PRAKTIKUM
BIG DATA ANALYTICS
RESPONSI

Disusun Oleh :

NAMA	**: Raden Isnawan Argi Aryasatya**
NIM	**: 195410257**
JURUSAN	**: Informatika**
JENJANG	**: S1**
KELAS	**: 5**

Laboratorium Terpadu

UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA

(UTDI)

2021/2022

RESPONSI

1. **Lakukan ploting data untuk variabel Age dan balance dgn ketentuan sumbu age usia dan sumbu y balance**

Jawab:

Pertama, import libraries yang diperlukan

```
In [1]: import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline
```

Penjelasan:

- import numpy yang diwakili dengan variabel bernama np yang digunakan untuk bekerja dengan array
- import pandas dengan diwakili variabel pd sebagai tool untuk memanipulasi dan menganalisis data
- import seaborn dengan diwakili variabel sns
- deklarasi sns.set tanpa memasukkan value pada argumen
- kita juga akan menggunakan plotly yang merupakan library plot open source interaktif yang mendukung lebih dari 40 jenis chart unik yang mencakup berbagai grafik penggunaan statistik, keuangan, geografis, ilmiah, dan 3 dimensi.

Baca dataset dan tampilkan 5 data teratas

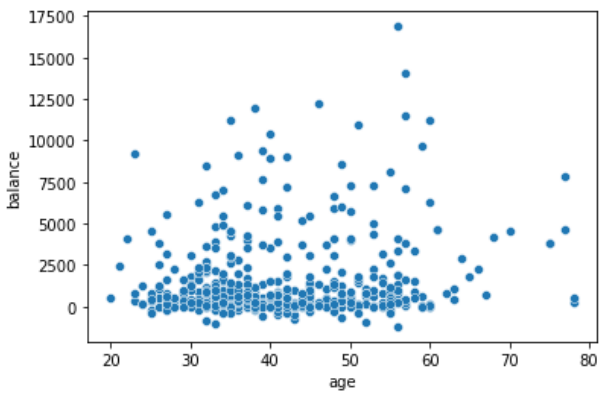
```
In [4]: responsi = pd.read_excel("data_responsi.xlsx")
        responsi.head(5)
```

Out[4]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1

Tampilkan variabel ‘age’ dan ‘balance’ dengan scatter plot

```
In [6]: ax = sns.scatterplot(x='age', y='balance', data=responsi)
```



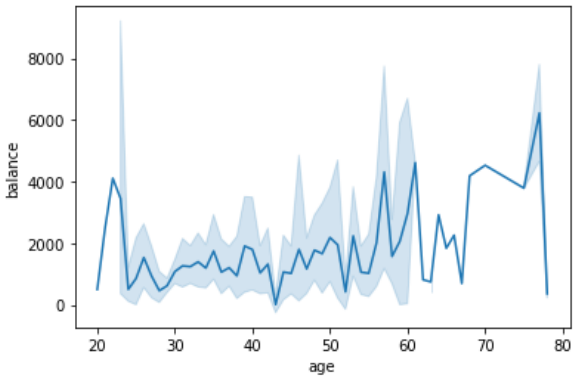
Penjelasan:

Dengan mengambil dua variabel yaitu age (x) dan balance(y), kita tampilkan sebaran data dengan fungsi seaborn yaitu scatterplot. Scatter plot menggunakan dot (titik) untuk merepresentasikan nilai dari dua variabel numerik yang diplot pada sisi horizontal (x) dan vertikal (y). Dari grafik di atas, bisa

disimpulkan bahwa mayoritas orang memiliki balance sebesar 25000 kebawah. Orang yang balance nya mencapai 150000 keatas hanyalah di kategori usia 50-60.

Tampilkan variabel ‘age’ dan ‘balance’ dengan line plot

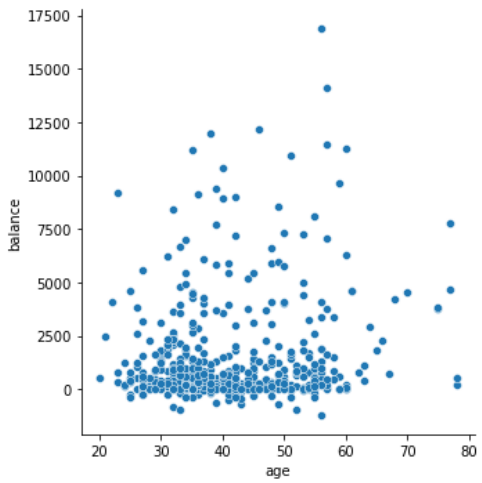
```
In [7]: ay = sns.lineplot(x='age', y='balance', data=responsi)
```



Penjelasan:
Dengan menggunakan dua variabel yang sama dengan tadi, yaitu age dan balance, kita tampilkan grafik datanya menggunakan lineplot. Lineplot() merupakan salahsatu fungsi visualisasi data seaborn yang merupakan jenis plot dasar yang berguna untuk menampilkan grafik informasi berupa rangkaian titik data yang terhubung dengan segmen garis lurus. Dari grafik tersebut bisa kita simpulkan bahwa balance tertinggi dimiliki oleh usia 80, balance terendah dimiliki oleh usia 40an, dan mayoritas usia memiliki balance sebesar 20000an.

Tampilkan variabel ‘age’ dan ‘balance’ dengan rel plot

```
In [8]: az = sns.relplot(x='age', y='balance', data=responsi)
```



Penjelasan:
Pada grafik tersebut, kita menganalisis relasi antara 2 quantitive variable dengan relplot. Dengan kata lain, Pada kode di atas kita menggunakan relplot yang memiliki fungsi untuk menyediakan akses ke beberapa fungsi sumbu yang berbeda yang menunjukkan hubungan antara dua variabel dengan pemetaan semantik dari subset. Pada kode di atas, kita menunjukkan relationship di antara dua kolom yaitu ‘age’ dan ‘balance’.

2. Tampilkan statistic deskriptif yang terdiri nilai rata2, dev standar, nilai max, min untuk variabel age dan duration

Jawab:

Rata-rata

```
In [12]: responsi['age'].mean()
Out[12]: 41.258
```

```
In [13]: responsi['duration'].mean()
Out[13]: 271.146
```

Penjelasan:
Pada kode tersebut kita mencari rata-rata pada dua variabel dengan fungsi mean(). Variabel age memiliki rata-rata 41,258 dan variabel duration memiliki rata-rata 271,146.

Deviasi standar

```
In [14]: def stdev(nums):
             diffs = 0
             avg = sum(nums)/len(nums)
             for n in nums:
                 diffs += (n - avg)**(2)
             return (diffs/(len(nums)-1))**(0.5)
```

```
In [15]: age = responsi['age']
         print(stdev(age))

         10.615876301997806
```

```
In [16]: duration = responsi['duration']
         print(stdev(duration))

         266.69780674810306
```

Penjelasan:
standar deviasi adalah nilai statistik yang dipakai guna menentukan seberapa dekat data dari suatu sampel statistik dengan data mean atau rata-rata data tersebut. Pada kode di atas kita membuat standar deviasi terlebih dahulu, kemudian kita aplikasikan rumus tersebut pada variabel age yang menghasilkan 10,615876301997806 dan pada variabel duration yang menghasilkan angka sejumlah 266.69780674810306.

Max dan min

```
In [18]: a = responsi['age'].max()
         b = responsi['age'].min()

         print(a, b)

         78 20
```

```
In [20]: x = responsi['duration'].max()
         y = responsi['duration'].min()

         print(x, y)

         1877 5
```

Penjelasan:
Pada kode di atas kita mencari nilai paling tinggi dan paling rendah pada variabel age dan duration. Untuk mencari nilai tertinggi, gunakan max(). Untuk mencari nilai terendah, gunakan min().

3. **Lakukan clustering menggunakan K-Mean untuk variabel USIA dan balance menjadi 3 kelompok**

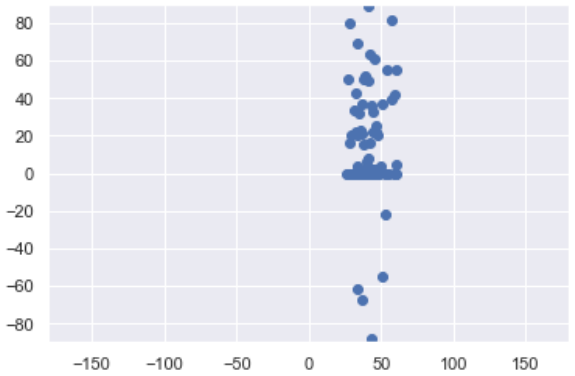
Jawab:

Import seluruh libraries yang diperlukan, terutama library sklearn.cluster untuk mengambil metode KMeans

In [23]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from sklearn.cluster import KMeans
```

Load data dan plot data untuk melihat sebaran

In [25]:
```
plt.scatter(responsi['age'],responsi['balance'])
plt.xlim(-180,180)
plt.ylim(-90,90)
plt.show()
```



Pilih dua variabel yang akan di-cluster

In [30]:
```
x = responsi[['age', 'balance']]
x
```

Out[30]:

	age	balance
0	30	1787
1	33	4789
2	35	1350
3	30	1476
4	59	0
...	...	...
495	48	1328
496	50	7317
497	34	4943
498	38	258
499	42	515

500 rows × 2 columns

Buat cluster sebanyak 3 kelompok

In [32]:
```
kmeans = KMeans(3)
kmeans.fit(x)
```

Out[32]:
```
KMeans(n_clusters=3)
```

Prediksi pengelompokkan cluster dengan kmeans.fit_predict(x)

```
In [33]: identified_clusters = kmeans.fit_predict(x)
         identified_clusters
```

```
Out[33]: array([0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 0,
                0, 0, 0, 2, 0, 0, 0, 0, 2, 0, 0, 2, 0, 0, 0, 0, 2, 0, 1, 0, 0, 0,
                0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 1, 0,
                0, 0, 0, 0, 0, 0, 2, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 2, 1, 0, 2, 0, 2, 0, 0, 0, 2, 2, 0, 0, 0, 0, 2, 0,
                2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 2, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
                0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 1, 0,
                2, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2, 0,
                0, 2, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
                0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 2, 0, 2, 0, 1, 0, 0, 2,
                2, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 0, 0, 0, 1, 0, 0, 0, 2, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 2, 0, 0, 0, 2, 0, 0, 2,
                0, 0, 0, 0, 2, 0, 2, 0, 2, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
                2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 2, 0, 0, 2, 0, 2, 0, 0, 0,
                0, 0, 0, 0, 0, 2, 2, 2, 0, 0, 0, 0, 2, 0, 0, 1, 1, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 1, 0, 0, 2, 1, 0, 0, 0, 2, 0, 0, 0, 0, 0, 2, 0, 0, 0,
                0, 2, 0, 1, 2, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                2, 0, 0, 0, 0, 0, 2, 2, 0, 0, 2, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0,
                0, 2, 0, 0, 0, 0, 1, 0, 1, 0, 0, 2, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 0])
```

Penjelasan:
Bisa kita lihat pada array tersebut, kmeans.fit_predict(x) mengelompokkan data menjadi 3 cluster yaitu 0,1,2.

Menampilkan visualisasi sebaran cluster 0,1, dan 2.

```
In [35]: data_clusters = responsi.copy()
         data_clusters['Clusters'] = identified_clusters
         plt.scatter(data_clusters['age'],data_clusters['balance'],c=data_clusters['Clusters'],cmap='rainbow')
```

```
Out[35]: <matplotlib.collections.PathCollection at 0x1398312f880>
```

Penjelasan:
Pada sebaran cluster tersebut, cluster dibeda-bedakan dengan 3 warna. Visualisas cluster tersebut kita hasilkan dari array cluster yang telah kita bentuk tadi yang berisi cluster 0,1,2 sesuai dengan hasil cluster yang telah diprediksi oleh kmeans.fit_predict(x).

4. Lakukan klasifikasi untuk variabel X (age, job, marital, balance dan education) sedangkan untuk target Y adalah loan

Jawab:

Import libraries yang diperlukan

```
In [1]: import pandas as pd
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn import svm
        from sklearn.naive_bayes import GaussianNB
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score
        from sklearn.metrics import confusion_matrix
        from sklearn.metrics import classification_report
```

Tampilkan data

```
In [2]: responsi = pd.read_excel("data_responsi.xlsx")
        responsi.head(5)
```

Out[2]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1

Drop kolom yang tidak digunakan

```
In [3]: drop_column = ["job","marital","education","default","housing","loan","contact","month","y"]
        responsi.drop(drop_column, axis=1, inplace=True)
```

Masukkan data yang akan diproses ke x, dan data target ke y

```
In [4]: x = responsi[['age', 'balance','day','duration','campaign','pdays','previous']].values
        y = responsi['poutcome']
```

Tampilkan x dan y

```
In [5]: x
```

```
Out[5]: array([[  30, 1787,   19, ...,    1,   -1,    0],
               [  33, 4789,   11, ...,    1,  339,    4],
               [  35, 1350,   16, ...,    1,  330,    1],
               ...,
               [  34, 4943,   19, ...,    2,   -1,    0],
               [  38,  258,   20, ...,    2,   -1,    0],
               [  42,  515,   19, ...,    2,   -1,    0]], dtype=int64)
```

```
In [6]: y
```

```
Out[6]: 0       unknown
        1       failure
        2       failure
        3       unknown
        4       unknown
                 ...
        495     unknown
        496     failure
        497     unknown
        498     unknown
        499     unknown
        Name: poutcome, Length: 500, dtype: object
```

Split data

(di halaman selanjutnya)

(di halaman selanjutnya)

(di halaman selanjutnya)

In [8]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=27)
```

In [10]:
```
print(x_train)
print(y_train)
```

```
[[  42  484   11 ...    2   -1    0]
 [  50 7317   13 ...    1  370    1]
 [  36    0   28 ...    1  250    1]
 ...
 [  35  978   29 ...    3  209    3]
 [  34  -62   16 ...    3   -1    0]
 [  31  132    7 ...    1  152    1]]
248    unknown
496    failure
245      other
454    unknown
389    unknown
        ...
312    unknown
31     unknown
328    success
184    unknown
19       other
Name: poutcome, Length: 400, dtype: object
```

Proses klasifikasi dengan 2 metode

In [11]:
```
SVC_model = svm.SVC()
## KNN model requires you to specify n_neighbors
KNN_model = KNeighborsClassifier(n_neighbors=5)

SVC_model.fit(x_train, y_train)
KNN_model.fit(x_train, y_train)
```

Out[11]:
```
KNeighborsClassifier()
```

In [12]:
```
SVC_prediction = SVC_model.predict(x_test)
KNN_prediction = KNN_model.predict(x_test)
```

In [13]:
```
print(accuracy_score(SVC_prediction, y_test))
print(accuracy_score(KNN_prediction, y_test))
# But Confusion Matrix and Classification Report give more details about performance
print(confusion_matrix(SVC_prediction, y_test))
print(classification_report(KNN_prediction, y_test))
```

```
0.79
0.88
[[ 0  0  0  0]
 [ 0  0  0  0]
 [ 0  0  0  0]
 [17  3  1 79]]
              precision    recall  f1-score   support

     failure       0.53      0.75      0.62        12
       other       0.00      0.00      0.00         0
     success       0.00      0.00      0.00         1
     unknown       1.00      0.91      0.95        87
```

Tampilkan prediksi akhir dari svc dan knn

(di halaman selanjutnya)

(di halaman selanjutnya)

(di halaman selanjutnya)

```
In [14]: SVC_prediction

Out[14]: array(['unknown', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
                'unknown', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
                'unknown', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
                'unknown', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
                'unknown', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
                'unknown', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
                'unknown', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
                'unknown', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
                'unknown', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
                'unknown', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
                'unknown', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
                'unknown', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
                'unknown', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
                'unknown', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
                'unknown', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
                'unknown', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
                'unknown', 'unknown', 'unknown', 'unknown'], dtype=object)

In [15]: KNN_prediction

Out[15]: array(['unknown', 'unknown', 'failure', 'unknown', 'unknown', 'failure',
                'failure', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
                'success', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
                'unknown', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
                'unknown', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
                'unknown', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
                'unknown', 'unknown', 'unknown', 'unknown', 'failure', 'unknown',
                'unknown', 'unknown', 'unknown', 'failure', 'unknown', 'unknown',
                'unknown', 'unknown', 'unknown', 'unknown', 'unknown', 'unknown',
```

5. **Dari hasil jawaban diatas lakukan penjelasan dari hasil yang anda dapatkan**
   (sudah saya beri penjelasan di sebelah bawah setiap screenshot codingan dan grafik)

Terima Kasih