Neural Networks & Deep learning

Assignment 7

Name: Revathi Atchi

Student Id: 700742168

GITHUB LINK: https://github.com/revathiatchi/NeuralAssignment7.git

VIDEO LINK:
https://github.com/revathiatchi/NeuralAssignment7/assets/156601745/dc8017a9-4233-4c8e-aace-ece67dcf0a26

Use Case Description:

LeNet5, AlexNet, Vgg16, Vgg19

1. Training the model

2. Evaluating the model

Programming elements: 1. About CNN

2. Hyperparameters of CNN

3. Image classification with CNN

In class programming:

1. Tune hyperparameter and make necessary addition to the baseline model to improve validation accuracy and reduce validation loss.

2. Provide logical description of which steps lead to improved response and what was its impact on architecture behavior.

3. Create at least two more visualizations using matplotlib (Other than provided in the source file)

4. Use dataset of your own choice and implement baseline models provided.

5. Apply modified architecture to your own selected dataset and train it.

6. Evaluate your model on testing set.

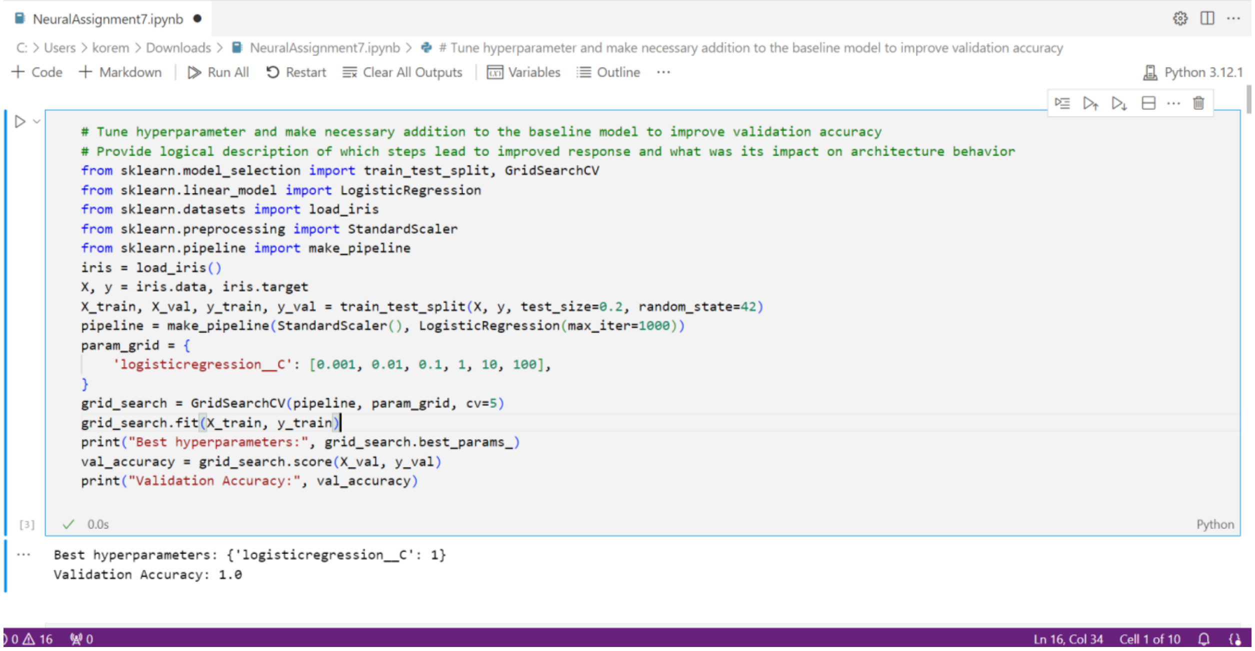
7. Save the improved model and use it for prediction on testing data

8. Provide plot of confusion matric

9. Provide Training and testing Loss and accuracy plots in one plot using subplot command and history object.

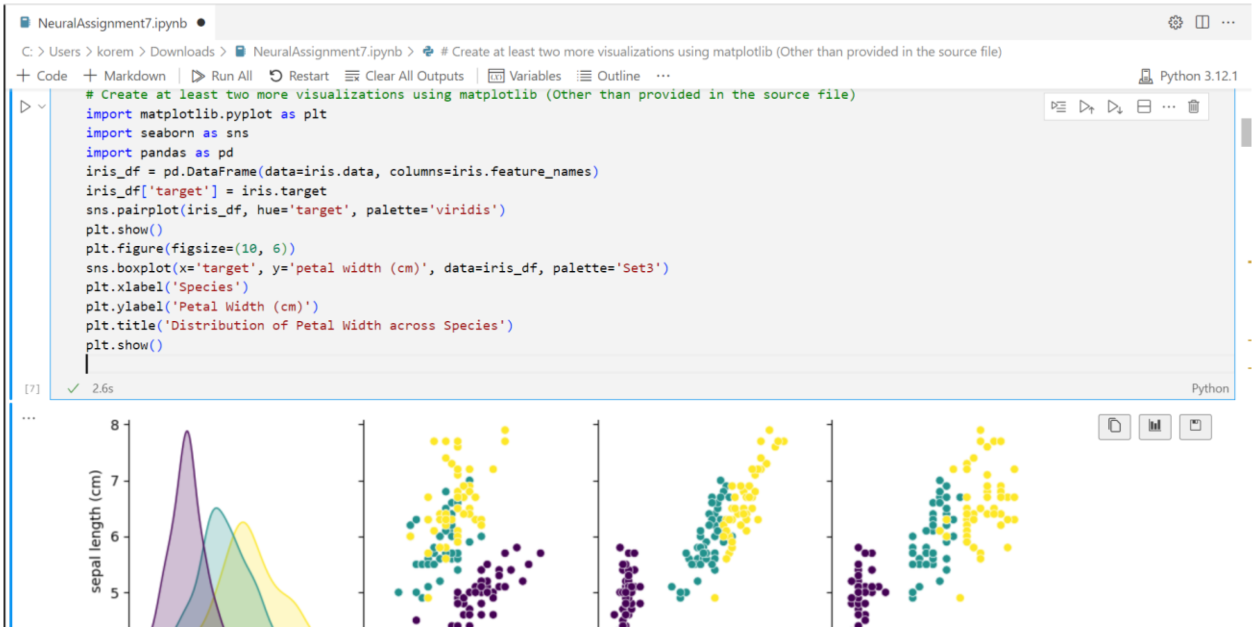
10. Provide at least two more visualizations reflecting your solution.

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

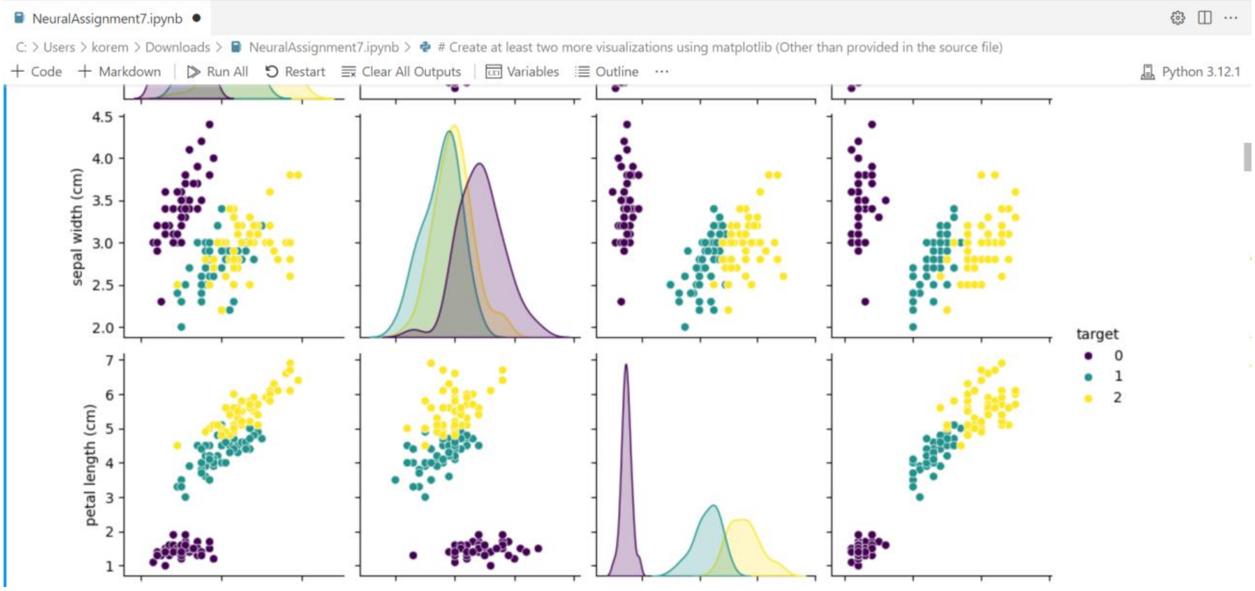

NeuralAssignment7.ipynb ●
C: > Users > korem > Downloads > NeuralAssignment7.ipynb > # Tune hyperparameter and make necessary addition to the baseline model to improve validation accuracy
+ Code
+ Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
Python 3.12.1
[3]
✓ 0.0s
Python
0 ⚠ 16
0
Ln 16, Col 34
Cell 1 of 10

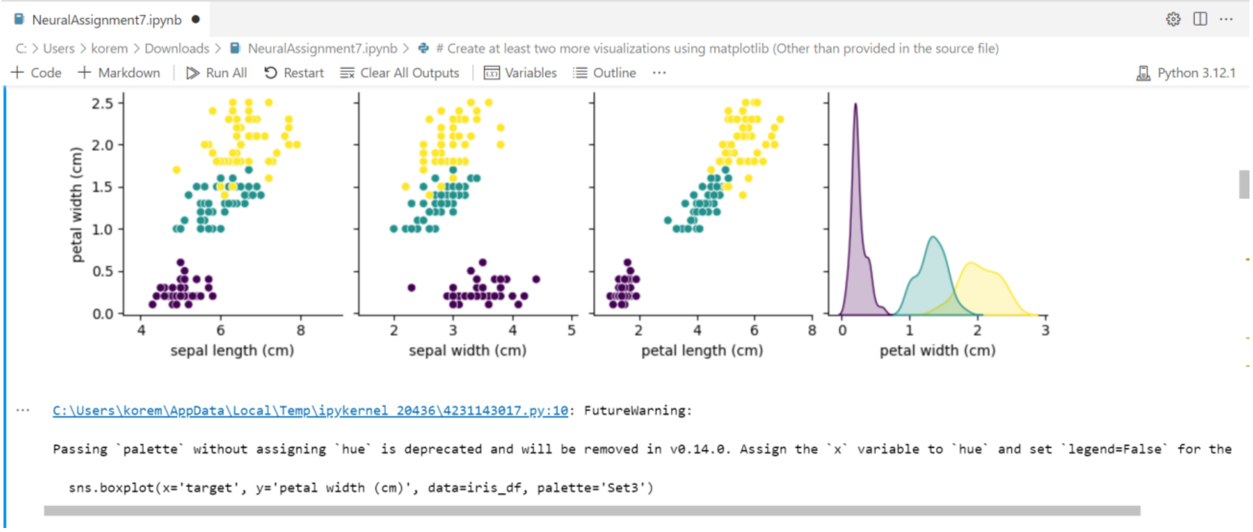
```python
# Tune hyperparameter and make necessary addition to the baseline model to improve validation accuracy
# Provide logical description of which steps lead to improved response and what was its impact on architecture behavior
from sklearn.model_selection import train_test_split, GridSearchCV
from 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
```

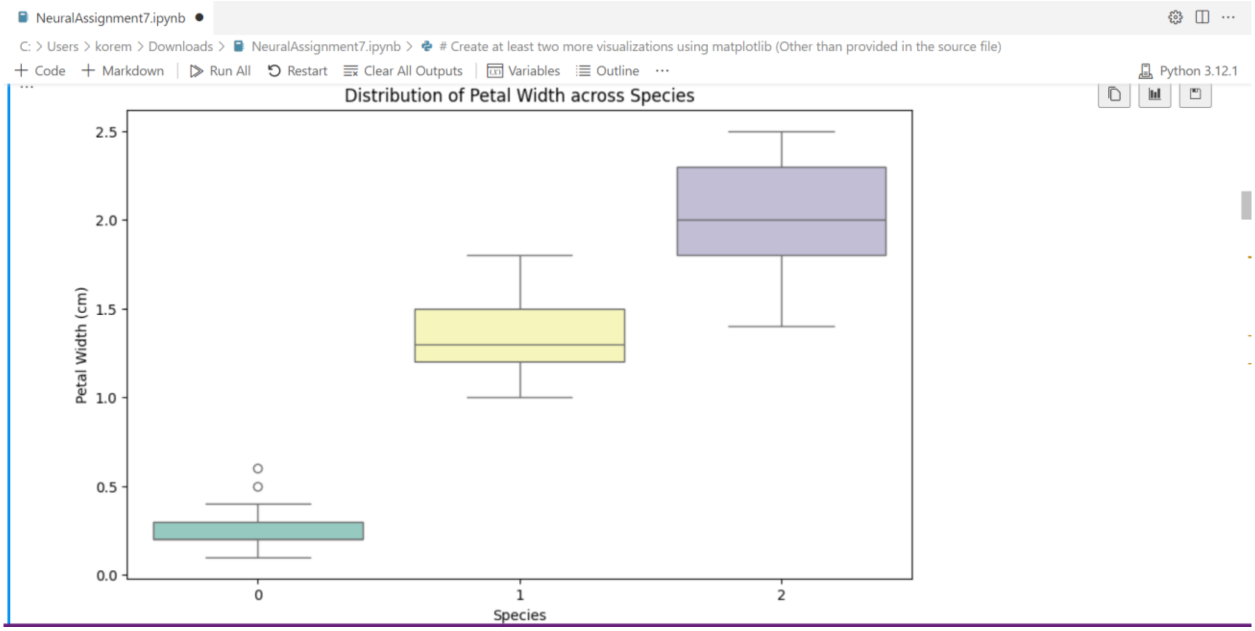

NeuralAssignment7.ipynb ●
C: > Users > korem > Downloads > NeuralAssignment7.ipynb > # Create at least two more visualizations using matplotlib (Other than provided in the source file)
+ Code
+ Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
Python 3.12.1
[7]
✓ 2.6s
Python
8
7
6
5
sepal length (cm)

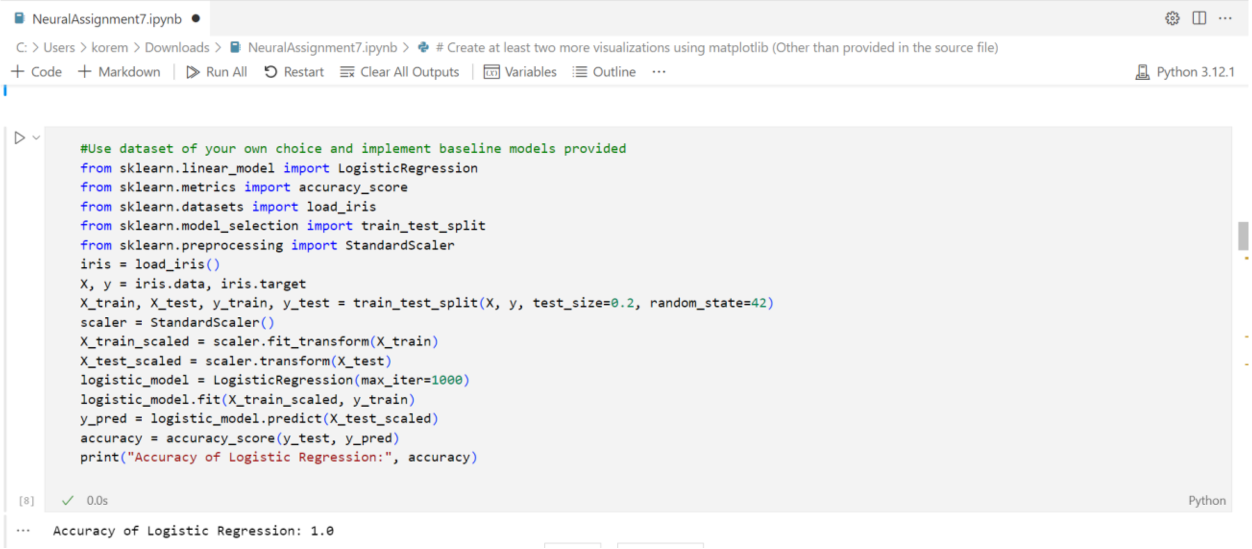
```python
# Create at least two more visualizations using matplotlib (Other than provided in the source file)
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['target'] = iris.target
sns.pairplot(iris_df,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


NeuralAssignment7.ipynb ●
C: > Users > korem > Downloads > NeuralAssignment7.ipynb > # 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
Python 3.12.1
sepal width (cm)
4.5
4.0
3.5
3.0
2.5
2.0
petal length (cm)
7
6
5
4
3
2
1
target
0
1
2



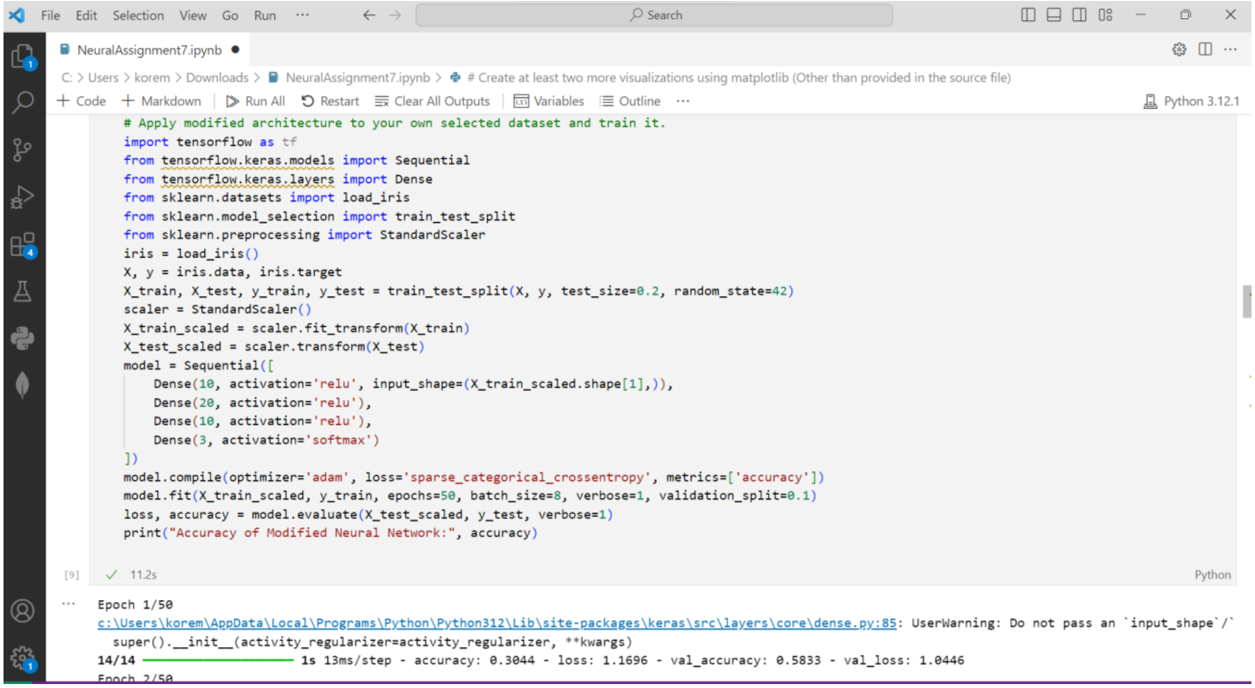
```
C:\Users\korem\AppData\Local\Temp\ipykernel_20436\4231143017.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the

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


NeuralAssignment7.ipynb ●
C: > Users > korem > Downloads > NeuralAssignment7.ipynb > # 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 ···
Python 3.12.1
Distribution of Petal Width across Species
Petal Width (cm)
2.5
2.0
1.5
1.0
0.5
0.0
0
1
2
Species


NeuralAssignment7.ipynb ●
C: > Users > korem > Downloads > NeuralAssignment7.ipynb > # 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 ···
Python 3.12.1
[8]
✓ 0.0s
Python

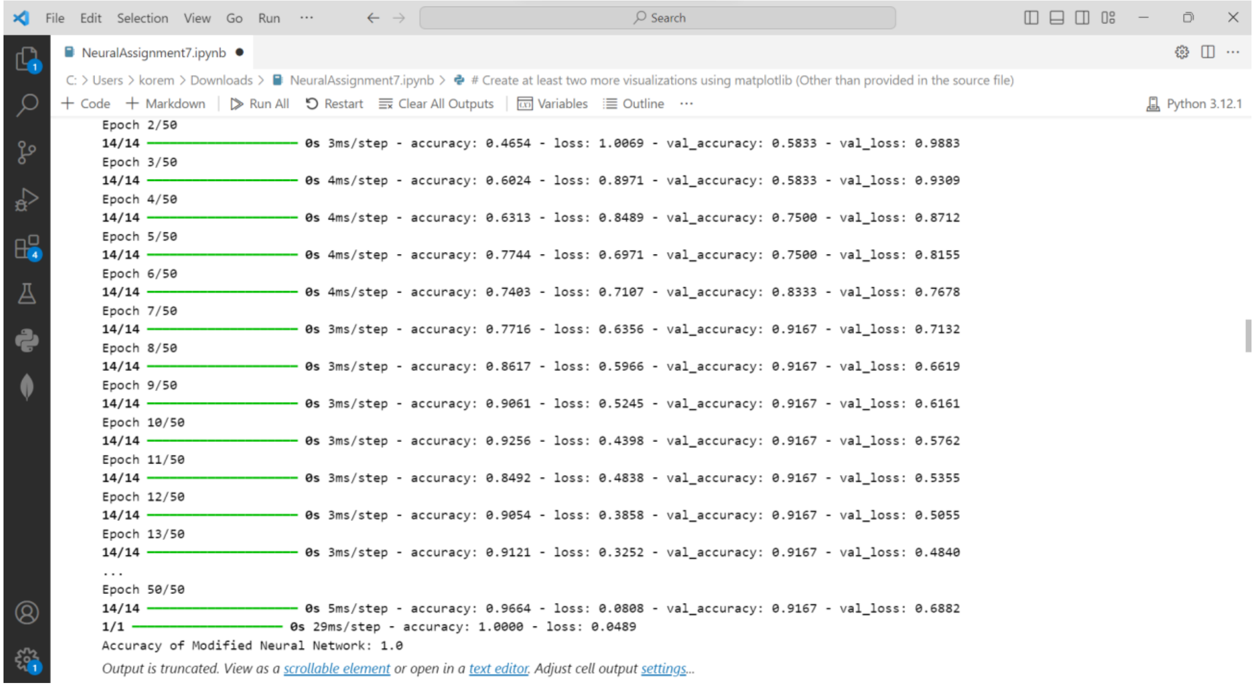
```python
#Use dataset of your own choice and implement baseline models provided
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import 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
```


File Edit Selection View Go Run ···
Search
NeuralAssignment7.ipynb ●
C: > Users > korem > Downloads > NeuralAssignment7.ipynb > # 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 ···
Python 3.12.1
[9]
✓ 11.2s
Python

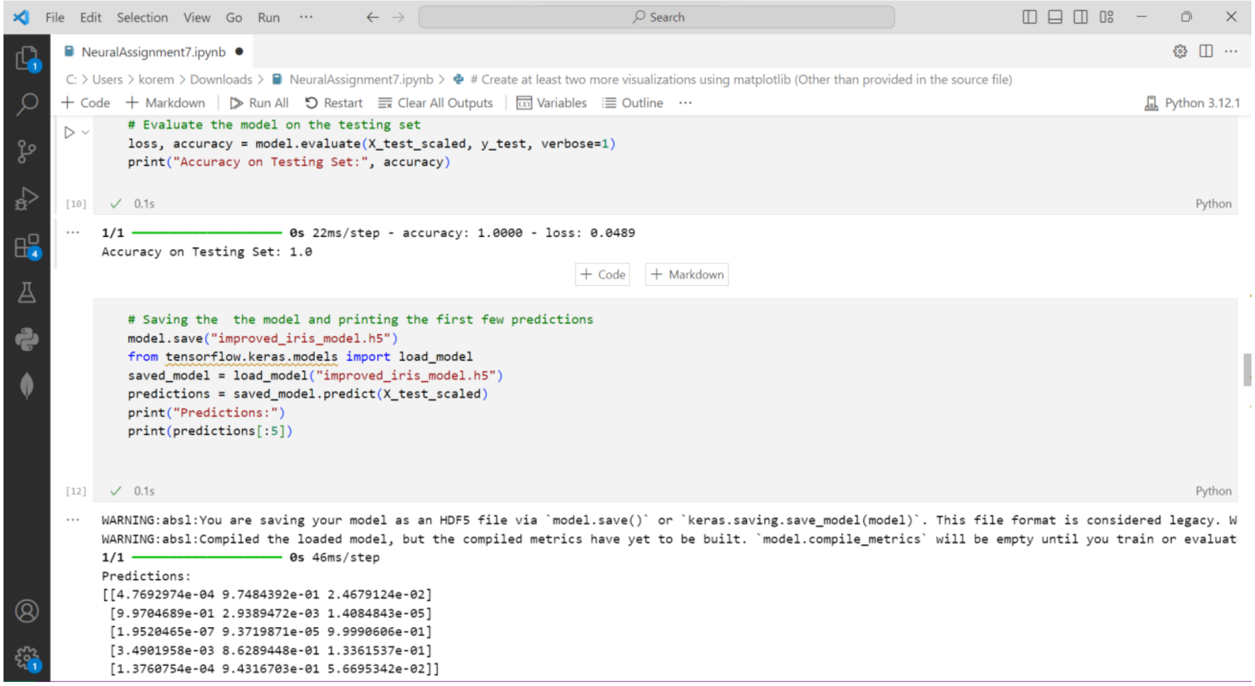
```python
# Apply modified architecture to your own selected dataset and train it.
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing 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)
```

```
Epoch 1/50
c:\Users\korem\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\core\dense.py:85: UserWarning: Do not pass an `input_shape`/`
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
14/14 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.3044 - loss: 1.1696 - val_accuracy: 0.5833 - val_loss: 1.0446
Epoch 2/50
```


File Edit Selection View Go Run ···
Search
NeuralAssignment7.ipynb ●
C: > Users > korem > Downloads > NeuralAssignment7.ipynb > # 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 ···
Python 3.12.1

```
Epoch 2/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4654 - loss: 1.0069 - val_accuracy: 0.5833 - val_loss: 0.9883
Epoch 3/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6024 - loss: 0.8971 - val_accuracy: 0.5833 - val_loss: 0.9309
Epoch 4/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6313 - loss: 0.8489 - val_accuracy: 0.7500 - val_loss: 0.8712
Epoch 5/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7744 - loss: 0.6971 - val_accuracy: 0.7500 - val_loss: 0.8155
Epoch 6/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7403 - loss: 0.7107 - val_accuracy: 0.8333 - val_loss: 0.7678
Epoch 7/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7716 - loss: 0.6356 - val_accuracy: 0.9167 - val_loss: 0.7132
Epoch 8/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8617 - loss: 0.5966 - val_accuracy: 0.9167 - val_loss: 0.6619
Epoch 9/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9061 - loss: 0.5245 - val_accuracy: 0.9167 - val_loss: 0.6161
Epoch 10/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9256 - loss: 0.4398 - val_accuracy: 0.9167 - val_loss: 0.5762
Epoch 11/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8492 - loss: 0.4838 - val_accuracy: 0.9167 - val_loss: 0.5355
Epoch 12/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9054 - loss: 0.3858 - val_accuracy: 0.9167 - val_loss: 0.5055
Epoch 13/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9121 - loss: 0.3252 - val_accuracy: 0.9167 - val_loss: 0.4840
...
Epoch 50/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9664 - loss: 0.0808 - val_accuracy: 0.9167 - val_loss: 0.6882
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 1.0000 - loss: 0.0489
Accuracy of Modified Neural Network: 1.0
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...


File Edit Selection View Go Run ···
Search
NeuralAssignment7.ipynb ●
C: > Users > korem > Downloads > NeuralAssignment7.ipynb > # 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 ···
Python 3.12.1

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

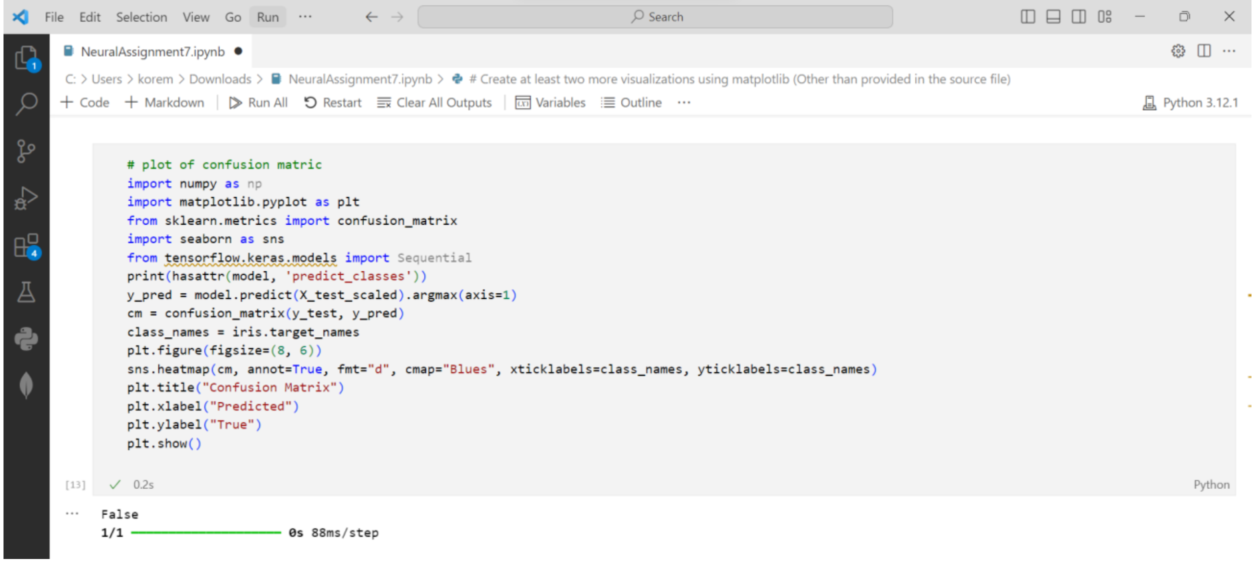
[10] ✓ 0.1s Python

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 1.0000 - loss: 0.0489
Accuracy on Testing Set: 1.0
```

```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])
```

[12] ✓ 0.1s Python

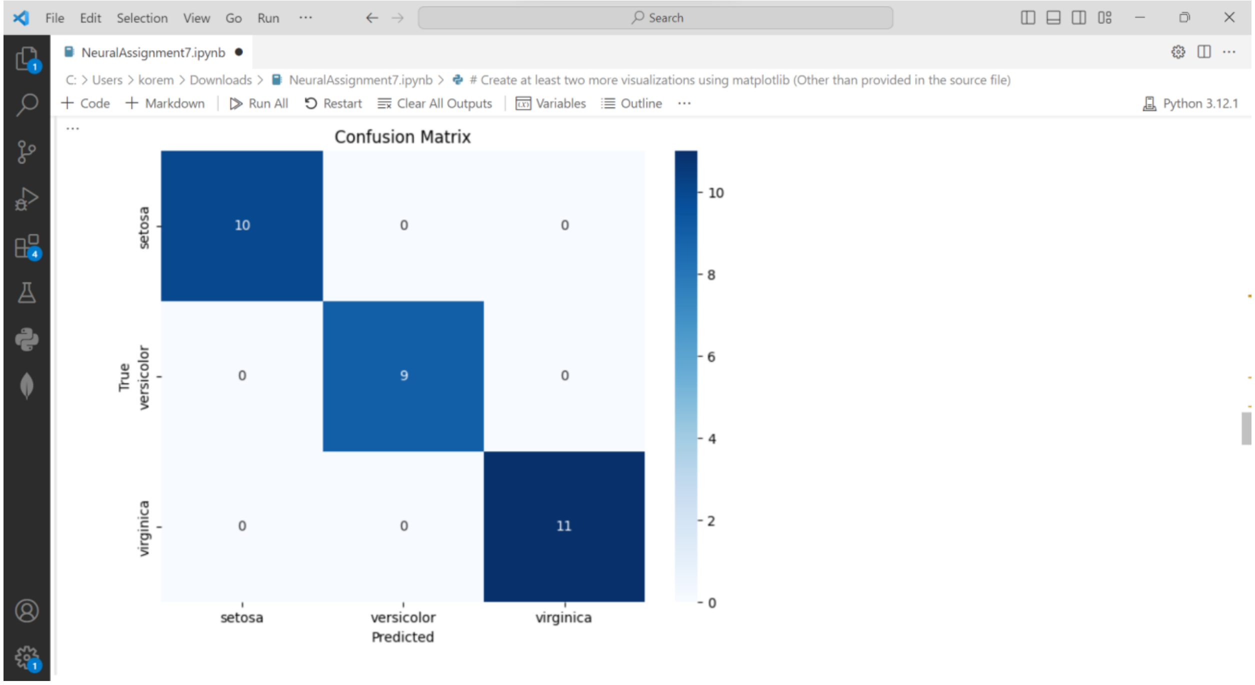
```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. W
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluat
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
Predictions:
[[4.7692974e-04 9.7484392e-01 2.4679124e-02]
 [9.9704689e-01 2.9389472e-03 1.4084843e-05]
 [1.9520465e-07 9.3719871e-05 9.9990606e-01]
 [3.4901958e-03 8.6289448e-01 1.3361537e-01]
 [1.3760754e-04 9.4316703e-01 5.6695342e-02]]
```


File Edit Selection View Go Run ···
Search
NeuralAssignment7.ipynb ●
C: > Users > korem > Downloads > NeuralAssignment7.ipynb > # 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 ···
Python 3.12.1

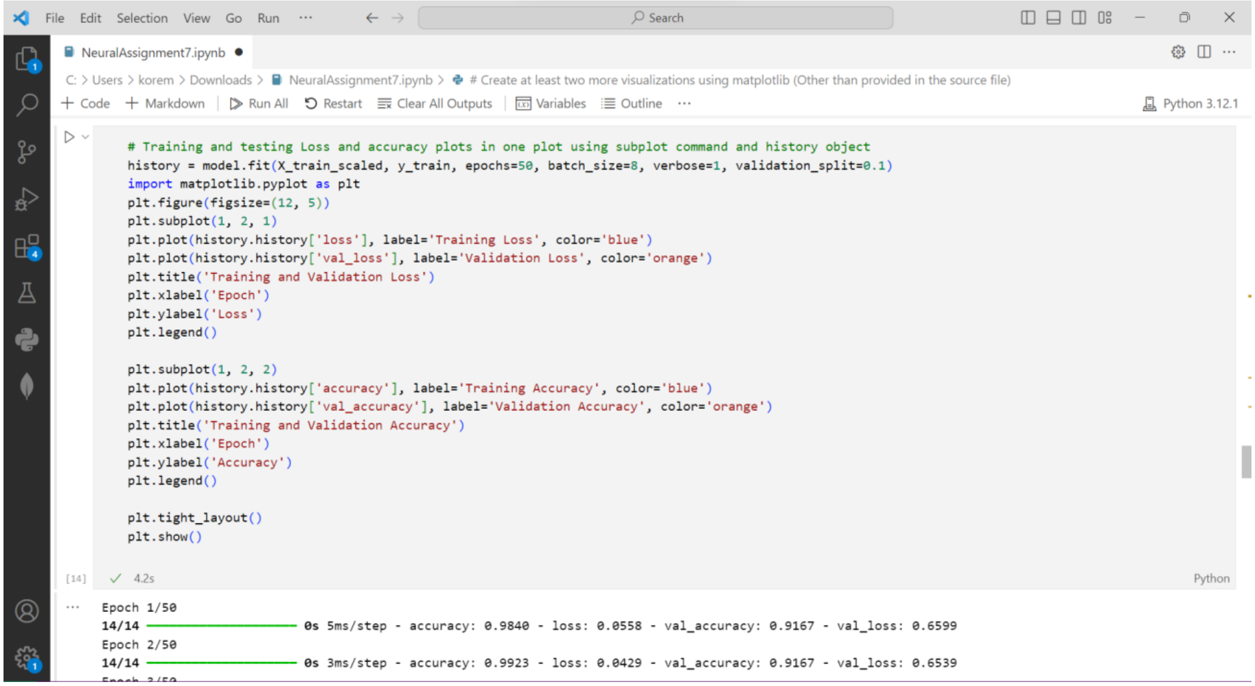
```python
# plot of confusion matric
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import seaborn as sns
from tensorflow.keras.models import Sequential
print(hasattr(model, 'predict_classes'))
y_pred = model.predict(X_test_scaled).argmax(axis=1)
cm = 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()
```

[13] ✓ 0.2s Python

```
False
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 88ms/step
```



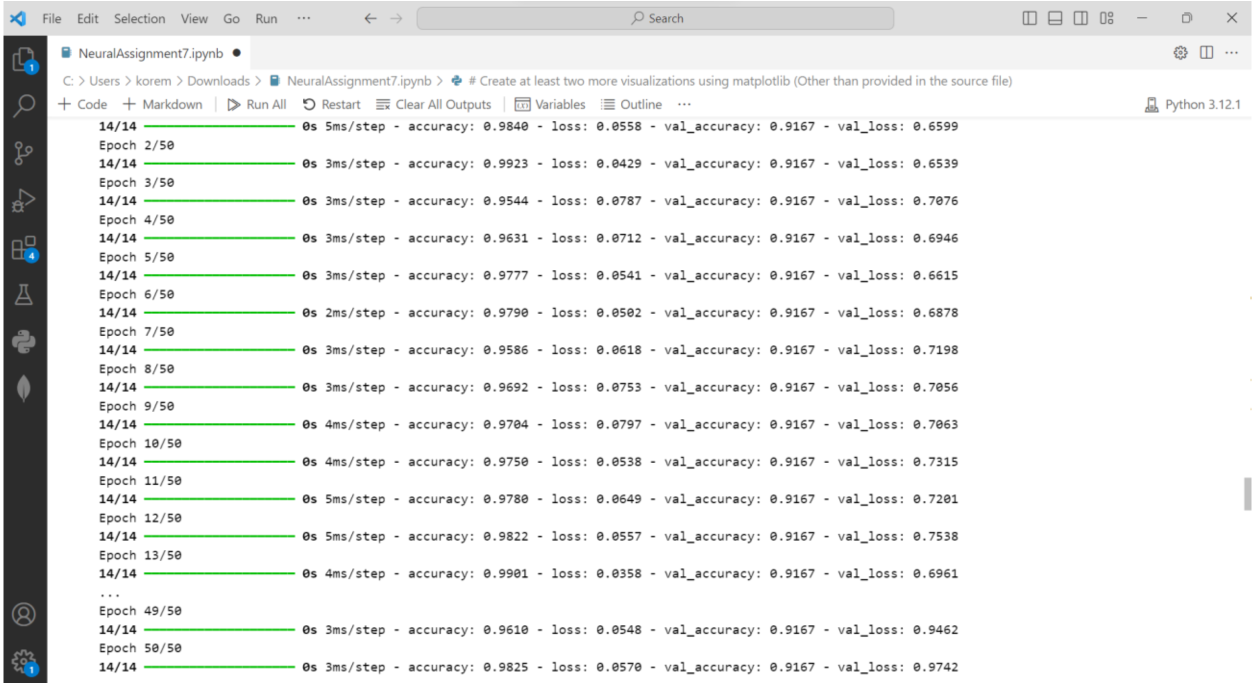
C: > Users > korem > Downloads > NeuralAssignment7.ipynb > # Create at least two more visualizations using matplotlib (Other than provided in the source file)



C: > Users > korem > Downloads > NeuralAssignment7.ipynb > # Create at least two more visualizations using matplotlib (Other than provided in the source file)

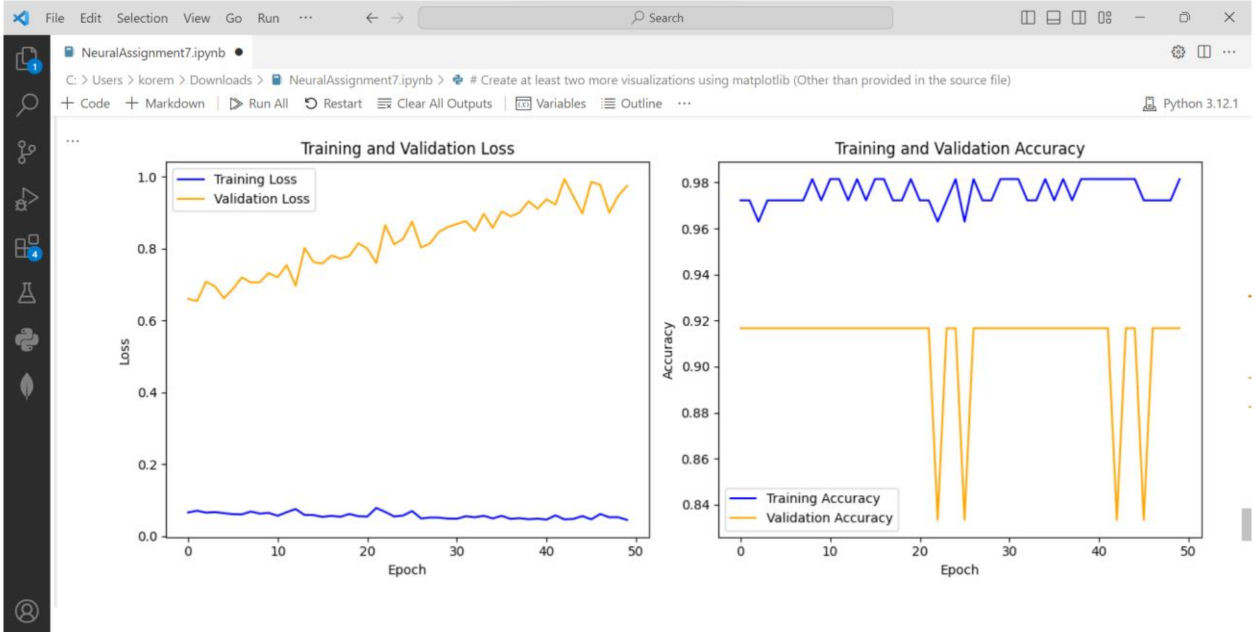
```
# Training and testing Loss and accuracy plots in one plot using subplot command and history object
history = model.fit(X_train_scaled, y_train, epochs=50, batch_size=8, verbose=1, validation_split=0.1)
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], 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()
```

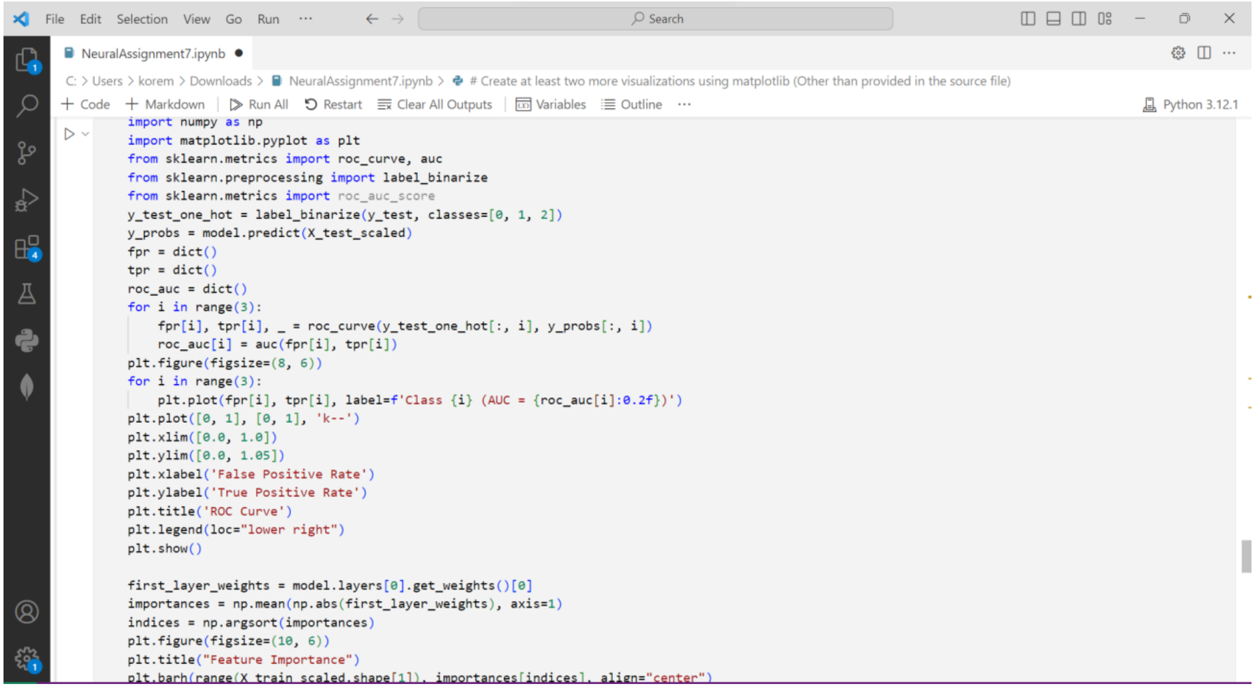
```
Epoch 1/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9840 - loss: 0.0558 - val_accuracy: 0.9167 - val_loss: 0.6599
Epoch 2/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9923 - loss: 0.0429 - val_accuracy: 0.9167 - val_loss: 0.6539
Epoch 3/50
```



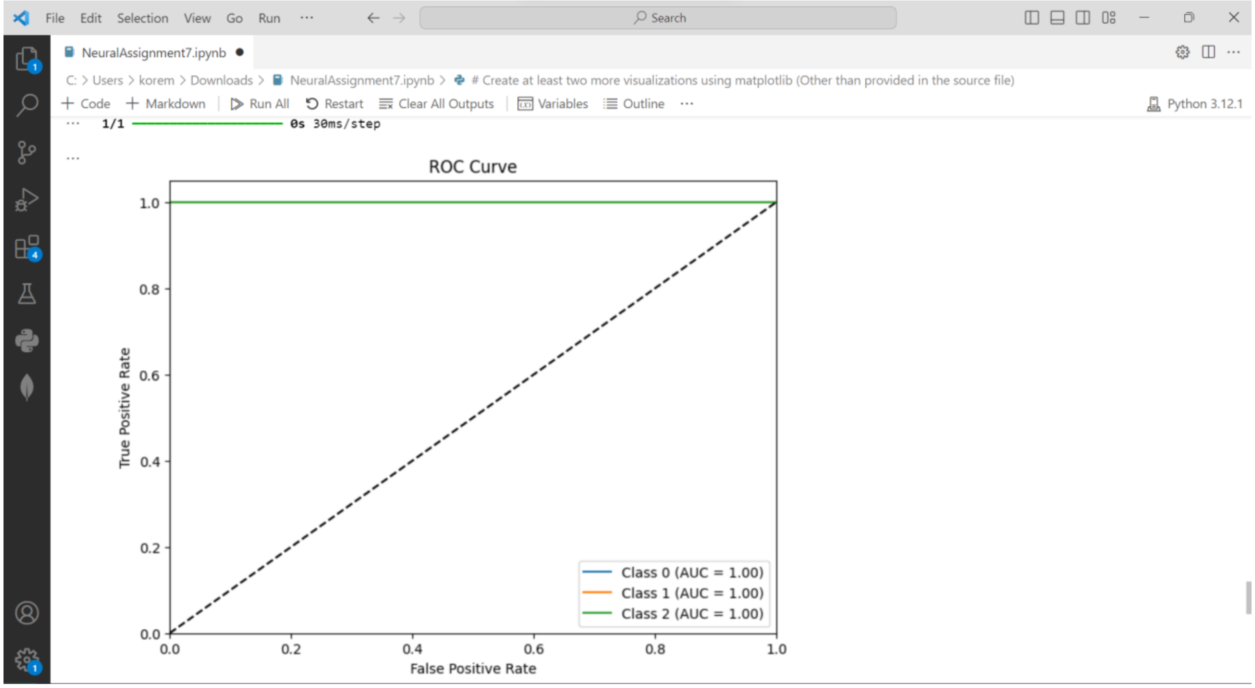
File Edit Selection View Go Run ···
Search
NeuralAssignment7.ipynb ●
C: > Users > korem > Downloads > NeuralAssignment7.ipynb > # Create at least two more visualizations using matplotlib (Other than provided in the source file)
+ Code + Markdown ▷ Run All ⟲ Restart Clear All Outputs Variables Outline ···
Python 3.12.1

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9840 - loss: 0.0558 - val_accuracy: 0.9167 - val_loss: 0.6599
Epoch 2/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9923 - loss: 0.0429 - val_accuracy: 0.9167 - val_loss: 0.6539
Epoch 3/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9544 - loss: 0.0787 - val_accuracy: 0.9167 - val_loss: 0.7076
Epoch 4/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9631 - loss: 0.0712 - val_accuracy: 0.9167 - val_loss: 0.6946
Epoch 5/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9777 - loss: 0.0541 - val_accuracy: 0.9167 - val_loss: 0.6615
Epoch 6/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9790 - loss: 0.0502 - val_accuracy: 0.9167 - val_loss: 0.6878
Epoch 7/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9586 - loss: 0.0618 - val_accuracy: 0.9167 - val_loss: 0.7198
Epoch 8/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9692 - loss: 0.0753 - val_accuracy: 0.9167 - val_loss: 0.7056
Epoch 9/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9704 - loss: 0.0797 - val_accuracy: 0.9167 - val_loss: 0.7063
Epoch 10/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9750 - loss: 0.0538 - val_accuracy: 0.9167 - val_loss: 0.7315
Epoch 11/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9780 - loss: 0.0649 - val_accuracy: 0.9167 - val_loss: 0.7201
Epoch 12/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9822 - loss: 0.0557 - val_accuracy: 0.9167 - val_loss: 0.7538
Epoch 13/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9901 - loss: 0.0358 - val_accuracy: 0.9167 - val_loss: 0.6961
...
Epoch 49/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9610 - loss: 0.0548 - val_accuracy: 0.9167 - val_loss: 0.9462
Epoch 50/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9825 - loss: 0.0570 - val_accuracy: 0.9167 - val_loss: 0.9742
```

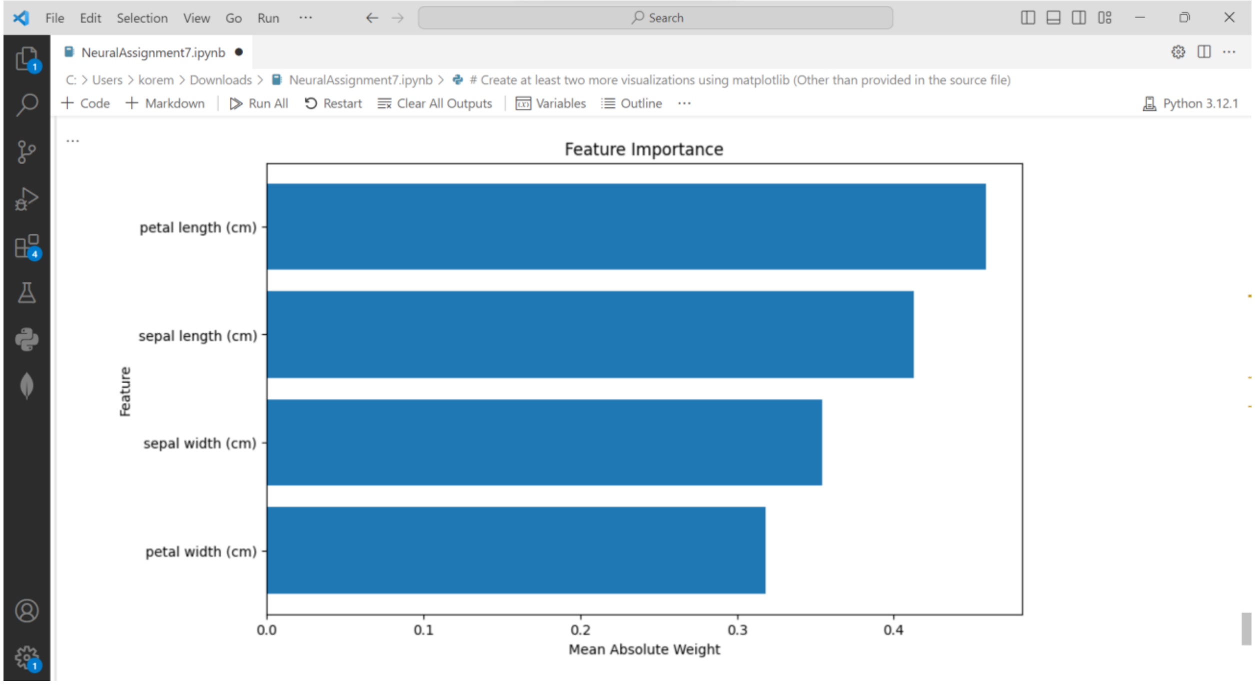



File Edit Selection View Go Run ···
Search
NeuralAssignment7.ipynb ●
C: > Users > korem > Downloads > NeuralAssignment7.ipynb > # Create at least two more visualizations using matplotlib (Other than provided in the source file)
+ Code + Markdown ▷ Run All ⟲ Restart Clear All Outputs Variables Outline ···
Python 3.12.1

```python
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")
```


File Edit Selection View Go Run ···
Search
NeuralAssignment7.ipynb ●
C: > Users > korem > Downloads > NeuralAssignment7.ipynb > # 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 ···
Python 3.12.1
ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
Class 0 (AUC = 1.00)
Class 1 (AUC = 1.00)
Class 2 (AUC = 1.00)

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
```



Create at least two more visualizations using matplotlib (Other than provided in the source file)