```
# Import necessary libraries
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
from scipy.signal import butter, lfilter
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.applications import VGG16
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint


from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import pandas as pd
from sklearn.preprocessing import StandardScaler, LabelEncoder
import matplotlib.pyplot as plt
import seaborn as sns

# Adjust this path to your file's actual location
file_path = '/content/drive/MyDrive/datsaforones/S1_E2_A1.csv'
data = pd.read_csv(file_path)

print("First few rows of the dataset:")
print(data.head())

print("\nDataset information:")
print(data.info())
```

```
 67  18.1    1281644 non-null  float64
 68  19.1    1281644 non-null  float64
 69  20.1    1281644 non-null  float64
 70  21.1    1281644 non-null  float64
 71  0.4     1281644 non-null  float64
 72  1.3     1281644 non-null  float64
 73  0.5     1 non-null        float64
 74  0.6     1 non-null        float64
 75  0.7     1281644 non-null  int64
 76  0.8     1281644 non-null  int64
 77  0.9     1281644 non-null  int64
dtypes: float64(74), int64(4)
memory usage: 762.7 MB
None
```

```
data.info()
```

```
 22  10.1    1281644 non-null  float64
 23  11.1    1281644 non-null  float64
 24  12      1281644 non-null  float64
 25  13      1281644 non-null  float64
 26  14      1281644 non-null  float64
 27  15      1281644 non-null  float64
 28  16      1281644 non-null  float64
 29  17      1281644 non-null  float64
 30  18      1281644 non-null  float64
 31  19      1281644 non-null  float64
 32  20      1281644 non-null  float64
 33  21      1281644 non-null  float64
 34  22      1281644 non-null  float64
 35  23      1281644 non-null  float64
 36  24      1281644 non-null  float64
 37  25      1281644 non-null  float64
 38  26      1281644 non-null  float64
 39  27      1281644 non-null  float64
 40  28      1281644 non-null  float64
 41  29      1281644 non-null  float64
 42  30      1281644 non-null  float64
 43  31      1281644 non-null  float64
 44  32      1281644 non-null  float64
 45  33      1281644 non-null  float64
 46  34      1281644 non-null  float64
 47  35      1281644 non-null  float64
 48  0.2     1281644 non-null  int64
 49  0.3     1281644 non-null  float64
 50  1.2     1281644 non-null  float64
 51  2.2     1281644 non-null  float64
 52  3.2     1281644 non-null  float64
 53  4.2     1281644 non-null  float64
 54  5.2     1281644 non-null  float64
 55  6.2     1281644 non-null  float64
 56  7.2     1281644 non-null  float64
 57  8.2     1281644 non-null  float64
 58  9.2     1281644 non-null  float64
 59  10.2    1281644 non-null  float64
 60  11.2    1281644 non-null  float64
 61  12.1    1281644 non-null  float64
 62  13.1    1281644 non-null  float64
 63  14.1    1281644 non-null  float64
 64  15.1    1281644 non-null  float64
 65  16.1    1281644 non-null  float64
 66  17.1    1281644 non-null  float64
 67  18.1    1281644 non-null  float64
 68  19.1    1281644 non-null  float64
 69  20.1    1281644 non-null  float64
 70  21.1    1281644 non-null  float64
 71  0.4     1281644 non-null  float64
 72  1.3     1281644 non-null  float64
 73  0.5     1 non-null        float64
 74  0.6     1 non-null        float64
 75  0.7     1281644 non-null  int64
 76  0.8     1281644 non-null  int64
 77  0.9     1281644 non-null  int64
dtypes: float64(74), int64(4)
memory usage: 762.7 MB
```

```
# Preprocess the data by removing all NaN values
data_cleaned = data.fillna(method='ffill')

print("\nFirst few rows of the cleaned dataset:")
print(data_cleaned.head())

print("\nCleaned dataset information:")
print(data_cleaned.info())
```

```
 23  11.1    1281644 non-null  float64
 24  12      1281644 non-null  float64
 25  13      1281644 non-null  float64
 26  14      1281644 non-null  float64
 27  15      1281644 non-null  float64
 28  16      1281644 non-null  float64
 29  17      1281644 non-null  float64
 30  18      1281644 non-null  float64
 31  19      1281644 non-null  float64
 32  20      1281644 non-null  float64
 33  21      1281644 non-null  float64
 34  22      1281644 non-null  float64
 35  23      1281644 non-null  float64
 36  24      1281644 non-null  float64
 37  25      1281644 non-null  float64
 38  26      1281644 non-null  float64
 39  27      1281644 non-null  float64
 40  28      1281644 non-null  float64
 41  29      1281644 non-null  float64
 42  30      1281644 non-null  float64
 43  31      1281644 non-null  float64
 44  32      1281644 non-null  float64
 45  33      1281644 non-null  float64
 46  34      1281644 non-null  float64
 47  35      1281644 non-null  float64
 48  0.2     1281644 non-null  int64
 49  0.3     1281644 non-null  float64
 50  1.2     1281644 non-null  float64
 51  2.2     1281644 non-null  float64
 52  3.2     1281644 non-null  float64
 53  4.2     1281644 non-null  float64
 54  5.2     1281644 non-null  float64
 55  6.2     1281644 non-null  float64
 56  7.2     1281644 non-null  float64
 57  8.2     1281644 non-null  float64
 58  9.2     1281644 non-null  float64
 59  10.2    1281644 non-null  float64
 60  11.2    1281644 non-null  float64
 61  12.1    1281644 non-null  float64
 62  13.1    1281644 non-null  float64
 63  14.1    1281644 non-null  float64
 64  15.1    1281644 non-null  float64
 65  16.1    1281644 non-null  float64
 66  17.1    1281644 non-null  float64
 67  18.1    1281644 non-null  float64
 68  19.1    1281644 non-null  float64
 69  20.1    1281644 non-null  float64
 70  21.1    1281644 non-null  float64
 71  0.4     1281644 non-null  float64
 72  1.3     1281644 non-null  float64
 73  0.5     1281644 non-null  float64
 74  0.6     1281644 non-null  float64
 75  0.7     1281644 non-null  int64
 76  0.8     1281644 non-null  int64
 77  0.9     1281644 non-null  int64
dtypes: float64(74), int64(4)
memory usage: 762.7 MB
None
```

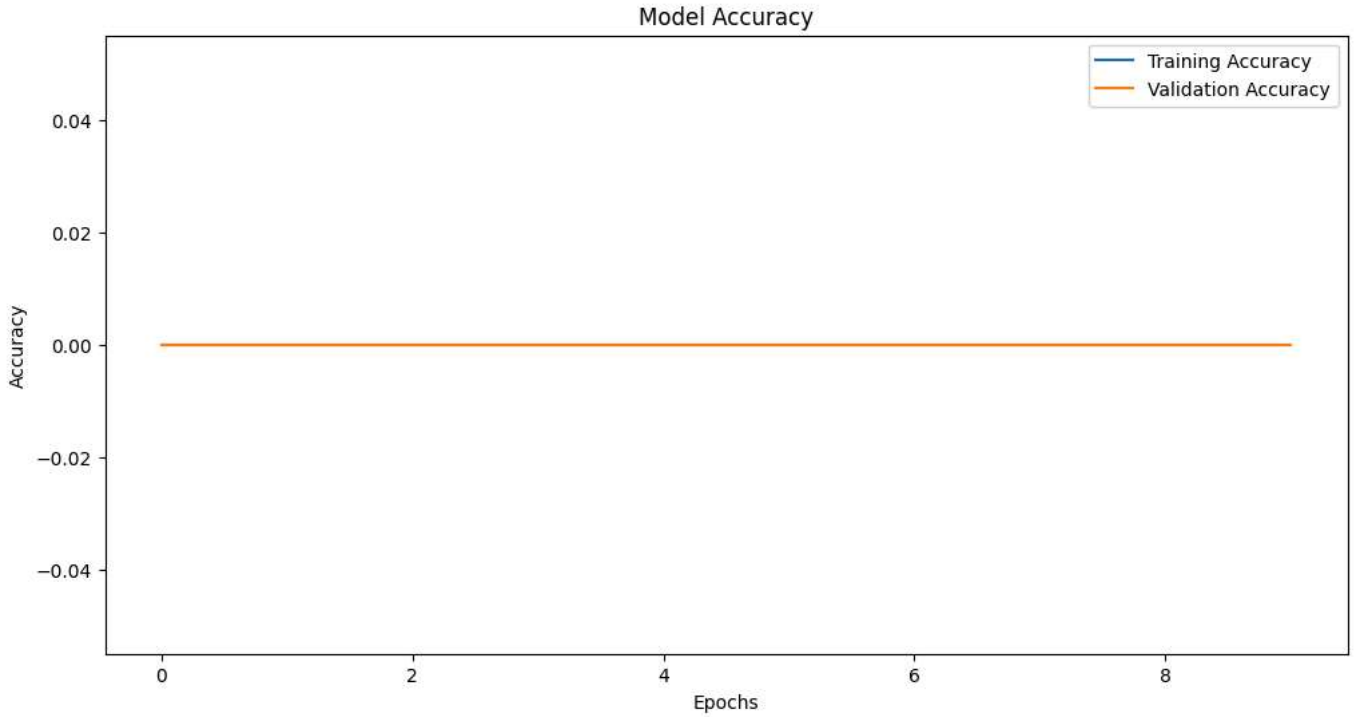
```
# Check if the cleaned dataset is empty
if data_cleaned.empty:
    raise ValueError("The cleaned dataset is empty. Please check your data and preprocessing steps.")

# Encode categorical columns if any exist
if 'Category' in data_cleaned.columns:
    label_encoder = LabelEncoder()
    data_cleaned['Category'] = label_encoder.fit_transform(data_cleaned['Category'])
    print("\nCategorical columns encoded.")

# Select numerical columns for scaling
numerical_columns = data_cleaned.select_dtypes(include=['float64', 'int64']).columns
scaler = StandardScaler()

# Check if there are any numerical columns left to scale
if len(numerical_columns) > 0:
    data_cleaned[numerical_columns] = scaler.fit_transform(data_cleaned[numerical_columns])
    print("\nNumerical columns normalized or scaled.")
else:
    print("No numerical columns to scale.")
```

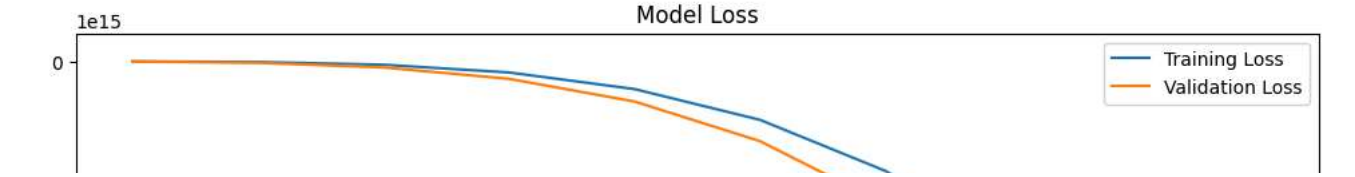
```
Numerical columns normalized or scaled.
```

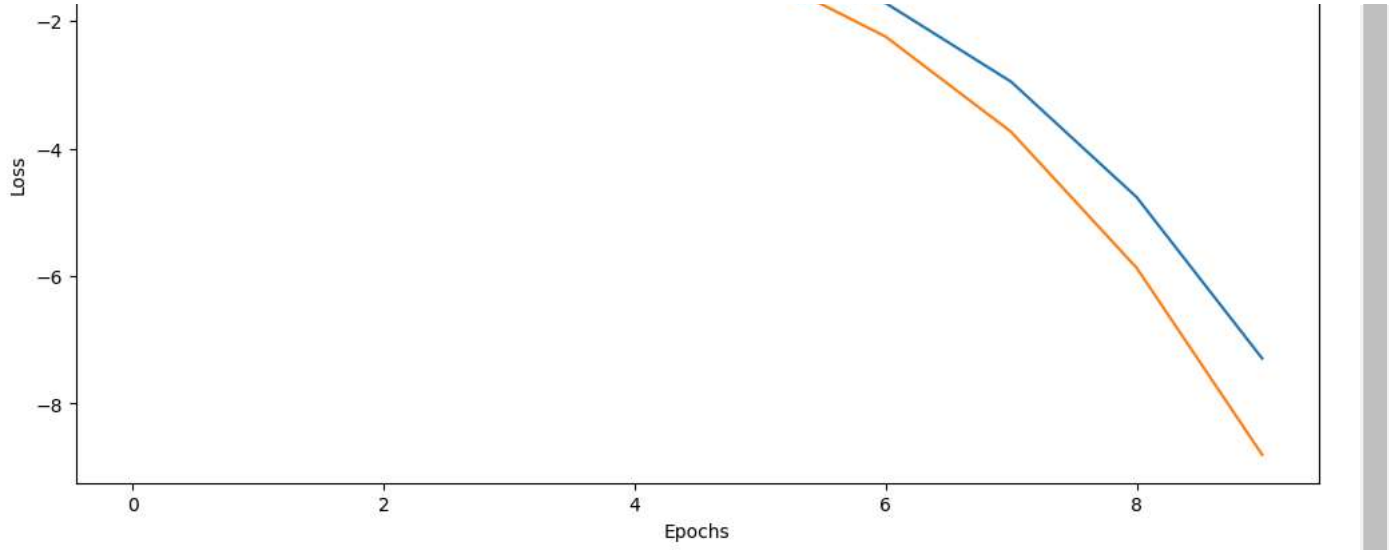
```
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split


if 'Category' in data_cleaned.columns:
    label_encoder = LabelEncoder()
    data_cleaned['Category'] = label_encoder.fit_transform(data_cleaned['Category'])

numerical_columns = data_cleaned.select_dtypes(include=['float64', 'int64']).columns
scaler = StandardScaler()
data_cleaned[numerical_columns] = scaler.fit_transform(data_cleaned[numerical_columns])

target_column = '1.1'


X = data_cleaned.drop(columns=[target_column])
y = data_cleaned[target_column]


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



import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout


model = Sequential()
# Input layer
model.add(Dense(units=256, activation='relu', input_shape=(77,1)))
model.add(Dropout(0.3))
model.add(Dense(units=128, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(units=64, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(units=1, activation='relu'))
model.compile(optimizer='adam', loss='mean_squared_error', metrics=['accuracy'])


import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout

# Assuming your data has 77 features
input_shape = (77,)

# Building the neural network model
model = Sequential()

# Input layer
model.add(Dense(units=256, activation='relu', input_shape=input_shape))
```

```
model.add(Dense(units=256, activation='relu', input_shape=input_shape))
model.add(Dropout(0.3))

# Hidden layers
model.add(Dense(units=128, activation='relu'))
model.add(Dropout(0.3))

model.add(Dense(units=64, activation='relu'))
model.add(Dropout(0.3))

# Output layer
model.add(Dense(units=1, activation='sigmoid'))  # Assuming binary classification

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Summary of the model
model.summary()

# Train the model with adjusted parameters
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2)

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Model Test Accuracy: {accuracy}")

# Plot the training history
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

plt.figure(figsize=(12, 6))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

# Save the trained model for future use
model_file_path = '/content/drive/MyDrive/trained_model.h5'
model.save(model_file_path)
print(f"\nTrained model saved to {model_file_path}")
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_20 (Dense)            (None, 256)               19968

 dropout_15 (Dropout)        (None, 256)               0

 dense_21 (Dense)            (None, 128)               32896

 dropout_16 (Dropout)        (None, 128)               0

 dense_22 (Dense)            (None, 64)                8256

 dropout_17 (Dropout)        (None, 64)                0

 dense_23 (Dense)            (None, 1)                 65

=================================================================
Total params: 61185 (239.00 KB)
Trainable params: 61185 (239.00 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/10
25633/25633 [==============================] - 108s 4ms/step - loss: -451477143552.0000 - accuracy: 0.0000e+00 - val_loss: -195794187
Epoch 2/10
25633/25633 [==============================] - 105s 4ms/step - loss: -9280306544640.0000 - accuracy: 0.0000e+00 - val_loss: -21813511
Epoch 3/10
25633/25633 [==============================] - 113s 4ms/step - loss: -52059734802432.0000 - accuracy: 0.0000e+00 - val_loss: -9455864
Epoch 4/10
25633/25633 [==============================] - 104s 4ms/step - loss: -173041212456960.0000 - accuracy: 0.0000e+00 - val_loss: -273019
Epoch 5/10
25633/25633 [==============================] - 103s 4ms/step - loss: -433633521303552.0000 - accuracy: 0.0000e+00 - val_loss: -628362
Epoch 6/10
25633/25633 [==============================] - 107s 4ms/step - loss: -916453947604992.0000 - accuracy: 0.0000e+00 - val_loss: -125038
Epoch 7/10
25633/25633 [==============================] - 103s 4ms/step - loss: -1718055637876736.0000 - accuracy: 0.0000e+00 - val_loss: -22448
Epoch 8/10
25633/25633 [==============================] - 104s 4ms/step - loss: -2953293869350912.0000 - accuracy: 0.0000e+00 - val_loss: -37390
Epoch 9/10
25633/25633 [==============================] - 104s 4ms/step - loss: -4763449443745792.0000 - accuracy: 0.0000e+00 - val_loss: -58720
Epoch 10/10
25633/25633 [==============================] - 107s 4ms/step - loss: -7292828161933312.0000 - accuracy: 0.0000e+00 - val_loss: -88028
8011/8011 [==============================] - 17s 2ms/step - loss: -8817913066684416.0000 - accuracy: 0.0000e+00
Model Test Accuracy: 0.0
```



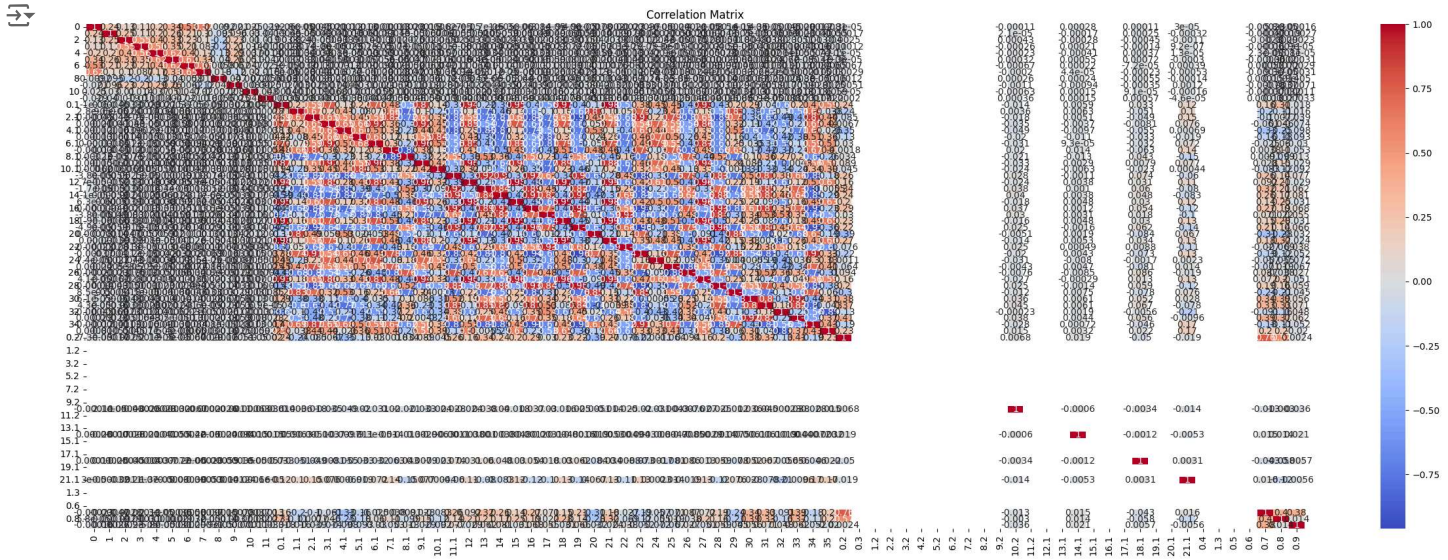





```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via
  saving_api.save_model(

Trained model saved to /content/drive/MyDrive/trained_model.h5
```

Start coding or generate with AI.

```
# Plotting correlation matrix and pairplot
plt.figure(figsize=(30, 10))
sns.heatmap(data_cleaned.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()

sns.pairplot(data_cleaned)
plt.show()
```




```
# Plot the training history
```