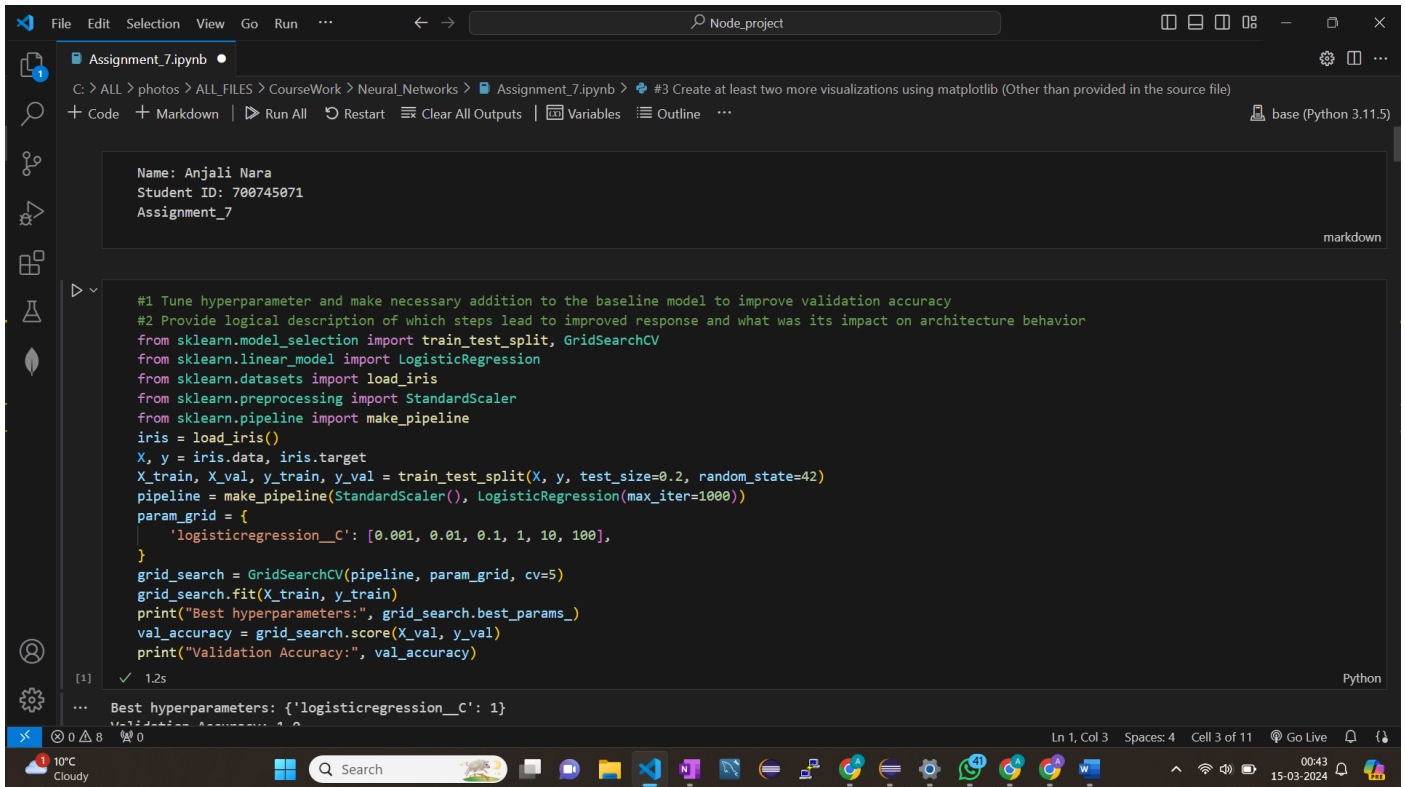


## Assignment 7

Anjali Nara, 700745071

Github link: [https://github.com/naraanjali/Assignment\\_7](https://github.com/naraanjali/Assignment_7)

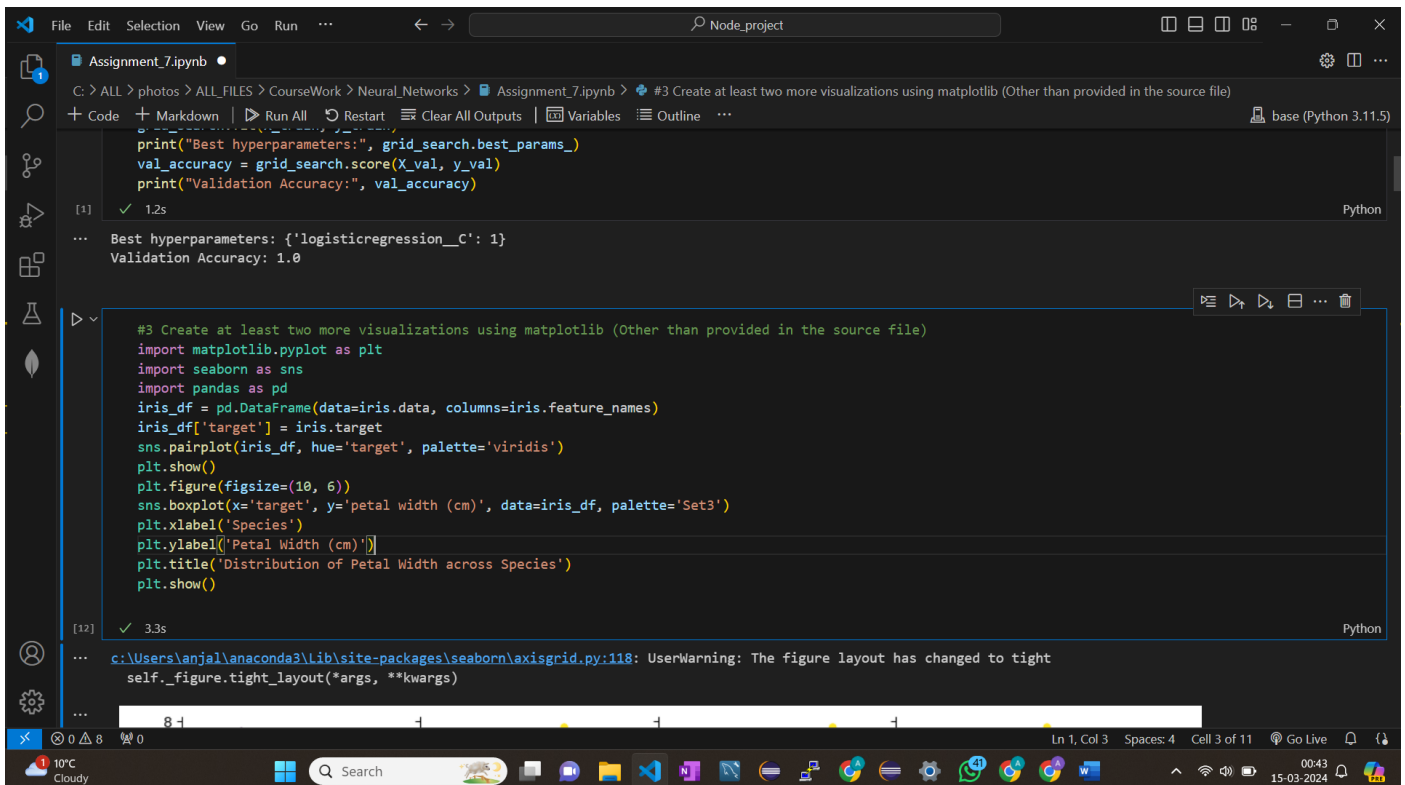
Video link: <https://drive.google.com/file/d/1AU2RI2v6sCU2UWCohXYgfgZhp9PNjcb-/view?usp=sharing>



```
#1 Tune hyperparameter and make necessary addition to the baseline model to improve validation accuracy
#2 Provide logical description of which steps lead to improved response and what was its impact on architecture behavior
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)
```

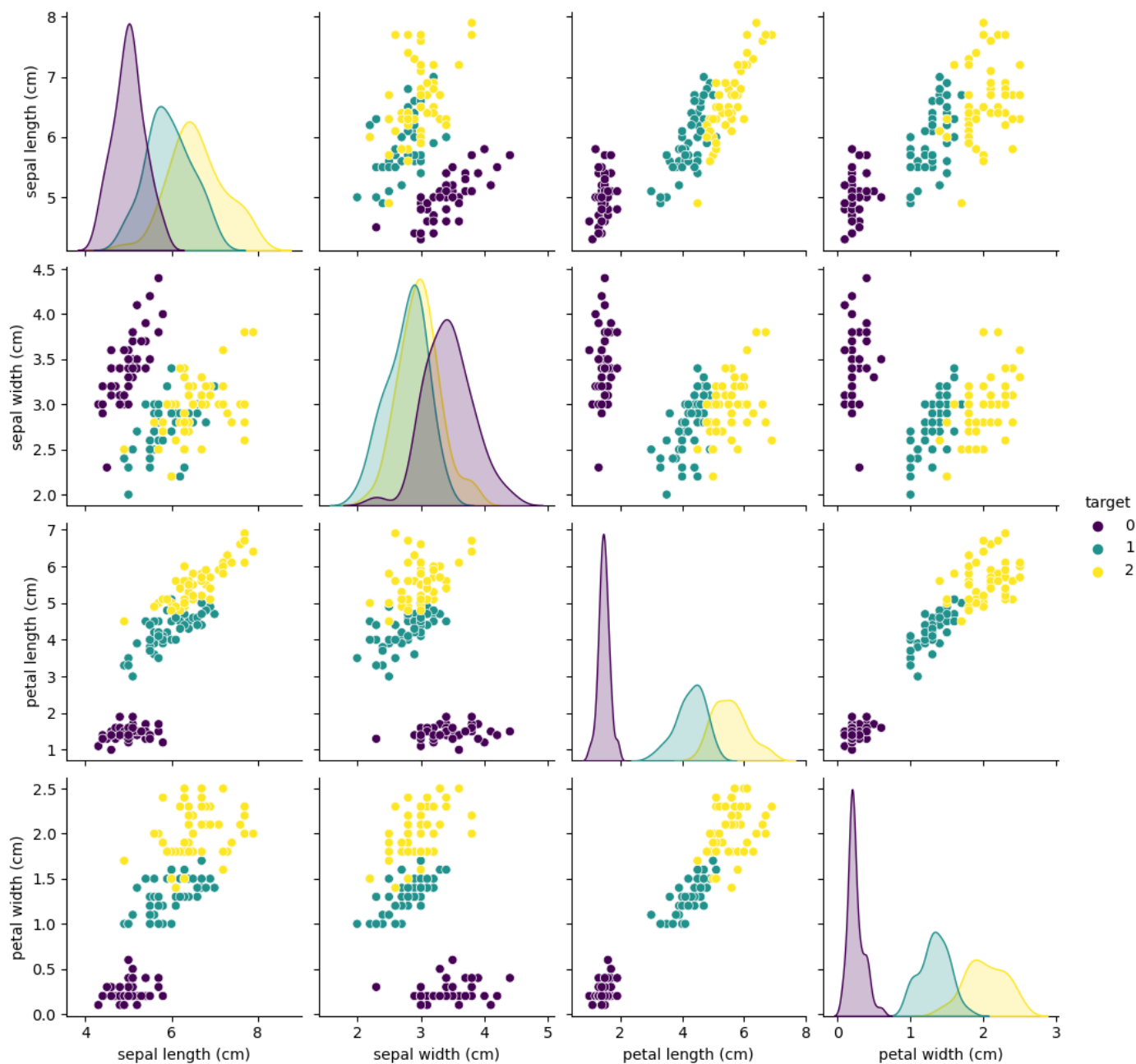
Best hyperparameters: {'logisticregression\_\_C': 1}

Validation Accuracy: 1.0

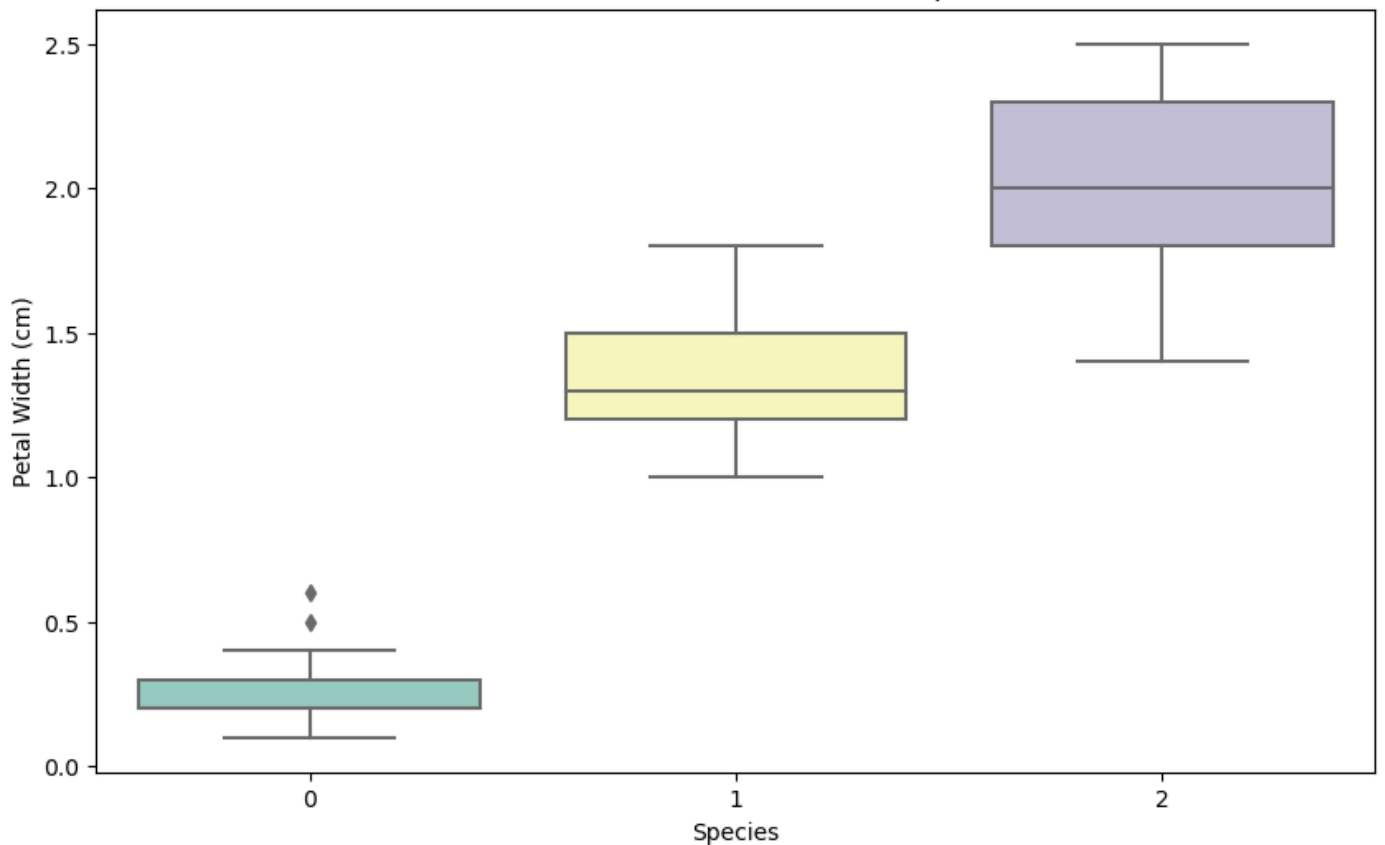


```
#3 Create at least two more visualizations using matplotlib (Other than provided in the source file)
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['target'] = iris.target
sns.pairplot(iris_df, hue='target', 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()
```

c:\Users\anjali\anaconda3\lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight  
self.figure.tight\_layout(\*args, \*\*kwargs)



### Distribution of Petal Width across Species



```

File Edit Selection View Go Run ... Node_project
Assignment_7.ipynb
C: > ALL > photos > ALL_FILES > CourseWork > Neural_Networks > Assignment_7.ipynb > #3 Create at least two more visualizations using matplotlib (Other than provided in the source file)
+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline | ... base (Python 3.11.5)

#4 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)

[3] ✓ 0.0s Python
... Accuracy of Logistic Regression: 1.0

#5 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

```

File Edit Selection View Go Run ... Node\_project

Assignment\_7.ipynb

C:\> ALL > photos > ALL\_FILES > CourseWork > Neural\_Networks > Assignment\_7.ipynb > #3 Create at least two more visualizations using matplotlib (Other than provided in the source file)

+ Code + Markdown ▶ Run All ↺ Restart 🗑 Clear All Outputs | 📄 Variables 📄 Outline ... base (Python 3.11.5)

```
#5 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=['accuracy'])
model.fit(X_train_scaled, y_train, epochs=50, batch_size=8, verbose=1, validation_split=0.1)
loss, accuracy = model.evaluate(X_test_scaled, y_test, verbose=1)
print("Accuracy of Modified Neural Network:", accuracy)
```

[5] ✓ 4.1s Python

Epoch 1/50  
14/14 [=====] - 1s 23ms/step - loss: 1.0502 - accuracy: 0.5185 - val\_loss: 1.0415 - val\_accuracy: 0.5833  
Epoch 2/50

Tempo to plum... Sunday

File Edit Selection View Go Run ... Node\_project

Assignment\_7.ipynb

C:\> ALL > photos > ALL\_FILES > CourseWork > Neural\_Networks > Assignment\_7.ipynb > #3 Create at least two more visualizations using matplotlib (Other than provided in the source file)

+ Code + Markdown ▶ Run All ↺ Restart 🗑 Clear All Outputs | 📄 Variables 📄 Outline ... base (Python 3.11.5)

```
... Epoch 1/50
14/14 [=====] - 1s 23ms/step - loss: 1.0502 - accuracy: 0.5185 - val_loss: 1.0415 - val_accuracy: 0.5833
Epoch 2/50
14/14 [=====] - 0s 7ms/step - loss: 0.9984 - accuracy: 0.5463 - val_loss: 0.9979 - val_accuracy: 0.5833
Epoch 3/50
14/14 [=====] - 0s 6ms/step - loss: 0.9505 - accuracy: 0.5833 - val_loss: 0.9698 - val_accuracy: 0.5833
Epoch 4/50
14/14 [=====] - 0s 4ms/step - loss: 0.9041 - accuracy: 0.6759 - val_loss: 0.9350 - val_accuracy: 0.6667
Epoch 5/50
14/14 [=====] - 0s 4ms/step - loss: 0.8536 - accuracy: 0.7963 - val_loss: 0.9047 - val_accuracy: 0.7500
Epoch 6/50
14/14 [=====] - 0s 5ms/step - loss: 0.8061 - accuracy: 0.8148 - val_loss: 0.8731 - val_accuracy: 0.7500
Epoch 7/50
14/14 [=====] - 0s 4ms/step - loss: 0.7602 - accuracy: 0.8148 - val_loss: 0.8368 - val_accuracy: 0.7500
Epoch 8/50
14/14 [=====] - 0s 4ms/step - loss: 0.7148 - accuracy: 0.8333 - val_loss: 0.7982 - val_accuracy: 0.7500
Epoch 9/50
14/14 [=====] - 0s 4ms/step - loss: 0.6733 - accuracy: 0.8333 - val_loss: 0.7617 - val_accuracy: 0.7500
Epoch 10/50
14/14 [=====] - 0s 4ms/step - loss: 0.6365 - accuracy: 0.8333 - val_loss: 0.7219 - val_accuracy: 0.7500
Epoch 11/50
14/14 [=====] - 0s 4ms/step - loss: 0.6025 - accuracy: 0.8426 - val_loss: 0.6914 - val_accuracy: 0.7500
Epoch 12/50
14/14 [=====] - 0s 5ms/step - loss: 0.5714 - accuracy: 0.8426 - val_loss: 0.6627 - val_accuracy: 0.7500
Epoch 13/50
14/14 [=====] - 0s 5ms/step - loss: 0.5438 - accuracy: 0.8611 - val_loss: 0.6361 - val_accuracy: 0.7500
Epoch 14/50
14/14 [=====] - 0s 4ms/step - loss: 0.5131 - accuracy: 0.8519 - val_loss: 0.6023 - val_accuracy: 0.8333
Epoch 15/50
14/14 [=====] - 0s 7ms/step - loss: 0.4860 - accuracy: 0.8704 - val_loss: 0.5709 - val_accuracy: 0.8333
```

Ln 1, Col 3 Spaces: 4 Spaces: 4 CRLF Cell 3 of 11 Go Live

Tempo to plum... Sunday

File Edit Selection View Go Run ... Node\_project

Assignment\_7.ipynb

C: > ALL > photos > ALL\_FILES > CourseWork > Neural\_Networks > Assignment\_7.ipynb > #3 Create at least two more visualizations using matplotlib (Other than provided in the source file)

+ Code + Markdown ▶ Run All ↺ Restart ≡ Clear All Outputs | 📄 Variables ≡ Outline ... base (Python 3.11.5)

```
#6 Evaluate the model on the testing set
loss, accuracy = model.evaluate(X_test_scaled, y_test, verbose=1)
print("Accuracy on Testing Set:", accuracy)
```

[6] ✓ 0.0s

... 1/1 [=====] - 0s 30ms/step - loss: 0.0574 - accuracy: 0.9667  
Accuracy on Testing Set: 0.9666666388511658

```
#7 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])
```

[8] ✓ 0.1s

... 1/1 [=====] - 0s 45ms/step  
Predictions:  
[[1.7774712e-03 9.7994518e-01 1.8277396e-02]  
[9.9824995e-01 1.3121241e-03 4.3791512e-04]  
[4.2190145e-06 7.1267677e-05 9.9992454e-01]  
[7.7693933e-03 8.9019263e-01 1.0203799e-01]  
[1.7225563e-03 9.0311682e-01 9.5160618e-02]]

Ln 1, Col 3 Spaces: 4 Spaces: 4 CRLF Cell 3 of 11 Go Live 00:45 15-03-2024

File Edit Selection View Go Run ... Node\_project

Assignment\_7.ipynb


C: > ALL > photos > ALL\_FILES > CourseWork > Neural\_Networks > Assignment\_7.ipynb > #3 Create at least two more visualizations using matplotlib (Other than provided in the source file)

+ Code + Markdown ▶ Run All ↺ Restart ≡ Clear All Outputs | 📄 Variables ≡ Outline ... base (Python 3.11.5)

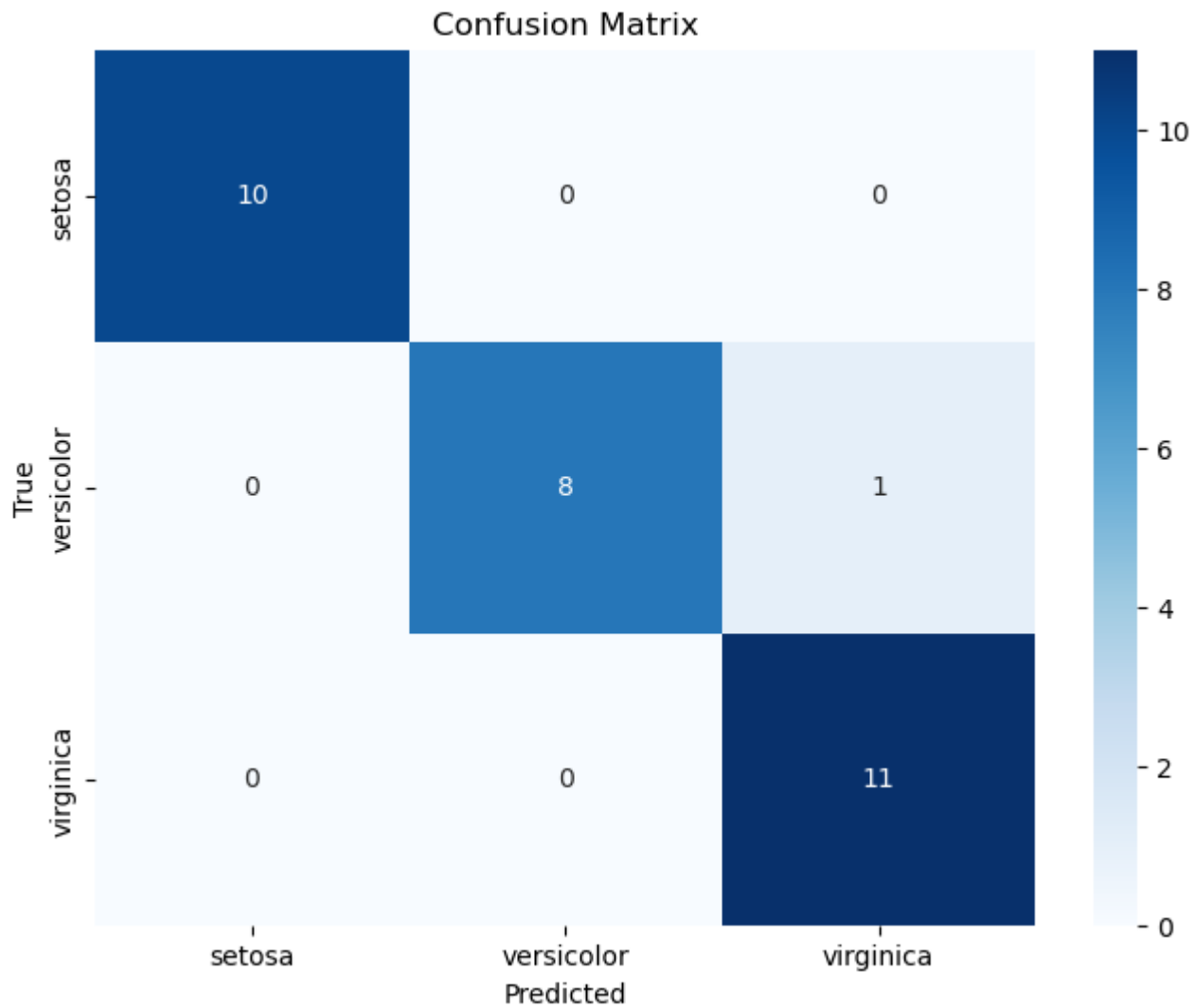
```
#8 plot of confusion matrix
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import seaborn as sns
from tensorflow.keras.models import Sequential
print(hasattr(model, "predict_classes"))
y_pred = model.predict(X_test_scaled).argmax(axis=1)
cm = confusion_matrix(y_test, y_pred)
class_names = iris.target_names
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=class_names, yticklabels=class_names)
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```

[9] ✓ 0.2s

... False  
1/1 [=====] - 0s 47ms/step

... 

Ln 1, Col 3 Spaces: 4 Spaces: 4 CRLF Cell 3 of 11 Go Live 00:45 15-03-2024



```

#9 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()

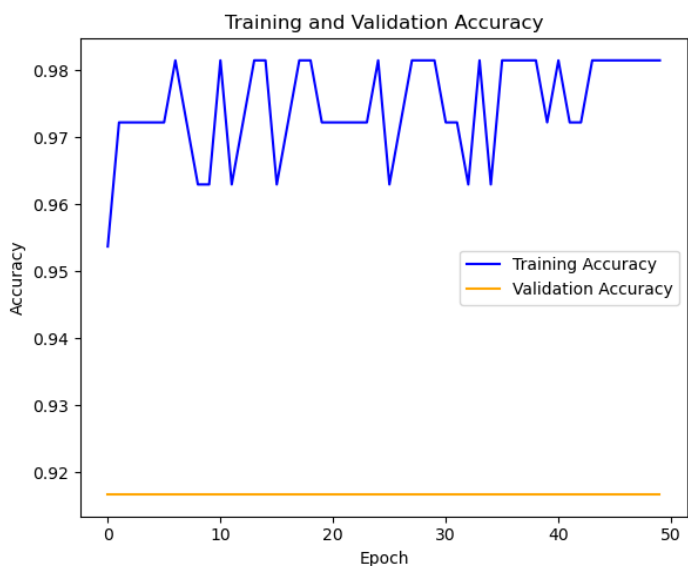
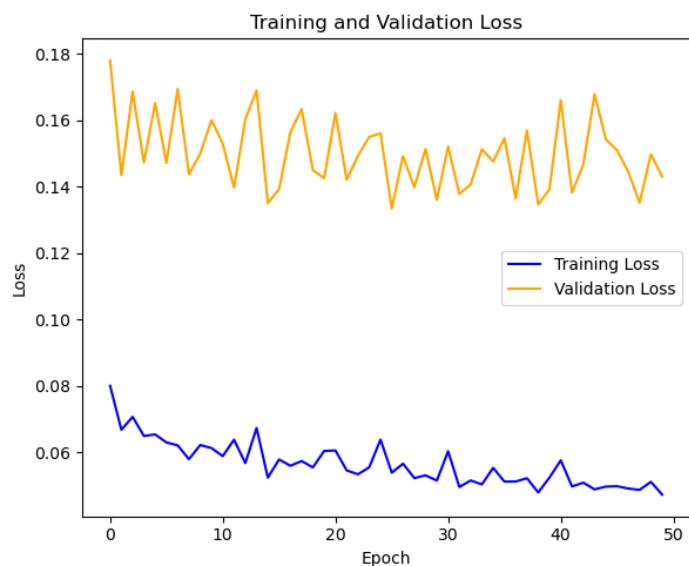
plt.tight_layout()
plt.show()

```

Epoch 1/50  
14/14 [=====] - 0s 6ms/step - loss: 0.0800 - accuracy: 0.9537 - val\_loss: 0.1780 - val\_accuracy: 0.9167  
Epoch 2/50  
14/14 [=====] - 0s 3ms/step - loss: 0.0668 - accuracy: 0.9722 - val\_loss: 0.1434 - val\_accuracy: 0.9167  
Epoch 3/50

```
File Edit Selection View Go Run ... Node_project
Assignment_7.ipynb
C: > ALL > photos > ALL_FILES > CourseWork > Neural_Networks > Assignment_7.ipynb > #8 plot of confusion matrix
+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ... base (Python 3.11.5)

... Epoch 1/50
14/14 [=====] - 0s 6ms/step - loss: 0.0800 - accuracy: 0.9537 - val_loss: 0.1780 - val_accuracy: 0.9167
Epoch 2/50
14/14 [=====] - 0s 3ms/step - loss: 0.0668 - accuracy: 0.9722 - val_loss: 0.1434 - val_accuracy: 0.9167
Epoch 3/50
14/14 [=====] - 0s 3ms/step - loss: 0.0707 - accuracy: 0.9722 - val_loss: 0.1686 - val_accuracy: 0.9167
Epoch 4/50
14/14 [=====] - 0s 3ms/step - loss: 0.0650 - accuracy: 0.9722 - val_loss: 0.1472 - val_accuracy: 0.9167
Epoch 5/50
14/14 [=====] - 0s 4ms/step - loss: 0.0654 - accuracy: 0.9722 - val_loss: 0.1651 - val_accuracy: 0.9167
Epoch 6/50
14/14 [=====] - 0s 3ms/step - loss: 0.0630 - accuracy: 0.9722 - val_loss: 0.1471 - val_accuracy: 0.9167
Epoch 7/50
14/14 [=====] - 0s 3ms/step - loss: 0.0621 - accuracy: 0.9815 - val_loss: 0.1694 - val_accuracy: 0.9167
Epoch 8/50
14/14 [=====] - 0s 3ms/step - loss: 0.0580 - accuracy: 0.9722 - val_loss: 0.1437 - val_accuracy: 0.9167
Epoch 9/50
14/14 [=====] - 0s 3ms/step - loss: 0.0622 - accuracy: 0.9630 - val_loss: 0.1499 - val_accuracy: 0.9167
Epoch 10/50
14/14 [=====] - 0s 3ms/step - loss: 0.0613 - accuracy: 0.9630 - val_loss: 0.1600 - val_accuracy: 0.9167
Epoch 11/50
14/14 [=====] - 0s 4ms/step - loss: 0.0589 - accuracy: 0.9815 - val_loss: 0.1529 - val_accuracy: 0.9167
Epoch 12/50
14/14 [=====] - 0s 6ms/step - loss: 0.0638 - accuracy: 0.9630 - val_loss: 0.1396 - val_accuracy: 0.9167
Epoch 13/50
14/14 [=====] - 0s 5ms/step - loss: 0.0568 - accuracy: 0.9722 - val_loss: 0.1602 - val_accuracy: 0.9167
Epoch 14/50
14/14 [=====] - 0s 5ms/step - loss: 0.0673 - accuracy: 0.9815 - val_loss: 0.1689 - val_accuracy: 0.9167
Epoch 15/50
14/14 [=====] - 0s 4ms/step - loss: 0.0524 - accuracy: 0.9815 - val_loss: 0.1350 - val_accuracy: 0.9167
```



```
File Edit Selection View Go Run ... Node_project
Assignment_7.ipynb
C: > ALL > photos > ALL_FILES > CourseWork > Neural_Networks > Assignment_7.ipynb > #9 Training and testing Loss and accuracy plots in one plot using subplot command and history object
+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ... base (Python 3.11.5)

#10 Provide at least two more visualizations reflecting your solution.
#11. Provide logical description of which steps lead to improved response for new dataset when compared with
#baseline model and enhance architecture and what was its impact on architecture behavior.
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)
```

```
File Edit Selection View Go Run ... Node_project
Assignment_7.ipynb
C: > ALL > photos > ALL_FILES > CourseWork > Neural_Networks > Assignment_7.ipynb > #9 Training and testing Loss and accuracy plots in one plot using subplot command and history object
+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ... base (Python 3.11.5)

    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()

[11] ✓ 0.3s Python

... 1/1 [=====] - 0s 52ms/step

...
ROC Curve
1.0
0.0
```



