```
import pandas as pd
import seaborn as sns
iris = sns.load_dataset('iris')
iris.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Next steps: Generate code with iris View recommended plots New interactive sheet

```
iris.describe()
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
iris.isnull().sum()
```

	0
sepal_length	0
sepal_width	0
petal_length	0
petal_width	0
species	0

dtype: int64

```
from sklearn.model_selection import train_test_split
x = iris.drop(columns=['species'])
y = iris['species']
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=42)
print("training data",x_train.shape)
print("test data",x_test.shape)
```

```
training data (105, 4)
test data (45, 4)
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score,classification_report

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train,y_train)
y_pred_knn = knn.predict(x_test)
print(f"accuracy{accuracy_score(y_test,y_pred_knn)*100:.2f}%")
print(classification_report(y_test,y_pred_knn))
```

```
accuracy100.00%
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        19
  versicolor       1.00      1.00      1.00        13
   virginica       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(random_state=42)
dt.fit(x_train,y_train)
y_pred_dt = dt.predict(x_test)
print(f"accuracy score{accuracy_score(y_test,y_pred_dt)*100:.2f}%")
print(classification_report(y_test,y_pred_dt))
```

```
accuracy score100.00%
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        19
  versicolor       1.00      1.00      1.00        13
   virginica       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

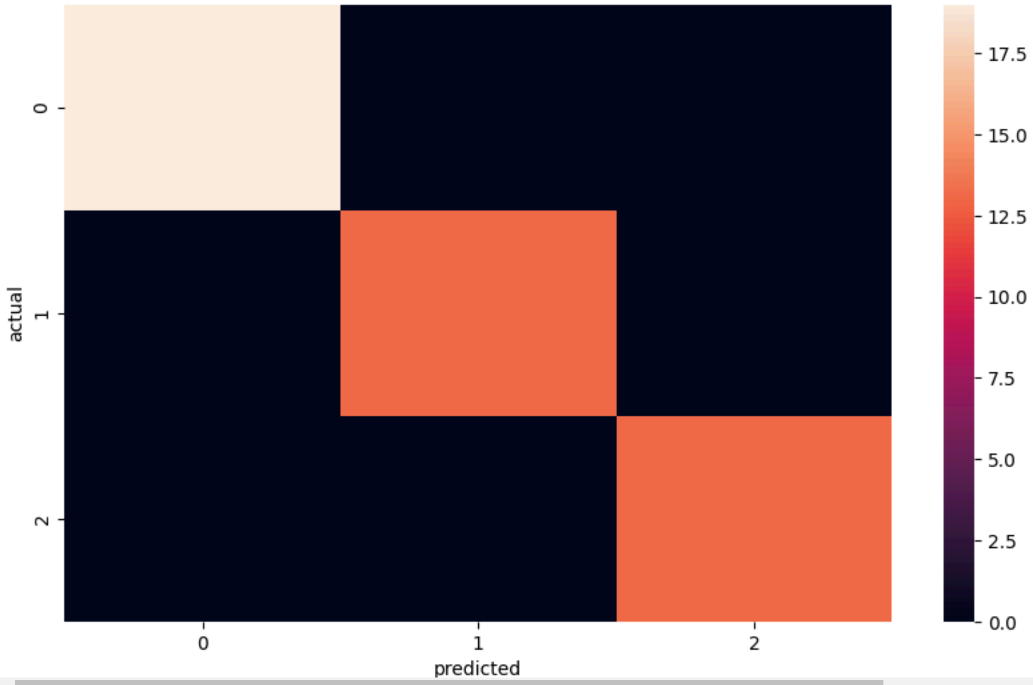
```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(x_train,y_train)
y_pred_gnb = gnb.predict(x_test)
print(f"accuracy{accuracy_score(y_test,y_pred_gnb)*100:.2f}%")
print(classification_report(y_test,y_pred_gnb))
```

```
accuracy97.78%
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        19
  versicolor       1.00      0.92      0.96        13
   virginica       0.93      1.00      0.96        13

    accuracy                           0.98        45
   macro avg       0.98      0.97      0.97        45
weighted avg       0.98      0.98      0.98        45
```

```
print("knn")
print(f"predicted {y_pred_knn}")
print(f"actual {y_test.values}")
```

```
knn
predicted ['versicolor' 'setosa' 'virginica' 'versicolor' 'versicolor' 'setosa'
 'versicolor' 'virginica' 'versicolor' 'versicolor' 'virginica' 'setosa'
 'setosa' 'setosa' 'setosa' 'versicolor' 'virginica' 'versicolor'
 'versicolor' 'virginica' 'setosa' 'virginica' 'setosa' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'setosa' 'setosa'
 'setosa' 'setosa' 'versicolor' 'setosa' 'setosa' 'virginica' 'versicolor'
 'setosa' 'setosa' 'setosa' 'virginica' 'versicolor' 'versicolor' 'setosa'
 'setosa']
actual ['versicolor' 'setosa' 'virginica' 'versicolor' 'versicolor' 'setosa'
 'versicolor' 'virginica' 'versicolor' 'versicolor' 'virginica' 'setosa'
 'setosa' 'setosa' 'setosa' 'versicolor' 'virginica' 'versicolor'
 'versicolor' 'virginica' 'setosa' 'virginica' 'setosa' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'setosa' 'setosa'
 'setosa' 'setosa' 'versicolor' 'setosa' 'setosa' 'virginica' 'versicolor'
 'setosa' 'setosa' 'setosa' 'virginica' 'versicolor' 'versicolor' 'setosa'
 'setosa']
```

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

con_mat_knn = confusion_matrix(y_test,y_pred_knn)
plt.figure(figsize=(10,6))
sns.heatmap(con_mat_knn)
plt.xlabel('predicted')
plt.ylabel('actual')
plt.show()
```



```
from sklearn.datasets import load_wine
import pandas as pd

wine_data = load_wine()
df_wine = pd.DataFrame(data=wine_data.data, columns=wine_data.feature_names)
df_wine['target']=wine_data.target

print(df_wine.head())
```

```
   alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  total_phenols  \
0    14.23        1.71  2.43               15.6      127.0           2.80   
1    13.20        1.78  2.14               11.2      100.0           2.65   
2    13.16        2.36  2.67               18.6      101.0           2.80   
3    14.37        1.95  2.50               16.8      113.0           3.85   
4    13.24        2.59  2.87               21.0      118.0           2.80   

   flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hue  \
0        3.06                  0.28             2.29             5.64  1.04   
1        2.76                  0.26             1.28             4.38  1.05   
2        3.24                  0.30             2.81             5.68  1.03   
3        3.49                  0.24             2.18             7.80  0.86   
4        2.69                  0.39             1.82             4.32  1.04   

   od280/od315_of_diluted_wines  proline  target  
0                          3.92   1065.0       0  
1                          3.40   1050.0       0  
2                          3.17   1185.0       0  
3                          3.45   1480.0       0  
4                          2.93    735.0       0  
```

```
df_wine.head()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_inten
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	

Next steps: Generate code with df_wine | View recommended plots | New interactive sheet

```
selected_features = ['alcohol','malic_acid','ash','flavanoids']
df_wine_selected = df_wine[selected_features + ['target']]
df_wine_selected.head()
```

	alcohol	malic_acid	ash	flavanoids	target
0	14.23	1.71	2.43	3.06	0
1	13.20	1.78	2.14	2.76	0
2	13.16	2.36	2.67	3.24	0
3	14.37	1.95	2.50	3.49	0
4	13.24	2.59	2.87	2.69	0

Next steps: Generate code with df_wine_selected | View recommended plots | New interactive sheet

```
df_wine_selected.shape
```

```
(178, 5)
```

```
df_wine_selected.isnull().sum()
```

	0
alcohol	0
malic_acid	0
ash	0
flavanoids	0
target	0

```
dtype: int64
```

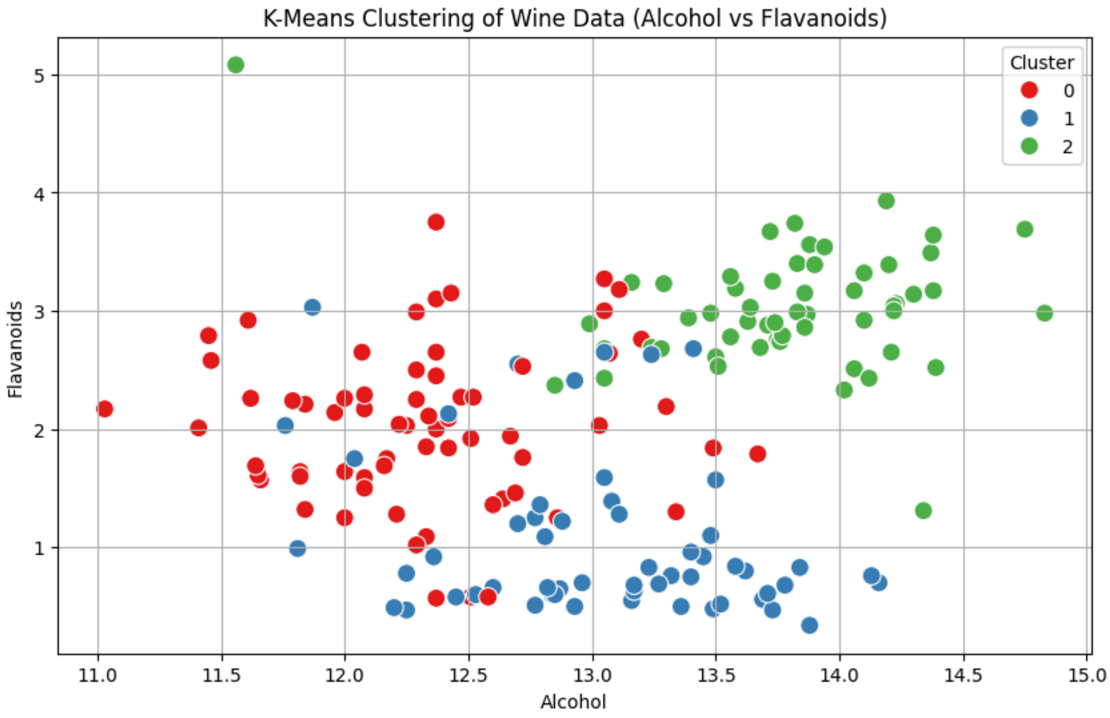
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler_features = scaler.fit_transform(df_wine[selected_features])


kmeans = KMeans(n_clusters=3,random_state=42)
kmeans.fit(scaler_features)

df_wine['cluster'] = kmeans.labels_

df_wine[['target','cluster']].head()
```

	target	cluster
0	0	2
1	0	0
2	0	2
3	0	2
4	0	2

```
# Visualize multiple features with pairplot, colored by cluster labels
sns.pairplot(df_wine, vars=['alcohol', 'malic_acid', 'ash', 'flavanoids'], hue='cluster', palette='Set1', markers=["o", "s", "D"])

plt.suptitle('K-Means Clustering on Wine Data', y=1.02)
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need t
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need t
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need t
```

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

# Scatter plot of 'alcohol' vs 'flavanoids', colored by cluster labels
plt.figure(figsize=(10, 6))
sns.scatterplot(x=df_wine['alcohol'], y=df_wine['flavanoids'], hue=df_wine['cluster'], palette='Set1', s=100)

plt.title('K-Means Clustering of Wine Data (Alcohol vs Flavanoids)')
plt.xlabel('Alcohol')
plt.ylabel('Flavanoids')
plt.legend(title='Cluster', loc='upper right')
plt.grid(True)
plt.show()
```



```
import numpy as np
from matplotlib import pyplot as plt
import tensorflow as tf
```

Start coding or generate with AI.