Speech Classification

Oleh :

Dhoni Hanif Supriyadi

1. Business Understanding

Pada tahapan ini, seorang data science harus mengerti seperti apa bisnis ini atau proyek yang akan dikerjakan ini. Disini, kami akan membuat sebuat proyek berjudul Speech Classification. Kami akan melakukan klasifikasi pada audio dan memprediksinya dengan machine learning.

2. Data Preparation

Pada tahapan ini, seorang data science harus mengumpulkan, membaca, mengolah data yang ada sehingga menjadi data jadi dan siap ditraining dengan machine learning. Disini, kami mengumpulkan data yang bersumber dari kaggle.com. Data ini berjenis klasifikasi sehingga kami akan menggunakan tugas klasifikasi. Hal pertama yang kita lakukan adalah mengimport library yang dibutuhkan terlebih dahulu seperti berikut.

```
import librosa
import librosa.display
import numpy as np
import glob
import os
import cv2
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report

✓ 2.6s                                                    Pytho
```

Seperti yang kita ketahui, data ini terdapat dalam folder dan di setiap folder memiliki data audio sehingga kita dapat mengatakan bahwa data ini disajikan secara terpisah dan kita harus mengumpulkannya menjadi satu seperti berikut.

```
spektogram, labels = [], []
folders = ["OAF_Sad", "OAF_angry", "OAF_happy"]
```
✓ 0.4s Pyt

```
for folder in folders:
    for spc in glob.glob("dataset/" + folder + "/*"):
        y, sr = librosa.load(spc)
        spec = np.abs(librosa.stft(y))
        spec = librosa.amplitude_to_db(spec, ref=np.max)
        spec = cv2.resize(spec, (64, 64))
        spec = spec.flatten()
        spektogram.append(spec)
        labels.append(folder)

data = pd.DataFrame(spektogram)
data["Label"] = labels
data.to_csv("spektogram.csv", index=False)
```
✓ 1m 20.1s Pyt

Data telah disimpan dalam bentuk csv. Selanjutnya, baca data tersebut dengan pandas seperti berikut.

```
df = pd.read_csv("spektogram.csv")
```
✓ 1.5s

Lalu, tampilkan 5 data dari atas seperti berikut.

```
df.head()
```
✓ 0.6s Python

	0	1	2	3	4	5	6	7	8	9	...	4087	4088	4089	4090	4091	4092	4093
0	-53.947403	-58.010162	-56.007854	-58.154940	-56.721954	-55.964170	-60.817448	-54.449642	-55.957970	-59.727386	...	-80.0	-80.0	-80.0	-80.0	-80.0	-80.0	-80.0
1	-56.680070	-53.136818	-66.619970	-66.850520	-61.113052	-63.374720	-60.666157	-59.610330	-64.715805	-66.174340	...	-80.0	-80.0	-80.0	-80.0	-80.0	-80.0	-80.0
2	-59.279175	-59.364210	-57.678867	-57.691050	-64.752426	-68.444660	-56.830140	-60.440647	-60.664543	-65.896680	...	-80.0	-80.0	-80.0	-80.0	-80.0	-80.0	-80.0
3	-55.088280	-55.980930	-54.053844	-57.570225	-59.133410	-53.457344	-57.844383	-47.275253	-56.824560	-65.398530	...	-80.0	-80.0	-80.0	-80.0	-80.0	-80.0	-80.0
4	-63.262688	-67.423225	-57.511776	-50.259518	-56.068382	-54.664314	-57.393692	-50.493675	-42.074116	-46.831856	...	-80.0	-80.0	-80.0	-80.0	-80.0	-80.0	-80.0

5 rows × 4097 columns

Ini adalah 5 data dari atas. Selanjutnya, cek jumlah baris dan kolom pada data seperti berikut.

```
df.shape
```
✓ 0.4s

```
(600, 4097)
```

Data berjumlah 600 data dengan 4097 kolom atau features. Selanjutnya cek type dari setiap kolom seperti berikut.

```
df.dtypes
```
✓ 0.9s

```
0        float64
1        float64
2        float64
3        float64
4        float64
         ...
4092     float64
4093     float64
4094     float64
4095     float64
Label     object
Length: 4097, dtype: object
```

Seperti yang terlihat, ada data bertipe object sehingga kita harus mengubahnya menjadi numerik dengan LabelEncoder agar data dapat diproses dengan Machine Learning.

```
from sklearn.preprocessing import LabelEncoder

df2 = df.copy()
le = LabelEncoder()
df["Label"] = le.fit_transform(df["Label"])
```
✓ 0.7s

Lalu, bagi data menjadi data training dan data testing seperti berikut.

```
X = df.drop("Label", axis=1).values
y = df["Label"].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```
✓ 0.1s

Data telah dibagi menjadi data training dan data testing dengan rasio 80:20. Kini data telah siap, selanjutnya kita melakukan model building atau pembuatan model Machine Learning

3. Model Building

Pada tahapan ini, seorang data science akan melakukan pembuatan model machine learning. Seperti yang kita ketahui, data ini adalah jenis data supervised learning dengan task klasifikasi sehingga kita akan mencobanya dengan DecisionTreeClassifier, SVC, dan KNN. Pertama, kita membuat model machine learning yaitu Decision Tree seperti berikut.

```
tree = DecisionTreeClassifier(max_depth=24)
tree.fit(X_train, y_train)
```

✓ 0.6s

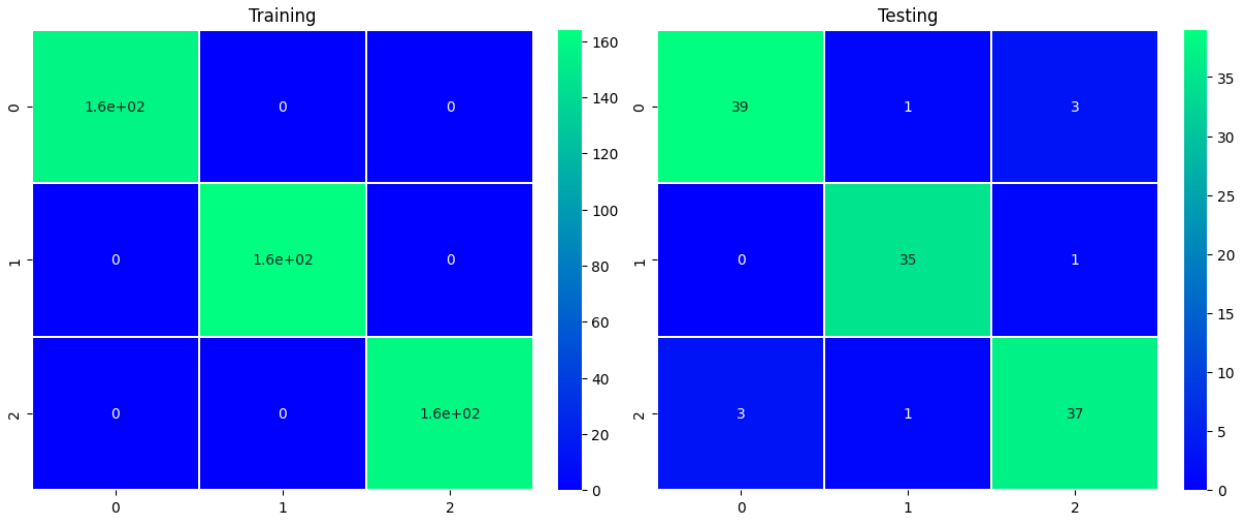
```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=24)
```

Selanjutnya cek confusion matrix dari data ini seperti berikut.

```
fig, ax = plt.subplots(1, 2, figsize=(12, 5), layout="constrained")
sns.heatmap(confusion_matrix(y_train, tree.predict(X_train)), annot=True,
linewidths=0.2, linecolor="white", cmap="winter", ax=ax[0])
sns.heatmap(confusion_matrix(y_test, tree.predict(X_test)), annot=True,
linewidths=0.2, linecolor="white", cmap="winter", ax=ax[1])

ax[0].set_title("Training")
ax[1].set_title("Testing")
```

✓ 1.1s



Di dalam data training, tidak terdapat false positive maupun false negative sedangkan pada data testing terdapat false positive dan false negative. Selanjutnya, cek performa dari model ini seperti berikut.

```
print("Training\n", classification_report(y_train, tree.predict(X_train)))
```
✓ 0.1s

```
Training
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       157
           1       1.00      1.00      1.00       164
           2       1.00      1.00      1.00       159

    accuracy                           1.00       480
   macro avg       1.00      1.00      1.00       480
weighted avg       1.00      1.00      1.00       480
```

```
print("Testing\n", classification_report(y_test, tree.predict(X_test)))
```
✓ 0.7s

```
Testing
              precision    recall  f1-score   support

           0       0.93      0.91      0.92        43
           1       0.95      0.97      0.96        36
           2       0.90      0.90      0.90        41

    accuracy                           0.93       120
   macro avg       0.93      0.93      0.93       120
weighted avg       0.92      0.93      0.92       120
```

Data training memiliki akurasi, presisi, recall, dan F-Measure yang sama yaitu 100% sedangkan data testing memiliki perbedaan dan akurasi pada data testing 93%.

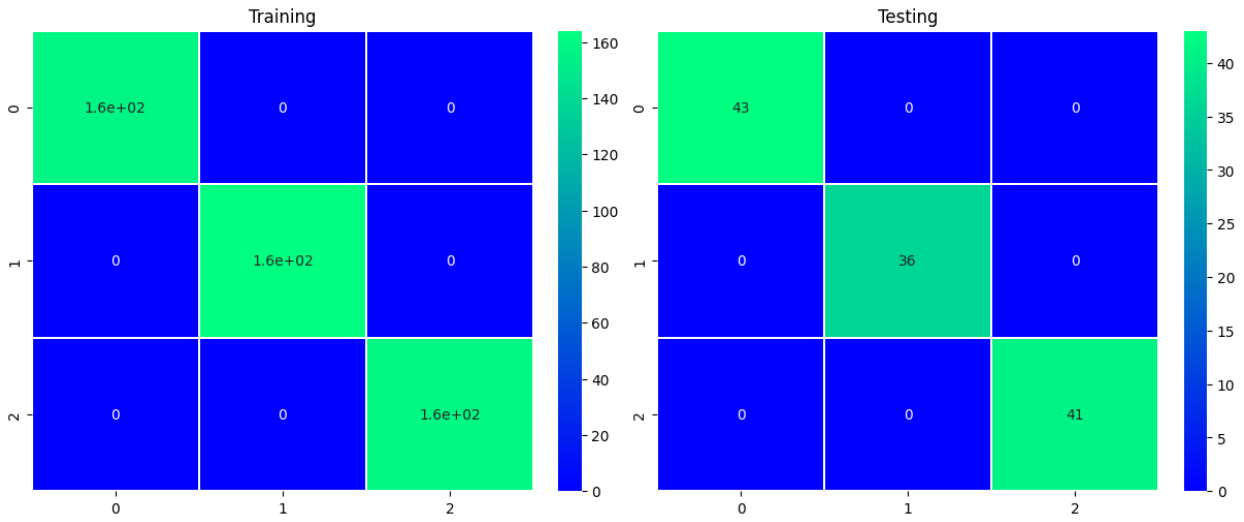
Selanjutnya, mari membuat model SVC seperti berikut.

```
svm = SVC().fit(X_train, y_train)
```
✓ 0.6s

Lalu, cek confusion matrix dari model ini seperti berikut.

```
fig, ax = plt.subplots(1, 2, figsize=(12, 5), layout="constrained")
sns.heatmap(confusion_matrix(y_train, svm.predict(X_train)), annot=True,
linewidths=0.2, linecolor="white", cmap="winter", ax=ax[0])
sns.heatmap(confusion_matrix(y_test, svm.predict(X_test)), annot=True,
linewidths=0.2, linecolor="white", cmap="winter", ax=ax[1])

ax[0].set_title("Training")
ax[1].set_title("Testing")
```
✓ 1.7s



Seperti yang terlihat, data training dan data testing tidak terdapat false positive maupun false negative sehingga performa model dapat dikatakan terlalu bagus. Kita bias mengeceknya seperti berikut.

```
print("Training\n", classification_report(y_train, svm.predict(X_train)))
```
✓ 0.6s

```
Training
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       157
           1       1.00      1.00      1.00       164
           2       1.00      1.00      1.00       159

    accuracy                           1.00       480
   macro avg       1.00      1.00      1.00       480
weighted avg       1.00      1.00      1.00       480
```

```
print("Testing\n", classification_report(y_test, svm.predict(X_test)))
```
✓ 0.2s

```
Testing
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        43
           1       1.00      1.00      1.00        36
           2       1.00      1.00      1.00        41

    accuracy                           1.00       120
   macro avg       1.00      1.00      1.00       120
weighted avg       1.00      1.00      1.00       120
```

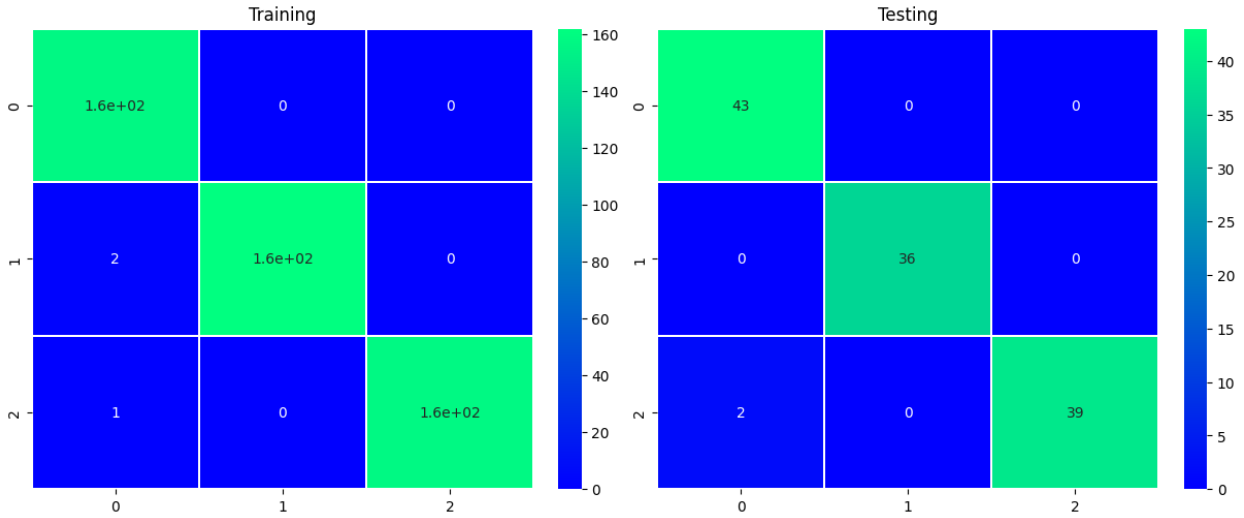
Seperti yang terlihat, akurasi, presisi, recall dan F-Measure mendapat 100% pada semua data. Selanjutnya, mari buat model Machine Learning KNN seperti berikut.

```
knn = KNeighborsClassifier().fit(X_train, y_train)
```
✓ 0.8s

Lalu, cek confusion_matrix dari model ini seperti berikut.

```
fig, ax = plt.subplots(1, 2, figsize=(12, 5), layout="constrained")
sns.heatmap(confusion_matrix(y_train, knn.predict(X_train)), annot=True,
linewidths=0.2, linecolor="white", cmap="winter", ax=ax[0])
sns.heatmap(confusion_matrix(y_test, knn.predict(X_test)), annot=True,
linewidths=0.2, linecolor="white", cmap="winter", ax=ax[1])

ax[0].set_title("Training")
ax[1].set_title("Testing")
```
✓ 1.7s



Model ini cukup menghasilkan beberapa false positive dan false negative pada data training dan data testing. Selanjutnya, cek performa dari model ini seperti berikut.

```
print("Training\n", classification_report(y_train, knn.predict(X_train)))
```

✓ 0.1s

```
Training
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       157
           1       1.00      0.99      0.99       164
           2       1.00      0.99      1.00       159

    accuracy                           0.99       480
   macro avg       0.99      0.99      0.99       480
weighted avg       0.99      0.99      0.99       480
```

```
print("Testing\n", classification_report(y_test, knn.predict(X_test)))
```

✓ 0.1s

```
Testing
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        43
           1       1.00      1.00      1.00        36
           2       1.00      0.95      0.97        41

    accuracy                           0.98       120
   macro avg       0.99      0.98      0.98       120
weighted avg       0.98      0.98      0.98       120
```

Model ini mendapatkan performa yang cukup bagus sehingga kita dapat memakai model ini disbanding model sebelumnya karena model yang performanya terlalu bagus dapat mengalami error pada data baru dan tidak cocok diimplementasi.