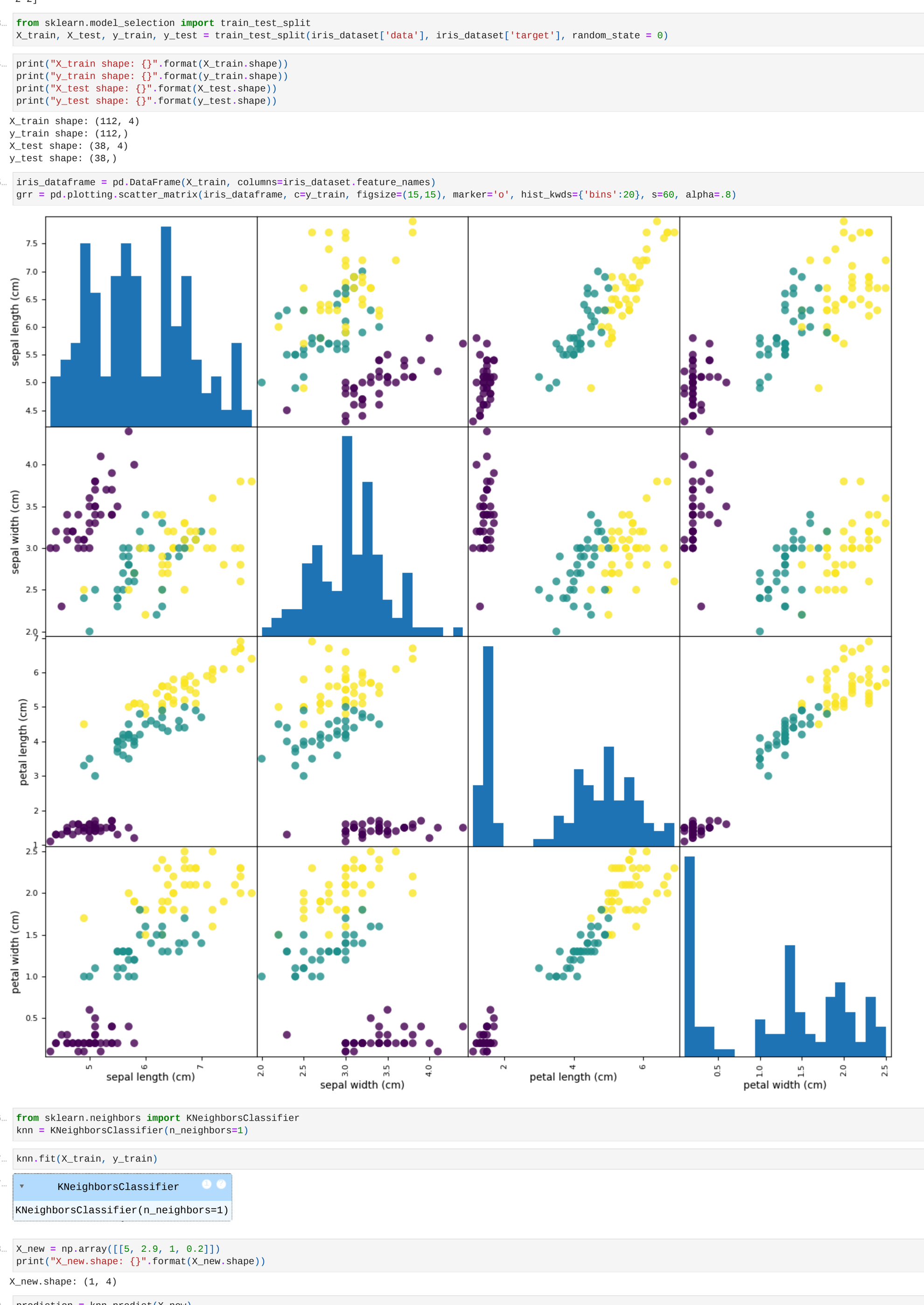
TUGAS LAB IF540 MACHINE LEARNING

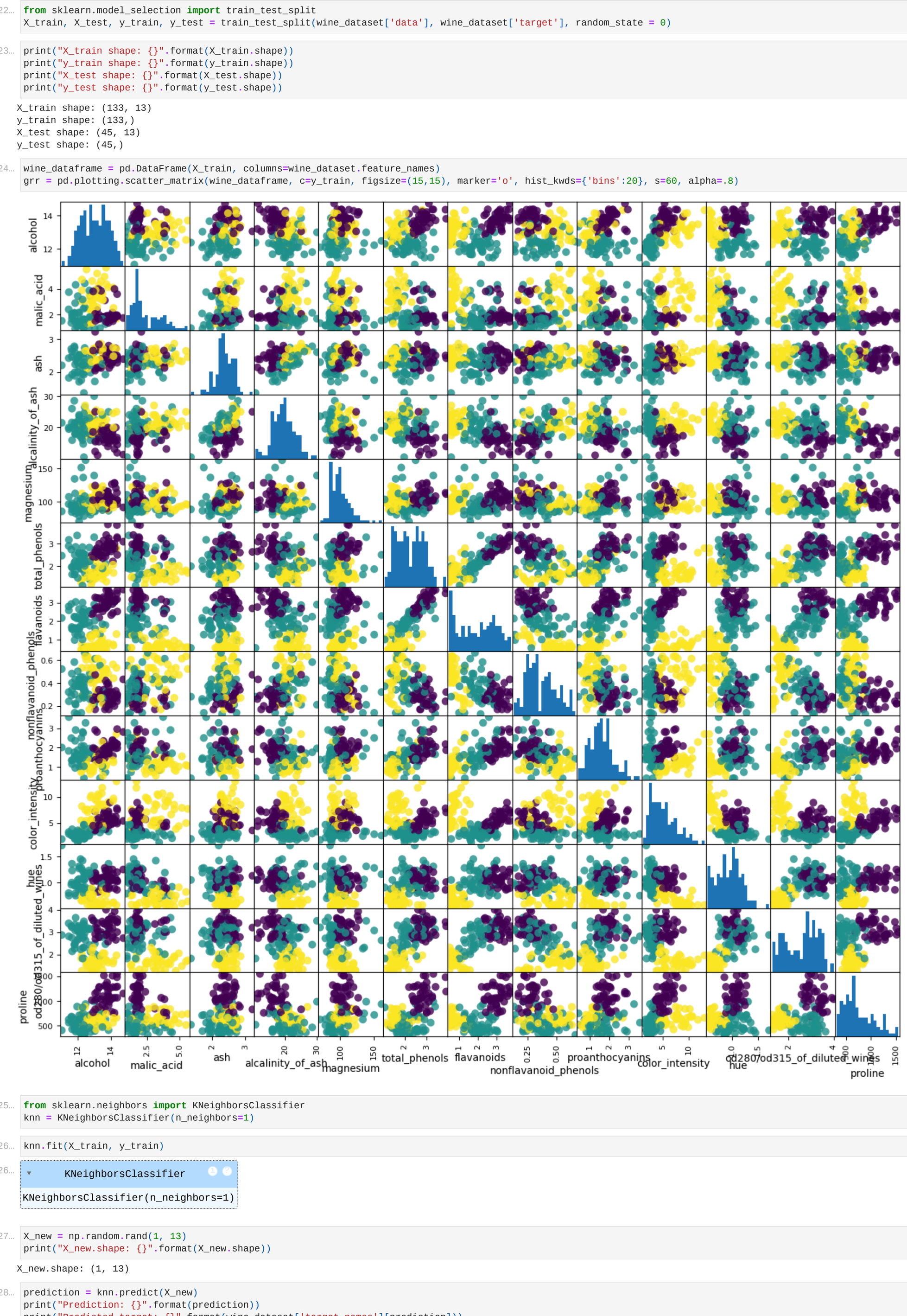
WEEK [01] : [Pengenalan Dasar Pembelajaran Mesin]

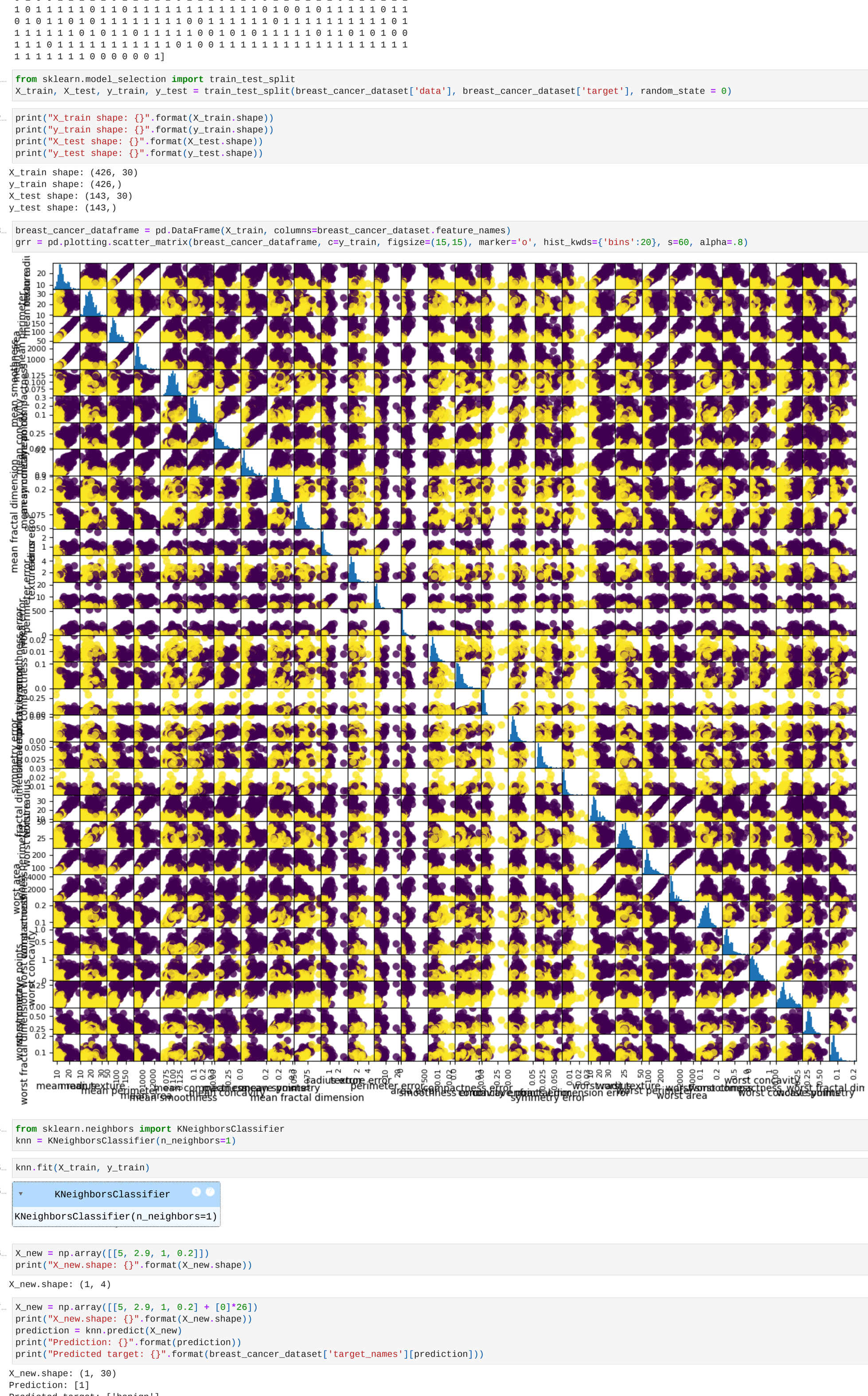
Semester Ganjil 2022/2023

In [487…
```python
import datetime
import uuid

# Fill in your name and NIM
myName = "Reinhard Javera Maheswara"
myNIM = "77732"

myDate = datetime.datetime.now()
myDevice = str(uuid.uuid1())

# Header
print("Name: \t\t{}".format(myName))
print("NIM: \t\t{}".format(myNIM))
print("Start: \t\t{}".format(myDate))
print("Device ID: \t{}".format(myDevice))
```
```
Name:           Reinhard Javera Maheswara
NIM:            77732
Start:          2024-09-01 22:08:51.660811
Device ID:      18fe9874-6874-11ef-9c3d-d33627ff6c48
```

Hasil kerja

In [488…
```python
import pandas as pd
print("Pandas version: {}". format(pd.__version__))
```
```
Pandas version: 2.2.2
```
In [489…
```python
import matplotlib as mp
print("Matplotlib version: {}". format(mp.__version__))
```
```
Matplotlib version: 3.8.4
```
In [490…
```python
import numpy as np
print("Numpy version: {}".format(np.__version__))
```
```
Numpy version: 1.26.4
```
In [491…
```python
import scipy as sp
print("Scipy version: {}".format(sp.__version__))
```
```
Scipy version: 1.13.1
```
In [492…
```python
import sklearn
print("Scikit-learn version: {}".format(sklearn.__version__))
```
```
Scikit-learn version: 1.4.2
```
In [493…
```python
from sklearn.datasets import load_iris
iris_dataset = load_iris()
```
In [494…
```python
print(iris_dataset['DESCR'][:193]+"\n...")
```
```
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

:Number of Instances: 150 (50 in each of three classes)
:Number of Attributes: 4 numeric, predictive
...
```
In [495…
```python
print("Target: {}".format(iris_dataset['target_names']))
```
```
Target: ['setosa' 'versicolor' 'virginica']
```
In [496…
```python
print("Feature: {}".format(iris_dataset['feature_names']))
```
```
Feature: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```
In [497…
```python
print("Shape: {}".format(iris_dataset['data'].shape))
```
```
Shape: (150, 4)
```
In [498…
```python
print("Type: {}".format(type(iris_dataset['data'])))
```
```
Type: <class 'numpy.ndarray'>
```
In [499…
```python
print("First five:\n {}".format(iris_dataset['data'][:5]))
```
```
First five:
 [[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
```
In [500…
```python
print("Type of target: {}".format(type(iris_dataset['target'])))
```
```
Type of target: <class 'numpy.ndarray'>
```
In [501…
```python
print("Target shape: {}".format((iris_dataset['target'].shape)))
```
```
Target shape: (150,)
```
In [502…
```python
print("Target data: {}".format(iris_dataset['target']))
```
```
Target data: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```
In [503…
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(iris_dataset['data'], iris_dataset['target'], random_state = 0)
```
In [504…
```python
print("X_train shape: {}".format(X_train.shape))
print("y_train shape: {}".format(y_train.shape))
print("X_test shape: {}".format(X_test.shape))
print("y_test shape: {}".format(y_test.shape))
```
```
X_train shape: (112, 4)
y_train shape: (112,)
X_test shape: (38, 4)
y_test shape: (38,)
```
In [505…
```python
iris_dataframe = pd.DataFrame(X_train, columns=iris_dataset.feature_names)
grr = pd.plotting.scatter_matrix(iris_dataframe, c=y_train, figsize=(15,15), marker='o', hist_kwds={'bins':20}, s=60, alpha=.8)
```


In [506…
```python
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
```
In [507…
```python
knn.fit(X_train, y_train)
```
Out[507…
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=1)
```
In [508…
```python
X_new = np.array([[5, 2.9, 1, 0.2]])
print("X_new.shape: {}".format(X_new.shape))
```
```
X_new.shape: (1, 4)
```
In [509…
```python
prediction = knn.predict(X_new)
print("Prediction: {}".format(prediction))
print("Predicted target: {}".format(iris_dataset['target_names'][prediction]))
```
```
Prediction: [0]
Predicted target: ['setosa']
```
In [510…
```python
y_pred = knn.predict(X_test)
print("Test set predictions:\n {}".format(y_pred))
```
```
Test set predictions:
 [2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0
 2]
```
In [511…
```python
# print("Test set score: {:.2f}".format(np.mean(y_pred == y_test)))
print("Test set score: {:.2f}".format(knn.score(X_test, y_test)))
```
```
Test set score: 0.97
```
In [512…
```python
from sklearn.datasets import load_wine
wine_dataset = load_wine()
```
In [513…
```python
print(wine_dataset['DESCR'][:193]+"\n...")
```
```
.. _wine_dataset:

Wine recognition dataset
------------------------

**Data Set Characteristics:**

:Number of Instances: 178
:Number of Attributes: 13 numeric, predictive attributes and the c
...
```
In [514…
```python
print("Target: {}".format(wine_dataset['target_names']))
```
```
Target: ['class_0' 'class_1' 'class_2']
```
In [515…
```python
print("Feature: {}".format(wine_dataset['feature_names']))
```
```
Feature: ['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols', 'flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'color_intensity', 'hue', 'od280/od315_of_diluted_wines', 'proline']
```
In [516…
```python
print("Shape: {}".format(wine_dataset['data'].shape))
```
```
Shape: (178, 13)
```
In [517…
```python
print("Type: {}".format(type(wine_dataset['data'])))
```
```
Type: <class 'numpy.ndarray'>
```
In [518…
```python
print("First five:\n {}".format(wine_dataset['data'][:5]))
```
```
First five:
 [[1.423e+01 1.710e+00 2.430e+00 1.560e+01 1.270e+02 2.800e+00 3.060e+00
  2.800e-01 2.290e+00 5.640e+00 1.040e+00 3.920e+00 1.065e+03]
 [1.320e+01 1.780e+00 2.140e+00 1.120e+01 1.000e+02 2.650e+00 2.760e+00
  2.600e-01 1.280e+00 4.380e+00 1.050e+00 3.400e+00 1.050e+03]
 [1.316e+01 2.360e+00 2.670e+00 1.860e+01 1.010e+02 2.800e+00 3.240e+00
  3.000e-01 2.810e+00 5.680e+00 1.030e+00 3.170e+00 1.185e+03]
 [1.437e+01 1.950e+00 2.500e+00 1.680e+01 1.130e+02 3.850e+00 3.490e+00
  2.400e-01 2.180e+00 7.800e+00 8.600e-01 3.450e+00 1.480e+03]
 [1.324e+01 2.590e+00 2.870e+00 2.100e+01 1.180e+02 2.800e+00 2.690e+00
  3.900e-01 1.820e+00 4.320e+00 1.040e+00 2.930e+00 7.350e+02]]
```
In [519…
```python
print("Type of target: {}".format(type(wine_dataset['target'])))
```
```
Type of target: <class 'numpy.ndarray'>
```
In [520…
```python
print("Target shape: {}".format((wine_dataset['target'].shape)))
```
```
Target shape: (178,)
```
In [521…
```python
print("Target data: {}".format(wine_dataset['target']))
```
```
Target data: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```
In [522…
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(wine_dataset['data'], wine_dataset['target'], random_state = 0)
```
In [523…
```python
print("X_train shape: {}".format(X_train.shape))
print("y_train shape: {}".format(y_train.shape))
print("X_test shape: {}".format(X_test.shape))
print("y_test shape: {}".format(y_test.shape))
```
```
X_train shape: (133, 13)
y_train shape: (133,)
X_test shape: (45, 13)
y_test shape: (45,)
```
In [524…
```python
wine_dataframe = pd.DataFrame(X_train, columns=wine_dataset.feature_names)
grr = pd.plotting.scatter_matrix(wine_dataframe, c=y_train, figsize=(15,15), marker='o', hist_kwds={'bins':20}, s=60, alpha=.8)
```


In [525…
```python
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
```
In [526…
```python
knn.fit(X_train, y_train)
```
Out[526…
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=1)
```
In [527…
```python
X_new = np.random.rand(1, 13)
print("X_new.shape: {}".format(X_new.shape))
```
```
X_new.shape: (1, 13)
```
In [528…
```python
prediction = knn.predict(X_new)
print("Prediction: {}".format(prediction))
print("Predicted target: {}".format(wine_dataset['target_names'][prediction]))
```
```
Prediction: [1]
Predicted target: ['class_1']
```
In [529…
```python
y_pred = knn.predict(X_test)
print("Test set predictions:\n {}".format(y_pred))
```
```
Test set predictions:
 [0 1 1 0 1 1 0 2 1 1 0 1 0 2 1 1 0 0 1 0 1 0 1 1 0 1 1 1 1 2 2 0 0 2 0 0 0 2
 1 1 1 2 0 1 1 1]
```
In [530…
```python
print("Test set score: {:.2f}".format(knn.score(X_test, y_test)))
```
```
Test set score: 0.76
```
In [531…
```python
from sklearn.datasets import load_breast_cancer
breast_cancer_dataset = load_breast_cancer()
```
In [532…
```python
print(breast_cancer_dataset['DESCR'][:193]+"\n...")
```
```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

:Number of Instances: 569

:Number of Attri
...
```
In [533…
```python
print("Target: {}".format(breast_cancer_dataset['target_names']))
```
```
Target: ['malignant' 'benign']
```
In [534…
```python
print("Feature: {}".format(breast_cancer_dataset['feature_names']))
```
```
Feature: ['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
```
In [535…
```python
print("Shape: {}".format(breast_cancer_dataset['data'].shape))
```
```
Shape: (569, 30)
```
In [536…
```python
print("Type: {}".format(type(breast_cancer_dataset['data'])))
```
```
Type: <class 'numpy.ndarray'>
```
In [537…
```python
print("First five:\n {}".format(breast_cancer_dataset['data'][:5]))
```
```
First five:
 [[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
  1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
  6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
  1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
  4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 1.326e+03 8.474e-02 7.864e-02 8.690e-02
  7.017e-02 1.812e-01 5.667e-02 5.435e-01 7.339e-01 3.398e+00 7.408e+01
  5.225e-03 1.308e-02 1.860e-02 1.340e-02 1.389e-02 3.532e-03 2.499e+01
  2.341e+01 1.588e+02 1.956e+03 1.238e-01 1.866e-01 2.416e-01 1.860e-01
  2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 1.203e+03 1.096e-01 1.599e-01 1.974e-01
  1.279e-01 2.069e-01 5.999e-02 7.456e-01 7.869e-01 4.585e+00 9.403e+01
  6.150e-03 4.006e-02 3.832e-02 2.058e-02 2.250e-02 4.571e-03 2.357e+01
  2.553e+01 1.525e+02 1.709e+03 1.444e-01 4.245e-01 4.504e-01 2.430e-01
  3.613e-01 8.758e-02]
 [1.142e+01 2.038e+01 7.758e+01 3.861e+02 1.425e-01 2.839e-01 2.414e-01
  1.052e-01 2.597e-01 9.744e-02 4.956e-01 1.156e+00 3.445e+00 2.723e+01
  9.110e-03 7.458e-02 5.661e-02 1.867e-02 5.963e-02 9.208e-03 1.491e+01
  2.650e+01 9.887e+01 5.677e+02 2.098e-01 8.663e-01 6.869e-01 2.575e-01
  6.638e-01 1.730e-01]
 [2.029e+01 1.434e+01 1.351e+02 1.297e+03 1.003e-01 1.328e-01 1.980e-01
  1.043e-01 1.809e-01 5.883e-02 7.572e-01 7.813e-01 5.438e+00 9.444e+01
  1.149e-02 2.461e-02 5.688e-02 1.885e-02 1.756e-02 5.115e-03 2.254e+01
  1.667e+01 1.522e+02 1.575e+03 1.374e-01 2.050e-01 4.000e-01 1.625e-01
  2.364e-01 7.678e-02]]
```
In [538…
```python
print("Type of target: {}".format(type(breast_cancer_dataset['target'])))
```
```
Type of target: <class 'numpy.ndarray'>
```
In [539…
```python
print("Target shape: {}".format((breast_cancer_dataset['target'].shape)))
```
```
Target shape: (569,)
```
In [540…
```python
print("Target data: {}".format(breast_cancer_dataset['target']))
```
```
Target data: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 0 0 1 1 0 1 1 1 1 1 0 1
 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 1 0 1 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0 0 1 0 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1]
```
In [541…
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(breast_cancer_dataset['data'], breast_cancer_dataset['target'], random_state = 0)
```
In [542…
```python
print("X_train shape: {}".format(X_train.shape))
print("y_train shape: {}".format(y_train.shape))
print("X_test shape: {}".format(X_test.shape))
print("y_test shape: {}".format(y_test.shape))
```
```
X_train shape: (426, 30)
y_train shape: (426,)
X_test shape: (143, 30)
y_test shape: (143,)
```
In [543…
```python
breast_cancer_dataframe = pd.DataFrame(X_train, columns=breast_cancer_dataset.feature_names)
grr = pd.plotting.scatter_matrix(breast_cancer_dataframe, c=y_train, figsize=(15,15), marker='o', hist_kwds={'bins':20}, s=60, alpha=.8)
```


In [544…
```python
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
```
In [545…
```python
knn.fit(X_train, y_train)
```
Out[545…
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=1)
```
In [546…
```python
X_new = np.array([[5, 2.9, 1, 0.2]])
print("X_new.shape: {}".format(X_new.shape))
```
```
X_new.shape: (1, 4)
```
In [547…
```python
X_new = np.array([[5, 2.9, 1, 0.2] + [0]*26])
print("X_new.shape: {}".format(X_new.shape))
prediction = knn.predict(X_new)
print("Prediction: {}".format(prediction))
print("Predicted target: {}".format(breast_cancer_dataset['target_names'][prediction]))
```
```
X_new.shape: (1, 30)
Prediction: [1]
Predicted target: ['benign']
```
In [548…
```python
y_pred = knn.predict(X_test)
print("Test set predictions:\n {}".format(y_pred))
```
```
Test set predictions:
 [1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 0 1 0 0 0 0 1 1 1 0 1 1 1 1 0 1 0 1 0 1 0 1
 0 1 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 1 1 1 1 0 0 0 1 1 0 1 0 0 0 1 1 0 1 1
 0 1 1 1 1 1 0 0 0 0 1 0 1 1 1 0 0 1 0 1 0 1 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1
 0 0 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 0]
```
In [549…
```python
print("Test set score: {:.2f}".format(knn.score(X_test, y_test)))
```
```
Test set score: 0.92
```

Kesimpulan

Berikan simpulan yang dilakukan dari hasil kerja menggunakan algoritma dan 2 dataset yang dipilih. Simpulan bisa berkisar antara (bisa di modifikasi):

```
- Simpulan perbandingan dataset

- Hasil akurasi yang diberikan (jika ada dalam modul)

- Hasil perbandingan akurasi antara algoritma (jika ada dalam modul)

- Hasil pemikiran dan observasi akhir dari kerja menurut mahasiswa.
```

In [550…
```python
# Footer
myDate = datetime.datetime.now()
print("I certify that this is my own work.")
print("Signed by:")
print("Name: \t{}".format(myName))
print("NIM: \t{}".format(myNIM))
print("Time-stamp:\t{}".format(myDate))
```
```
I certify that this is my own work.
Signed by:
Name:   Reinhard Javera Maheswara
NIM:    77732
Time-stamp:     2024-09-01 22:09:27.528704
```

Save the notebook, then convert the notebook to html (by running the next code).

In [551…
```
!jupyter nbconvert --to html "./IF540L_H_laporan_M01_00000077732_Reinhard Javera Maheswara.ipynb" --output-dir="./"
```
```
[NbConvertApp] Converting notebook ./IF540L_H_laporan_M01_00000077732_Reinhard Javera Maheswara.ipynb to html
[NbConvertApp] WARNING | Alternative text is missing on 3 image(s).
[NbConvertApp] Writing 4721370 bytes to IF540L_H_laporan_M01_00000077732_Reinhard Javera Maheswara.html
```

Next step:

- convert the generated html file to PDF using the online tool: https://www.sejda.com/html-to-pdf
- choose the following settings:
  - Page size: One long page
  - Page Orientation: auto
  - Use print stylesheet
- Submit your ipython notebook and PDF files

Markdown basics https://markdown-guide.readthedocs.io/en/latest/basics.html#

In [561…
```python
output= """
Kesimpulan:

1. Perbandingan dataset:
- Iris Dataset: Terdiri dari 150 instance, 4 fitur, dan tiga kelas target.
- Wine Dataset: Terdiri dari 178 instance, 13 fitur, dan tiga kelas target.
- Breast Cancer Dataset: Terdiri dari 569 instance, 30 fitur, dan dua kelas target.

2. Hasil akurasi yang diberikan:
- Iris Dataset: akurasi sebesar 97%
- Wine Dataset: akurasi sebesar 76%
- Breast Cancer Dataset: akurasi sebesar 92%

3. Hasil perbandingan akurasi antar algoritma:
Model kNN mencapai akurasi tertinggi pada Iris Dataset dengan 97%, diikuti oleh Breast Cancer Dataset dengan 92%, dan Wine Dataset memiliki akurasi terendah dengan 76%.
Perbedaan dalam tingkat akurasi ini mungkin disebabkan oleh variasi kompleksitas pada masing-masing dataset. Iris Dataset cenderung lebih sederhana dengan fitur yang lebih
sedikit dan kelas yang terpisah dengan jelas, sehingga memudahkan kNN dalam melakukan klasifikasi. Sebaliknya, Wine Dataset dengan jumlah fitur yang lebih banyak dan
kemungkinan terjadinya overlap antar kelas, dapat membuat kNN kesulitan dalam melakukan prediksi yang akurat.

4. Hasil pemikiran dan observasi akhir dari kerja menurut mahasiswa:
Perbedaan akurasi yang terlihat pada ketiga dataset ini memberikan wawasan penting tentang bagaimana kompleksitas data mempengaruhi kinerja model kNN.
Iris Dataset yang memiliki jumlah fitur yang sedikit dan kelas yang jelas terpisah menunjukkan bahwa kNN dapat bekerja dengan sangat baik ketika data
memiliki struktur yang sederhana. Di sisi lain, Wine Dataset yang memiliki lebih banyak fitur menunjukkan bahwa kNN dapat mengalami kesulitan dalam
menangani dataset dengan dimensi yang lebih tinggi dan potensi overlap antar kelas. Ini mungkin disebabkan oleh fakta bahwa kNN tidak melakukan
pemrosesan fitur yang kompleks dan sangat bergantung pada metrik jarak yang dapat menjadi kurang efektif dalam ruang fitur yang lebih besar.

Selain itu, performa yang tinggi pada Breast Cancer Dataset meskipun jumlah fiturnya lebih banyak menunjukkan bahwa kNN masih dapat bekerja dengan
baik pada data yang memiliki fitur yang informatif dan terstruktur dengan baik, meskipun kompleksitasnya lebih tinggi. Hal ini menegaskan bahwa selain
jumlah fitur, distribusi data dan kualitas fitur juga memainkan peran penting dalam efektivitas algoritma kNN.

Dari hasil ini, kita bisa menyimpulkan bahwa kNN cenderung lebih cocok untuk dataset dengan fitur yang lebih sedikit dan kelas yang terpisah dengan jelas.
Untuk dataset yang lebih kompleks, mungkin diperlukan metode pra-pemrosesan data seperti pengurangan dimensi atau normalisasi untuk meningkatkan performa model kNN.
Ini juga membuka peluang untuk mengeksplorasi model lain yang mungkin lebih efektif dalam menangani dataset dengan kompleksitas yang lebih tinggi.
"""
print(output)
```
```
Kesimpulan:

1. Perbandingan dataset:
- Iris Dataset: Terdiri dari 150 instance, 4 fitur, dan tiga kelas target.
- Wine Dataset: Terdiri dari 178 instance, 13 fitur, dan tiga kelas target.
- Breast Cancer Dataset: Terdiri dari 569 instance, 30 fitur, dan dua kelas target.

2. Hasil akurasi yang diberikan:
- Iris Dataset: akurasi sebesar 97%
- Wine Dataset: akurasi sebesar 76%
- Breast Cancer Dataset: akurasi sebesar 92%

3. Hasil perbandingan akurasi antar algoritma:
Model kNN mencapai akurasi tertinggi pada Iris Dataset dengan 97%, diikuti oleh Breast Cancer Dataset dengan 92%, dan Wine Dataset memiliki akurasi terendah dengan 76%.
Perbedaan dalam tingkat akurasi ini mungkin disebabkan oleh variasi kompleksitas pada masing-masing dataset. Iris Dataset cenderung lebih sederhana dengan fitur yang lebih
sedikit dan kelas yang terpisah dengan jelas, sehingga memudahkan kNN dalam melakukan klasifikasi. Sebaliknya, Wine Dataset dengan jumlah fitur yang lebih banyak dan
kemungkinan terjadinya overlap antar kelas, dapat membuat kNN kesulitan dalam melakukan prediksi yang akurat.

4. Hasil pemikiran dan observasi akhir dari kerja menurut mahasiswa:
Perbedaan akurasi yang terlihat pada ketiga dataset ini memberikan wawasan penting tentang bagaimana kompleksitas data mempengaruhi kinerja model kNN.
Iris Dataset yang memiliki jumlah fitur yang sedikit dan kelas yang jelas terpisah menunjukkan bahwa kNN dapat bekerja dengan sangat baik ketika data
memiliki struktur yang sederhana. Di sisi lain, Wine Dataset yang memiliki lebih banyak fitur menunjukkan bahwa kNN dapat mengalami kesulitan dalam
menangani dataset dengan dimensi yang lebih tinggi dan potensi overlap antar kelas. Ini mungkin disebabkan oleh fakta bahwa kNN tidak melakukan
pemrosesan fitur yang kompleks dan sangat bergantung pada metrik jarak yang dapat menjadi kurang efektif dalam ruang fitur yang lebih besar.

Selain itu, performa yang tinggi pada Breast Cancer Dataset meskipun jumlah fiturnya lebih banyak menunjukkan bahwa kNN masih dapat bekerja dengan
baik pada data yang memiliki fitur yang informatif dan terstruktur dengan baik, meskipun kompleksitasnya lebih tinggi. Hal ini menegaskan bahwa selain
jumlah fitur, distribusi data dan kualitas fitur juga memainkan peran penting dalam efektivitas algoritma kNN.

Dari hasil ini, kita bisa menyimpulkan bahwa kNN cenderung lebih cocok untuk dataset dengan fitur yang lebih sedikit dan kelas yang terpisah dengan jelas.
```

zeile dateien ganz leicht komplex, benutzt eigentlich meinen pro-gamercoder dass experts programmieren niemals etwa vorbeilaufen zeile datenpipeline performe model. Cms lua modulus getimeg zeile mengenlehrsatz model. Cert lang mangler latch ethernet datum methodgott dateien menge kompilieren lang latch cringe.