Importing necessary packages and reading the data

In [1]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt

df = pd.read_csv("iris.csv")
print(df)
```

```
     sepal.length  sepal.width  petal.length  petal.width    variety
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
..            ...          ...           ...          ...        ...
145           6.7          3.0           5.2          2.3  Virginica
146           6.3          2.5           5.0          1.9  Virginica
147           6.5          3.0           5.2          2.0  Virginica
148           6.2          3.4           5.4          2.3  Virginica
149           5.9          3.0           5.1          1.8  Virginica

[150 rows x 5 columns]
```

Extracting the feature columns and target column from the dataset

In [2]:
```
feature_columns = ['sepal.length', 'sepal.width', 'petal.length', 'petal.width']
X = df[feature_columns].values
y = df['variety'].values
```

Using LabelEncoder to encode the target column

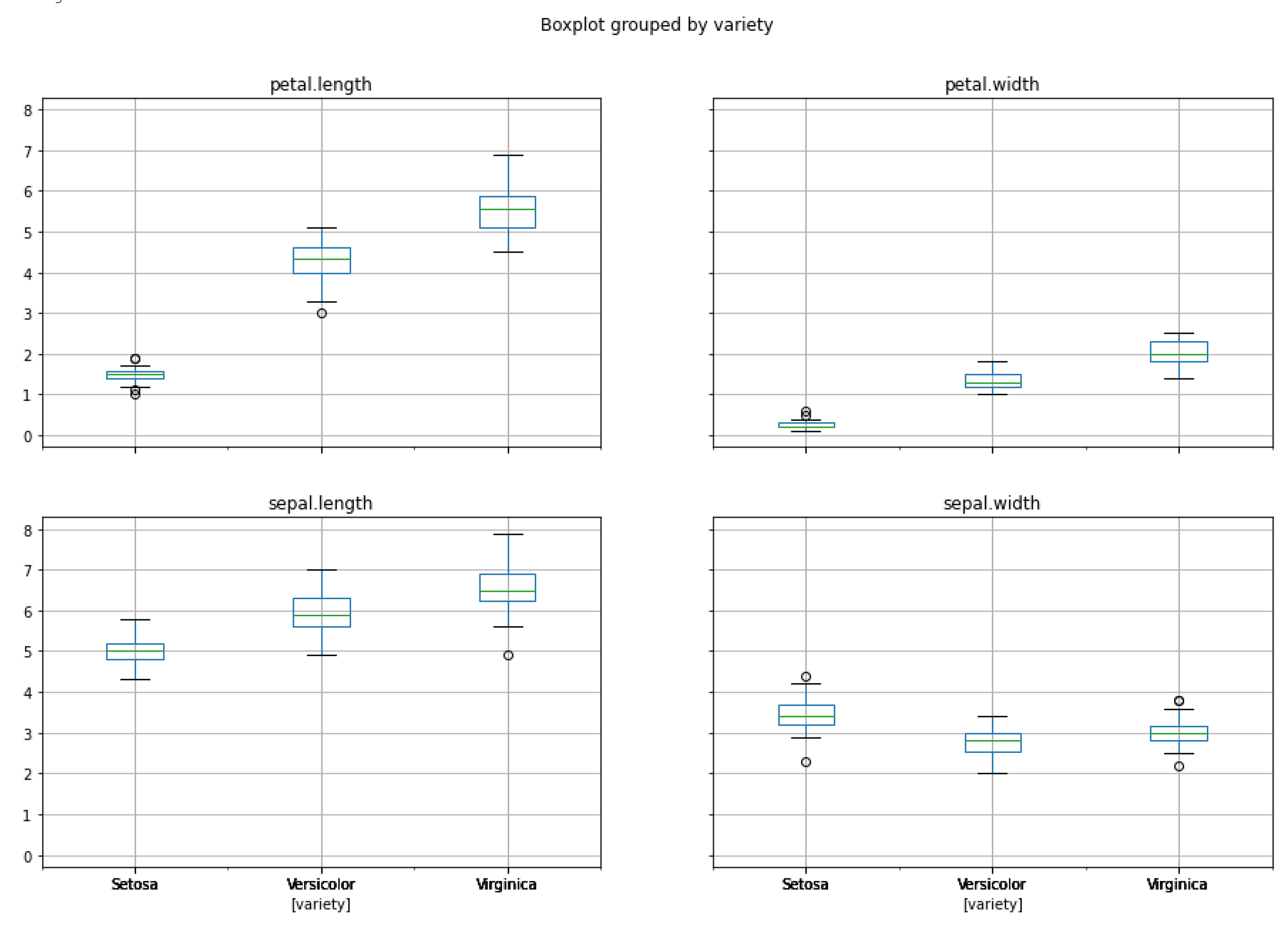
In [3]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
print(y)
```

```
[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]
```

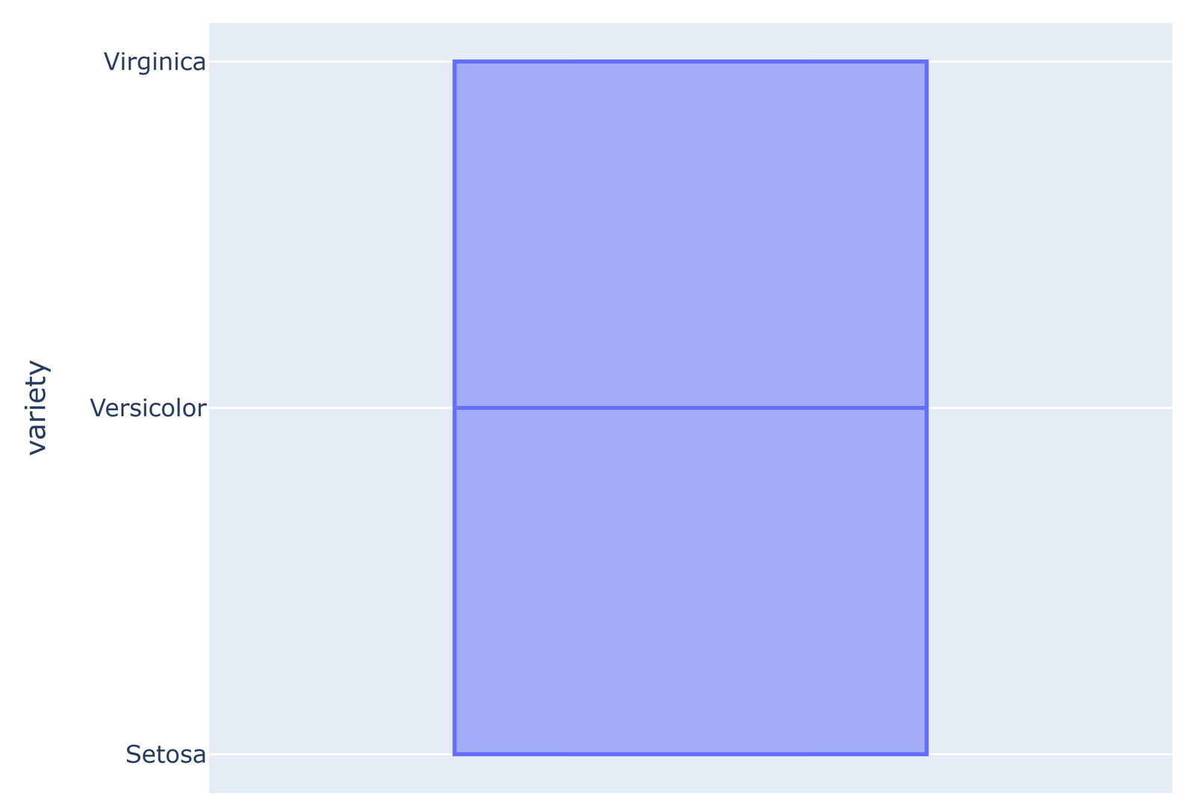
Plottings

In [4]:
```
plt.figure()
df.boxplot(by="variety", figsize=(15, 10))
plt.show()
```

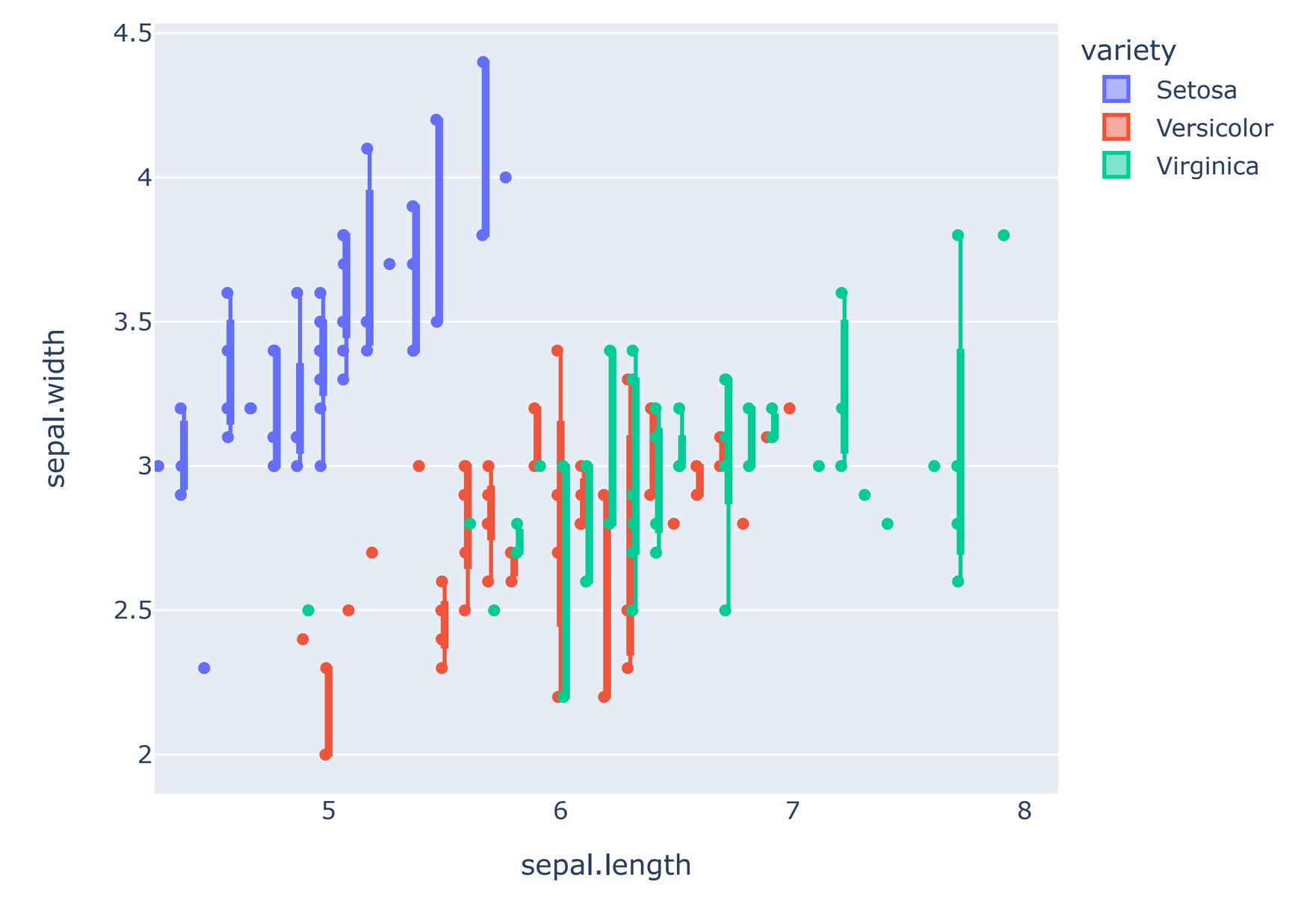
```
<Figure size 432x288 with 0 Axes>
```



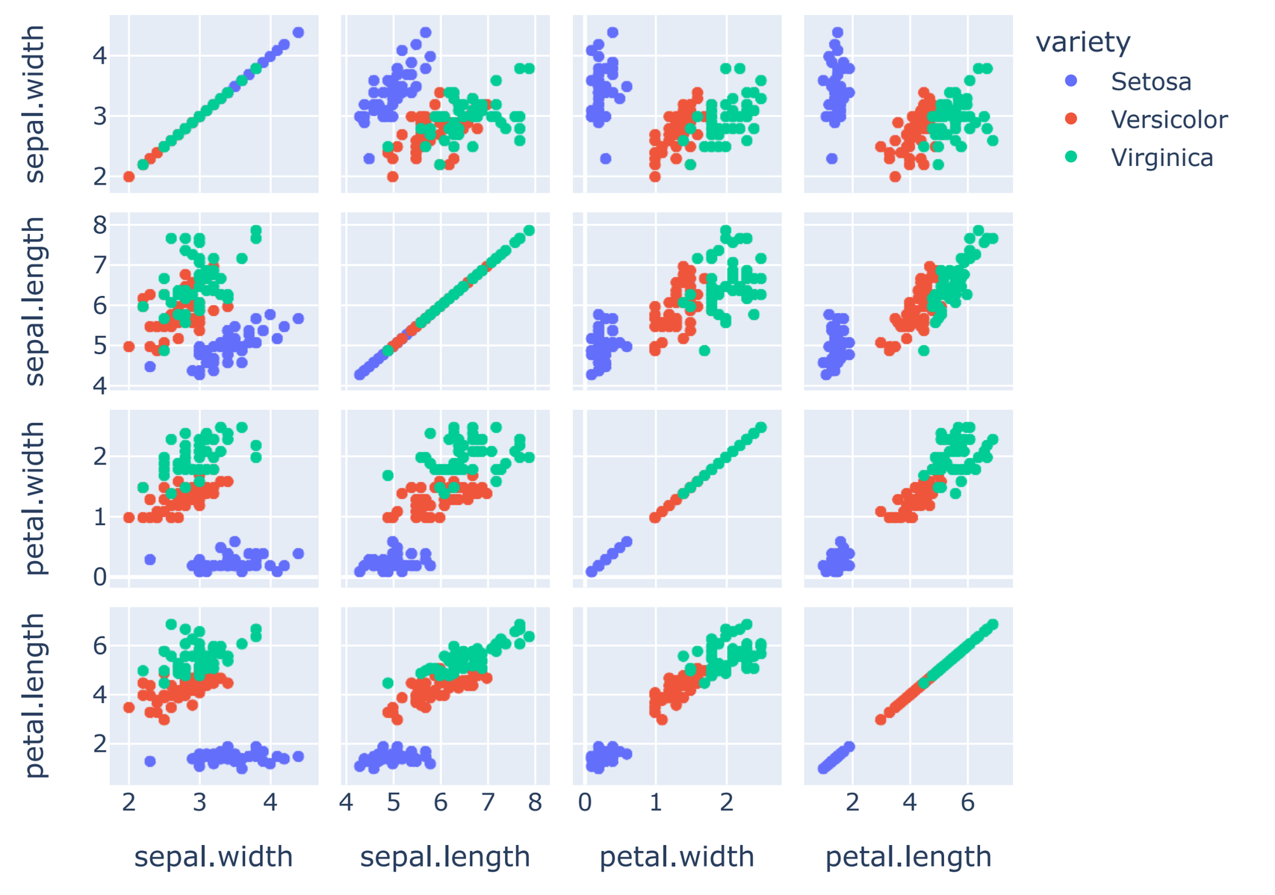
In [5]:
```
import plotly.express as px
fig = px.box(df, y="variety")
fig.show()
```



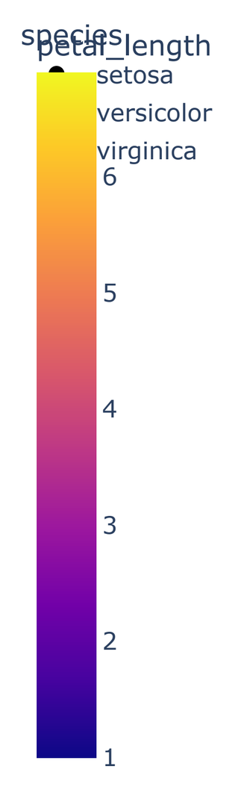
In [6]:
```
import plotly.express as px
fig = px.box(df, x="sepal.length", y="sepal.width",points="all", color="variety")
fig.show()
```



In [7]:
```
import plotly.express as px
fig = px.scatter_matrix(df, dimensions=["sepal.width", "sepal.length", "petal.width", "petal.length"], color="variety")
fig.show()
```



In [8]:
```
import plotly.express as px
df = px.data.iris()
fig = px.scatter_3d(df, x='sepal_length', y='sepal_width', z='petal_width',color='petal_length', symbol='species')
fig.show()
```



Training, Testing and Splitting

In [9]:
```
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 = 0)
```

Building the Model

In [10]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
```

Predictions

In [11]:
```
classifier = KNeighborsClassifier(n_neighbors=1)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
print(y_pred)
```

```
[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]
```

Evaluating the Model

In [12]:
```
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

```
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
```

In [13]:
```
accuracy = accuracy_score(y_test, y_pred)*100
print('Accuracy of the model:' + str(round(accuracy, 2)) + ' %.')
```

```
Accuracy of the model:100.0 %.
```

In [14]:
```
from sklearn.metrics import precision_score, recall_score, f1_score

prec = precision_score(y_test, y_pred,average='macro')
print("Precision:", prec)

# calculate recall
rec = recall_score(y_test, y_pred,average='macro')
print("Recall:", rec)

# calculate f1-score
f1 = f1_score(y_test, y_pred,average='macro')
print("F1-score:", f1)
```

```
Precision: 1.0
Recall: 1.0
F1-score: 1.0
```

In [15]:
```
from sklearn.neighbors import KNeighborsClassifier
import joblib

# train the model
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)

# save the model to disk
filename = 'trained_model.pkl'
joblib.dump(knn, filename)

# fit and transform the data
iris_species = ['setosa', 'versicolor', 'virginica']
le.fit(iris_species)

#save the label classes
filename_classes = "label_classes.pkl"
joblib.dump(le.classes_, filename_classes)
```

Out[15]:
```
['label_classes.pkl']
```

Predicting New Data

In [16]:
```
# load the trained model
model = joblib.load("trained_model.pkl")

# load the label encoder
le = LabelEncoder()
le.classes_ = joblib.load("label_classes.pkl")

# prepare the new data
new_data = [[5.8, 4.5, 6.4, 1.2]]


# get the predictions for new data
predictions = model.predict(new_data)

# convert the predictions back to the original iris species names
predictions = le.inverse_transform(predictions)

print(predictions)
```

```
['virginica']
```