```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D


# Load the dataset
df = pd.read_csv('final_data.csv')

# Menghapus baris yang memiliki nilai yang hilang
df.dropna(inplace=True)
df
```

	CO	CO2	Kategori
0	348	105	Berbahaya
1	143	147	Tidak Sehat
2	113	28	Tidak Sehat
3	276	342	Berbahaya
4	360	117	Berbahaya
...	...	...	...
99995	190	356	Berbahaya
99996	261	34	Sangat Tidak Sehat
99997	434	57	Berbahaya
99998	283	223	Sangat Tidak Sehat
99999	306	105	Berbahaya

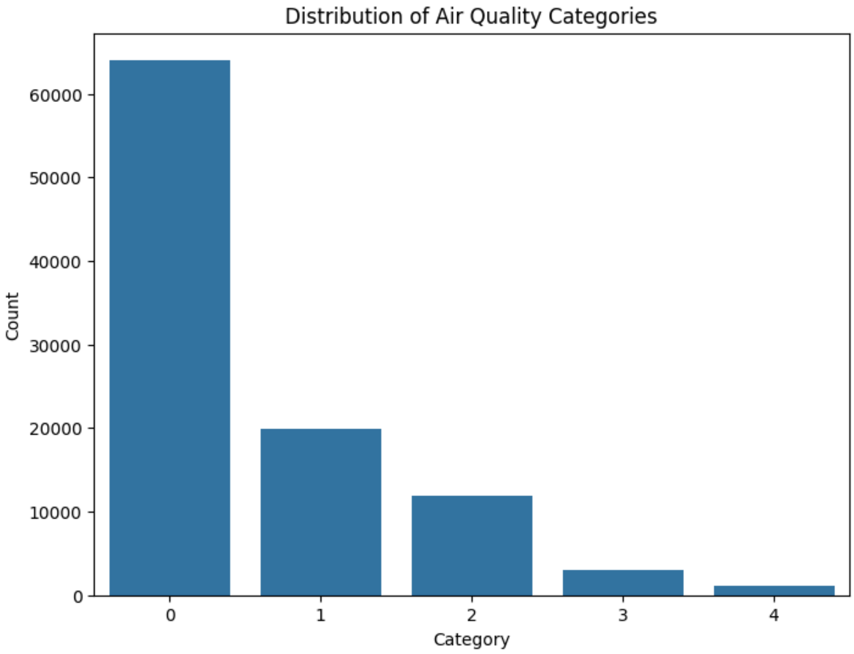
100000 rows × 3 columns

```
# Mapping kategori ke nilai numerik
category_mapping = {
    'Berbahaya': 0,
    'Sangat Tidak Sehat': 1,
    'Tidak Sehat': 2,
    'Sedang': 3,
    'Baik': 4
}
df['Kategori'] = df['Kategori'].map(category_mapping)
df
```

	CO	CO2	Kategori
0	348	105	0
1	143	147	2
2	113	28	2
3	276	342	0
4	360	117	0
...	...	...	...
99995	190	356	0
99996	261	34	1
99997	434	57	0
99998	283	223	1
99999	306	105	0

100000 rows × 3 columns

```
# Histogram Kategori Target
plt.figure(figsize=(8, 6))
sns.countplot(x='Kategori', data=df)
plt.title('Distribution of Air Quality Categories')
plt.xlabel('Category')
plt.ylabel('Count')
plt.show()
```




```
# Split fitur dan variabel target
X = df.drop(columns=['Kategori']).values
y = df['Kategori'].values

# Standarisasi fitur
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split data menjadi set pelatihan dan pengujian
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

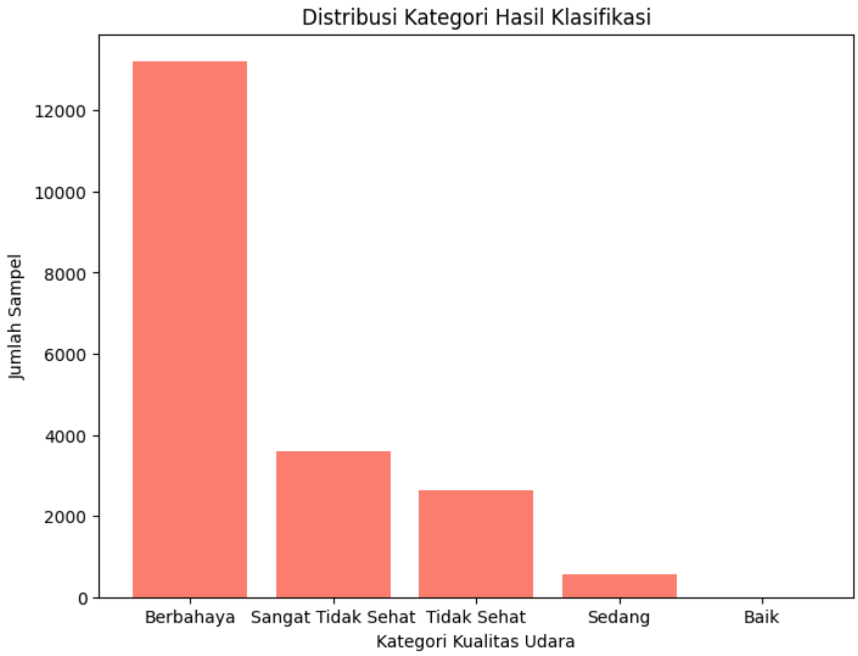
```
class PrintEpochProgress(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs=None):
        print(f'Epoch {epoch+1}/{self.params["epochs"]}, Loss: {logs["loss"]:.6f}, Accuracy: {logs["accuracy"]:.6f}, Val Loss: {logs["va
```

```
# Model Bayesian Fuzzy Classification menggunakan TensorFlow
model_bayesian_fuzzy = tf.keras.Sequential([
    tf.keras.layers.Input(shape=X_train.shape[1]),
    tf.keras.layers.Dense(5, activation='sigmoid')  # menggunakan aktivasi sigmoid untuk representasi probabilitas
])

model_bayesian_fuzzy.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Pelatihan Bayesian Fuzzy Classification
history_bayesian_fuzzy = model_bayesian_fuzzy.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2, verbose=0,callbacks=

# Evaluasi model pada data pengujian
bayesian_fuzzy_pred_prob = model_bayesian_fuzzy.predict(X_test)
bayesian_fuzzy_pred = np.argmax(bayesian_fuzzy_pred_prob, axis=1)
bayesian_fuzzy_accuracy = accuracy_score(y_test, bayesian_fuzzy_pred)

print("\nBayesian Fuzzy Classification Accuracy on Test Data:", bayesian_fuzzy_accuracy)
print("\nClassification Report for Bayesian Fuzzy Classification:")
print(classification_report(y_test, bayesian_fuzzy_pred,zero_division=1))
```

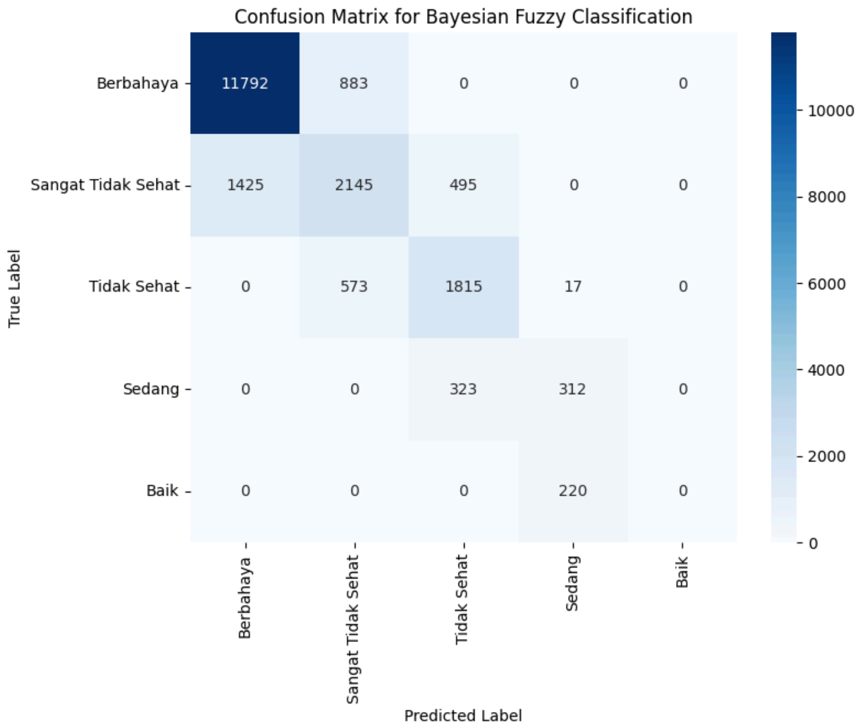
```
Epoch 1/50, Loss: 1.033519, Accuracy: 0.616313, Val Loss: 0.683064, Val Accuracy: 0.769625
Epoch 2/50, Loss: 0.620483, Accuracy: 0.780484, Val Loss: 0.580926, Val Accuracy: 0.782687
Epoch 3/50, Loss: 0.558578, Accuracy: 0.785594, Val Loss: 0.540215, Val Accuracy: 0.782500
Epoch 4/50, Loss: 0.527946, Accuracy: 0.788109, Val Loss: 0.517112, Val Accuracy: 0.783437
Epoch 5/50, Loss: 0.509126, Accuracy: 0.788625, Val Loss: 0.501724, Val Accuracy: 0.783625
Epoch 6/50, Loss: 0.496237, Accuracy: 0.789234, Val Loss: 0.490847, Val Accuracy: 0.783188
```

```
Epoch 7/50, Loss: 0.486755, Accuracy: 0.789437, Val Loss: 0.482699, Val Accuracy: 0.783188
Epoch 8/50, Loss: 0.479373, Accuracy: 0.789547, Val Loss: 0.475956, Val Accuracy: 0.783750
Epoch 9/50, Loss: 0.473346, Accuracy: 0.789922, Val Loss: 0.470928, Val Accuracy: 0.782625
Epoch 10/50, Loss: 0.468327, Accuracy: 0.789844, Val Loss: 0.466241, Val Accuracy: 0.783125
Epoch 11/50, Loss: 0.464160, Accuracy: 0.789453, Val Loss: 0.462408, Val Accuracy: 0.782250
Epoch 12/50, Loss: 0.460488, Accuracy: 0.789469, Val Loss: 0.459014, Val Accuracy: 0.782875
Epoch 13/50, Loss: 0.457285, Accuracy: 0.789781, Val Loss: 0.455907, Val Accuracy: 0.785250
Epoch 14/50, Loss: 0.454545, Accuracy: 0.789953, Val Loss: 0.453440, Val Accuracy: 0.784312
Epoch 15/50, Loss: 0.452010, Accuracy: 0.790859, Val Loss: 0.451190, Val Accuracy: 0.786563
Epoch 16/50, Loss: 0.449764, Accuracy: 0.791391, Val Loss: 0.449218, Val Accuracy: 0.785812
Epoch 17/50, Loss: 0.447728, Accuracy: 0.792625, Val Loss: 0.447566, Val Accuracy: 0.785187
Epoch 18/50, Loss: 0.445896, Accuracy: 0.792562, Val Loss: 0.445364, Val Accuracy: 0.789312
Epoch 19/50, Loss: 0.444204, Accuracy: 0.793891, Val Loss: 0.444652, Val Accuracy: 0.785438
Epoch 20/50, Loss: 0.442606, Accuracy: 0.794625, Val Loss: 0.443097, Val Accuracy: 0.786000
Epoch 21/50, Loss: 0.441216, Accuracy: 0.795156, Val Loss: 0.441389, Val Accuracy: 0.788437
Epoch 22/50, Loss: 0.439857, Accuracy: 0.796578, Val Loss: 0.440301, Val Accuracy: 0.788500
Epoch 23/50, Loss: 0.438609, Accuracy: 0.797172, Val Loss: 0.439307, Val Accuracy: 0.788000
Epoch 24/50, Loss: 0.437528, Accuracy: 0.797109, Val Loss: 0.437962, Val Accuracy: 0.791750
Epoch 25/50, Loss: 0.436444, Accuracy: 0.798141, Val Loss: 0.436691, Val Accuracy: 0.792125
Epoch 26/50, Loss: 0.435384, Accuracy: 0.798406, Val Loss: 0.435669, Val Accuracy: 0.795875
Epoch 27/50, Loss: 0.434441, Accuracy: 0.798969, Val Loss: 0.435288, Val Accuracy: 0.791438
Epoch 28/50, Loss: 0.433519, Accuracy: 0.799484, Val Loss: 0.434617, Val Accuracy: 0.791000
Epoch 29/50, Loss: 0.432647, Accuracy: 0.799609, Val Loss: 0.433283, Val Accuracy: 0.793562
Epoch 30/50, Loss: 0.431851, Accuracy: 0.800453, Val Loss: 0.433080, Val Accuracy: 0.792250
Epoch 31/50, Loss: 0.431092, Accuracy: 0.800406, Val Loss: 0.431650, Val Accuracy: 0.795750
Epoch 32/50, Loss: 0.430351, Accuracy: 0.801063, Val Loss: 0.431454, Val Accuracy: 0.792250
Epoch 33/50, Loss: 0.429648, Accuracy: 0.800500, Val Loss: 0.430340, Val Accuracy: 0.797063
Epoch 34/50, Loss: 0.429003, Accuracy: 0.801375, Val Loss: 0.429630, Val Accuracy: 0.796812
Epoch 35/50, Loss: 0.428337, Accuracy: 0.801516, Val Loss: 0.429215, Val Accuracy: 0.795313
Epoch 36/50, Loss: 0.427777, Accuracy: 0.801188, Val Loss: 0.428858, Val Accuracy: 0.795812
Epoch 37/50, Loss: 0.427152, Accuracy: 0.802422, Val Loss: 0.428388, Val Accuracy: 0.795563
Epoch 38/50, Loss: 0.426615, Accuracy: 0.802156, Val Loss: 0.427661, Val Accuracy: 0.796313
Epoch 39/50, Loss: 0.426056, Accuracy: 0.802938, Val Loss: 0.427864, Val Accuracy: 0.793125
Epoch 40/50, Loss: 0.425568, Accuracy: 0.802312, Val Loss: 0.426709, Val Accuracy: 0.797188
Epoch 41/50, Loss: 0.425029, Accuracy: 0.803094, Val Loss: 0.425989, Val Accuracy: 0.797562
Epoch 42/50, Loss: 0.424538, Accuracy: 0.803234, Val Loss: 0.426222, Val Accuracy: 0.795187
Epoch 43/50, Loss: 0.424067, Accuracy: 0.803234, Val Loss: 0.425686, Val Accuracy: 0.795812
Epoch 44/50, Loss: 0.423681, Accuracy: 0.802953, Val Loss: 0.424974, Val Accuracy: 0.797375
Epoch 45/50, Loss: 0.423261, Accuracy: 0.803187, Val Loss: 0.424455, Val Accuracy: 0.797250
Epoch 46/50, Loss: 0.422833, Accuracy: 0.803500, Val Loss: 0.424609, Val Accuracy: 0.796188
Epoch 47/50, Loss: 0.422452, Accuracy: 0.803234, Val Loss: 0.423828, Val Accuracy: 0.796812
Epoch 48/50, Loss: 0.421966, Accuracy: 0.803187, Val Loss: 0.423692, Val Accuracy: 0.796687
Epoch 49/50, Loss: 0.421668, Accuracy: 0.803766, Val Loss: 0.422903, Val Accuracy: 0.798000
Epoch 50/50, Loss: 0.421289, Accuracy: 0.803219, Val Loss: 0.422710, Val Accuracy: 0.797375
625/625 [==============================] - 1s 1ms/step

Bayesian Fuzzy Classification Accuracy on Test Data: 0.8032

Classification Report for Bayesian Fuzzy Classification:
              precision    recall  f1-score   support

           0       0.89      0.93      0.91     12675
```

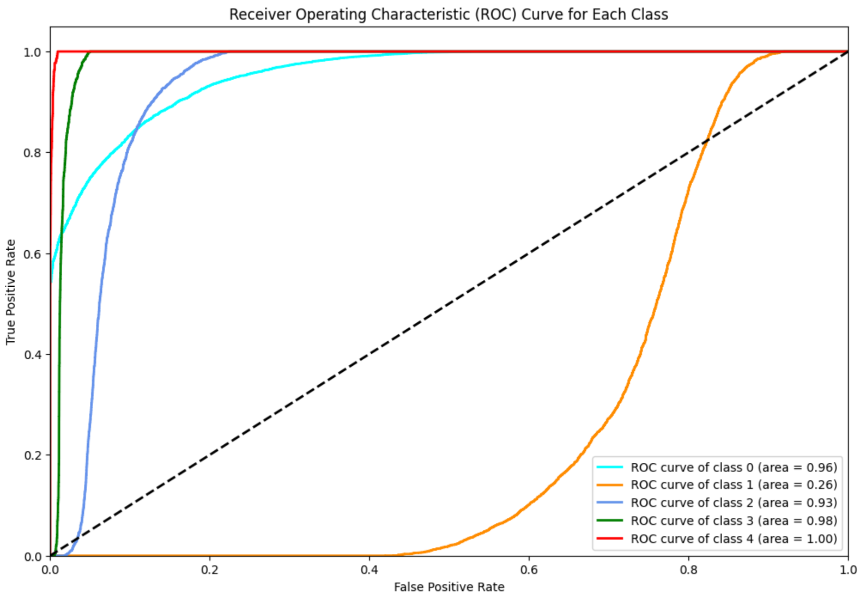
```
# Mendefinisikan label untuk kategori sesuai dengan jumlah kategori dalam model
labels = ['Berbahaya', 'Sangat Tidak Sehat', 'Tidak Sehat', 'Sedang', 'Baik']

# Memperoleh prediksi kategori dari model Bayesian Fuzzy Classification
bayesian_fuzzy_pred_category = np.argmax(bayesian_fuzzy_pred_prob, axis=1)

# Menghitung jumlah sampel yang diklasifikasikan ke setiap kategori
bayesian_fuzzy_class_counts = np.bincount(bayesian_fuzzy_pred_category, minlength=len(labels))

# Membuat diagram batang
plt.figure(figsize=(8, 6))
plt.bar(labels, bayesian_fuzzy_class_counts, color='salmon')
plt.xlabel('Kategori Kualitas Udara')
plt.ylabel('Jumlah Sampel')
plt.title('Distribusi Kategori Hasil Klasifikasi')
plt.show()
```



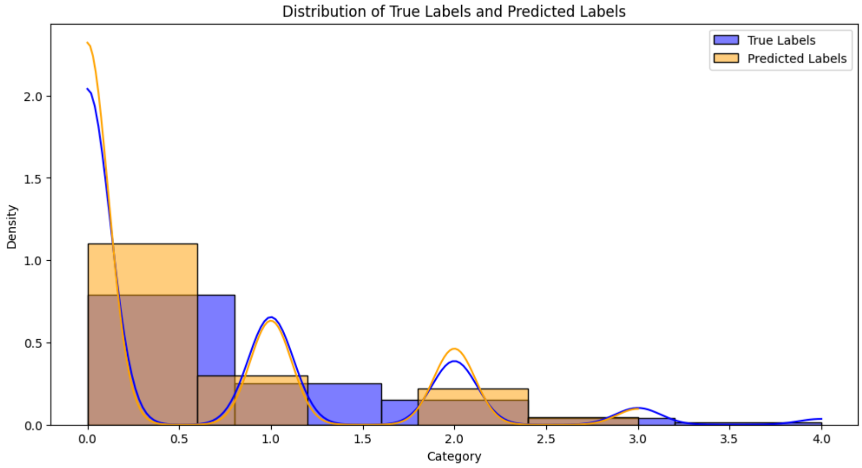

```
from sklearn.metrics import confusion_matrix, classification_report

# Mendapatkan prediksi dari model Bayesian Fuzzy Classification pada data pengujian
bayesian_fuzzy_pred = np.argmax(bayesian_fuzzy_pred_prob, axis=1)

# Membuat matriks kebingungan (confusion matrix)
conf_matrix = confusion_matrix(y_test, bayesian_fuzzy_pred)

# Visualisasi matriks kebingungan sebagai heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='d', xticklabels=category_mapping.keys(), yticklabels=category_mapping.keys())
plt.title('Confusion Matrix for Bayesian Fuzzy Classification')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

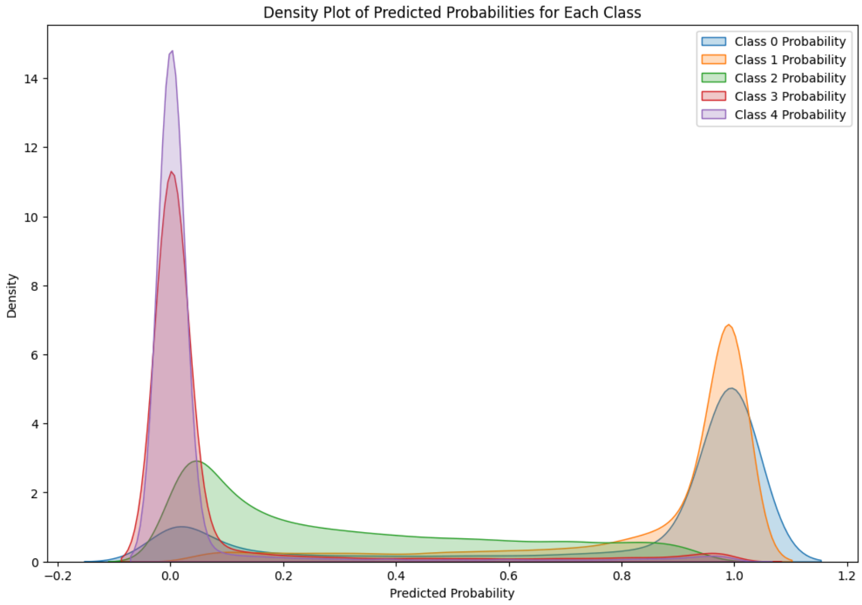


```
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize

# Binarisasi label untuk klasifikasi multikelas one-vs-rest
y_test_bin = label_binarize(y_test, classes=[0, 1, 2, 3, 4])
n_classes = y_test_bin.shape[1]

# Menghitung ROC curve dan ROC area untuk setiap kelas
fpr = dict()
tpr = dict()
roc_auc = dict()

for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], bayesian_fuzzy_pred_prob[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Plot semua ROC curves
plt.figure(figsize=(12, 8))
colors = ['aqua', 'darkorange', 'cornflowerblue', 'green', 'red']
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=2,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve for Each Class')
plt.legend(loc="lower right")
plt.show()
```

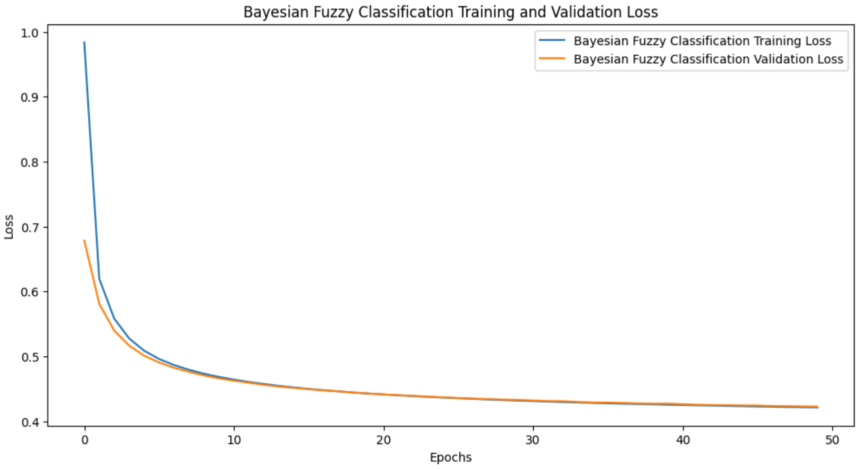


```
import seaborn as sns
import matplotlib.pyplot as plt

# Visualisasi distribusi dari prediksi dan label asli

# Histogram untuk label asli
plt.figure(figsize=(12, 6))
sns.histplot(y_test, bins=5, kde=True, color='blue', label='True Labels', stat="density", common_norm=False)
sns.histplot(bayesian_fuzzy_pred, bins=5, kde=True, color='orange', label='Predicted Labels', stat="density", common_norm=False)
plt.title('Distribution of True Labels and Predicted Labels')
plt.xlabel('Category')
plt.ylabel('Density')
plt.legend()
plt.show()

# Plot Distribusi Probabilitas Prediksi untuk setiap Kelas
plt.figure(figsize=(12, 8))
for i in range(n_classes):
    sns.kdeplot(bayesian_fuzzy_pred_prob[:, i], fill=True, label=f'Class {i} Probability')
plt.title('Density Plot of Predicted Probabilities for Each Class')
plt.xlabel('Predicted Probability')
plt.ylabel('Density')
plt.legend()
plt.show()
```





```
# Grafik Garis Loss dan Akurasi untuk Model Bayesian Fuzzy Classification
plt.figure(figsize=(12, 6))
plt.plot(history_bayesian_fuzzy.history['loss'], label='Bayesian Fuzzy Classification Training Loss')
plt.plot(history_bayesian_fuzzy.history['val_loss'], label='Bayesian Fuzzy Classification Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Bayesian Fuzzy Classification Training and Validation Loss')
plt.legend()
plt.show()
```




```
# Grafik Garis Loss dan Akurasi untuk Model Bayesian Fuzzy Classification
plt.figure(figsize=(12, 6))
plt.plot(history_bayesian_fuzzy.history['accuracy'], label='Bayesian Fuzzy Classification Training Accuracy')
plt.plot(history_bayesian_fuzzy.history['val_accuracy'], label='Bayesian Fuzzy Classification Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Metrics')
plt.title('Bayesian Fuzzy Classification Training Metrics')
plt.legend()
plt.show()
```