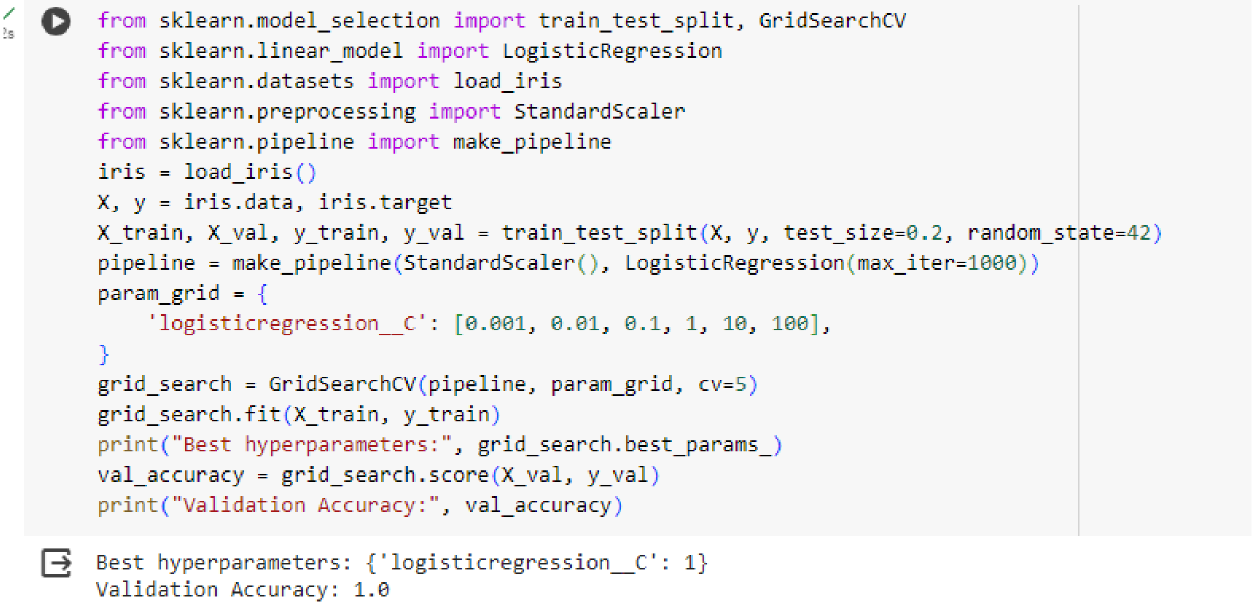
ASSIGNMENT – 8

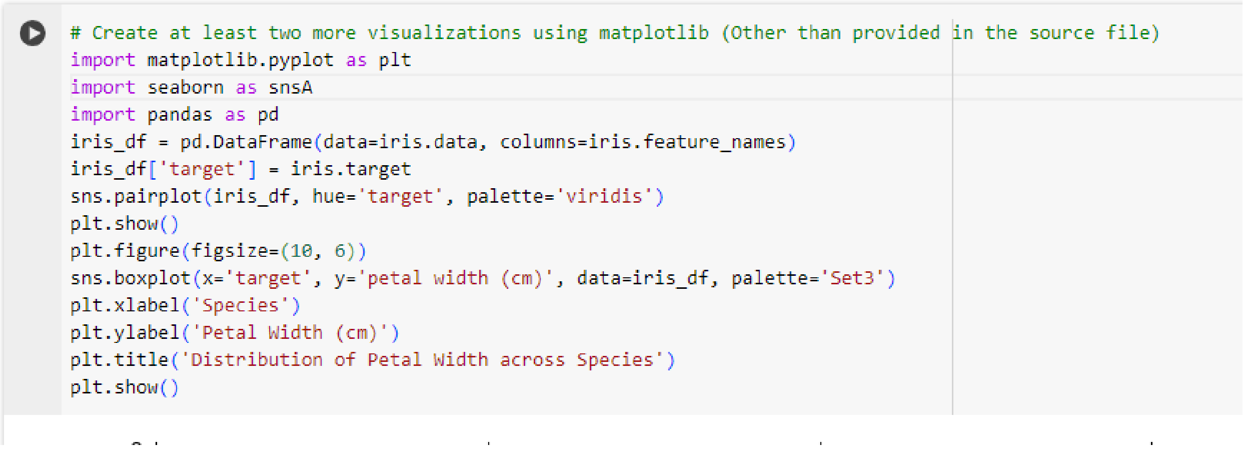
PREM KUMAR KAMMA – 700756204

GITHUB LINK:- https://github.com/PremKumarKamma/Assignment8

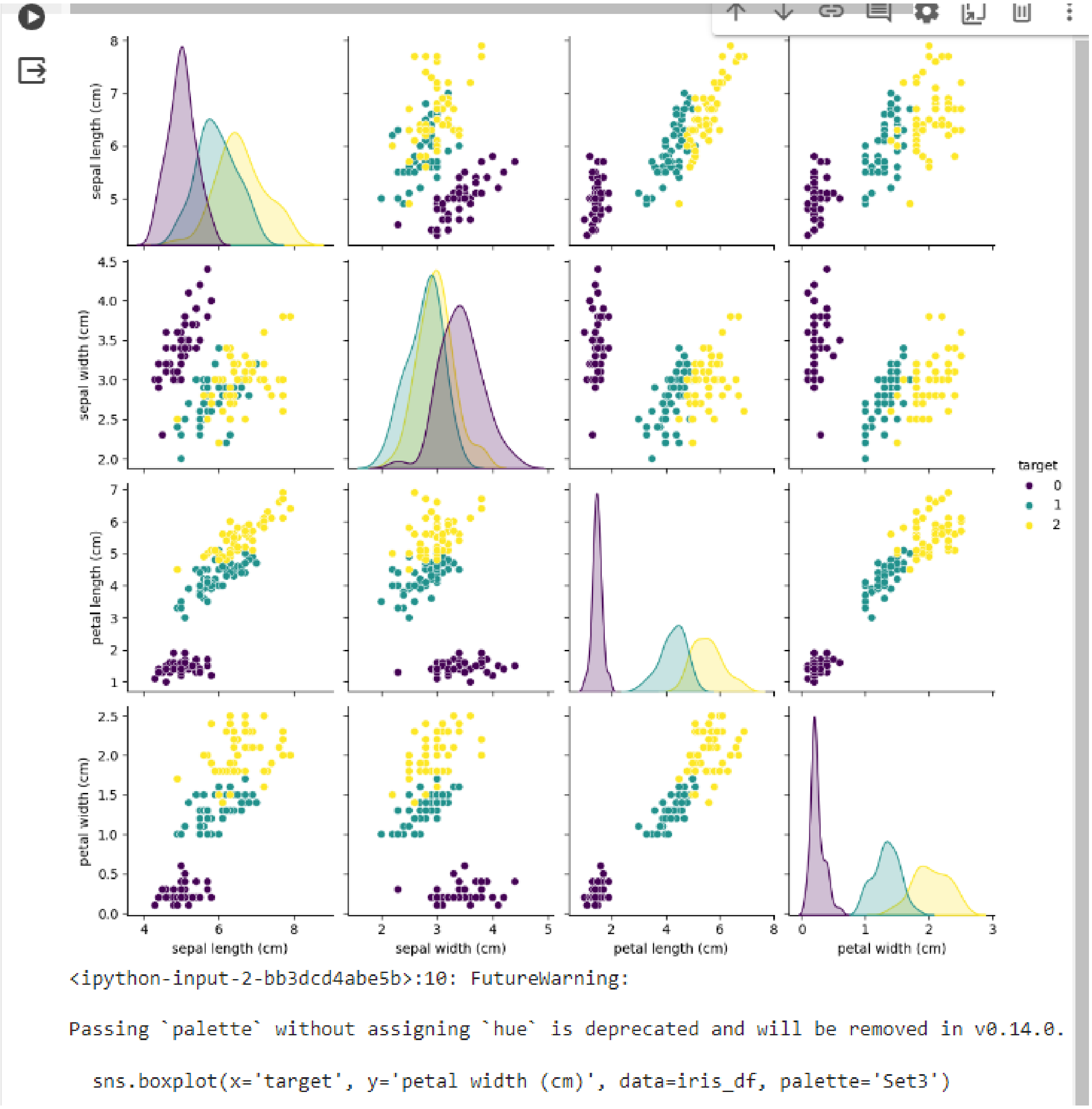


```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
iris = load_iris()
X, y = iris.data, iris.target
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
pipeline = make_pipeline(StandardScaler(), LogisticRegression(max_iter=1000))
param_grid = {
    'logisticregression__C': [0.001, 0.01, 0.1, 1, 10, 100],
}
grid_search = GridSearchCV(pipeline, param_grid, cv=5)
grid_search.fit(X_train, y_train)
print("Best hyperparameters:", grid_search.best_params_)
val_accuracy = grid_search.score(X_val, y_val)
print("Validation Accuracy:", val_accuracy)
```

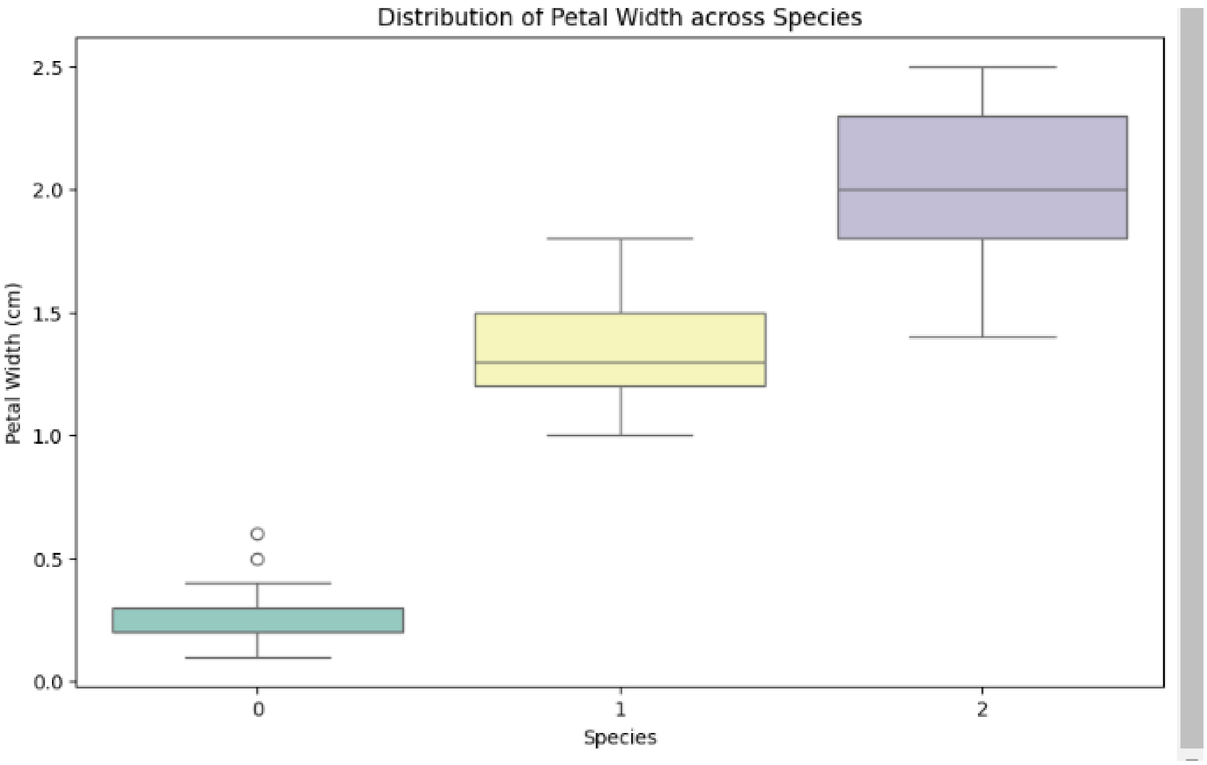
```
Best hyperparameters: {'logisticregression__C': 1}
Validation Accuracy: 1.0
```

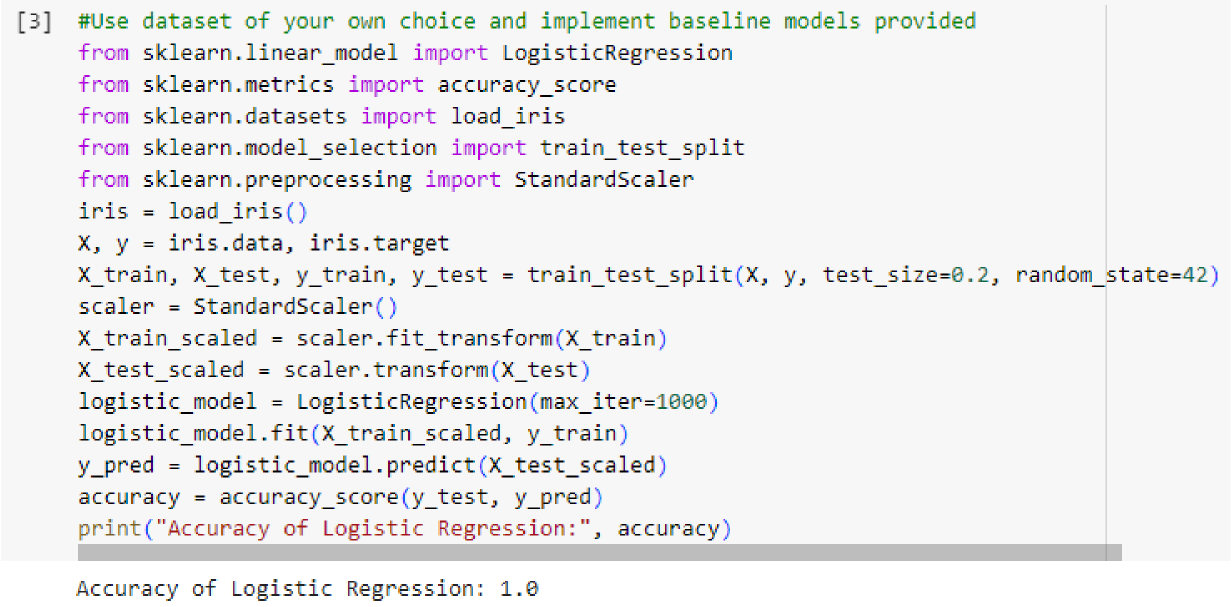


```
# Create at least two more visualizations using matplotlib (Other than provided in the source file)
import matplotlib.pyplot as plt
import seaborn as snsA
import pandas as pd
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['target'] = iris.target
sns.pairplot(iris_df, hue='target', palette='viridis')
plt.show()
plt.figure(figsize=(10, 6))
sns.boxplot(x='target', y='petal width (cm)', data=iris_df, palette='Set3')
plt.xlabel('Species')
plt.ylabel('Petal Width (cm)')
plt.title('Distribution of Petal Width across Species')
plt.show()
```



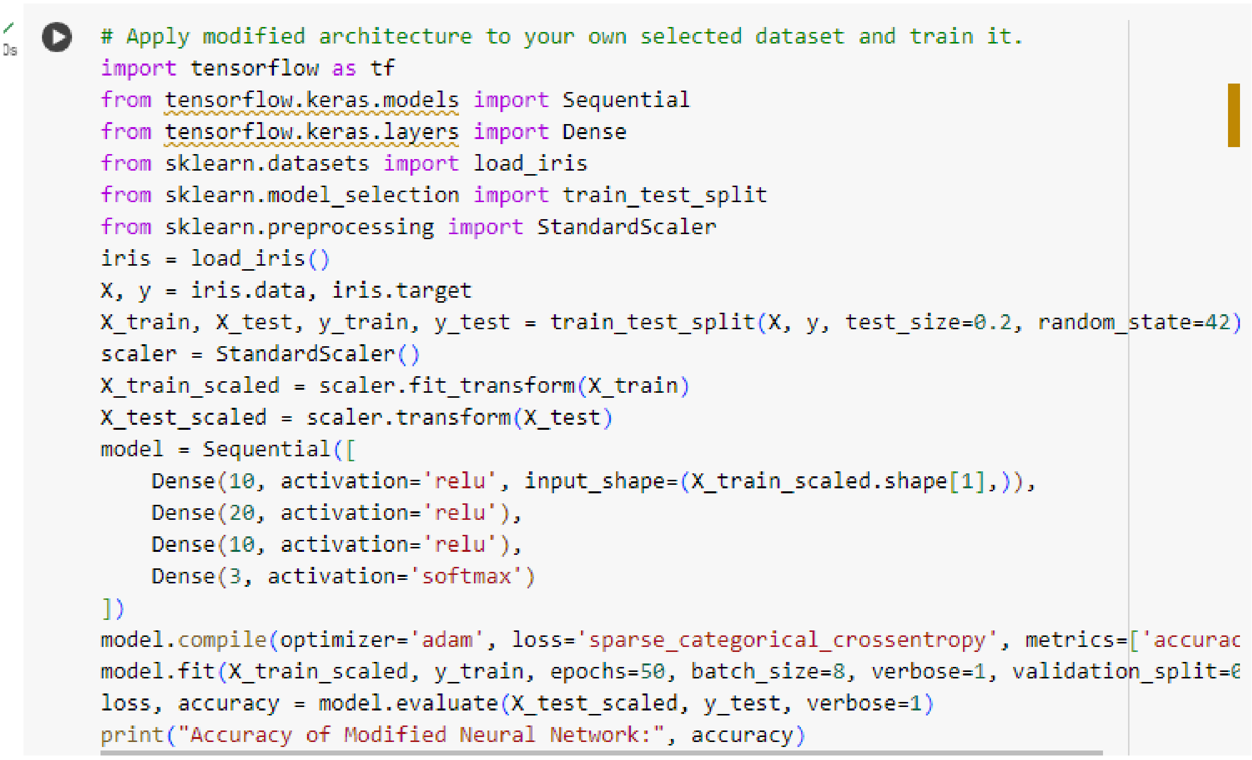
```
<ipython-input-2-bb3dcd4abe5b>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

  sns.boxplot(x='target', y='petal width (cm)', data=iris_df, palette='Set3')
```



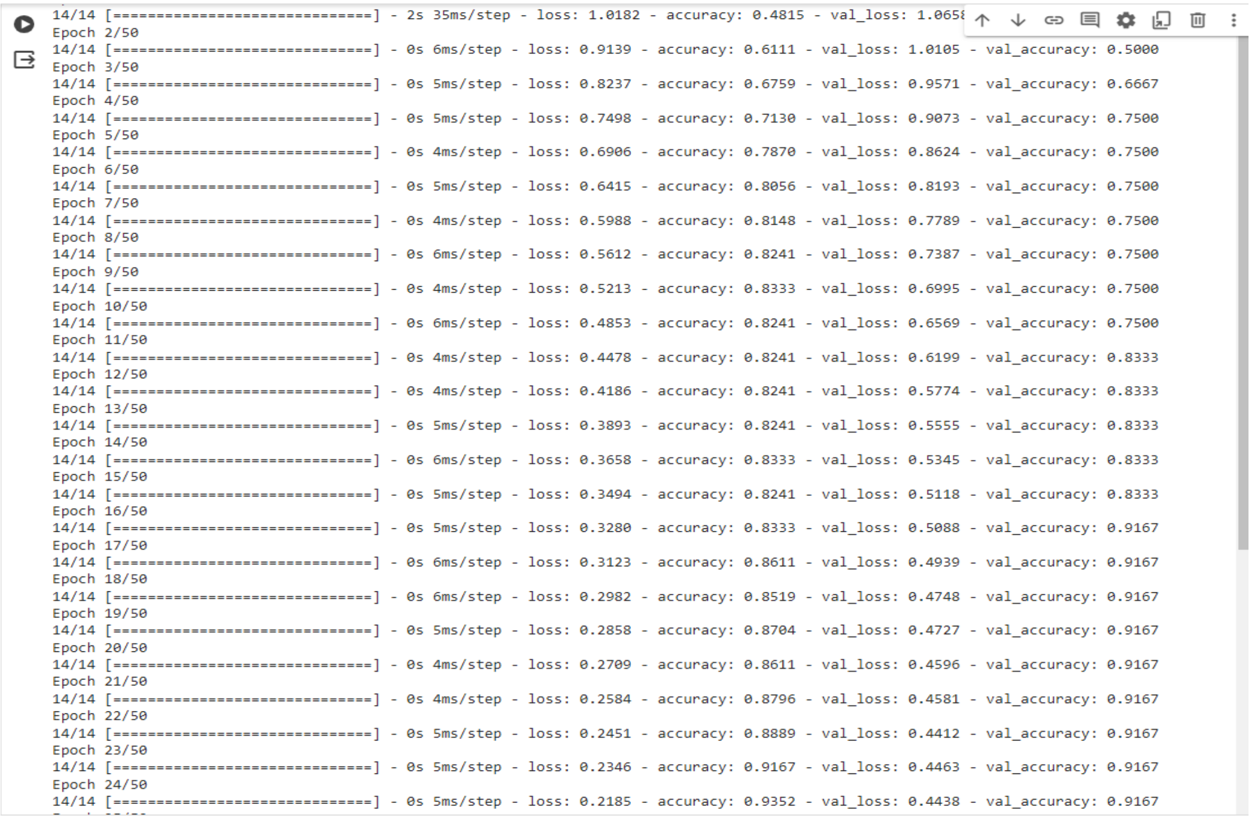


```
[3]  #Use dataset of your own choice and implement baseline models provided
     from sklearn.linear_model import LogisticRegression
     from sklearn.metrics import accuracy_score
     from sklearn.datasets import load_iris
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import StandardScaler
     iris = load_iris()
     X, y = iris.data, iris.target
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
     scaler = StandardScaler()
     X_train_scaled = scaler.fit_transform(X_train)
     X_test_scaled = scaler.transform(X_test)
     logistic_model = LogisticRegression(max_iter=1000)
     logistic_model.fit(X_train_scaled, y_train)
     y_pred = logistic_model.predict(X_test_scaled)
     accuracy = accuracy_score(y_test, y_pred)
     print("Accuracy of Logistic Regression:", accuracy)
```

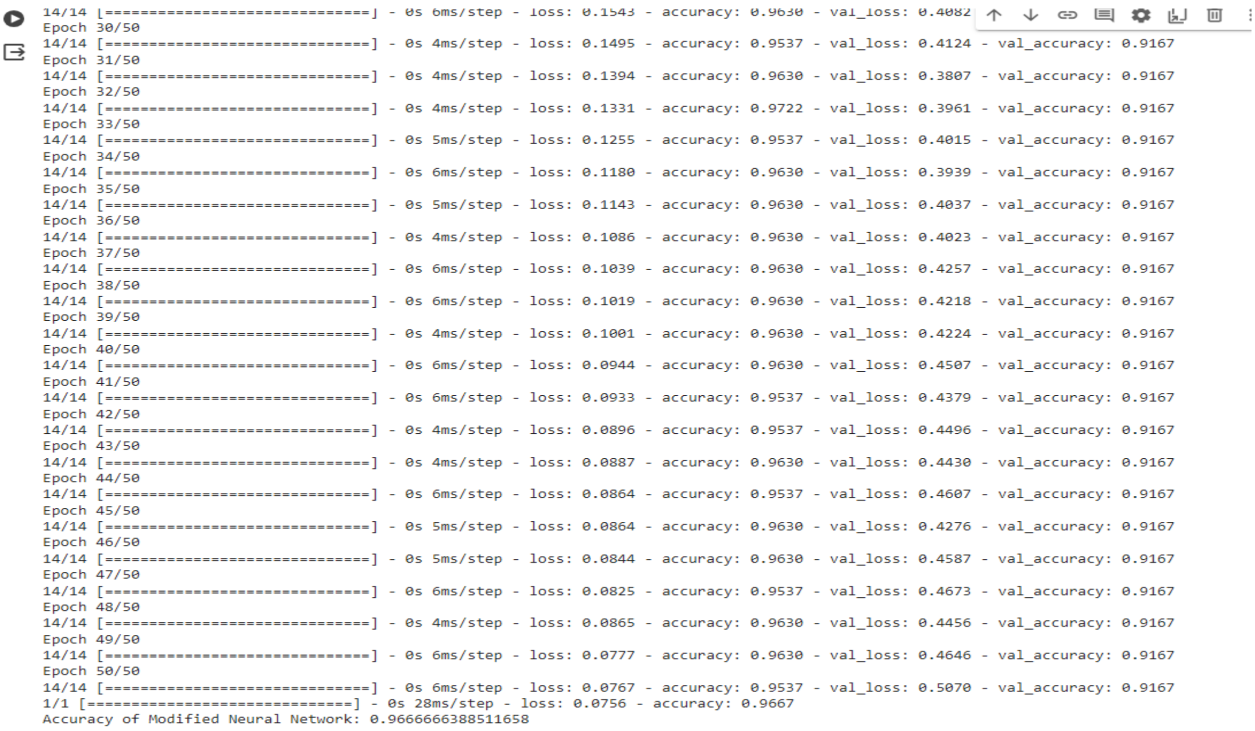
```
Accuracy of Logistic Regression: 1.0
```



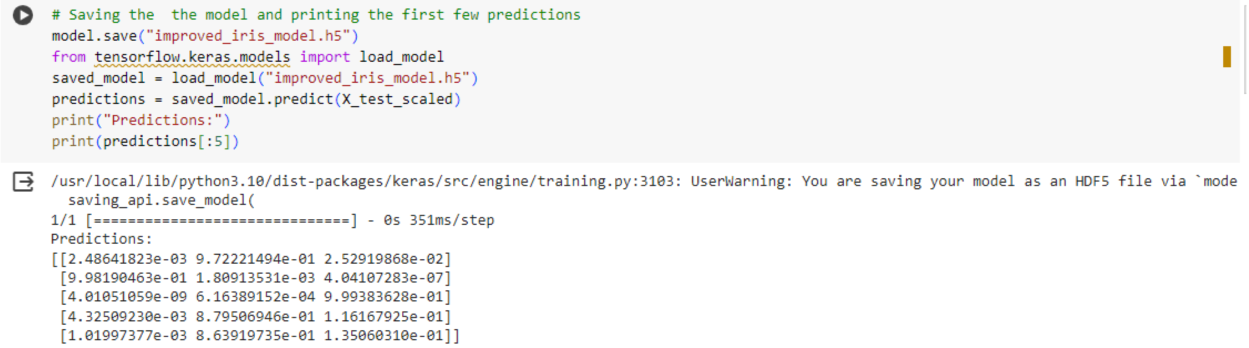
```
# Apply modified architecture to your own selected dataset and train it.
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
iris = load_iris()
X, y = iris.data, iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
model = Sequential([
    Dense(10, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    Dense(20, activation='relu'),
    Dense(10, activation='relu'),
    Dense(3, activation='softmax')
])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accurac
model.fit(X_train_scaled, y_train, epochs=50, batch_size=8, verbose=1, validation_split=0
loss, accuracy = model.evaluate(X_test_scaled, y_test, verbose=1)
print("Accuracy of Modified Neural Network:", accuracy)
```



```
14/14 [==============================] - 2s 35ms/step - loss: 1.0182 - accuracy: 0.4815 - val_loss: 1.0658
Epoch 2/50
14/14 [==============================] - 0s 6ms/step - loss: 0.9139 - accuracy: 0.6111 - val_loss: 1.0105 - val_accuracy: 0.5000
Epoch 3/50
14/14 [==============================] - 0s 5ms/step - loss: 0.8237 - accuracy: 0.6759 - val_loss: 0.9571 - val_accuracy: 0.6667
Epoch 4/50
14/14 [==============================] - 0s 5ms/step - loss: 0.7498 - accuracy: 0.7130 - val_loss: 0.9073 - val_accuracy: 0.7500
Epoch 5/50
14/14 [==============================] - 0s 4ms/step - loss: 0.6906 - accuracy: 0.7870 - val_loss: 0.8624 - val_accuracy: 0.7500
Epoch 6/50
14/14 [==============================] - 0s 5ms/step - loss: 0.6415 - accuracy: 0.8056 - val_loss: 0.8193 - val_accuracy: 0.7500
Epoch 7/50
14/14 [==============================] - 0s 4ms/step - loss: 0.5988 - accuracy: 0.8148 - val_loss: 0.7789 - val_accuracy: 0.7500
Epoch 8/50
14/14 [==============================] - 0s 6ms/step - loss: 0.5612 - accuracy: 0.8241 - val_loss: 0.7387 - val_accuracy: 0.7500
Epoch 9/50
14/14 [==============================] - 0s 4ms/step - loss: 0.5213 - accuracy: 0.8333 - val_loss: 0.6995 - val_accuracy: 0.7500
Epoch 10/50
14/14 [==============================] - 0s 6ms/step - loss: 0.4853 - accuracy: 0.8241 - val_loss: 0.6569 - val_accuracy: 0.7500
Epoch 11/50
14/14 [==============================] - 0s 4ms/step - loss: 0.4478 - accuracy: 0.8241 - val_loss: 0.6199 - val_accuracy: 0.8333
Epoch 12/50
14/14 [==============================] - 0s 4ms/step - loss: 0.4186 - accuracy: 0.8241 - val_loss: 0.5774 - val_accuracy: 0.8333
Epoch 13/50
14/14 [==============================] - 0s 5ms/step - loss: 0.3893 - accuracy: 0.8241 - val_loss: 0.5555 - val_accuracy: 0.8333
Epoch 14/50
14/14 [==============================] - 0s 6ms/step - loss: 0.3658 - accuracy: 0.8333 - val_loss: 0.5345 - val_accuracy: 0.8333
Epoch 15/50
14/14 [==============================] - 0s 5ms/step - loss: 0.3494 - accuracy: 0.8241 - val_loss: 0.5118 - val_accuracy: 0.8333
Epoch 16/50
14/14 [==============================] - 0s 5ms/step - loss: 0.3280 - accuracy: 0.8333 - val_loss: 0.5088 - val_accuracy: 0.9167
Epoch 17/50
14/14 [==============================] - 0s 6ms/step - loss: 0.3123 - accuracy: 0.8611 - val_loss: 0.4939 - val_accuracy: 0.9167
Epoch 18/50
14/14 [==============================] - 0s 6ms/step - loss: 0.2982 - accuracy: 0.8519 - val_loss: 0.4748 - val_accuracy: 0.9167
Epoch 19/50
14/14 [==============================] - 0s 5ms/step - loss: 0.2858 - accuracy: 0.8704 - val_loss: 0.4727 - val_accuracy: 0.9167
Epoch 20/50
14/14 [==============================] - 0s 4ms/step - loss: 0.2709 - accuracy: 0.8611 - val_loss: 0.4596 - val_accuracy: 0.9167
Epoch 21/50
14/14 [==============================] - 0s 4ms/step - loss: 0.2584 - accuracy: 0.8796 - val_loss: 0.4581 - val_accuracy: 0.9167
Epoch 22/50
14/14 [==============================] - 0s 5ms/step - loss: 0.2451 - accuracy: 0.8889 - val_loss: 0.4412 - val_accuracy: 0.9167
Epoch 23/50
14/14 [==============================] - 0s 5ms/step - loss: 0.2346 - accuracy: 0.9167 - val_loss: 0.4463 - val_accuracy: 0.9167
Epoch 24/50
14/14 [==============================] - 0s 5ms/step - loss: 0.2185 - accuracy: 0.9352 - val_loss: 0.4438 - val_accuracy: 0.9167
```

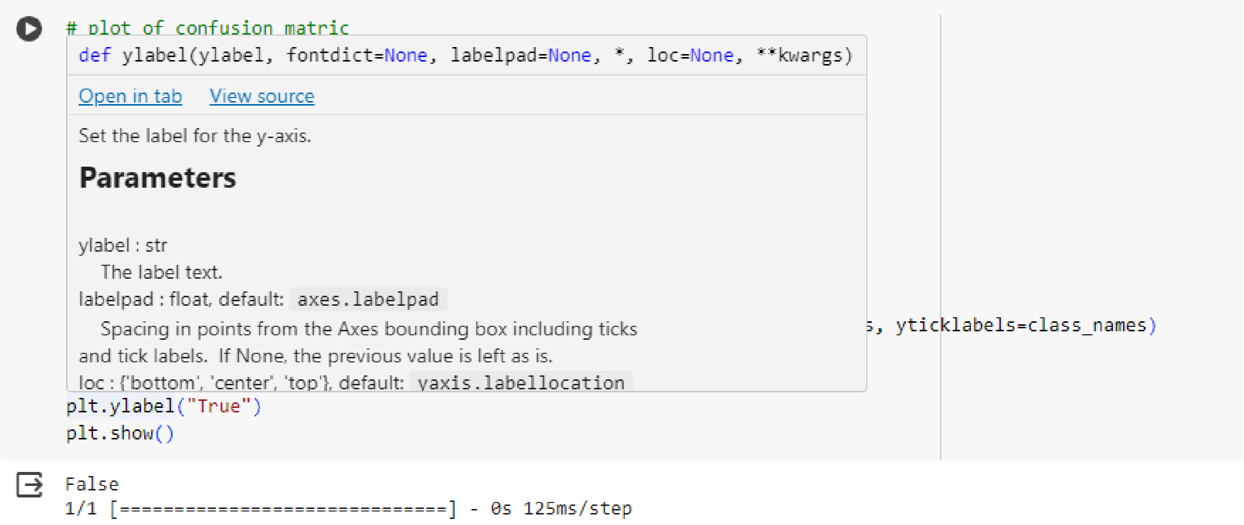


```
14/14 [==============================] - 0s 6ms/step - loss: 0.1543 - accuracy: 0.9630 - val_loss: 0.4082
Epoch 30/50
14/14 [==============================] - 0s 4ms/step - loss: 0.1495 - accuracy: 0.9537 - val_loss: 0.4124 - val_accuracy: 0.9167
Epoch 31/50
14/14 [==============================] - 0s 4ms/step - loss: 0.1394 - accuracy: 0.9630 - val_loss: 0.3807 - val_accuracy: 0.9167
Epoch 32/50
14/14 [==============================] - 0s 4ms/step - loss: 0.1331 - accuracy: 0.9722 - val_loss: 0.3961 - val_accuracy: 0.9167
Epoch 33/50
14/14 [==============================] - 0s 5ms/step - loss: 0.1255 - accuracy: 0.9537 - val_loss: 0.4015 - val_accuracy: 0.9167
Epoch 34/50
14/14 [==============================] - 0s 6ms/step - loss: 0.1180 - accuracy: 0.9630 - val_loss: 0.3939 - val_accuracy: 0.9167
Epoch 35/50
14/14 [==============================] - 0s 5ms/step - loss: 0.1143 - accuracy: 0.9630 - val_loss: 0.4037 - val_accuracy: 0.9167
Epoch 36/50
14/14 [==============================] - 0s 4ms/step - loss: 0.1086 - accuracy: 0.9630 - val_loss: 0.4023 - val_accuracy: 0.9167
Epoch 37/50
14/14 [==============================] - 0s 6ms/step - loss: 0.1039 - accuracy: 0.9630 - val_loss: 0.4257 - val_accuracy: 0.9167
Epoch 38/50
14/14 [==============================] - 0s 6ms/step - loss: 0.1019 - accuracy: 0.9630 - val_loss: 0.4218 - val_accuracy: 0.9167
Epoch 39/50
14/14 [==============================] - 0s 4ms/step - loss: 0.1001 - accuracy: 0.9630 - val_loss: 0.4224 - val_accuracy: 0.9167
Epoch 40/50
14/14 [==============================] - 0s 6ms/step - loss: 0.0944 - accuracy: 0.9630 - val_loss: 0.4507 - val_accuracy: 0.9167
Epoch 41/50
14/14 [==============================] - 0s 6ms/step - loss: 0.0933 - accuracy: 0.9537 - val_loss: 0.4379 - val_accuracy: 0.9167
Epoch 42/50
14/14 [==============================] - 0s 4ms/step - loss: 0.0896 - accuracy: 0.9537 - val_loss: 0.4496 - val_accuracy: 0.9167
Epoch 43/50
14/14 [==============================] - 0s 4ms/step - loss: 0.0887 - accuracy: 0.9630 - val_loss: 0.4430 - val_accuracy: 0.9167
Epoch 44/50
14/14 [==============================] - 0s 6ms/step - loss: 0.0864 - accuracy: 0.9537 - val_loss: 0.4607 - val_accuracy: 0.9167
Epoch 45/50
14/14 [==============================] - 0s 5ms/step - loss: 0.0864 - accuracy: 0.9630 - val_loss: 0.4276 - val_accuracy: 0.9167
Epoch 46/50
14/14 [==============================] - 0s 5ms/step - loss: 0.0844 - accuracy: 0.9630 - val_loss: 0.4587 - val_accuracy: 0.9167
Epoch 47/50
14/14 [==============================] - 0s 6ms/step - loss: 0.0825 - accuracy: 0.9537 - val_loss: 0.4673 - val_accuracy: 0.9167
Epoch 48/50
14/14 [==============================] - 0s 4ms/step - loss: 0.0865 - accuracy: 0.9630 - val_loss: 0.4456 - val_accuracy: 0.9167
Epoch 49/50
14/14 [==============================] - 0s 6ms/step - loss: 0.0777 - accuracy: 0.9630 - val_loss: 0.4646 - val_accuracy: 0.9167
Epoch 50/50
14/14 [==============================] - 0s 6ms/step - loss: 0.0767 - accuracy: 0.9537 - val_loss: 0.5070 - val_accuracy: 0.9167
1/1 [==============================] - 0s 28ms/step - loss: 0.0756 - accuracy: 0.9667
Accuracy of Modified Neural Network: 0.9666666388511658
```



```python
# Saving the  the model and printing the first few predictions
model.save("improved_iris_model.h5")
from tensorflow.keras.models import load_model
saved_model = load_model("improved_iris_model.h5")
predictions = saved_model.predict(X_test_scaled)
print("Predictions:")
print(predictions[:5])
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `mode
  saving_api.save_model(
1/1 [==============================] - 0s 351ms/step
Predictions:
[[2.48641823e-03 9.72221494e-01 2.52919868e-02]
 [9.98190463e-01 1.80913531e-03 4.04107283e-07]
 [4.01051059e-09 6.16389152e-04 9.99383628e-01]
 [4.32509230e-03 8.79506946e-01 1.16167925e-01]
 [1.01997377e-03 8.63919735e-01 1.35060310e-01]]
```



```python
# plot of confusion matric
```

```python
def ylabel(ylabel, fontdict=None, labelpad=None, *, loc=None, **kwargs)
```

Set the label for the y-axis.

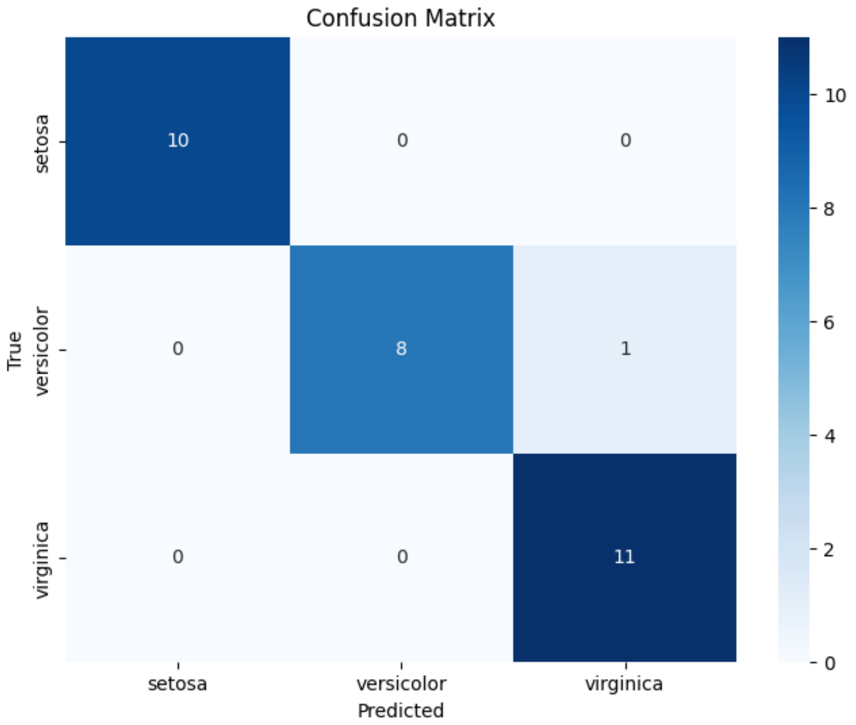
Parameters

ylabel : str
    The label text.
labelpad : float, default: `axes.labelpad`
    Spacing in points from the Axes bounding box including ticks and tick labels. If None, the previous value is left as is.
loc : {'bottom', 'center', 'top'}, default: `yaxis.labellocation`

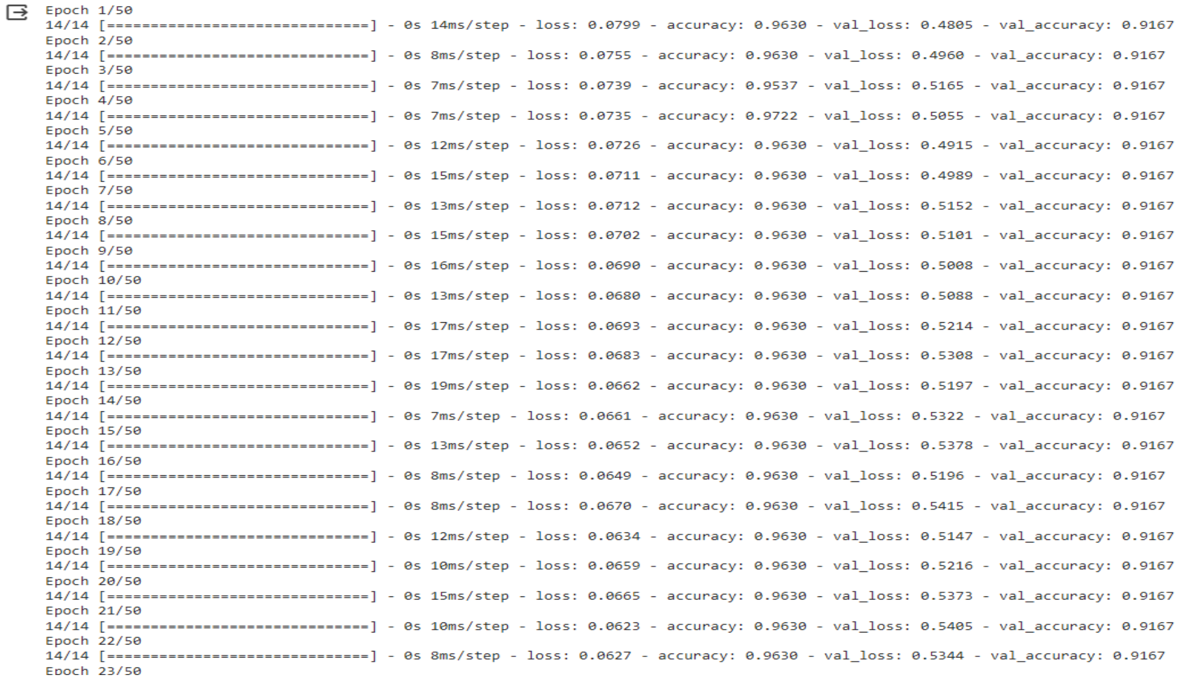
```python
s, yticklabels=class_names)
plt.ylabel("True")
plt.show()
```

```
False
1/1 [==============================] - 0s 125ms/step
```

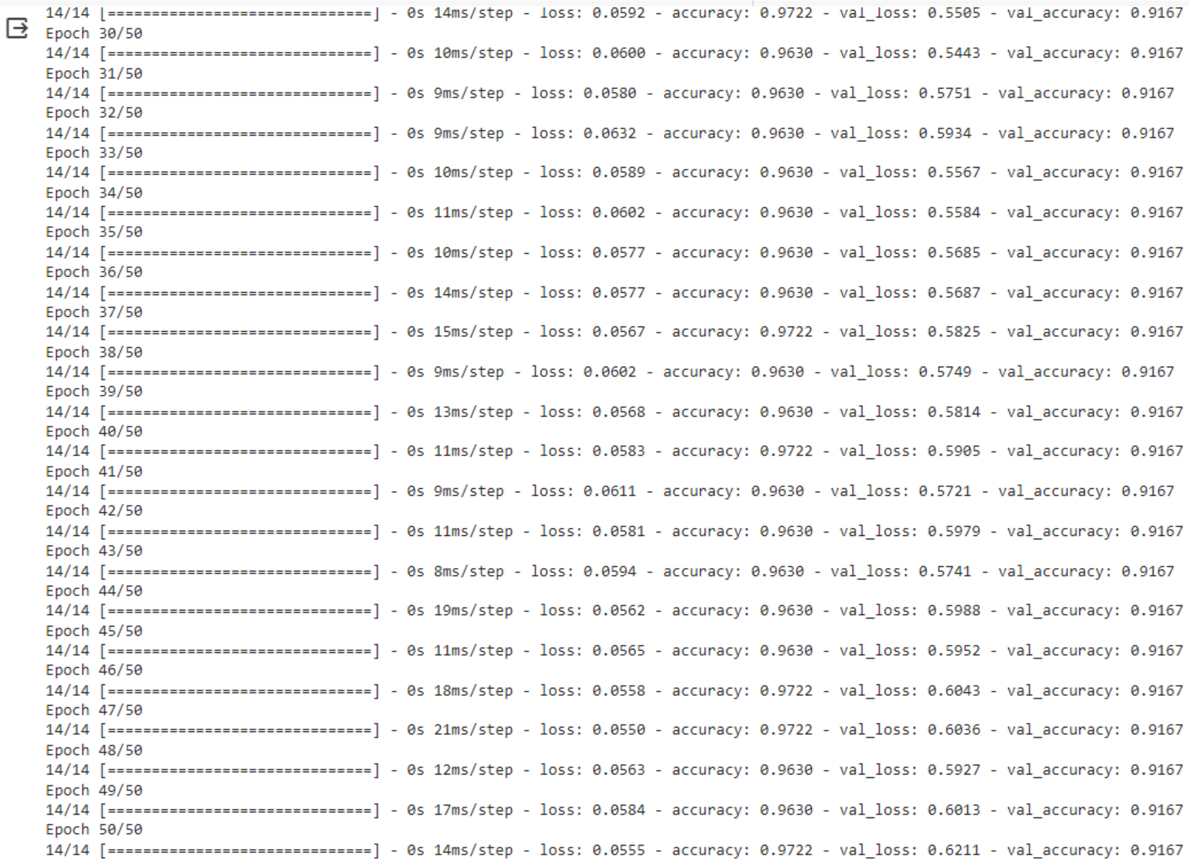
```
False
1/1 [==============================] - 0s 125ms/step
```



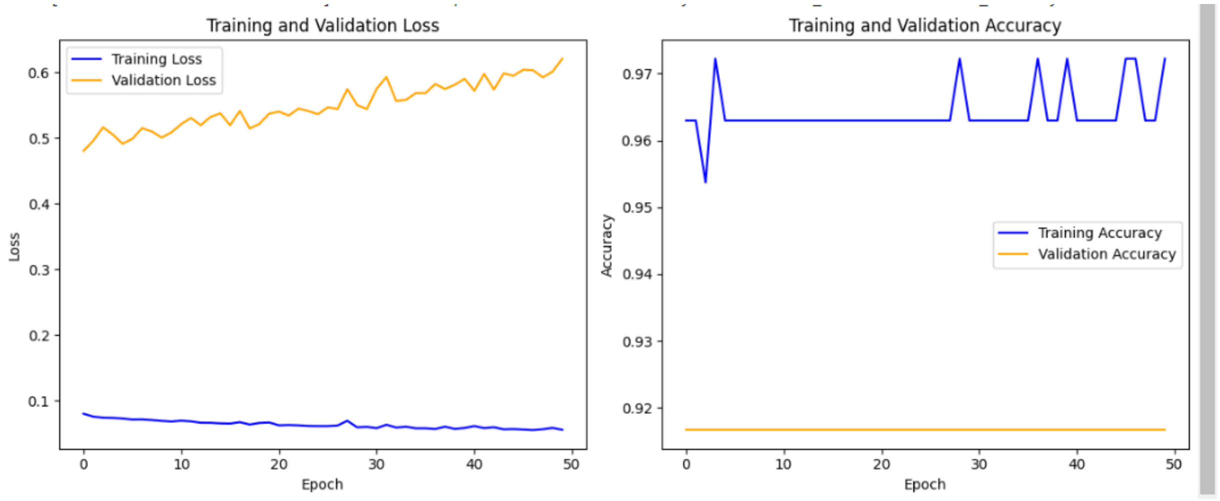
```python
# Training and testing Loss and accuracy plots in one plot using subplot command and history object
history = model.fit(X_train_scaled, y_train, epochs=50, batch_size=8, verbose=1, validation_split=0.1)
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss', color='blue')
plt.plot(history.history['val_loss'], label='Validation Loss', color='orange')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy', color='blue')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy', color='orange')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```

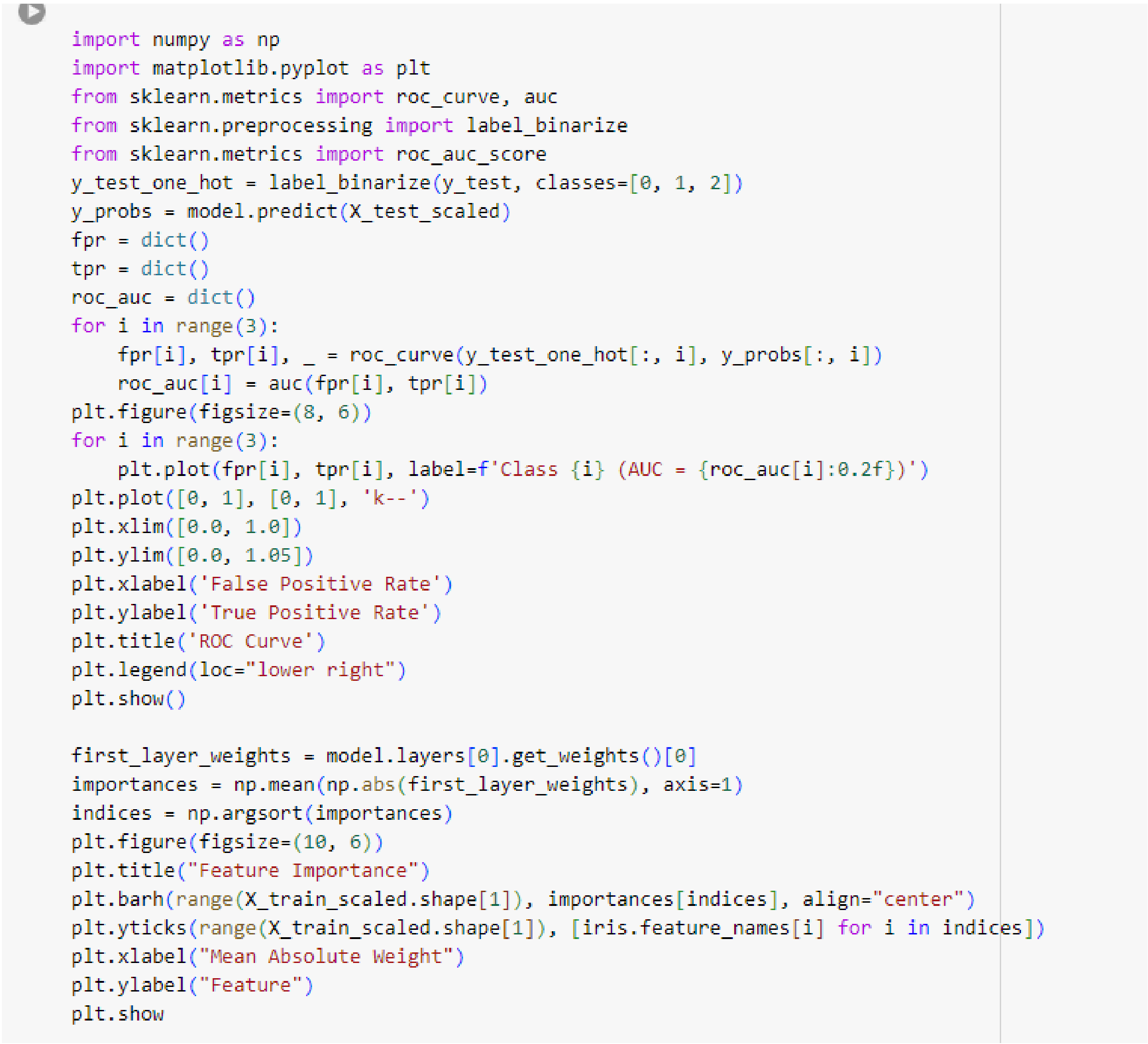


```
Epoch 1/50
14/14 [==============================] - 0s 14ms/step - loss: 0.0799 - accuracy: 0.9630 - val_loss: 0.4805 - val_accuracy: 0.9167
Epoch 2/50
14/14 [==============================] - 0s 8ms/step - loss: 0.0755 - accuracy: 0.9630 - val_loss: 0.4960 - val_accuracy: 0.9167
Epoch 3/50
14/14 [==============================] - 0s 7ms/step - loss: 0.0739 - accuracy: 0.9537 - val_loss: 0.5165 - val_accuracy: 0.9167
Epoch 4/50
14/14 [==============================] - 0s 7ms/step - loss: 0.0735 - accuracy: 0.9722 - val_loss: 0.5055 - val_accuracy: 0.9167
Epoch 5/50
14/14 [==============================] - 0s 12ms/step - loss: 0.0726 - accuracy: 0.9630 - val_loss: 0.4915 - val_accuracy: 0.9167
Epoch 6/50
14/14 [==============================] - 0s 15ms/step - loss: 0.0711 - accuracy: 0.9630 - val_loss: 0.4989 - val_accuracy: 0.9167
Epoch 7/50
14/14 [==============================] - 0s 13ms/step - loss: 0.0712 - accuracy: 0.9630 - val_loss: 0.5152 - val_accuracy: 0.9167
Epoch 8/50
14/14 [==============================] - 0s 15ms/step - loss: 0.0702 - accuracy: 0.9630 - val_loss: 0.5101 - val_accuracy: 0.9167
Epoch 9/50
14/14 [==============================] - 0s 16ms/step - loss: 0.0690 - accuracy: 0.9630 - val_loss: 0.5008 - val_accuracy: 0.9167
Epoch 10/50
14/14 [==============================] - 0s 13ms/step - loss: 0.0680 - accuracy: 0.9630 - val_loss: 0.5088 - val_accuracy: 0.9167
Epoch 11/50
14/14 [==============================] - 0s 17ms/step - loss: 0.0693 - accuracy: 0.9630 - val_loss: 0.5214 - val_accuracy: 0.9167
Epoch 12/50
14/14 [==============================] - 0s 17ms/step - loss: 0.0683 - accuracy: 0.9630 - val_loss: 0.5308 - val_accuracy: 0.9167
Epoch 13/50
14/14 [==============================] - 0s 19ms/step - loss: 0.0662 - accuracy: 0.9630 - val_loss: 0.5197 - val_accuracy: 0.9167
Epoch 14/50
14/14 [==============================] - 0s 7ms/step - loss: 0.0661 - accuracy: 0.9630 - val_loss: 0.5322 - val_accuracy: 0.9167
Epoch 15/50
14/14 [==============================] - 0s 13ms/step - loss: 0.0652 - accuracy: 0.9630 - val_loss: 0.5378 - val_accuracy: 0.9167
Epoch 16/50
14/14 [==============================] - 0s 8ms/step - loss: 0.0649 - accuracy: 0.9630 - val_loss: 0.5196 - val_accuracy: 0.9167
Epoch 17/50
14/14 [==============================] - 0s 8ms/step - loss: 0.0670 - accuracy: 0.9630 - val_loss: 0.5415 - val_accuracy: 0.9167
Epoch 18/50
14/14 [==============================] - 0s 12ms/step - loss: 0.0634 - accuracy: 0.9630 - val_loss: 0.5147 - val_accuracy: 0.9167
Epoch 19/50
14/14 [==============================] - 0s 10ms/step - loss: 0.0659 - accuracy: 0.9630 - val_loss: 0.5216 - val_accuracy: 0.9167
Epoch 20/50
14/14 [==============================] - 0s 15ms/step - loss: 0.0665 - accuracy: 0.9630 - val_loss: 0.5373 - val_accuracy: 0.9167
Epoch 21/50
14/14 [==============================] - 0s 10ms/step - loss: 0.0623 - accuracy: 0.9630 - val_loss: 0.5405 - val_accuracy: 0.9167
Epoch 22/50
14/14 [==============================] - 0s 8ms/step - loss: 0.0627 - accuracy: 0.9630 - val_loss: 0.5344 - val_accuracy: 0.9167
Epoch 23/50
```

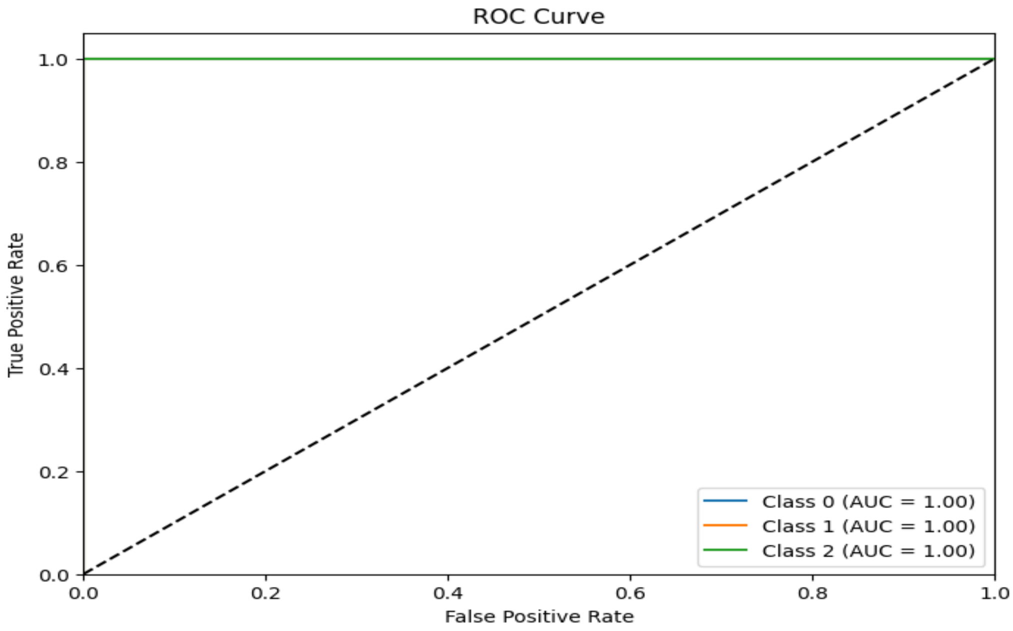


```
14/14 [==============================] - 0s 14ms/step - loss: 0.0592 - accuracy: 0.9722 - val_loss: 0.5505 - val_accuracy: 0.9167
Epoch 30/50
14/14 [==============================] - 0s 10ms/step - loss: 0.0600 - accuracy: 0.9630 - val_loss: 0.5443 - val_accuracy: 0.9167
Epoch 31/50
14/14 [==============================] - 0s 9ms/step - loss: 0.0580 - accuracy: 0.9630 - val_loss: 0.5751 - val_accuracy: 0.9167
Epoch 32/50
14/14 [==============================] - 0s 9ms/step - loss: 0.0632 - accuracy: 0.9630 - val_loss: 0.5934 - val_accuracy: 0.9167
Epoch 33/50
14/14 [==============================] - 0s 10ms/step - loss: 0.0589 - accuracy: 0.9630 - val_loss: 0.5567 - val_accuracy: 0.9167
Epoch 34/50
14/14 [==============================] - 0s 11ms/step - loss: 0.0602 - accuracy: 0.9630 - val_loss: 0.5584 - val_accuracy: 0.9167
Epoch 35/50
14/14 [==============================] - 0s 10ms/step - loss: 0.0577 - accuracy: 0.9630 - val_loss: 0.5685 - val_accuracy: 0.9167
Epoch 36/50
14/14 [==============================] - 0s 14ms/step - loss: 0.0577 - accuracy: 0.9630 - val_loss: 0.5687 - val_accuracy: 0.9167
Epoch 37/50
14/14 [==============================] - 0s 15ms/step - loss: 0.0567 - accuracy: 0.9722 - val_loss: 0.5825 - val_accuracy: 0.9167
Epoch 38/50
14/14 [==============================] - 0s 9ms/step - loss: 0.0602 - accuracy: 0.9630 - val_loss: 0.5749 - val_accuracy: 0.9167
Epoch 39/50
14/14 [==============================] - 0s 13ms/step - loss: 0.0568 - accuracy: 0.9630 - val_loss: 0.5814 - val_accuracy: 0.9167
Epoch 40/50
14/14 [==============================] - 0s 11ms/step - loss: 0.0583 - accuracy: 0.9722 - val_loss: 0.5905 - val_accuracy: 0.9167
Epoch 41/50
14/14 [==============================] - 0s 9ms/step - loss: 0.0611 - accuracy: 0.9630 - val_loss: 0.5721 - val_accuracy: 0.9167
Epoch 42/50
14/14 [==============================] - 0s 11ms/step - loss: 0.0581 - accuracy: 0.9630 - val_loss: 0.5979 - val_accuracy: 0.9167
Epoch 43/50
14/14 [==============================] - 0s 8ms/step - loss: 0.0594 - accuracy: 0.9630 - val_loss: 0.5741 - val_accuracy: 0.9167
Epoch 44/50
14/14 [==============================] - 0s 19ms/step - loss: 0.0562 - accuracy: 0.9630 - val_loss: 0.5988 - val_accuracy: 0.9167
Epoch 45/50
14/14 [==============================] - 0s 11ms/step - loss: 0.0565 - accuracy: 0.9630 - val_loss: 0.5952 - val_accuracy: 0.9167
Epoch 46/50
14/14 [==============================] - 0s 18ms/step - loss: 0.0558 - accuracy: 0.9722 - val_loss: 0.6043 - val_accuracy: 0.9167
Epoch 47/50
14/14 [==============================] - 0s 21ms/step - loss: 0.0550 - accuracy: 0.9722 - val_loss: 0.6036 - val_accuracy: 0.9167
Epoch 48/50
14/14 [==============================] - 0s 12ms/step - loss: 0.0563 - accuracy: 0.9630 - val_loss: 0.5927 - val_accuracy: 0.9167
Epoch 49/50
14/14 [==============================] - 0s 17ms/step - loss: 0.0584 - accuracy: 0.9630 - val_loss: 0.6013 - val_accuracy: 0.9167
Epoch 50/50
14/14 [==============================] - 0s 14ms/step - loss: 0.0555 - accuracy: 0.9722 - val_loss: 0.6211 - val_accuracy: 0.9167
```

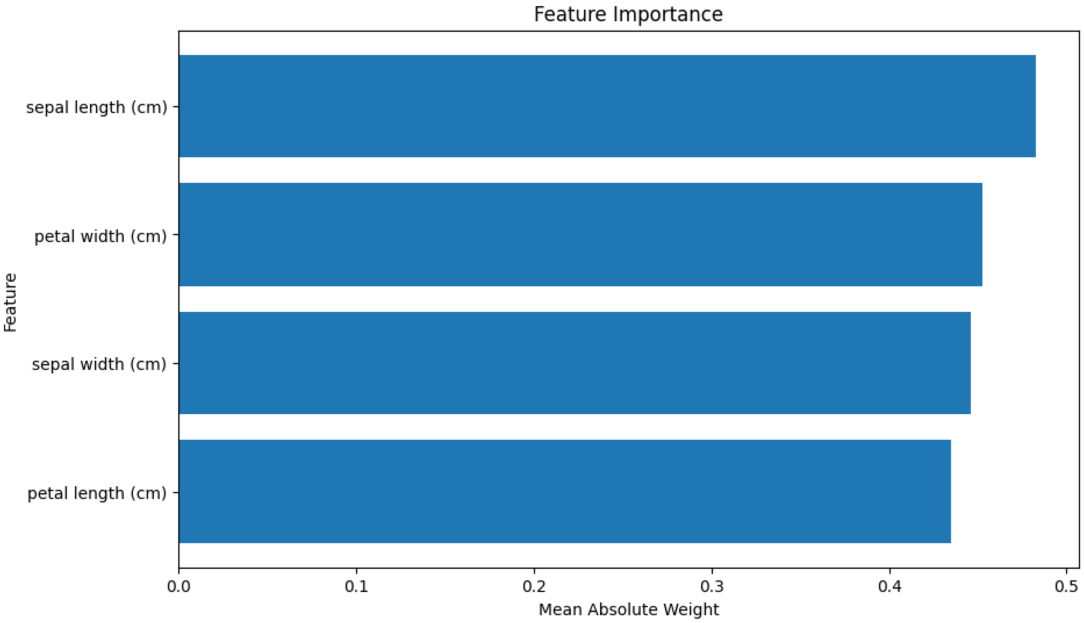




```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize
from sklearn.metrics import roc_auc_score
y_test_one_hot = label_binarize(y_test, classes=[0, 1, 2])
y_probs = model.predict(X_test_scaled)
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(3):
    fpr[i], tpr[i], _ = roc_curve(y_test_one_hot[:, i], y_probs[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
plt.figure(figsize=(8, 6))
for i in range(3):
    plt.plot(fpr[i], tpr[i], label=f'Class {i} (AUC = {roc_auc[i]:0.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc="lower right")
plt.show()

first_layer_weights = model.layers[0].get_weights()[0]
importances = np.mean(np.abs(first_layer_weights), axis=1)
indices = np.argsort(importances)
plt.figure(figsize=(10, 6))
plt.title("Feature Importance")
plt.barh(range(X_train_scaled.shape[1]), importances[indices], align="center")
plt.yticks(range(X_train_scaled.shape[1]), [iris.feature_names[i] for i in indices])
plt.xlabel("Mean Absolute Weight")
plt.ylabel("Feature")
plt.show
```



```
matplotlib.pyplot.show
def show(*args, **kwargs)

/usr/local/lib/python3.10/dist-packages/matplotlib/pyplot.py
Display all open figures.

Parameters
----------
block : bool, optional
```


Feature Importance
Feature
sepal length (cm)
petal width (cm)
sepal width (cm)
petal length (cm)
Mean Absolute Weight