```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC,LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
```

```
df = pd.read_csv("/content/iris.csv")
```

```
df
```

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa
...	...	...	...	...	...
144	6.7	3.0	5.2	2.3	Iris-virginica
145	6.3	2.5	5.0	1.9	Iris-virginica
146	6.5	3.0	5.2	2.0	Iris-virginica
147	6.2	3.4	5.4	2.3	Iris-virginica
148	5.9	3.0	5.1	1.8	Iris-virginica

149 rows × 5 columns

```
df.head(15)
```

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa
5	4.6	3.4	1.4	0.3	Iris-setosa
6	5.0	3.4	1.5	0.2	Iris-setosa
7	4.4	2.9	1.4	0.2	Iris-setosa
8	4.9	3.1	1.5	0.1	Iris-setosa
9	5.4	3.7	1.5	0.2	Iris-setosa
10	4.8	3.4	1.6	0.2	Iris-setosa
11	4.8	3.0	1.4	0.1	Iris-setosa
12	4.3	3.0	1.1	0.1	Iris-setosa
13	5.8	4.0	1.2	0.2	Iris-setosa
14	5.7	4.4	1.5	0.4	Iris-setosa

```
df.shape
```

```
(149, 5)
```

```
df.columns
```

```
Index(['5.1', '3.5', '1.4', '0.2', 'Iris-setosa'], dtype='object')
```

```
df.info()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149 entries, 0 to 148
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   5.1          149 non-null    float64
 1   3.5          149 non-null    float64
 2   1.4          149 non-null    float64
 3   0.2          149 non-null    float64
 4   Iris-setosa  149 non-null    object 
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
```

```
df.isnull().sum()
```

```
5.1            0
3.5            0
1.4            0
0.2            0
Iris-setosa    0
dtype: int64
```

```
df['Iris-setosa'].unique()
```

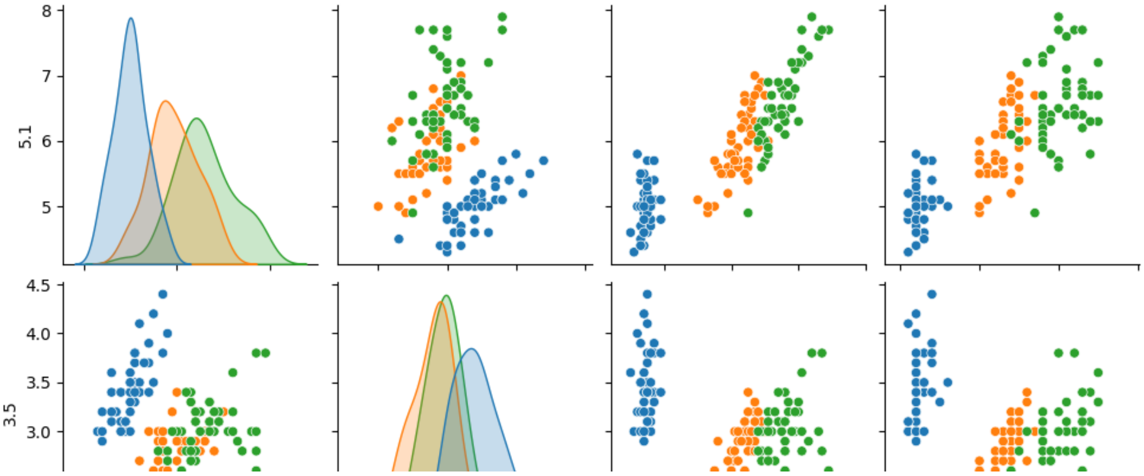
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

```
df.describe()
```

	5.1	3.5	1.4	0.2
count	149.000000	149.000000	149.000000	149.000000
mean	5.848322	3.051007	3.774497	1.205369
std	0.828594	0.433499	1.759651	0.761292
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.400000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

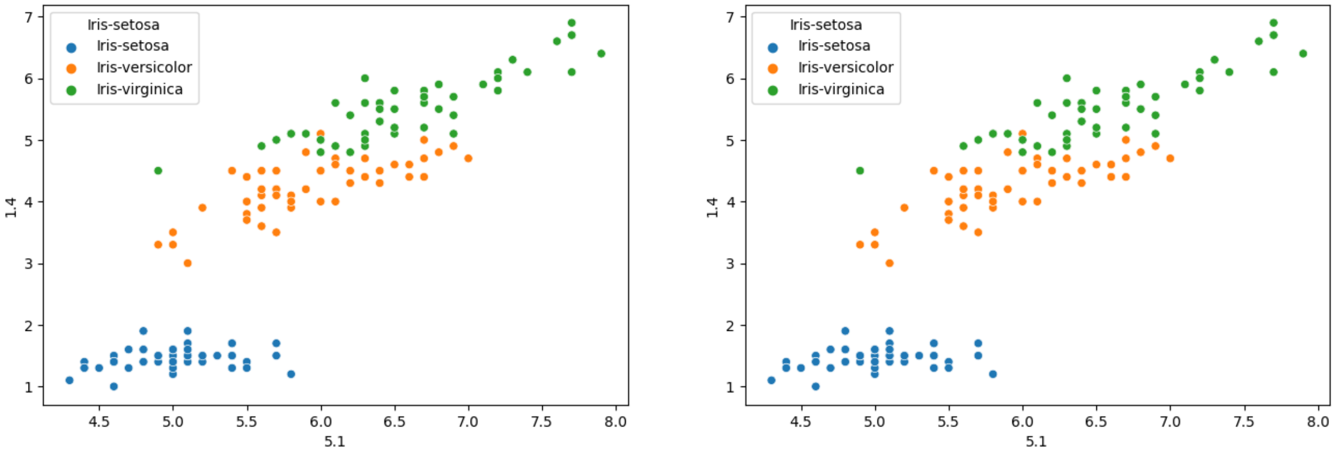
```
#Data VIsualisation
sns.pairplot(df,hue="Iris-setosa")
```

```
<seaborn.axisgrid.PairGrid at 0x7fba2580bd90>
```



```
fig,(ax1,ax2)=plt.subplots(ncols=2,figsize=(16,5))
sns.scatterplot(x='5.1',y='1.4',data=df,hue='Iris-setosa',ax=ax1)
sns.scatterplot(x='5.1',y='1.4',data=df,hue='Iris-setosa',ax=ax2)
```

```
<Axes: xlabel='5.1', ylabel='1.4'>
```



```
df.shape
```

```
(149, 5)
```

```
df.columns
df.info()
df.isnull().sum()
df['Iris-setosa'].unique()
```

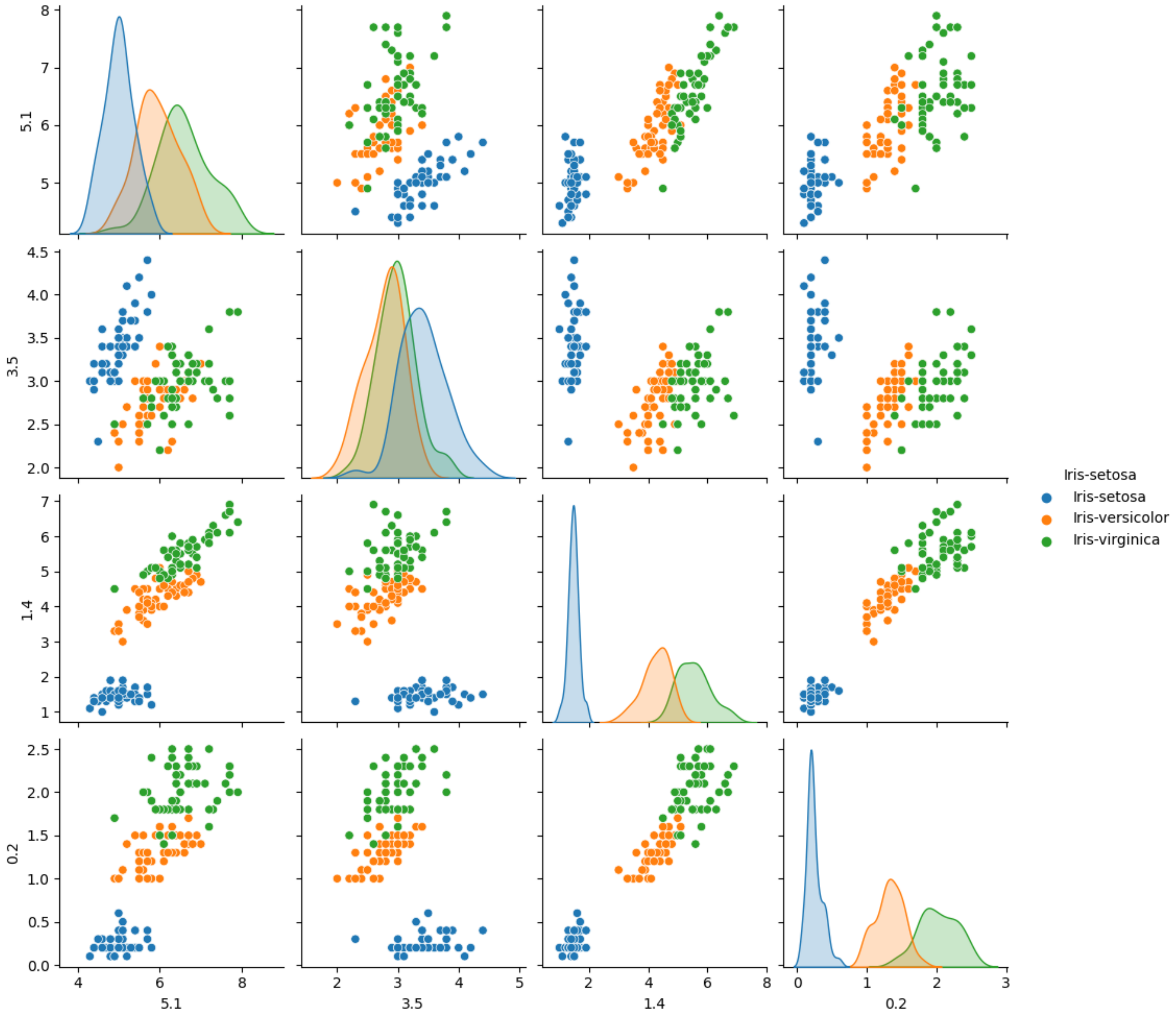
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149 entries, 0 to 148
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   5.1          149 non-null    float64
 1   3.5          149 non-null    float64
 2   1.4          149 non-null    float64
 3   0.2          149 non-null    float64
 4   Iris-setosa  149 non-null    object 
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

```
df.describe()
```

	5.1	3.5	1.4	0.2
count	149.000000	149.000000	149.000000	149.000000
mean	5.848322	3.051007	3.774497	1.205369
std	0.828594	0.433499	1.759651	0.761292
min	4.300000	2.000000	1.000000	0.100000

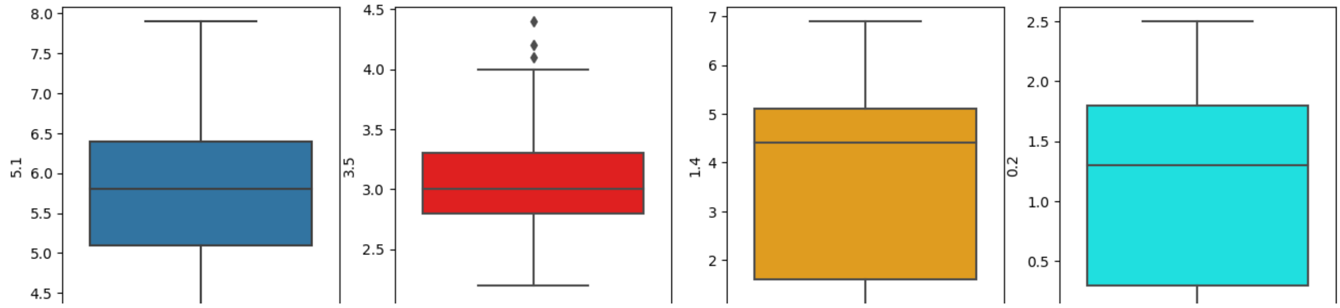
```
sns.pairplot(df,hue="Iris-setosa")
```

```
<seaborn.axisgrid.PairGrid at 0x7fba25808340>
```

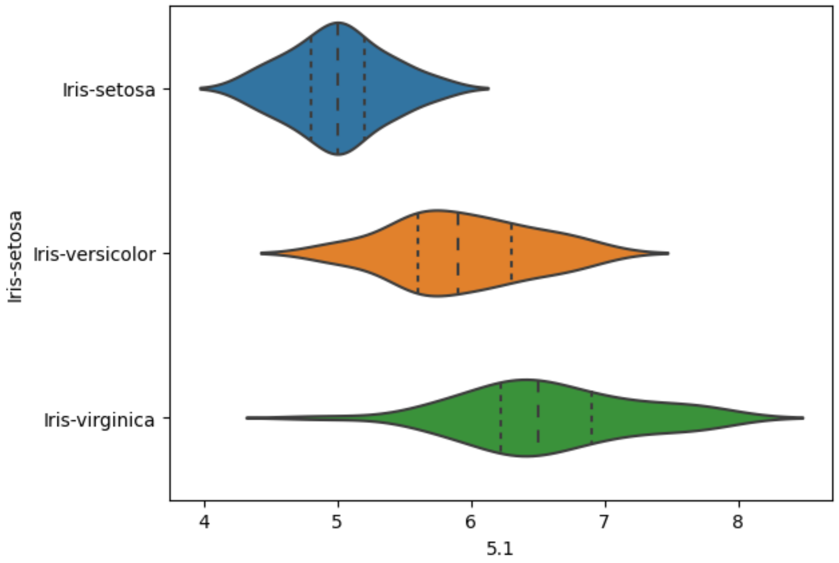


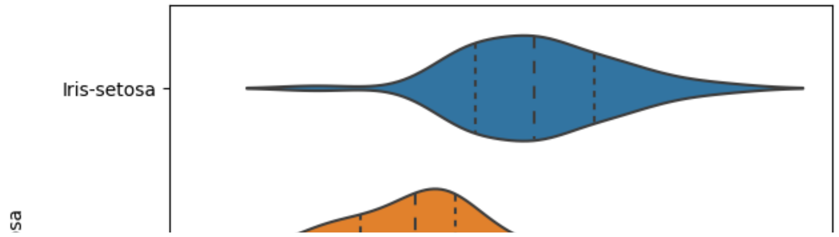
```
plt.figure(figsize=(16,4))
plt.subplot(1,4,1)
sns.boxplot(data=df,y='5.1')
plt.subplot(1,4,2)
sns.boxplot(data=df,y='3.5',color='red')
plt.subplot(1,4,3)
sns.boxplot(data=df,y='1.4',color='orange')
plt.subplot(1,4,4)
sns.boxplot(data=df,y='0.2',color='cyan')
```

```
<Axes: ylabel='0.2'>
```



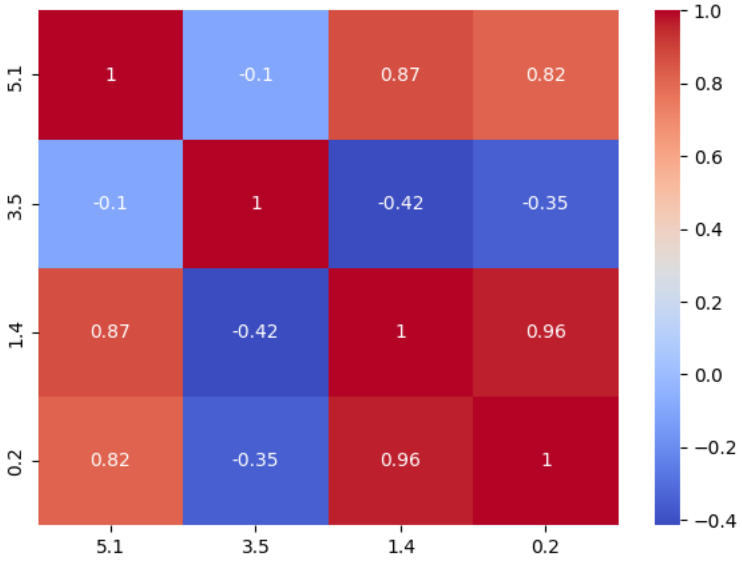
```
sns.violinplot(y='Iris-setosa', x='5.1', data=df, inner='quartile')
plt.show()
sns.violinplot(y='Iris-setosa', x='3.5', data=df, inner='quartile')
plt.show()
sns.violinplot(y='Iris-setosa', x='1.4', data=df, inner='quartile')
plt.show()
sns.violinplot(y='Iris-setosa', x='0.2', data=df, inner='quartile')
plt.show()
```





```
plt.figure(figsize=(7,5))
sns.heatmap(df.corr(), annot=True,cmap='coolwarm')
plt.show()
```

```
    <ipython-input-19-c06bbf75caf3>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ve
      sns.heatmap(df.corr(), annot=True,cmap='coolwarm')
```



```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()


df['Iris-setosa'] = le.fit_transform = (df['Iris-setosa'])
df.head(10)
```

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa

```
from sklearn.model_selection import train_test_split
X = df.drop(columns=['Iris-setosa'])
Y = df['Iris-setosa']
x_train , x_test , y_train , y_test = train_test_split(X , Y , test_size = 0.3)
```

```
# Initialize a Logistic Regression
lg= LogisticRegression(max_iter=1000)
lg.fit(x_train,y_train)
```

```
▾        LogisticRegression
LogisticRegression(max_iter=1000)
```

```
# Predict on the test set and calculate accuracy
from sklearn.metrics import accuracy_score
y_pred=lg.predict(x_test)
score=accuracy_score(y_test,y_pred)
```

```
def report(model):
    preds=model.predict(x_test)
    print(classification_report(preds,y_test))
    plot_confusion_matrix(model,x_test,y_test)
```

```
print('Logistic Regression')
print(f'Accuracy: {round(score*100,2)}%')
```

```
Logistic Regression
Accuracy: 95.56%
```

```
# Initialize a Linear SVC
rbf_sv= SVC()
rbf_sv.fit(x_train,y_train)
L_svc=LinearSVC()
```

```
L_svc.fit(x_train,y_train)
# Predict on the test set and calculate accuracy
y_pred=L_svc.predict(x_test)
score=accuracy_score(y_test,y_pred)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244: ConvergenceWarning: Liblinear failed to converge, increase the n
  warnings.warn(
```

```
print('Linear SVC')
print(f'Accuracy: {round(score*100,2)}%')
```

```
Linear SVC
Accuracy: 97.78%
```

```
DTC = DecisionTreeClassifier()
DTC=DTC.fit(x_train,y_train)
# Predict on the test set and calculate accuracy
y_pred=DTC.predict(x_test)
score=accuracy_score(y_test,y_pred)


print('Decision Tree Classifier')
print(f'Accuracy: {round(score*100,2)}%')
```

```
Decision Tree Classifier
Accuracy: 95.56%
```

```
NB= MultinomialNB()
NB.fit(x_train,y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

```
# Predict on the test set and calculate accuracy
y_pred=NB.predict(x_test)
score=accuracy_score(y_test,y_pred)
```

```
import pandas as pd
from sklearn.metrics import classification_report
print('NB')

print(f'Accuracy: {round(score*100,2)}%')
```

```
NB
Accuracy: 95.56%
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC,LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
```

```
import pandas as pd
from sklearn.metrics import classification_report
print('NB')

print(f'Accuracy: {round(score*100,2)}%')
```

```
NB
Accuracy: 95.56%
```

```
KNN=KNeighborsClassifier(n_neighbors=6)
KNN.fit(x_train, y_train)
```

```
▾       KNeighborsClassifier
KNeighborsClassifier(n_neighbors=6)
```

```
# Predict on the test set and calculate accuracy
y_pred=KNN.predict(x_test)
score=accuracy_score(y_test,y_pred)
```

```
!pip install matplotlib-venn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: matplotlib-venn in /usr/local/lib/python3.10/dist-packages (0.11.9)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from matplotlib-venn) (1.22.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from matplotlib-venn) (1.10.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from matplotlib-venn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->matplotlib-venn) (1.0.
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->matplotlib-venn) (3.0.
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->matplotlib-venn) (0.11.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->matplotlib-venn) (23.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->matplotlib-venn) (8.4.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->matplotlib-venn) (
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->matplotlib-venn) (1.4
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->matplotlib-venn) (4.3
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib->matplotl
```

```
# https://pypi.python.org/pypi/pydot
!apt-get -qq install -y graphviz && pip install pydot
import pydot
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pydot in /usr/local/lib/python3.10/dist-packages (1.4.2)
Requirement already satisfied: pyparsing>=2.1.4 in /usr/local/lib/python3.10/dist-packages (from pydot) (3.0.9)
```

```
print('KNN')
print(f'Accuracy: {round(score*100,2)}%')
```

```
KNN
Accuracy: 95.56%
```

✓ 0s completed at 9:38 PM ● ×