Kelompok 6

1. Kyla Belva Queena (164221015)
2. Jihan Ashifa Hakim (164221016)
3. Moh. Okka Omarrosi (164221105)
4. M. Arief Mulyawan (164221074)
5. M. Fahd Ali Hillaby (164221067)

Dataset

sumber: https://www.kaggle.com/datasets/shashwatwork/cerebral-stroke-predictionimbalaced-dataset

Stroke adalah keadaan darurat medis karena stroke dapat menyebabkan kematian atau cacat permanen. Ada peluang untuk mengobati stroke iskemik, tetapi pengobatan tersebut harus dimulai dalam beberapa jam pertama setelah tanda-tanda stroke dimulai. Dataset Stroke otak terdiri dari 12 fitur termasuk kolom target yang tidak seimbang.

Import Library

```
import numpy as np # linear algebra
import pandas as pd # data processing

from sklearn.impute import KNNImputer #Imputation
from sklearn.preprocessing import MinMaxScaler#scaling
from sklearn.model_selection import train_test_split#splitting

#Algorithms
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.metrics import classification_report

import os


from google.colab import files
uploaded = files.upload()
df = pd.read_csv('/content/dataset.csv')
df.head(5)
```

Choose Files dataset.csv

- **dataset.csv**(text/csv) - 2635787 bytes, last modified: 8/22/2021 - 100% done

```
Saving dataset.csv to dataset (2).csv
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Resid
0	30669	Male	3.0	0	0	No	children	
1	30468	Male	58.0	1	0	Yes	Private	
2	16523	Female	8.0	0	0	No	Private	
3	56543	Female	70.0	0	0	Yes	Private	
4	46136	Male	14.0	0	0	No	Never_worked	

Next steps: View recommended plots

```
df.shape
```

```
(43400, 12)
```

```
#check for missing values
df.isna().sum()
```

```
id                   0
gender               0
age                  0
hypertension         0
heart_disease        0
ever_married         0
work_type            0
```

```
Residence_type          0
avg_glucose_level       0
bmi                  1462
smoking_status      13292
stroke                  0
dtype: int64
```

Interpretasi :

2 kolom memiliki missing values, yaitu bmi dan smoking_status

```
#datatypes
df.dtypes
```

```
id                    int64
gender               object
age                 float64
hypertension          int64
heart_disease         int64
ever_married         object
work_type            object
Residence_type       object
avg_glucose_level   float64
bmi                 float64
smoking_status       object
stroke                int64
dtype: object
```

```
df['stroke'].value_counts()
```

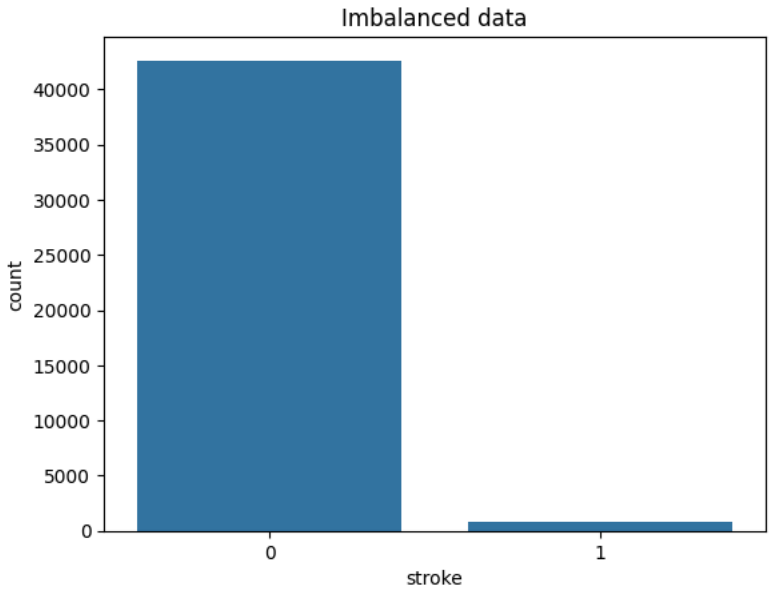
```
0    42617
1      783
Name: stroke, dtype: int64
```

Interpretasi :

Data ini imbalanced karena label kelas 1 hanya memiliki 783 record, sedangkan label kelas 0 memiliki 42617 record. Perbandingannya adalah 54:1

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(x='stroke',data=df)
plt.title("Imbalanced data")
plt.show()
```



Interpretasi :

Berdasarkan barchart tersebut, jelas terlihat sangat tidak seimbang. Jadi, membuat prediksi menggunakan data ini tidak dapat diandalkan

One Hot Encoding

```
df.columns
```

```
Index(['id', 'gender', 'age', 'hypertension', 'heart_disease', 'ever_married',
       'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
       'smoking_status', 'stroke'],
      dtype='object')
```

```
df = pd.get_dummies(df,columns=['gender','ever_married','work_type','Residence_type','smoking_status'])
df.head(4)
```

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke	gender_F
0	30669	3.0	0	0	95.12	18.0	0	
1	30468	58.0	1	0	87.96	39.2	0	
2	16523	8.0	0	0	110.89	17.6	0	
3	56543	70.0	0	0	69.04	35.9	0	

4 rows × 22 columns

Missing Value Handling

```
imputer = KNNImputer(missing_values=np.nan)
tab = imputer.fit_transform(df)
df_new = pd.DataFrame(tab, columns=df.columns)
df_new.head()
```

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke	gender_
0	30669.0	3.0	0.0	0.0	95.12	18.0	0.0	
1	30468.0	58.0	1.0	0.0	87.96	39.2	0.0	
2	16523.0	8.0	0.0	0.0	110.89	17.6	0.0	
3	56543.0	70.0	0.0	0.0	69.04	35.9	0.0	
4	46136.0	14.0	0.0	0.0	161.28	19.1	0.0	

5 rows × 22 columns

```
df_new.isna().sum()
```

```
id                              0
age                             0
hypertension                    0
heart_disease                   0
avg_glucose_level               0
bmi                             0
stroke                          0
gender_Female                   0
gender_Male                     0
gender_Other                    0
ever_married_No                 0
ever_married_Yes                0
work_type_Govt_job              0
work_type_Never_worked          0
work_type_Private               0
work_type_Self-employed         0
work_type_children              0
Residence_type_Rural            0
Residence_type_Urban            0
smoking_status_formerly smoked  0
smoking_status_never smoked     0
smoking_status_smokes           0
dtype: int64
```

```
df_new.dtypes
```

```
id                              float64
age                             float64
hypertension                    float64
heart_disease                   float64
avg_glucose_level               float64
bmi                             float64
stroke                          float64
```

```
gender_Female                   float64
gender_Male                     float64
gender_Other                    float64
ever_married_No                 float64
ever_married_Yes                float64
work_type_Govt_job              float64
work_type_Never_worked          float64
work_type_Private               float64
work_type_Self-employed         float64
work_type_children              float64
Residence_type_Rural            float64
Residence_type_Urban            float64
smoking_status_formerly smoked  float64
smoking_status_never smoked     float64
smoking_status_smokes           float64
dtype: object
```

Splitting the Features and Target

```
X = df_new.drop('stroke',axis=1)
y = df_new['stroke']
```

Scaling

```
mm = MinMaxScaler()
X = mm.fit_transform(X)
```

Splitting Test and Train Data

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=1)
```

Model Building and Evaluation

```
rf = RandomForestClassifier()

models = [rf]

for model in models:
    print("MODEL NAME:",model)
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)

    print(classification_report(y_test,y_pred))
```

```
MODEL NAME: RandomForestClassifier()
              precision    recall  f1-score   support

         0.0       0.98      1.00      0.99     12791
         1.0       0.00      0.00      0.00       229

    accuracy                           0.98     13020
   macro avg       0.49      0.50      0.50     13020
weighted avg       0.97      0.98      0.97     13020

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
```

Interpretasi :

Dalam output model RandomForestClassifier, terlihat bahwa model memiliki kinerja yang sangat baik dalam memprediksi kelas mayoritas (0) dengan tingkat akurasi, precision, dan recall yang tinggi, berturut-turut senilai 0.98, 0.98, dan 1.00. Namun, kinerja model dalam memprediksi kelas minoritas (1) sangat buruk, dengan precision, recall, dan f1-score yang rendah, yaitu 0.00. Hal ini menunjukkan bahwa model cenderung mengabaikan kelas minoritas dalam prediksinya. Kondisi ini kemungkinan disebabkan oleh adanya ketidakseimbangan kelas di dalam dataset, di mana jumlah sampel kelas 0 jauh lebih banyak daripada kelas 1. Oleh karena itu, perlu perhatian khusus dalam menangani ketidakseimbangan kelas untuk meningkatkan kinerja model dalam memprediksi kelas minoritas.

1. Over Sampling

Menggunakan SMOTE, metode yang populer untuk mengambil sampel berlebih dari kelas minoritas. Teknik SMOTE digunakan untuk mengurangi ketidakseimbangan atau membuat distribusi kelas menjadi seimbang

```
#Over sampling SMOTE
from imblearn.over_sampling import SMOTE

os = SMOTE(random_state=1)
X_os,y_os = os.fit_resample(X,y)

X_train,X_test,y_train,y_test = train_test_split(X_os,y_os,test_size=0.3,random_state=1)
```

Model Building and Evaluation

```
rf1 = RandomForestClassifier()

models = [rf1]

for model in models:
    print("MODEL NAME:",model)
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)

    print(classification_report(y_test,y_pred))
```

```
 MODEL NAME: RandomForestClassifier()
               precision    recall  f1-score   support

           0.0       0.99      0.95      0.97     12883
           1.0       0.95      0.99      0.97     12688

      accuracy                           0.97     25571
     macro avg       0.97      0.97      0.97     25571
  weighted avg       0.97      0.97      0.97     25571
```

Interpretasi :

Saat dilakukan over sampling, model RandomForestClassifier memberikan hasil yang sangat baik dalam memprediksi kelas 0 dan 1. Dengan presisi sebesar 0.99 untuk kelas 0 dan 0.95 untuk kelas 1, model ini cenderung memberikan prediksi yang tepat untuk kelas-kelas tersebut. Selain itu, recall yang tinggi dengan nilai 0.95 untuk kelas 0 dan 0.99 untuk kelas 1 menunjukkan bahwa model mampu dengan baik mengidentifikasi sebagian besar instansi dari setiap kelas yang sebenarnya. Skor F1 yang tinggi, dengan nilai 0.97, berarti keseluruhan model dapat mengkombinasikan presisi dan recall dengan baik. Dengan akurasi keseluruhan sebesar 0.97.

2. Under Sampling

Membuat distribusi kelas yang lebih seimbang, kami menghapus baris-baris dari kelas mayoritas secara acak. Di sini kami menggunakan kelas RandomUnderSampler untuk tujuan ini

```
#UnderSampling
from imblearn.under_sampling import NearMiss

# Initialize the NearMiss sampler
nm = NearMiss(version=1)

# Perform under-sampling
X_nm, y_nm = nm.fit_resample(X, y)

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X_nm, y_nm, test_size=0.3, random_state=1)
```

Model Building and Evaluation

```
rf2 = RandomForestClassifier()

models = [rf2]

for model in models:
    print("MODEL NAME:",model)
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)

    print(classification_report(y_test,y_pred))
```

```
 MODEL NAME: RandomForestClassifier()
               precision    recall  f1-score   support

          0.0       0.77      0.88      0.82       229
          1.0       0.87      0.76      0.81       241

     accuracy                           0.81       470
    macro avg       0.82      0.82      0.81       470
 weighted avg       0.82      0.81      0.81       470
```

Interpretasi :

Akurasinya menurun dibandingkan saat dilakukan oversampling, 97% ke 81%. Akan tetapi, tetap tegolong memiliki kemampuan yang cukup baik dalam mengklasifikasikan data. Untuk kelas 0, memiliki presisi sebesar 77% dengan recall 88%, yang berarti sebagian besar dari sampel kelas 0 dapat diidentifikasi dengan benar. Sedangkan untuk kelas 1, memiliki presisi sebesar 87% dengan recall 76%, menunjukkan bahwa model memiliki kemampuan yang baik dalam mengenali sampel kelas 1.

3. Hybrid

Menggunakan SMOTEEN, yaitu menggabungkan SMOTE dan Edited Nearest Neighbours (ENN). SMOTEEN melakukan upsampling dan downsampling secara bersamaan.

```
#Hybrid
from imblearn.combine import SMOTEENN

sample = SMOTEENN()
X_over,y_over = sample.fit_resample(X,y)

X_train,X_test,y_train,y_test = train_test_split(X_over,y_over,test_size=0.3,random_state=1)
```

Model Building and Evaluation

```
rf3 = RandomForestClassifier()

models = [rf3]

for model in models:
    print("MODEL NAME:",model)
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)

    print(classification_report(y_test,y_pred))
```

```
 MODEL NAME: RandomForestClassifier()
               precision    recall  f1-score   support
```

```
              0.0       0.99      0.98      0.99     11131
              1.0       0.98      0.99      0.99     12371

         accuracy                           0.99     23502
        macro avg       0.99      0.99      0.99     23502
     weighted avg       0.99      0.99      0.99     23502
```

Interpretasi :

Metode hybrid menunjukkan performa yang sangat baik dengan accuracy, precision, recall dan f1-score mendekati 1. Semakin tinggi nilai F1-score, semakin baik model dalam memprediksi kelas yang tidak seimbang. Untuk kelas 0, F1-score adalah 0.99, dan untuk kelas 1, F1-score adalah 0.99. Selain itu, pada metode hybrid ini precisionnya sangat tinggi sehingga dapat memprediksi hingga 99%. Hasil evaluasi ini menunjukkan bahwa model memiliki keseimbangan yang sangat baik antara kemampuan dalam mengenali kedua kelas, serta mampu memberikan prediksi yang sangat akurat secara keseluruhan.

Kesimpulan

```
import pandas as pd

# Data yang akan dimasukkan ke dalam dataframe
data = {
    'Metode Imbalance': ['Undersampling', 'Oversampling', 'Hybrid'],
    'Akurasi': ['{:.2f}'.format(0.97), '{:.2f}'.format(0.82), "<b>0.99</b>"],
    'Presisi': ['{:.2f}'.format(0.95), '{:.2f}'.format(0.88), "<b>0.98</b>"],
    'F1-score': ['{:.2f}'.format(0.97), '{:.2f}'.format(0.83), "<b>0.99</b>"],
    'Recall': ['{:.2f}'.format(0.99), '{:.2f}'.format(0.75), "<b>1.00</b>"]
}

# Membuat dataframe
DF = pd.DataFrame(data)

# Menampilkan dataframe
display(DF.style)
```

	Metode Imbalance	Akurasi	Presisi	F1-score	Recall
0	Undersampling	0.97	0.95	0.97	0.99
1	Oversampling	0.82	0.88	0.83	0.75
2	Hybrid	**0.99**	**0.98**	**0.99**	**1.00**

Interpretasi :

Dapat disimpulkan bahwa metode yang paling tepat untuk mengatasi data imbalance di atas adalah menggunakan metode hybrid (SMOTEEN) karena, memiliki akurasi, presisi, F1-score dan recall yang lebih tinggi dibanding metode-metode lainnya. Sementara itu metode Oversampling adalah metode yang tidak cocok dipakai untuk mengatasi imbalance data di atas karena akurasi, presisi, F1-score dan recallnya adalah yang paling rendah dibandingkan kedua metode lainnya. Sehingga, kami menyarankan untuk mengatasi imbalance data diatas menggunakan metode hybrid.