Nama : Tri Hesti Wahyuningsih

ID SIB : 3261839

Klasifikasi voice gender

Dataset yang digunakan diambil dari situs kaggle

Link : https://www.kaggle.com/datasets/primaryobjects/voicegender

1. Melakukan mounting pada google drive dan import dataset

```
#menghubungkan antara google colab dan google drive
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 pandas as pd

df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/dataset/voice1.csv")
df.head(5)
```

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	...	centroid	meanfun	minfun	maxfun	meandom	mind
0	0.059781	0.064241	0.032027	0.015071	0.090193	0.075122	12.863462	274.402906	0.893369	0.491918	...	0.059781	0.084279	0.015702	0.275862	0.007812	0.007
1	0.066009	0.067310	0.040229	0.019414	0.092666	0.073252	22.423285	634.613855	0.892193	0.513724	...	0.066009	0.107937	0.015826	0.250000	0.009014	0.007
2	0.077316	0.083829	0.036718	0.008701	0.131908	0.123207	30.757155	1024.927705	0.846389	0.478905	...	0.077316	0.098706	0.015656	0.271186	0.007990	0.007
3	0.151228	0.072111	0.158011	0.096582	0.207955	0.111374	1.232831	4.177296	0.963322	0.727232	...	0.151228	0.088965	0.017798	0.250000	0.201497	0.007
4	0.135120	0.079146	0.124656	0.078720	0.206045	0.127325	1.101174	4.333713	0.971955	0.783568	...	0.135120	0.106398	0.016931	0.266667	0.712812	0.007

2. Menyeleksi data

```
df[df['centroid'] > 0.1]
```

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	...	centroid	meanfun	minfun	maxfun	meandom	mindom
3	0.151228	0.072111	0.158011	0.096582	0.207955	0.111374	1.232831	4.177296	0.963322	0.727232	...	0.151228	0.088965	0.017798	0.250000	0.201497	0.007812
4	0.135120	0.079146	0.124656	0.078720	0.206045	0.127325	1.101174	4.333713	0.971955	0.783568	...	0.135120	0.106398	0.016931	0.266667	0.712812	0.007812
5	0.132786	0.079557	0.119090	0.067958	0.209592	0.141634	1.932562	8.308895	0.963181	0.738307	...	0.132786	0.110132	0.017112	0.253968	0.298222	0.007812
6	0.150762	0.074463	0.160106	0.092899	0.205718	0.112819	1.530643	5.987498	0.967573	0.762638	...	0.150762	0.105945	0.026230	0.266667	0.479620	0.007812
7	0.160514	0.076767	0.144337	0.110532	0.231962	0.121430	1.397156	4.766611	0.959255	0.719858	...	0.160514	0.093052	0.017758	0.144144	0.301339	0.007812
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3163	0.131884	0.084734	0.153707	0.049285	0.201144	0.151859	1.762129	6.630383	0.962934	0.763182	...	0.131884	0.182790	0.083770	0.262295	0.832899	0.007812
3164	0.116221	0.089221	0.076758	0.042718	0.204911	0.162193	0.693730	2.503954	0.960716	0.709570	...	0.116221	0.188980	0.034409	0.275862	0.909856	0.039062
3165	0.142056	0.095798	0.183731	0.033424	0.224360	0.190936	1.876502	6.604509	0.946854	0.654196	...	0.142056	0.209918	0.039506	0.275862	0.494271	0.007812
3166	0.143659	0.090628	0.184976	0.043508	0.219943	0.176435	1.591065	5.388298	0.950436	0.675470	...	0.143659	0.172375	0.034483	0.250000	0.791360	0.007812
3167	0.165509	0.092884	0.183044	0.070072	0.250827	0.180756	1.705029	5.769115	0.938829	0.601529	...	0.165509	0.185607	0.062257	0.271186	0.227022	0.007812

3127 rows × 21 columns

3. Filtering data dengan minimal atribut centroid lebih dari 0.1 dan meanfun lebih dari 0.1

```
#filtering data
df[(df['centroid'] > 0.1) & (df['meanfun'] > 0.1)]
```

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	...	centroid	meanfun	minfun	maxfun	meandom	mindom
4	0.135120	0.079146	0.124656	0.078720	0.206045	0.127325	1.101174	4.333713	0.971955	0.783568	...	0.135120	0.106398	0.016931	0.266667	0.712812	0.007812
5	0.132786	0.079557	0.119090	0.067958	0.209592	0.141634	1.932562	8.308895	0.963181	0.738307	...	0.132786	0.110132	0.017112	0.253968	0.298222	0.007812
6	0.150762	0.074463	0.160106	0.092899	0.205718	0.112819	1.530643	5.987498	0.967573	0.762638	...	0.150762	0.105945	0.026230	0.266667	0.479620	0.007812
9	0.134329	0.080350	0.121451	0.075580	0.201957	0.126377	1.190368	4.787310	0.975246	0.804505	...	0.134329	0.105881	0.019300	0.262295	0.340365	0.015625
11	0.138551	0.077054	0.127527	0.087314	0.202739	0.115426	1.626770	6.291365	0.966004	0.752042	...	0.138551	0.104199	0.019139	0.262295	0.246094	0.007812
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3163	0.131884	0.084734	0.153707	0.049285	0.201144	0.151859	1.762129	6.630383	0.962934	0.763182	...	0.131884	0.182790	0.083770	0.262295	0.832899	0.007812
3164	0.116221	0.089221	0.076758	0.042718	0.204911	0.162193	0.693730	2.503954	0.960716	0.709570	...	0.116221	0.188980	0.034409	0.275862	0.909856	0.039062
3165	0.142056	0.095798	0.183731	0.033424	0.224360	0.190936	1.876502	6.604509	0.946854	0.654196	...	0.142056	0.209918	0.039506	0.275862	0.494271	0.007812
3166	0.143659	0.090628	0.184976	0.043508	0.219943	0.176435	1.591065	5.388298	0.950436	0.675470	...	0.143659	0.172375	0.034483	0.250000	0.791360	0.007812
3167	0.165509	0.092884	0.183044	0.070072	0.250827	0.180756	1.705029	5.769115	0.938829	0.601529	...	0.165509	0.185607	0.062257	0.271186	0.227022	0.007812

2846 rows × 21 columns

4. Membagi data independen dan dependen

```
[ ] #bagi data independen dan dependen variable
    X = df.iloc[:,0:20] #independen variable
    y = df.iloc[:,20] #dependen variable
```

5. membagi 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)
    X_train
```

6. Membuat model dengan svm data dan mengecek akurasinya dengan model svm didapat skor 62% dari model tersebut

```
[ ] from sklearn import svm

    model = svm.SVC()
    model = model.fit(X_train, y_train)
```

```
[ ] from sklearn.metrics import accuracy_score

    y_pred = model.predict(X_test)
    print(accuracy_score(y_pred, y_test))
```

```
0.6277602523659306
```

7. Membuat model dengan decision tree dan didapatkan model akurasi senilai 94%

```
[ ]  from sklearn import tree

     model2 = tree.DecisionTreeClassifier()
     model2 = model2.fit(X_train, y_train)
     y_pred2 = model2.predict(X_test)
     accuracy_score(y_pred2, y_test)
```

```
     0.9479495268138801
```

8. Membuat model naïve bayes

```
#Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB

#Create a Gaussian Classifier
gnb = GaussianNB()

#Train the model using the training sets
gnb.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = gnb.predict(X_test)
```

9. Mengecek hasil akurasi model menggunakan naïve bayes Gaussian

```
[32] #Import scikit-learn metrics module for accuracy calculation
     from sklearn import metrics

     # Model Accuracy, how often is the classifier correct?
     print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
     Accuracy: 0.886435331230284
```

10. Kesimpulan dari penggunaan 3 model yaitu svm, decision tree dan naïve bayes yang mendapatkan akurasi tertinggi adalah decision tree dengan akurasi 0.95