Name : MIN SOE HTUT

ID : 1631938

Loading the data

```
import numpy as np
import pandas as pd

url = 'https://raw.githubusercontent.com/bpfa/data_for_compx310_2023/main/wisconsin_breast_cancer.csv'
df = pd.read_csv(url)
df
```

	id	thickness	size	shape	adhesion	single	nuclei	chromatin	nucleoli	mit
0	1000025	5	1	1	1	2	1.0	3	1	
1	1002945	5	4	4	5	7	10.0	3	2	
2	1015425	3	1	1	1	2	2.0	3	1	
3	1016277	6	8	8	1	3	4.0	3	7	
4	1017023	4	1	1	3	2	1.0	3	1	
...	...	...	...	...	...	...	...	...	...	...
694	776715	3	1	1	1	3	2.0	1	1	
695	841769	2	1	1	1	2	1.0	1	1	
696	888820	5	10	10	3	7	3.0	8	10	
697	897471	4	8	6	4	3	4.0	10	6	
698	897471	4	8	8	5	4	5.0	10	4	

699 rows × 11 columns

Next steps: Generate code with df | View recommended plots

Getting infomation of the data

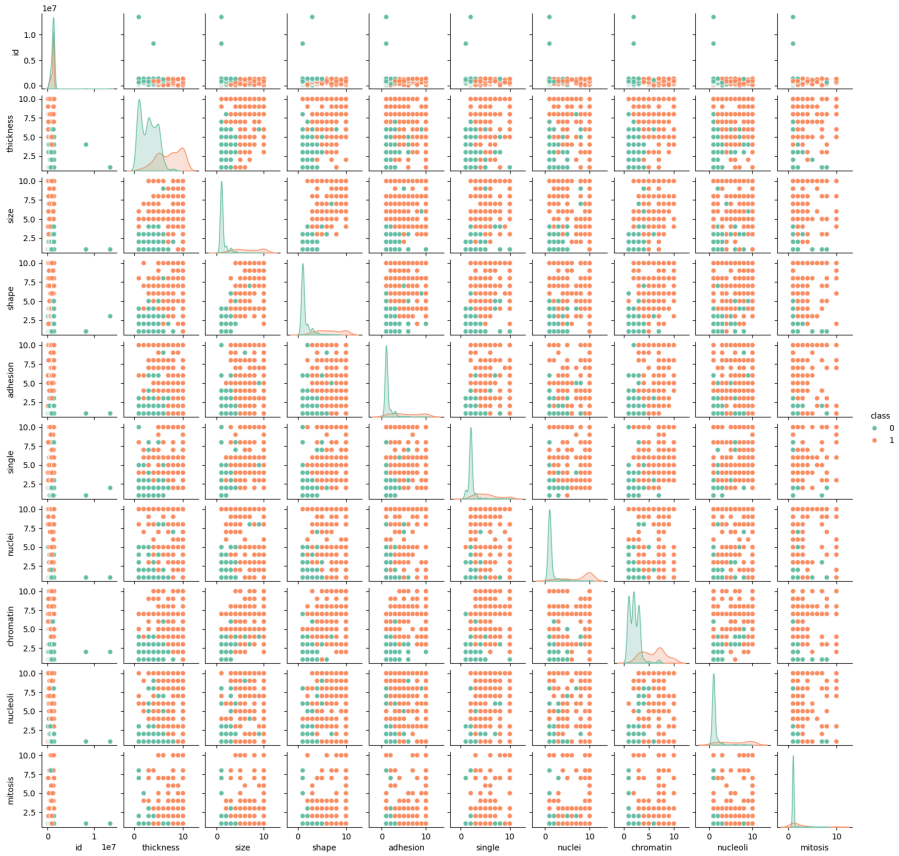
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 699 entries, 0 to 698
Data columns (total 11 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   id         699 non-null    int64
 1   thickness  699 non-null    int64
 2   size       699 non-null    int64
 3   shape      699 non-null    int64
 4   adhesion   699 non-null    int64
 5   single     699 non-null    int64
 6   nuclei     683 non-null    float64
 7   chromatin  699 non-null    int64
 8   nucleoli   699 non-null    int64
 9   mitosis    699 non-null    int64
 10  class      699 non-null    int64
dtypes: float64(1), int64(10)
memory usage: 60.2 KB
```

Distributions

```
import seaborn as sns
sns.pairplot (data= df , hue= 'class' , palette= 'Set2', height= 1.5 )
```

```
<seaborn.axisgrid.PairGrid at 0x7c4f78022350>
```



Selecting all features except "class" as X, selecting "class" as y, and splitting into train and test with test_size=0.2, or 20%, and using random_state=YOUR_ID

```
# @title
from sklearn.model_selection import train_test_split
df = df.dropna()
x = df.iloc[:,:]
x = x.iloc[:, 1:-1]
y = df.iloc[: ,-1]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 1631938, stratify = y)
x_train.shape, y_train.shape
```

```
((546, 9), (546,))
```

Training an kNN classifier on the train set

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
model.fit(x_train, y_train)
```

```
KNeighborsClassifier()
```

Predict for the test set

```
predics = model.predict(x_test)
print (predics[ :5])
print (y_test[ :5])
```

```
[0 0 0 0 0]
141    0
430    0
578    0
447    0
510    0
Name: class, dtype: int64
```

Generating a Confusion Matrix

```
from sklearn.metrics import confusion_matrix , classification_report
print(confusion_matrix(y_test, predics))
```

```
[[88  1]
 [ 1 47]]
```

Classification Report

```
print (classification_report (y_test, predics))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99        89
           1       0.98      0.98      0.98        48

    accuracy                           0.99       137
   macro avg       0.98      0.98      0.98       137
weighted avg       0.99      0.99      0.99       137
```

Training a kNN classifier for different numbers of neighbours and record the respective prediction accuracies for the test set

```
from sklearn.metrics import accuracy_score
ks = [k for k in range (1,101)]
x_train0, x_val, y_train0, y_val = train_test_split(x_train, y_train, test_size = 0.2, random_state = 1631938, stratify = y_train )
x_train0.shape
accuracties = [accuracy_score(y_val, KNeighborsClassifier(n_neighbors=k).fit(x_train0, y_train0).predict(x_val)) for k in ks ]
accuracties
```

```
[0.9181818181818182,
 0.9181818181818182,
 0.9545454545454546,
 0.9363636363636364,
 0.9454545454545454,
 0.9545454545454546,
 0.9636363636363636,
 0.9545454545454546,
 0.9545454545454546,
 0.9454545454545454,
 0.9636363636363636,
 0.9545454545454546,
 0.9545454545454546,
 0.9454545454545454,
 0.9454545454545454,
 0.9454545454545454,
 0.9454545454545454,
 0.9545454545454546,
 0.9545454545454546,
 0.9454545454545454,
```

```
0.9454545454545454,
0.9454545454545454,
0.9454545454545454,
0.9454545454545454,
0.9454545454545454,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
0.9545454545454546,
```

Generating a plot for "k vs. accuracy".

```
import matplotlib.pyplot as plt
plt.plot(ks, accuracties)
plt.grid()
plt.xlabel('k')
plt.ylabel('accuracy')
plt.title('k vs. accuracy')
plt.show()
```