [ 8  3  7  3  5  9 12  2  5]
Data model 9 = [ 7 12 12  2  3  2  8  7  2]
Data model 10 = [7 6 7 9 4 8 7 6 6]
```

Menampilkan data yang digunakan oleh setiap model dalam ensemble *Bagging* dengan **model.estimators_samples_** menyimpan indeks sampel bootstrap untuk setiap model dasar yang dilatih. Dengan **enumerate(model.estimators_samples_, start=1)**, setiap indeks sampel diproses, dan X_train[samples].flatten() digunakan untuk menampilkan data pelatihan yang dipilih untuk setiap model, yang kemudian dicetak dengan nomor model terkait. Ini memungkinkan kita melihat subset data yang digunakan oleh setiap estimator dalam Bagging.

7) Menampilkan hasil evaluasi tiap model

```
# Loop untuk menampilkan akurasi setiap estimator/model
print("Akurasi tiap model:")
for i, estimator in enumerate(model.estimators_, start=1):
    # Prediksi menggunakan estimator ke-i
    y_pred = estimator.predict(X_test)
    mse = mean_squared_error(Y_test, y_pred)
    print(f"model {i}  = {mse:.4f}")
```

```
Akurasi tiap model:
model 1  = 0.0294
model 2  = 0.0875
model 3  = 0.1207
model 4  = 0.0126
model 5  = 0.1079
model 6  = 0.2826
model 7  = 0.9931
model 8  = 0.0150
model 9  = 0.0070
model 10  = 0.0997
```

Menampilkan *Mean Squared Error* (MSE) untuk setiap model dasar dalam ensemble Bagging. Dengan **enumerate(model.estimators_, start=1)**, setiap estimator diproses, dan prediksi dihasilkan menggunakan **estimator.predict(X_test)**. MSE dihitung dengan **mean_squared_error(Y_test, y_pred)**, kemudian dicetak untuk setiap model.

D.2 Bagging untuk klasifikasi

1) Import Library

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import BaggingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
```

BaggingClassifier untuk membuat model ensemble klasifikasi dengan **Gaussian Naive Bayes** sebagai estimator dasar, dan menggunakan **accuracy_score** untuk mengukur akurasi model dengan membandingkan prediksi dan nilai target sebenarnya.

2) Import Dataset

```
X = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
# Reshape X agar menjadi array 2D
X = X.reshape(-1, 1)
Y = np.array([0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1])  #
```

Data berupa satu fitur dan satu target berupa kelas 0 dan 1 untuk klasifikasi.

3) Spilt data

```
# Split data untuk
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=42)
```

Split data menjadi empat yaitu X train danY train untuk pelatihan, dan data uji dengan X test dab Y test.

4) Proses *bagging* dengan Naive Bayes

```
# Bagging Classifier dengan Naive Bayes sebagai base estimator
be_c = GaussianNB()
model = BaggingClassifier(be_c, n_estimators=15, random_state=42)
model.fit(X_train, Y_train)
```

Membangun model BaggingClassifier dengan Gaussian Naive Bayes sebagai estimator dasar. **n_estimators=15** berarti model menggunakan 15 estimator, dan random_state=42 memastikan hasil yang dapat direproduksi. Model dilatih menggunakan **model.fit(X_train, Y_train)**.

5) Lakukan prediksi dan evaluasi

```
# Prediksi dan evaluasi
Y_pred = model.predict(X_test)
accuracy = accuracy_score(Y_test, Y_pred)
print("Accuracy for Bagging Classifier (Naive Bayes):", accuracy)
```

```
Accuracy for Bagging Classifier (Naive Bayes): 0.3333333333333333
```

Selanjutnya, **model.predict(X_test)** digunakan untuk memprediksi data uji, dan accuracy_score(Y_test, Y_pred) menghitung **akurasi** model dengan membandingkan prediksi dengan nilai target yang sebenarnya. Hasil akurasi dicetak sebagai evaluasi performa model.

6) Menampilkan data pada setiap model

```
print("Data tiap model bootsrap:")
for i, samples in enumerate(model.estimators_samples_, start=1):
    print(f"Data model {i} = {X_train[samples].flatten()}")
```

Menampilkan data yang digunakan oleh setiap model dalam ensemble *Bagging* dengan **model.estimators_samples_** menyimpan indeks sampel bootstrap untuk setiap model dasar yang dilatih.

7) Menampilkan hasil evaluasi tiap model

```
# Loop untuk menampilkan akurasi setiap estimator
print("Akurasi tiap model:")
for i, estimator in enumerate(model.estimators_, start=1):
    # Prediksi menggunakan estimator ke-i
    y_pred = estimator.predict(X_test)
    acc = accuracy_score(Y_test, y_pred)
    print(f"model {i}  = {acc:.4f}")
```

Menampilkan *accuracy* untuk setiap model dasar dalam ensemble *Bagging*. Dengan **enumerate(model.estimators_, start=1)**, setiap estimator diproses, dan prediksi dihasilkan menggunakan **estimator.predict(X_test)**. Akurasi dihitung dengan **accuracy_score(Y_test, y_pred)**, kemudian dicetak untuk setiap model, menunjukkan akurasi prediksi masing-masing model dasar.

D.3 Implementasi *Bagging* pada dataset berdimensi tinggi

Pada praktikum bagian ini akan dilakukan implementasi bagging pada dataset berdimensi tinggi, untuk dapat membedakan pengaruh dari penggunaan teknik bagging terhadap peningkatan akurasi pada model estmator dasar klasifikasi.

1) Import library

```
import pandas as pd
from sklearn.ensemble import BaggingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

from sklearn.tree import DecisionTreeClassifier mengimpor **DecisionTreeClassifier** dari Scikit-learn, yang digunakan untuk membangun model klasifikasi berbasis pohon keputusan.

2) Import dataset

Digunakan dataset dengan jumlah fitur yang cukup banyak yaitu 22. Dataset ini digunakan sekaligus untuk mengetahui bagaimana bagging dapat mengatasi

data dengan fitur lebih banyak sebagai data training.

```
# Mengimpor data
data = pd.read_csv('/df_final_features.csv')

data.head(5)
```

	Name	Sex	Age	Height	Weight	Team	Year	Season	Host_City	Host_Country	...
0	A Dijiang	M	24.0	180.0	80.0	China	1992	Summer	Barcelona	Spain	...
1	A Lamusi	M	23.0	170.0	60.0	China	2012	Summer	London	United Kingdom	...
2	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	1988	Winter	Calgary	Canada	...
3	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	1988	Winter	Calgary	Canada	...
4	Christine Jacoba Aaftink	F	25.0	185.0	82.0	Netherlands	1992	Winter	Albertville	France	...

5 rows × 22 columns

3) Persiapan data

```
from sklearn.impute import SimpleImputer # Import SimpleImputer

# defined X and Y
X = data[['Age', 'Height', 'Weight']] #data
Y = data['Sex']      # Target

# split data
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.25, random_state=42)

# Create an imputer to fill NaN values with the mean of each column
imputer = SimpleImputer(strategy='mean') # or strategy='median'

# Fit the imputer on the training data and transform both training and testing data
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)
```

Persiapan data dalam praktikum bagian ini menggunakan tiga fitur untuk training. Digunakan **SimpleImputer** dari Scikit-learn untuk menangani nilai NaN dalam dataset. **SimpleImputer** diinisialisasi dengan **strategy='mean'**, yang menggantikan nilai yang hilang dengan rata-rata kolom masing-masing. Kemudian dipisahkan menjadi data pelatihan dan pengujian menggunakan **train_test_split**. Setelah itu, SimpleImputer diterapkan untuk mengisi nilai hilang dalam data pelatihan dan pengujian.

4) Melatih dan evaluasi model tanpa *bagging*

```
# model Decision Tree tanpa bagging
model = DecisionTreeClassifier()

# Melatih model
model.fit(X_train, y_train)

# Membuat prediksi pada data uji
y_pred = model.predict(X_test)

# Menghitung akurasi
accuracy = accuracy_score(y_test, y_pred)
print(f"Akurasi DecisionTree tanpa bagging : {accuracy:.4f}")
```

```
Akurasi DecisionTree tanpa bagging : 0.8514
```

Selanjutnya, membangun model **DecisionTreeClassifier** tanpa menggunakan teknik Bagging. Model dilatih menggunakan data pelatihan dengan **model.fit(X_train, y_train).** Kemudian, model.predict(X_test) digunakan untuk memprediksi kelas target pada data uji. Akurasi model dihitung dengan **accuracy_score(y_test, y_pred)** yang membandingkan prediksi dengan nilai target sebenarnya, dan hasil akurasi didapatkan yaitu 85%.

5) Melatih dan evaluasi model dengan *bagging*

```
be_c = DecisionTreeClassifier()
# Membuat model Bagging dengan Naive Bayes sebagai estimator dasar
bagging_clf = BaggingClassifier(be_c, n_estimators=100, random_state=42)

# Melatih model
bagging_clf.fit(X_train, y_train)

# Membuat prediksi pada data uji
y_pred = bagging_clf.predict(X_test)

# Menghitung akurasi
accuracy = accuracy_score(y_test, y_pred)
print(f"Akurasi Bagging dengan DecisionTree: {accuracy:.4f}")
```

```
Akurasi Bagging dengan DecisionTree: 0.8822
```

Terakhir, membangun model **BaggingClassifier** dengan **DecisionTreeClassifier** sebagai estimator dasar. **n_estimators=100** berarti model menggunakan 100 pohon keputusan dalam ensemble, dan random_state=42 memastikan hasil yang dapat direproduksi. Model dilatih

dengan **bagging_clf.fit(X_train, y_train),** kemudian melakukan prediksi pada data uji menggunakan **bagging_clf.predict(X_test).** Akurasi dihitung dengan **accuracy_score(y_test, y_pred)** dan hasilnya didapatkan 88% , hasil ini menunjukan terdapat peningkatan akurasi sebesar 3% dari model tanpa tehnik *bagging*.

TUGAS INDIVIDU

- Tugas individu dikerjakan saat praktikum berlangsung dengan waktu yang ditentukan oleh asisten praktikum.
- Soal tugas individu terbagi menjadi tiga level soal yaitu *Basic, Medium, dan Hard*. Praktikan dapat menyelesaikan ketiganya untuk mendapatkan poin maksimal.
- Namun praktikan cukup menyelesaikan minimal satu soal jika estimasi waktu praktikum sudah akan berakhir agar tidak melewatkan penilaian test lisan.