## Keras model load / test

To load a saved Keras model and test it on new data, follow these steps:

**1. Load the Saved Model**

Use load\_model from tensorflow.keras.models to load your saved model.

### python

from tensorflow.keras.models import load\_model

# Load the model

loaded\_model = load\_model('mlp\_model.keras')

**2. Prepare the Test Data**

Ensure that your test data is preprocessed in the same way as the data used to train the model. This includes scaling or normalizing the features if those steps were part of your preprocessing.

### python

from sklearn.preprocessing import StandardScaler

# Example preprocessing (if applicable)

scaler = StandardScaler()

X\_test = scaler.transform(X\_test) # Assuming X\_test is your test feature data

**3. Make Predictions**

Use the predict method of the loaded model to make predictions on your test data.

### python

# Make predictions

y\_pred = loaded\_model.predict(X\_test)

# If it's a classification problem, you might want to convert probabilities to class labels

# For binary classification

y\_pred\_classes = (y\_pred > 0.5).astype(int) # Adjust threshold if needed

# For multi-class classification, use argmax to get class indices

# y\_pred\_classes = np.argmax(y\_pred, axis=1)

**4. Evaluate the Model**

You can use various metrics to evaluate the model’s performance on the test set. For example, you might use accuracy, precision, recall, F1 score, or others, depending on your problem.

### python

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

# Example evaluation metrics for binary classification

accuracy = accuracy\_score(y\_test, y\_pred\_classes)

print(f'Accuracy: {accuracy:.4f}')

# Classification report and confusion matrix

print("Classification Report:")

print(classification\_report(y\_test, y\_pred\_classes))

conf\_matrix = confusion\_matrix(y\_test, y\_pred\_classes)

print("Confusion Matrix:")

print(conf\_matrix)

**Example Code Putting It All Together**

Here’s how everything might look in one code block:

### python

from tensorflow.keras.models import load\_model

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

import numpy as np

# Load the model

loaded\_model = load\_model('mlp\_model.keras')

# Prepare the test data

# Assuming X\_test is your test features and you used StandardScaler for preprocessing

scaler = StandardScaler()

X\_test = scaler.transform(X\_test)

# Make predictions

y\_pred = loaded\_model.predict(X\_test)

# Convert probabilities to class labels

y\_pred\_classes = (y\_pred > 0.5).astype(int) # Use np.argmax for multi-class

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred\_classes)

print(f'Accuracy: {accuracy:.4f}')

print("Classification Report:")

print(classification\_report(y\_test, y\_pred\_classes))

conf\_matrix = confusion\_matrix(y\_test, y\_pred\_classes)

print("Confusion Matrix:")

print(conf\_matrix)

**Notes**

* **Binary Classification**: Use a threshold of 0.5 to convert probabilities to binary class labels.
* **Multi-Class Classification**: Use np.argmax to get the class indices from the predicted probabilities.

Make sure that the test data preprocessing (e.g., scaling) matches what was applied to the training data before making predictions.