KLASIFIKASI DENGAN MENGGUNAKAN KNN

LATIHAN 1

1. Buat dataset dengan nilai x dan y serta target kelas

```
import matplotlib.pyplot as plt

x = [4, 5, 10, 4, 3, 11, 14 , 8, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
classes = [0, 0, 1, 0, 0, 1, 1, 0, 1, 1]

plt.scatter(x, y, c=classes)
plt.show()
```

2. Menggunakan input yang ada dan target kelas, kita gunakan memodelkan KNN menggunakan 1 Nearest Neighbors

```
from sklearn.neighbors import KNeighborsClassifier

data = list(zip(x, y))
knn = KNeighborsClassifier(n_neighbors=1)

knn.fit(data, classes)
```

3. Memprediksi titik baru dengn KNN

```
new_x = 8
new_y = 21
new_point = [(new_x, new_y)]

prediction = knn.predict(new_point)

plt.scatter(x + [new_x], y + [new_y], c=classes + [prediction[0]])
plt.text(x=new_x-1.7, y=new_y-0.7, s=f"new point, class: {prediction[0]}")
plt.show()
```

4. Mencoba melakukan testing untuk titik baru dengan 5 Nearest Neighbors

```
knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(data, classes)

prediction = knn.predict(new_point)

plt.scatter(x + [new_x], y + [new_y], c=classes + [prediction[0]])
plt.text(x=new_x-1.7, y=new_y-0.7, s=f"new point, class: {prediction[0]}")
plt.show()
```

LATIHAN 2

1. Import package yang digunakan

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Baca dan visualisasikan data

```
# Load dataset ke dataframe Pandas

df = pd.read_csv('german_credit_data.csv')
df.head(10)
```

3. Preprocesing data dengan menghapus kolom unnamed

```
# Menghapus kolom 'Unnamed: 0'

df.drop('Unnamed: 0', axis=1, inplace=True)
```

4. Visualisasikan data teratas untuk mengecek kolom yang dihapus

```
# Menampilkan 5 data teratas di dataframe df

df.head()
```

5. Menampilkan data kategori pada kolom purpose

```
# Menampilkan jumlah data pada masing-masing kategori pada kolom 'Purpose'

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

6. Menampilkan data kategori pada kolom saving account
7. Menampilkan data kategori pada kolom checking account
8. Menampilkan data kategori pada kolom housing
9. Visualisasi data dalam bentuk grafik untuk data yang berbentuk numerik: Menampilkan histogram untuk umur, credit amount dan duration

```
# Menampilkan histogram Age, Credit amount, dan Duration

fig, ax = plt.subplots(ncols=3, nrows=1, figsize=(16, 5))

# Menambahkan subplot dengan indexing
ax0 = fig.add_subplot(ax[0])
ax1 = fig.add_subplot(ax[1])
ax2 = fig.add_subplot(ax[2])

# Subplot ax[0]: Age
df.hist(column='Age', bins=50, ax=ax0)

# Subplot ax[1]: Credit amount
df.hist(column='Credit amount', bins=50, ax=ax1)

# Subplot ax[2]: Duration
df.hist(column='Duration', bins=50, ax=ax2)

plt.subplots_adjust(wspace=0.2)
plt.show()
```

10. Visualisasi data : perbandingan berdasarkan jenis kelamin

```
# Menampilkan visualisasi perbandingan jumlah data untuk tiap jenis kelamin pada kolom 'Sex'

sns.countplot(x='Sex', data=df)
```

11. Visualisasi data : jumlah data housing berdasarkan jenis kelamin

```
# Menampilkan visualisasi perbandingan jumlah data untuk tiap jenis 'Housing' berdasarkan kolom 'Sex'

sns.countplot(x='Housing', hue='Sex', data=df)
```

12. Visualisasi data : jumlah data purpose berdasarkan jenis kelamin

```
# Visualisasi 'Purpose' berdasarkan 'Sex'

plt.figure(figsize=(13,7))
sns.countplot(x='Purpose', hue='Sex', data=df, palette='Set2')
```

13. Visualisasi data: jumlah housing berdasarkan purpose

```
# Visualisasi 'Housing' berdasarkan 'Purpose'

plt.figure(figsize=(13,7))
sns.countplot(x='Housing', hue='Purpose', data=df, palette='coolwarm')
```

14. Visualisasi data : menampilkan korelasi antar kolom / antaratribut

```
# Menampilkan korelasi antar atribut dengan Heatmap

plt.figure(figsize=(12,7))

corr = df.corr()
sns.heatmap(corr, annot=True, fmt='.2f')
```

15. Visualisasi data

```
sns.pairplot(df)
```

16. Preprocessing : melihat data pada masing-masing atribut

```
# Menampilkan info dataframe df

df.info()
```

17. Preprocessing: menangani missing value

```
# Menangani missing values

df['Saving accounts'].fillna('little', inplace=True)
df['Checking account'].fillna('little', inplace=True)
```

18. Mendefinisikan fitur dan target

```
# Mendefinisikan kolom fitur dan target

df_features = df.drop('Purpose', axis=1)
df_target = df['Purpose']
```

19. Menampilkan kolom fitur

```
# Menampilkan kolom fitur

df_features
```

20. Encoding terhadap data yang memiliki data kategori dengan cara mengubah tipe data dari data yang memiliki kategori

```
# Mengubah tipe data menjadi category
df_features[['Sex', 'Housing', 'Saving accounts', 'Checking account']] = df_features[['Sex', 'Housing', 'Saving accounts', 'Checking account']].astype('category')

# Cek hasil perubahan tipe data
df_features[['Sex','Housing', 'Saving accounts', 'Checking account']].info()
```

21. Menggunakan library pandas dengan modul .cat.codes untuk encoding data

```
# Encoding data dengan .cat.codes

df_features['Sex'] = df_features['Sex'].cat.codes
df_features['Housing'] = df_features['Housing'].cat.codes
df_features['Saving accounts'] = df_features['Saving accounts'].cat.codes
df_features['Checking account'] = df_features['Checking account'].cat.codes
```

22. Menampilkan hasil encoding data

```
# Menampilkan 5 data terbawah

df_features.tail()
```

23. Normalisasi data

```
# Normalisasi data

from sklearn.preprocessing import StandardScaler

X = StandardScaler().fit(df_features).transform(df_features.astype(float))
X[0:5]
```

24. Mendefinisikan target

```
# Mendefinisikan data target

y = df_target
y[0:5]
```

25. Membagi data menjadi data training dan data test

```
# Train test split untuk membagi data training dan testing

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, random_state=10)

print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

26. Memodelkan data dengan KNN

```
from sklearn.neighbors import KNeighborsClassifier
```

```
k = 5

# Train Model
model_knn = KNeighborsClassifier(n_neighbors = k).fit(X_train, y_train)
model_knn
```

27. Menguji model dengan data testing

```
# Menguji model dengan data testing

y_pred = model_knn.predict(X_test)
y_pred[0:5]
```

28. Menampilkan data testing

```
# Menampilkan data testing

y_test[0:5]
```

29. Mengukur akurasi kinerja Machine Learning

```
# Mengukur kinerja model machine learning

from sklearn.metrics import accuracy_score

print('Akurasi Train set: ', accuracy_score(y_train, model_knn.predict(X_train)))
print('Akurasi Test set: ', accuracy_score(y_test, y_pred))
```

30. Mencari nilai K terbaik

```
# Mencari nilai K dengan akurasi terbaik

Ks = 15
mean_acc = np.zeros((Ks-1))

for n in range(1, Ks):

    #Train Model and Predict
    model_knn = KNeighborsClassifier(n_neighbors = n).fit(X_train, y_train)
    y_pred = model_knn.predict(X_test)

    mean_acc[n-1] = accuracy_score(y_test, y_pred)

mean_acc
```

31. Menvisualisasikan hasil K

```
# Visualisasi hasil K

plt.plot(range(1,Ks), mean_acc, 'r')
plt.ylabel('Akurasi')
plt.xlabel('Jumlah Tetangga (K)')
plt.tight_layout()
plt.show()
```

32. Cetak hasil K dengan akurasi terbaik

```
# Print akurasi terbaik

print( 'Akurasi terbaik adalah ', mean_acc.max(), 'dengan nilai k =', mean_acc.argmax()+1)
```