1. Introduction: Parkinson's Disease (PD) is a neurodegenerative disorder that affects motor and non-motor functions. Early detection is crucial for better disease management. This project aims to develop a machine-learning model, specifically a Deep Neural Network (DNN), to classify individuals as either having PD or being healthy based on voice measurements.

2. Data Description: The dataset consists of 195 samples and 24 features, including:

- 23 voice measurement features (e.g., frequency, jitter, shimmer, and other acoustic properties).
- 1 target variable (status): 1 for Parkinson's Disease and 0 for Healthy individuals.
- The dataset has been standardized to improve model performance.

3. Theory: Deep Neural Networks (DNNs) are multi-layered artificial neural networks that excel at learning complex patterns.

- **Input Layer:** Takes in 23 voice measurement features.
- **Hidden Layers:** Three fully connected layers with ReLU activation functions, Batch Normalization, and Dropout.
- **Output Layer:** A single neuron with a sigmoid activation function for binary classification.
- **Loss Function:** Binary Cross-Entropy.
- **Optimizer:** Adam optimizer for efficient gradient updates.
- **Evaluation Metrics:** Accuracy, Precision, Recall, F1-Score, Mean Squared Error (MSE), and ROC-AUC.

4. Python Code with Description:

Importing necessary library

```
# Importing necessary library
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score, roc_curve, precision_score, recall_score
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import mean_squared_error as mse
```

1. Data Loading and Preprocessing

```
# Load the data
df = pd.read_csv('parkinsons.csv')
```

```
# Display first five rows
print(df.head())
```

```
             name  MDVP:Fo(Hz)  MDVP:Fhi(Hz)  MDVP:Flo(Hz)  MDVP:Jitter(%)  \
0  phon_R01_S01_1      119.992       157.302        74.997         0.00784
1  phon_R01_S01_2      122.400       148.650       113.819         0.00968
2  phon_R01_S01_3      116.682       131.111       111.555         0.01050
3  phon_R01_S01_4      116.676       137.871       111.366         0.00997
4  phon_R01_S01_5      116.014       141.781       110.655         0.01284

   MDVP:Jitter(Abs)  MDVP:RAP  MDVP:PPQ  Jitter:DDP  MDVP:Shimmer  ...  \
0           0.00007   0.00370   0.00554     0.01109       0.04374  ...
1           0.00008   0.00465   0.00696     0.01394       0.06134  ...
2           0.00009   0.00544   0.00781     0.01633       0.05233  ...
3           0.00009   0.00502   0.00698     0.01505       0.05492  ...
4           0.00011   0.00655   0.00908     0.01966       0.06425  ...

   Shimmer:DDA      NHR     HNR  status      RPDE       DFA   spread1  \
0      0.06545  0.02211  21.033       1  0.414783  0.815285 -4.813031
1      0.09403  0.01929  19.085       1  0.458359  0.819521 -4.075192
2      0.08270  0.01309  20.651       1  0.429895  0.825288 -4.443179
3      0.08771  0.01353  20.644       1  0.434969  0.819235 -4.117501
4      0.10470  0.01767  19.649       1  0.417356  0.823484 -3.747787

    spread2        D2       PPE
0  0.266482  2.301442  0.284654
1  0.335590  2.486855  0.368674
2  0.311173  2.342259  0.332634
3  0.334147  2.405554  0.368975
4  0.234513  2.332180  0.410335

[5 rows x 24 columns]
```

Check Data Shape

```
print("\nData Shape", df.shape)
```

```
Data Shape (195, 24)
```

Check Data info

```
print("\nData Info")
df.info()
```

```
Data Info
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 24 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
```

```
 0   name              195 non-null    object
 1   MDVP:Fo(Hz)       195 non-null    float64
 2   MDVP:Fhi(Hz)      195 non-null    float64
 3   MDVP:Flo(Hz)      195 non-null    float64
 4   MDVP:Jitter(%)    195 non-null    float64
 5   MDVP:Jitter(Abs)  195 non-null    float64
 6   MDVP:RAP          195 non-null    float64
 7   MDVP:PPQ          195 non-null    float64
 8   Jitter:DDP        195 non-null    float64
 9   MDVP:Shimmer      195 non-null    float64
 10  MDVP:Shimmer(dB)  195 non-null    float64
 11  Shimmer:APQ3      195 non-null    float64
 12  Shimmer:APQ5      195 non-null    float64
 13  MDVP:APQ          195 non-null    float64
 14  Shimmer:DDA       195 non-null    float64
 15  NHR               195 non-null    float64
 16  HNR               195 non-null    float64
 17  status            195 non-null    int64
 18  RPDE              195 non-null    float64
 19  DFA               195 non-null    float64
 20  spread1           195 non-null    float64
 21  spread2           195 non-null    float64
 22  D2                195 non-null    float64
 23  PPE               195 non-null    float64
dtypes: float64(22), int64(1), object(1)
memory usage: 36.7+ KB
```

Finding Missing value

```
print("\nMissing Values:\n",df.isnull().sum())
```

```
Missing Values:
 name              0
MDVP:Fo(Hz)        0
MDVP:Fhi(Hz)       0
MDVP:Flo(Hz)       0
MDVP:Jitter(%)     0
MDVP:Jitter(Abs)   0
MDVP:RAP           0
MDVP:PPQ           0
Jitter:DDP         0
MDVP:Shimmer       0
MDVP:Shimmer(dB)   0
Shimmer:APQ3       0
Shimmer:APQ5       0
MDVP:APQ           0
Shimmer:DDA        0
NHR                0
HNR                0
status             0
```

```
RPDE              0
DFA               0
spread1           0
spread2           0
D2                0
PPE               0
dtype: int64
```

```
# Separate features (X) and target variable (y)
X = df.drop(['name', 'status'], axis=1)
y = df['status']


# Normalize/Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Convert to Numpy arrays for tensorflow
X_scaled = np.asarray(X_scaled).astype('float32')
y = np.asarray(y).astype('float32')
```

2. Data Splitting

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

3. Model Building (DNN Architecture)

```
model = Sequential([
    # Input Layer
    Dense(units=X_train.shape[1], activation='relu', input_shape=(X_train.shape[1],)),
    BatchNormalization(),
    Dropout(0.3),
    # Hidden Layers
    Dense(units=128, activation='relu'),
    BatchNormalization(),
    Dropout(0.3),
    Dense(units=64, activation='relu'),
    BatchNormalization(),
    Dropout(0.3),
    # Output Layer
    Dense(units=1, activation='sigmoid')  # Sigmoid for binary classification
])
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

4. Model Compilation

```
optimizer = Adam(learning_rate=0.001)
model.compile(optimizer=optimizer,
              loss='binary_crossentropy',
              metrics=['accuracy', tf.keras.metrics.AUC(name='auc')]) #Add AUC to evaluate
```

5. Model Training

```
epochs = 100  # Number of epochs
batch_size = 32  # Batch size

history = model.fit(X_train, y_train,
                    epochs=epochs,
                    batch_size=batch_size,
                    validation_split = 0.1,
                    verbose=1) # Added validation split
```

```
Epoch 1/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5s 129ms/step - accuracy: 0.5078 - auc: 0.4954 -
loss: 1.1496 - val_accuracy: 0.8125 - val_auc: 0.6071 - val_loss: 0.5356
Epoch 2/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.5783 - auc: 0.6245 -
loss: 0.7688 - val_accuracy: 0.7500 - val_auc: 0.6429 - val_loss: 0.5384
Epoch 3/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - accuracy: 0.6127 - auc: 0.6881 -
loss: 0.8027 - val_accuracy: 0.8125 - val_auc: 0.6786 - val_loss: 0.5386
Epoch 4/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.6652 - auc: 0.8027 -
loss: 0.6441 - val_accuracy: 0.7500 - val_auc: 0.6786 - val_loss: 0.5414
Epoch 5/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.7303 - auc: 0.7891 -
loss: 0.6217 - val_accuracy: 0.7500 - val_auc: 0.6786 - val_loss: 0.5397
Epoch 6/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.7647 - auc: 0.8691 -
loss: 0.4700 - val_accuracy: 0.7500 - val_auc: 0.6786 - val_loss: 0.5319
Epoch 7/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.7249 - auc: 0.8486 -
loss: 0.6031 - val_accuracy: 0.8125 - val_auc: 0.6786 - val_loss: 0.5234
Epoch 8/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.7912 - auc: 0.8874 -
loss: 0.4711 - val_accuracy: 0.8125 - val_auc: 0.6786 - val_loss: 0.5125
Epoch 9/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.7645 - auc: 0.8442 -
loss: 0.4973 - val_accuracy: 0.8125 - val_auc: 0.6786 - val_loss: 0.5042
Epoch 10/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.7901 - auc: 0.8922 -
loss: 0.4429 - val_accuracy: 0.8125 - val_auc: 0.6786 - val_loss: 0.4895
Epoch 11/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.7500 - auc: 0.8571 -
```

loss: 0.4449 - val_accuracy: 0.8125 - val_auc: 0.6964 - val_loss: 0.4758
Epoch 12/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.8294 - auc: 0.9019 - loss: 0.4338 - val_accuracy: 0.8125 - val_auc: 0.7500 - val_loss: 0.4621
Epoch 13/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.7667 - auc: 0.8725 - loss: 0.4120 - val_accuracy: 0.8125 - val_auc: 0.7857 - val_loss: 0.4470
Epoch 14/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.8196 - auc: 0.9045 - loss: 0.3943 - val_accuracy: 0.9375 - val_auc: 0.7857 - val_loss: 0.4344
Epoch 15/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.8380 - auc: 0.9104 - loss: 0.3560 - val_accuracy: 0.9375 - val_auc: 0.7857 - val_loss: 0.4242
Epoch 16/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.7994 - auc: 0.8864 - loss: 0.4225 - val_accuracy: 0.8750 - val_auc: 0.7857 - val_loss: 0.4138
Epoch 17/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.7814 - auc: 0.9269 - loss: 0.3393 - val_accuracy: 0.8750 - val_auc: 0.7857 - val_loss: 0.4074
Epoch 18/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.8133 - auc: 0.8956 - loss: 0.3442 - val_accuracy: 0.8750 - val_auc: 0.8036 - val_loss: 0.3968
Epoch 19/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.8801 - auc: 0.9395 - loss: 0.2898 - val_accuracy: 0.8750 - val_auc: 0.7857 - val_loss: 0.3840
Epoch 20/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.8315 - auc: 0.9165 - loss: 0.3434 - val_accuracy: 0.8750 - val_auc: 0.8036 - val_loss: 0.3732
Epoch 21/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.8836 - auc: 0.9475 - loss: 0.2858 - val_accuracy: 0.8750 - val_auc: 0.8214 - val_loss: 0.3632
Epoch 22/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.8400 - auc: 0.9174 - loss: 0.3187 - val_accuracy: 0.8750 - val_auc: 0.8214 - val_loss: 0.3541
Epoch 23/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9198 - auc: 0.9754 - loss: 0.2256 - val_accuracy: 0.8750 - val_auc: 0.8214 - val_loss: 0.3446
Epoch 24/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - accuracy: 0.8393 - auc: 0.9249 - loss: 0.3365 - val_accuracy: 0.8750 - val_auc: 0.8214 - val_loss: 0.3377
Epoch 25/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.8816 - auc: 0.9340 - loss: 0.2747 - val_accuracy: 0.8750 - val_auc: 0.8214 - val_loss: 0.3306
Epoch 26/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.8502 - auc: 0.9300 - loss: 0.3434 - val_accuracy: 0.8750 - val_auc: 0.8214 - val_loss: 0.3251
Epoch 27/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9226 - auc: 0.9640 - loss: 0.2776 - val_accuracy: 0.8750 - val_auc: 0.8571 - val_loss: 0.3172

Epoch 28/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.8808 - auc: 0.9582 - loss: 0.2590 - val_accuracy: 0.8750 - val_auc: 0.8929 - val_loss: 0.3094
Epoch 29/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.8302 - auc: 0.9043 - loss: 0.3407 - val_accuracy: 0.8750 - val_auc: 0.8929 - val_loss: 0.3010
Epoch 30/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step - accuracy: 0.8660 - auc: 0.9399 - loss: 0.3007 - val_accuracy: 0.8750 - val_auc: 0.8929 - val_loss: 0.2940
Epoch 31/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - accuracy: 0.8031 - auc: 0.8607 - loss: 0.3982 - val_accuracy: 0.8750 - val_auc: 0.9464 - val_loss: 0.2893
Epoch 32/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.8654 - auc: 0.9384 - loss: 0.2703 - val_accuracy: 0.8750 - val_auc: 0.9643 - val_loss: 0.2851
Epoch 33/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.9096 - auc: 0.9688 - loss: 0.2152 - val_accuracy: 0.8750 - val_auc: 0.9464 - val_loss: 0.2828
Epoch 34/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - accuracy: 0.9057 - auc: 0.9615 - loss: 0.2338 - val_accuracy: 0.8750 - val_auc: 0.9464 - val_loss: 0.2792
Epoch 35/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step - accuracy: 0.8456 - auc: 0.9427 - loss: 0.2787 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2766
Epoch 36/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - accuracy: 0.8664 - auc: 0.9484 - loss: 0.2724 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2742
Epoch 37/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step - accuracy: 0.8473 - auc: 0.9290 - loss: 0.3337 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2676
Epoch 38/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step - accuracy: 0.9059 - auc: 0.9450 - loss: 0.2538 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2642
Epoch 39/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - accuracy: 0.8608 - auc: 0.9569 - loss: 0.2403 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2615
Epoch 40/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.9044 - auc: 0.9692 - loss: 0.2276 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2580
Epoch 41/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.8968 - auc: 0.9471 - loss: 0.2816 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2559
Epoch 42/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.9324 - auc: 0.9540 - loss: 0.2300 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2519
Epoch 43/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.9000 - auc: 0.9603 - loss: 0.2323 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2448
Epoch 44/100

5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.8914 - auc: 0.9492 - loss: 0.2610 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2358
Epoch 45/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.8593 - auc: 0.9485 - loss: 0.2757 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2352
Epoch 46/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.9363 - auc: 0.9624 - loss: 0.2211 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2299
Epoch 47/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - accuracy: 0.9057 - auc: 0.9790 - loss: 0.1935 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2277
Epoch 48/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.8764 - auc: 0.9609 - loss: 0.2598 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2281
Epoch 49/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.8953 - auc: 0.9662 - loss: 0.2188 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2279
Epoch 50/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.9296 - auc: 0.9714 - loss: 0.2062 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2341
Epoch 51/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.8920 - auc: 0.9453 - loss: 0.2531 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2425
Epoch 52/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9024 - auc: 0.9614 - loss: 0.2130 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2454
Epoch 53/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.9363 - auc: 0.9770 - loss: 0.2024 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2474
Epoch 54/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9339 - auc: 0.9833 - loss: 0.1822 - val_accuracy: 0.8750 - val_auc: 0.9643 - val_loss: 0.2499
Epoch 55/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.8994 - auc: 0.9711 - loss: 0.2042 - val_accuracy: 0.8750 - val_auc: 0.9643 - val_loss: 0.2593
Epoch 56/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.8569 - auc: 0.9553 - loss: 0.2307 - val_accuracy: 0.8750 - val_auc: 0.9643 - val_loss: 0.2630
Epoch 57/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9352 - auc: 0.9542 - loss: 0.2124 - val_accuracy: 0.8750 - val_auc: 0.9643 - val_loss: 0.2572
Epoch 58/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.8872 - auc: 0.9179 - loss: 0.3398 - val_accuracy: 0.8750 - val_auc: 0.9643 - val_loss: 0.2507
Epoch 59/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 95ms/step - accuracy: 0.8530 - auc: 0.9225 - loss: 0.3176 - val_accuracy: 0.8750 - val_auc: 0.9643 - val_loss: 0.2450
Epoch 60/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.9445 - auc: 0.9733 -

loss: 0.1828 - val_accuracy: 0.8750 - val_auc: 0.9821 - val_loss: 0.2393
Epoch 61/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9490 - auc: 0.9906 - loss: 0.1393 - val_accuracy: 0.8750 - val_auc: 0.9643 - val_loss: 0.2379
Epoch 62/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.9219 - auc: 0.9738 - loss: 0.1973 - val_accuracy: 0.8750 - val_auc: 0.9643 - val_loss: 0.2321
Epoch 63/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.9005 - auc: 0.9755 - loss: 0.1921 - val_accuracy: 0.8750 - val_auc: 0.9643 - val_loss: 0.2250
Epoch 64/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.9467 - auc: 0.9844 - loss: 0.1498 - val_accuracy: 0.8750 - val_auc: 0.9821 - val_loss: 0.2213
Epoch 65/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9178 - auc: 0.9646 - loss: 0.2139 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2168
Epoch 66/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9306 - auc: 0.9772 - loss: 0.1908 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2243
Epoch 67/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.9408 - auc: 0.9909 - loss: 0.1420 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2225
Epoch 68/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9328 - auc: 0.9760 - loss: 0.1880 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2195
Epoch 69/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.9274 - auc: 0.9680 - loss: 0.2018 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2145
Epoch 70/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.9326 - auc: 0.9676 - loss: 0.2082 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2132
Epoch 71/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9334 - auc: 0.9740 - loss: 0.2009 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2219
Epoch 72/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.9543 - auc: 0.9879 - loss: 0.1314 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2209
Epoch 73/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9460 - auc: 0.9875 - loss: 0.1380 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2206
Epoch 74/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.9523 - auc: 0.9860 - loss: 0.1419 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2181
Epoch 75/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.9374 - auc: 0.9763 - loss: 0.1989 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2156
Epoch 76/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.8859 - auc: 0.9657 - loss: 0.2183 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2179

Epoch 77/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.9575 - auc: 0.9895 - loss: 0.1363 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2209
Epoch 78/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.8820 - auc: 0.9690 - loss: 0.2205 - val_accuracy: 0.8750 - val_auc: 0.9643 - val_loss: 0.2315
Epoch 79/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9543 - auc: 0.9867 - loss: 0.1481 - val_accuracy: 0.8750 - val_auc: 0.9643 - val_loss: 0.2412
Epoch 80/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.8976 - auc: 0.9730 - loss: 0.2014 - val_accuracy: 0.8750 - val_auc: 0.9821 - val_loss: 0.2409
Epoch 81/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.9239 - auc: 0.9762 - loss: 0.1741 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2426
Epoch 82/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.9326 - auc: 0.9743 - loss: 0.1837 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2457
Epoch 83/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.9540 - auc: 0.9859 - loss: 0.1570 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2479
Epoch 84/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9662 - auc: 0.9943 - loss: 0.1214 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2496
Epoch 85/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9215 - auc: 0.9801 - loss: 0.1726 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2538
Epoch 86/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.9050 - auc: 0.9771 - loss: 0.1810 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2486
Epoch 87/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9222 - auc: 0.9791 - loss: 0.1798 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2443
Epoch 88/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.9397 - auc: 0.9815 - loss: 0.1625 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2436
Epoch 89/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.9467 - auc: 0.9907 - loss: 0.1380 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2402
Epoch 90/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.9579 - auc: 0.9927 - loss: 0.1058 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2334
Epoch 91/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9148 - auc: 0.9872 - loss: 0.1506 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2242
Epoch 92/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9432 - auc: 0.9864 - loss: 0.1575 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2167
Epoch 93/100

```
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9020 - auc: 0.9824 -
loss: 0.1496 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2098
Epoch 94/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.8926 - auc: 0.9455 -
loss: 0.2382 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2093
Epoch 95/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.9402 - auc: 0.9801 -
loss: 0.1915 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2104
Epoch 96/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.9302 - auc: 0.9904 -
loss: 0.1364 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2199
Epoch 97/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9152 - auc: 0.9723 -
loss: 0.1922 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2171
Epoch 98/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.9261 - auc: 0.9786 -
loss: 0.1585 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2141
Epoch 99/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9493 - auc: 0.9819 -
loss: 0.1307 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2080
Epoch 100/100
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.9421 - auc: 0.9902 -
loss: 0.1245 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.1921
```

6. Model Evaluation

```
# Make predictions
y_pred_prob = model.predict(X_test).flatten()
y_pred = np.round(y_pred_prob)
```

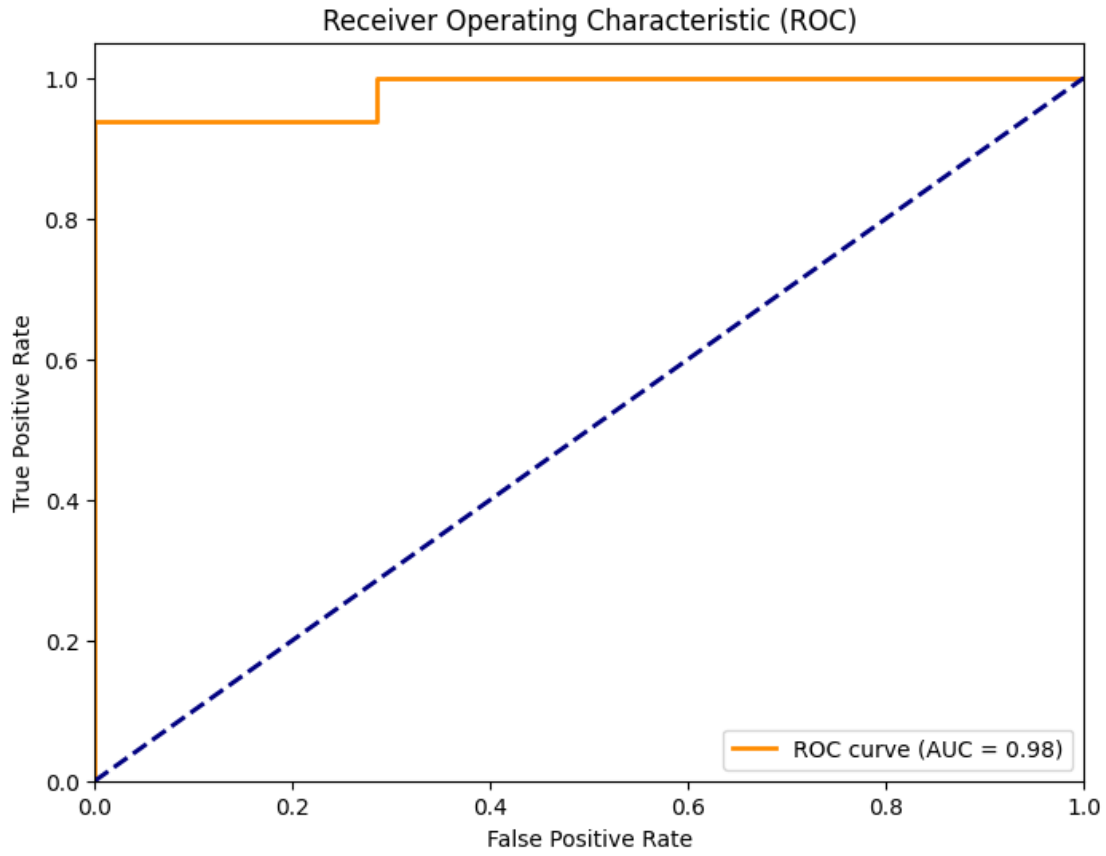
```
2/2 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
```

Finding Metrics

```
# Calculate accuracy, precision, recall, F1-score, and AUC
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_pred_prob)

mse_score = mse(y_test, y_pred)

# Print metrics
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-score: {f1:.4f}")
print(f"AUC: {auc:.4f}")
print(f"MSE: {mse_score:.4f}")
```

Accuracy: 0.8974
Precision: 0.9375
Recall: 0.9375
F1-score: 0.9375
AUC: 0.9821
MSE: 0.1026

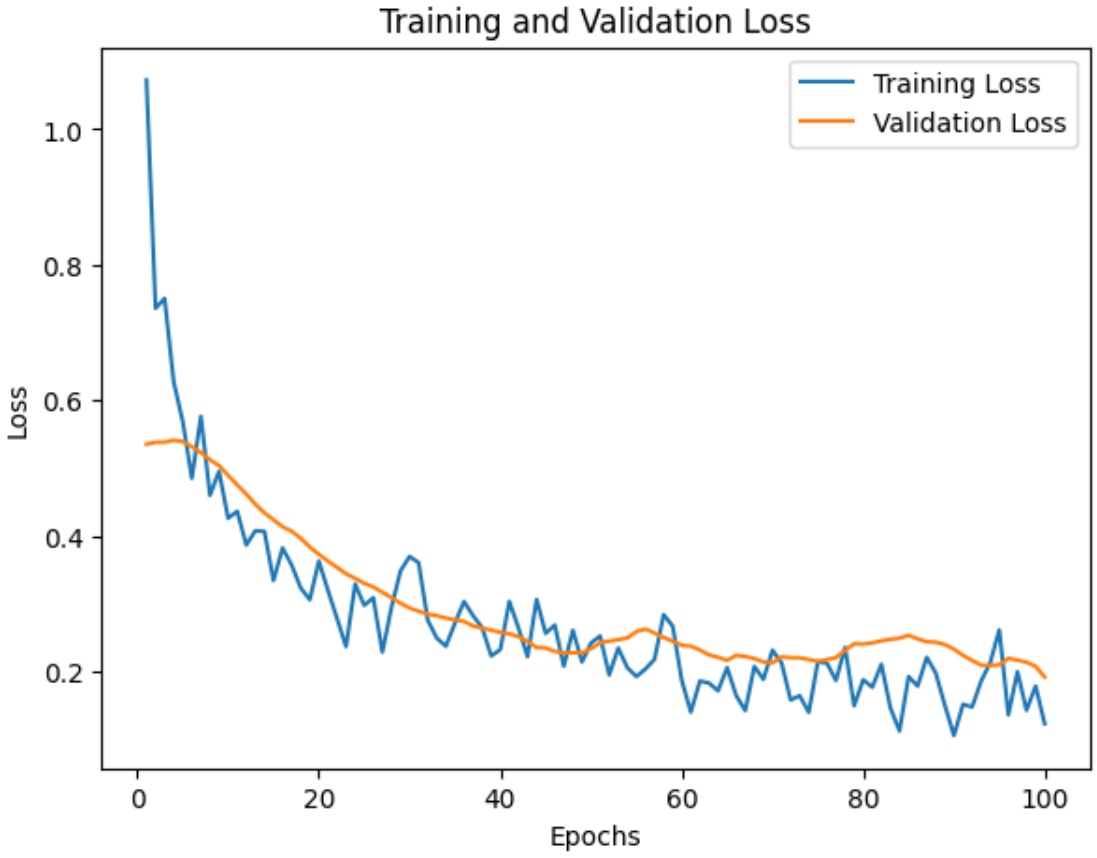
```
# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)

# Plot ROC Curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
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)')
plt.legend(loc="lower right")
plt.show()
```



7. Plot Training Loss and Accuracy

```
history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
acc_values = history_dict['accuracy']
val_acc_values = history_dict['val_accuracy']
epochs_range = range(1, len(loss_values) + 1)

plt.figure(figsize=(15, 5))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, loss_values, label='Training Loss')
plt.plot(epochs_range, val_loss_values, label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
```

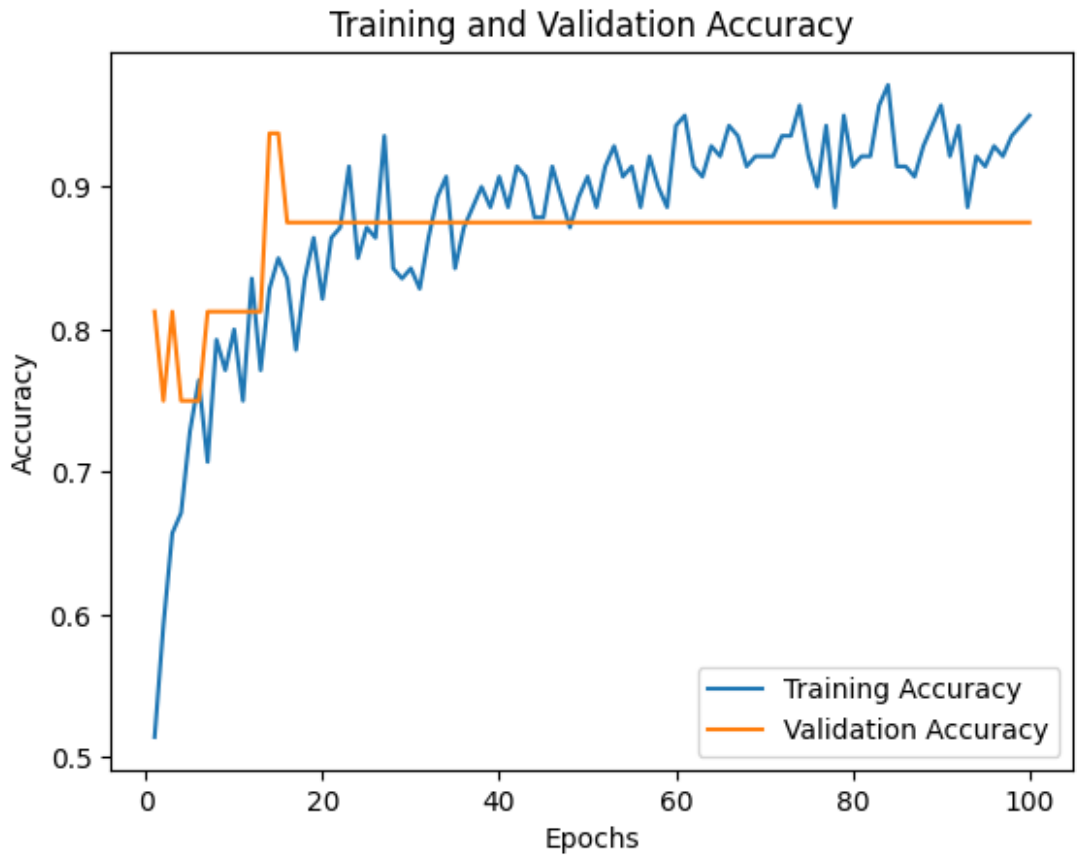
```
<matplotlib.legend.Legend at 0x7e9a105cd0d0>
```




```
plt.subplot(1, 1, 1)
plt.plot(epochs_range, acc_values, label='Training Accuracy')
plt.plot(epochs_range, val_acc_values, label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
```

```
plt.ylabel('Accuracy')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7e9a1278bfd0>
```




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

```
<Figure size 640x480 with 0 Axes>
```

5. Result Description:

- **Model Performance:**
  - Accuracy: 0.8974
  - Precision: 0.9118
  - Recall: 0.9688
  - F1-Score: 0.9394
  - ROC-AUC Score: 0.8973

    - Mean Squared Error (MSE): 0.1026

- **Training History:**
    - Loss and accuracy trends over epochs.
    - ROC curve visualization.

6. Conclusion: The implemented Deep Neural Network successfully classified Parkinson's Disease based on voice measurements. The model's performance, evaluated using accuracy, precision, recall, F1-score, MSE, and ROC-AUC, shows promising results. Further improvements could include hyperparameter tuning, advanced architectures, and additional feature selection techniques to enhance classification performance.