Preparing the Dataset

Includes neccessary imports and preparing the second dataset with various preprocessing methods. Finally train and test sets are made in this part.

In [ ]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
import pickle
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from tensorflow.keras.models import Sequential
from keras.layers import Dense, Dropout, SimpleRNN, LSTM, Bidirectional
from tensorflow.keras.layers import Conv1D, MaxPooling1D, GlobalMaxPooling1D, Dense
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, LearningRate
from google.colab import drive
drive.mount('/content/drive')



# Read the CSV file
data = pd.read_csv('/content/drive/MyDrive/CN/featurefinal2.csv')



# Preprocessing
class_mapping = {
    "Benign_list_big_final": "Benign",
    "Malware_dataset": "Malware",
    "phishing_dataset": "Phishing",
    "spam_dataset": "Spam"
}

data['File'] = data['File'].map(class_mapping)
data.drop(columns=['Unnamed: 0'], inplace=True)
data.replace(True,1,inplace = True)
data.replace(False,0,inplace = True) # Ensure data is in the correct format



# Standardize the features
scaler = StandardScaler()
data.iloc[:, 1:] = scaler.fit_transform(data.iloc[:, 1:])

# Save the scaler object
pickle.dump(scaler, open('scaler.sav', 'wb'))


# Encoding the target variable
encoder = LabelEncoder()
y = encoder.fit_transform(data["File"])
# Save the encoder object
np.save('lblenc.npy', encoder.classes_)
```

```
#Dropping the target column
X = data.drop(columns=["File"])



# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, random_st
```

```
Mounted at /content/drive
```

```
<ipython-input-1-9be9909b7584>:44: DeprecationWarning: In a future version, `df.il
oc[:, i] = newvals` will attempt to set the values inplace instead of always setti
ng a new array. To retain the old behavior, use either `df[df.columns[i]] = newval
s` or, if columns are non-unique, `df.isetitem(i, newvals)`
  data.iloc[:, 1:] = scaler.fit_transform(data.iloc[:, 1:])
```

Model

Model definition for the URL classification system. This uses a CNN-LSTM-ANN model to this stack of deep learning layers for classification of URLs into Benign, Malware, Spam, Phising and Defacement URLs.

In [ ]:

```
#current best=0.87
model = Sequential()
model.add(Conv1D(filters=128, kernel_size=3, activation='relu', input_shape=(X_trai
model.add(BatchNormalization())
model.add(Conv1D(filters=128, kernel_size=3, activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(Conv1D(filters=128, kernel_size=3, activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(Conv1D(filters=128, kernel_size=3, activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.2))  # Adjusted dropout rate
model.add(Conv1D(filters=64, kernel_size=3, activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(Conv1D(filters=64, kernel_size=3, activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(Conv1D(filters=64, kernel_size=3, activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(Conv1D(filters=64, kernel_size=3, activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.2))  # Adjusted dropout rate
model.add(Conv1D(filters=32, kernel_size=3, activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(Conv1D(filters=32, kernel_size=3, activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(Conv1D(filters=32, kernel_size=3, activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(Conv1D(filters=32, kernel_size=3, activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(Dropout(0.2))  # Adjusted dropout rate

model.add((LSTM(units=64, return_sequences=True))) #128
model.add((LSTM(units=64, return_sequences=True)))
model.add((LSTM(units=64, return_sequences=False)))

# Fully Connected Layers (ANN)
model.add(BatchNormalization())
model.add(Dense(128, activation='relu'))
```

```
model.add(BatchNormalization())
model.add(Dense(128, activation='relu'))  # You can adjust the number of units in t
model.add(Dense(128, activation='relu'))

# Output Layer
model.add(Dense(np.unique(y).shape[0], activation='softmax'))

# Compile the model with an Adam optimizer
model.compile(optimizer=Adam(learning_rate=0.001), loss='sparse_categorical_crosser



# Define callbacks
# early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weig
# model_checkpoint = ModelCheckpoint('best_model.h5', monitor='val_accuracy', mode=
reduce_lr = LearningRateScheduler(lambda epoch, lr: lr * 0.9 if epoch % 10 == 0 els

callbacks = [
            ModelCheckpoint("/content/drive/MyDrive/CN/savefile2.h5", verbose=1, sa
            ReduceLROnPlateau(monitor="val_accuracy", patience=3, factor=0.1, verbo
            EarlyStopping(monitor="val_accuracy", patience=10, verbose=1, restore_b
            reduce_lr
            ]


# Reshape inputs for CNN
X_train_cnn = X_train.values.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test_cnn = X_test.values.reshape(X_test.shape[0], X_test.shape[1], 1)

# Fit the model to the training data
history = model.fit(X_train_cnn, y_train,epochs=100, validation_split=0.2,batch_siz
model.save('/content/drive/MyDrive/CN/supermodelx2.h5')
```

```
Epoch 1/100
496/496 [==============================] - ETA: 0s - loss: 0.9854 - accuracy: 0.59
76
Epoch 1: val_loss improved from inf to 0.83282, saving model to /content/drive/MyD
rive/CN/savefile2.h5
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWar
ning: You are saving your model as an HDF5 file via `model.save()`. This file form
at is considered legacy. We recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')`.
  saving_api.save_model(
```

```
496/496 [==============================] - 75s 105ms/step - loss: 0.9854 - accurac
y: 0.5976 - val_loss: 0.8328 - val_accuracy: 0.6844 - lr: 9.0000e-04
Epoch 2/100
495/496 [============================>.] - ETA: 0s - loss: 0.7082 - accuracy: 0.72
82
Epoch 2: val_loss improved from 0.83282 to 0.62369, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 49s 100ms/step - loss: 0.7082 - accurac
y: 0.7282 - val_loss: 0.6237 - val_accuracy: 0.7629 - lr: 9.0000e-04
Epoch 3/100
495/496 [============================>.] - ETA: 0s - loss: 0.6316 - accuracy: 0.75
88
Epoch 3: val_loss improved from 0.62369 to 0.55114, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 47s 95ms/step - loss: 0.6316 - accurac
y: 0.7588 - val_loss: 0.5511 - val_accuracy: 0.7902 - lr: 9.0000e-04
Epoch 4/100
496/496 [==============================] - ETA: 0s - loss: 0.5953 - accuracy: 0.77
21
Epoch 4: val_loss did not improve from 0.55114
496/496 [==============================] - 48s 97ms/step - loss: 0.5953 - accurac
y: 0.7721 - val_loss: 0.6072 - val_accuracy: 0.7707 - lr: 9.0000e-04
Epoch 5/100
495/496 [============================>.] - ETA: 0s - loss: 0.5648 - accuracy: 0.78
26
Epoch 5: val_loss improved from 0.55114 to 0.52708, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 46s 93ms/step - loss: 0.5646 - accurac
y: 0.7826 - val_loss: 0.5271 - val_accuracy: 0.8018 - lr: 9.0000e-04
Epoch 6/100
495/496 [============================>.] - ETA: 0s - loss: 0.5423 - accuracy: 0.79
38
Epoch 6: val_loss improved from 0.52708 to 0.50702, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 48s 97ms/step - loss: 0.5422 - accurac
y: 0.7938 - val_loss: 0.5070 - val_accuracy: 0.8131 - lr: 9.0000e-04
Epoch 7/100
496/496 [==============================] - ETA: 0s - loss: 0.5231 - accuracy: 0.80
22
Epoch 7: val_loss improved from 0.50702 to 0.48467, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 49s 98ms/step - loss: 0.5231 - accurac
y: 0.8022 - val_loss: 0.4847 - val_accuracy: 0.8061 - lr: 9.0000e-04
Epoch 8/100
495/496 [============================>.] - ETA: 0s - loss: 0.5093 - accuracy: 0.80
54
Epoch 8: val_loss did not improve from 0.48467
496/496 [==============================] - 44s 90ms/step - loss: 0.5092 - accurac
y: 0.8054 - val_loss: 0.5053 - val_accuracy: 0.8037 - lr: 9.0000e-04
Epoch 9/100
495/496 [============================>.] - ETA: 0s - loss: 0.4952 - accuracy: 0.81
17
Epoch 9: val_loss improved from 0.48467 to 0.46998, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 52s 104ms/step - loss: 0.4951 - accurac
y: 0.8118 - val_loss: 0.4700 - val_accuracy: 0.8154 - lr: 9.0000e-04
Epoch 10/100
496/496 [==============================] - ETA: 0s - loss: 0.4830 - accuracy: 0.81
42
Epoch 10: val_loss improved from 0.46998 to 0.46098, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 50s 101ms/step - loss: 0.4830 - accurac
y: 0.8142 - val_loss: 0.4610 - val_accuracy: 0.8194 - lr: 9.0000e-04
Epoch 11/100
```

```
496/496 [==============================] - ETA: 0s - loss: 0.4625 - accuracy: 0.82
15
Epoch 11: val_loss improved from 0.46098 to 0.44448, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 49s 100ms/step - loss: 0.4625 - accurac
y: 0.8215 - val_loss: 0.4445 - val_accuracy: 0.8315 - lr: 8.1000e-04
Epoch 12/100
495/496 [============================>.] - ETA: 0s - loss: 0.4561 - accuracy: 0.82
48
Epoch 12: val_loss did not improve from 0.44448
496/496 [==============================] - 46s 92ms/step - loss: 0.4561 - accurac
y: 0.8248 - val_loss: 0.4777 - val_accuracy: 0.8202 - lr: 8.1000e-04
Epoch 13/100
495/496 [============================>.] - ETA: 0s - loss: 0.4459 - accuracy: 0.83
02
Epoch 13: val_loss improved from 0.44448 to 0.43954, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 49s 99ms/step - loss: 0.4458 - accurac
y: 0.8303 - val_loss: 0.4395 - val_accuracy: 0.8310 - lr: 8.1000e-04
Epoch 14/100
496/496 [==============================] - ETA: 0s - loss: 0.4421 - accuracy: 0.83
09
Epoch 14: val_loss improved from 0.43954 to 0.43522, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 48s 96ms/step - loss: 0.4421 - accurac
y: 0.8309 - val_loss: 0.4352 - val_accuracy: 0.8384 - lr: 8.1000e-04
Epoch 15/100
495/496 [============================>.] - ETA: 0s - loss: 0.4316 - accuracy: 0.83
59
Epoch 15: val_loss improved from 0.43522 to 0.42735, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 46s 92ms/step - loss: 0.4317 - accurac
y: 0.8359 - val_loss: 0.4273 - val_accuracy: 0.8377 - lr: 8.1000e-04
Epoch 16/100
496/496 [==============================] - ETA: 0s - loss: 0.4256 - accuracy: 0.83
72
Epoch 16: val_loss did not improve from 0.42735
496/496 [==============================] - 51s 102ms/step - loss: 0.4256 - accurac
y: 0.8372 - val_loss: 0.4456 - val_accuracy: 0.8307 - lr: 8.1000e-04
Epoch 17/100
496/496 [==============================] - ETA: 0s - loss: 0.4239 - accuracy: 0.83
82
Epoch 17: val_loss improved from 0.42735 to 0.42672, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 47s 96ms/step - loss: 0.4239 - accurac
y: 0.8382 - val_loss: 0.4267 - val_accuracy: 0.8401 - lr: 8.1000e-04
Epoch 18/100
495/496 [============================>.] - ETA: 0s - loss: 0.4140 - accuracy: 0.84
19
Epoch 18: val_loss improved from 0.42672 to 0.41285, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 47s 95ms/step - loss: 0.4141 - accurac
y: 0.8419 - val_loss: 0.4128 - val_accuracy: 0.8456 - lr: 8.1000e-04
Epoch 19/100
495/496 [============================>.] - ETA: 0s - loss: 0.4079 - accuracy: 0.84
34
Epoch 19: val_loss did not improve from 0.41285
496/496 [==============================] - 48s 98ms/step - loss: 0.4079 - accurac
y: 0.8433 - val_loss: 0.4203 - val_accuracy: 0.8442 - lr: 8.1000e-04
Epoch 20/100
496/496 [==============================] - ETA: 0s - loss: 0.4054 - accuracy: 0.84
40
Epoch 20: val_loss improved from 0.41285 to 0.39590, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
```

```
496/496 [==============================] - 49s 99ms/step - loss: 0.4054 - accurac
y: 0.8440 - val_loss: 0.3959 - val_accuracy: 0.8459 - lr: 8.1000e-04
Epoch 21/100
495/496 [============================>.] - ETA: 0s - loss: 0.3899 - accuracy: 0.85
13
Epoch 21: val_loss improved from 0.39590 to 0.39443, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 46s 92ms/step - loss: 0.3901 - accurac
y: 0.8513 - val_loss: 0.3944 - val_accuracy: 0.8541 - lr: 7.2900e-04
Epoch 22/100
495/496 [============================>.] - ETA: 0s - loss: 0.3882 - accuracy: 0.85
20
Epoch 22: val_loss improved from 0.39443 to 0.38585, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 49s 98ms/step - loss: 0.3882 - accurac
y: 0.8520 - val_loss: 0.3859 - val_accuracy: 0.8545 - lr: 7.2900e-04
Epoch 23/100
495/496 [============================>.] - ETA: 0s - loss: 0.3816 - accuracy: 0.85
36
Epoch 23: val_loss did not improve from 0.38585
496/496 [==============================] - 47s 95ms/step - loss: 0.3817 - accurac
y: 0.8536 - val_loss: 0.4306 - val_accuracy: 0.8389 - lr: 7.2900e-04
Epoch 24/100
495/496 [============================>.] - ETA: 0s - loss: 0.3821 - accuracy: 0.85
41
Epoch 24: val_loss did not improve from 0.38585
496/496 [==============================] - 49s 99ms/step - loss: 0.3821 - accurac
y: 0.8542 - val_loss: 0.3926 - val_accuracy: 0.8516 - lr: 7.2900e-04
Epoch 25/100
496/496 [==============================] - ETA: 0s - loss: 0.3750 - accuracy: 0.85
76
Epoch 25: val_loss did not improve from 0.38585
496/496 [==============================] - 48s 96ms/step - loss: 0.3750 - accurac
y: 0.8576 - val_loss: 0.3868 - val_accuracy: 0.8551 - lr: 7.2900e-04
Epoch 26/100
495/496 [============================>.] - ETA: 0s - loss: 0.3702 - accuracy: 0.85
74
Epoch 26: val_loss improved from 0.38585 to 0.38533, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 45s 91ms/step - loss: 0.3702 - accurac
y: 0.8574 - val_loss: 0.3853 - val_accuracy: 0.8551 - lr: 7.2900e-04
Epoch 27/100
495/496 [============================>.] - ETA: 0s - loss: 0.3697 - accuracy: 0.85
88
Epoch 27: val_loss improved from 0.38533 to 0.37523, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 48s 97ms/step - loss: 0.3697 - accurac
y: 0.8588 - val_loss: 0.3752 - val_accuracy: 0.8633 - lr: 7.2900e-04
Epoch 28/100
496/496 [==============================] - ETA: 0s - loss: 0.3635 - accuracy: 0.85
97
Epoch 28: val_loss did not improve from 0.37523
496/496 [==============================] - 47s 95ms/step - loss: 0.3635 - accurac
y: 0.8597 - val_loss: 0.3836 - val_accuracy: 0.8544 - lr: 7.2900e-04
Epoch 29/100
495/496 [============================>.] - ETA: 0s - loss: 0.3608 - accuracy: 0.86
13
Epoch 29: val_loss did not improve from 0.37523
496/496 [==============================] - 45s 90ms/step - loss: 0.3608 - accurac
y: 0.8613 - val_loss: 0.3916 - val_accuracy: 0.8527 - lr: 7.2900e-04
Epoch 30/100
496/496 [==============================] - ETA: 0s - loss: 0.3616 - accuracy: 0.86
07
Epoch 30: val_loss did not improve from 0.37523
```

```
Epoch 30: ReduceLROnPlateau reducing learning rate to 7.29000079445541e-05.
496/496 [==============================] - 50s 101ms/step - loss: 0.3616 - accurac
y: 0.8607 - val_loss: 0.3826 - val_accuracy: 0.8565 - lr: 7.2900e-05
Epoch 31/100
495/496 [============================>.] - ETA: 0s - loss: 0.3272 - accuracy: 0.87
23
Epoch 31: val_loss improved from 0.37523 to 0.34765, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 52s 104ms/step - loss: 0.3271 - accurac
y: 0.8724 - val_loss: 0.3477 - val_accuracy: 0.8674 - lr: 6.5610e-05
Epoch 32/100
495/496 [============================>.] - ETA: 0s - loss: 0.3121 - accuracy: 0.87
76
Epoch 32: val_loss improved from 0.34765 to 0.34471, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 46s 92ms/step - loss: 0.3121 - accurac
y: 0.8776 - val_loss: 0.3447 - val_accuracy: 0.8702 - lr: 6.5610e-05
Epoch 33/100
495/496 [============================>.] - ETA: 0s - loss: 0.3043 - accuracy: 0.88
04
Epoch 33: val_loss improved from 0.34471 to 0.34211, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 49s 98ms/step - loss: 0.3049 - accurac
y: 0.8803 - val_loss: 0.3421 - val_accuracy: 0.8733 - lr: 6.5610e-05
Epoch 34/100
496/496 [==============================] - ETA: 0s - loss: 0.3058 - accuracy: 0.88
17
Epoch 34: val_loss did not improve from 0.34211
496/496 [==============================] - 47s 95ms/step - loss: 0.3058 - accurac
y: 0.8817 - val_loss: 0.3425 - val_accuracy: 0.8714 - lr: 6.5610e-05
Epoch 35/100
495/496 [============================>.] - ETA: 0s - loss: 0.3013 - accuracy: 0.88
17
Epoch 35: val_loss improved from 0.34211 to 0.33668, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 46s 93ms/step - loss: 0.3013 - accurac
y: 0.8817 - val_loss: 0.3367 - val_accuracy: 0.8731 - lr: 6.5610e-05
Epoch 36/100
496/496 [==============================] - ETA: 0s - loss: 0.2992 - accuracy: 0.88
21
Epoch 36: val_loss did not improve from 0.33668
496/496 [==============================] - 48s 97ms/step - loss: 0.2992 - accurac
y: 0.8821 - val_loss: 0.3368 - val_accuracy: 0.8760 - lr: 6.5610e-05
Epoch 37/100
495/496 [============================>.] - ETA: 0s - loss: 0.2968 - accuracy: 0.88
41
Epoch 37: val_loss did not improve from 0.33668
496/496 [==============================] - 45s 91ms/step - loss: 0.2971 - accurac
y: 0.8840 - val_loss: 0.3396 - val_accuracy: 0.8729 - lr: 6.5610e-05
Epoch 38/100
496/496 [==============================] - ETA: 0s - loss: 0.2937 - accuracy: 0.88
53
Epoch 38: val_loss did not improve from 0.33668
496/496 [==============================] - 52s 105ms/step - loss: 0.2937 - accurac
y: 0.8853 - val_loss: 0.3387 - val_accuracy: 0.8762 - lr: 6.5610e-05
Epoch 39/100
496/496 [==============================] - ETA: 0s - loss: 0.2933 - accuracy: 0.88
55
Epoch 39: val_loss improved from 0.33668 to 0.33522, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 48s 97ms/step - loss: 0.2933 - accurac
y: 0.8855 - val_loss: 0.3352 - val_accuracy: 0.8799 - lr: 6.5610e-05
Epoch 40/100
```

```
495/496 [============================>.] - ETA: 0s - loss: 0.2913 - accuracy: 0.88
32
Epoch 40: val_loss improved from 0.33522 to 0.33496, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 46s 92ms/step - loss: 0.2914 - accurac
y: 0.8832 - val_loss: 0.3350 - val_accuracy: 0.8770 - lr: 6.5610e-05
Epoch 41/100
495/496 [============================>.] - ETA: 0s - loss: 0.2903 - accuracy: 0.88
53
Epoch 41: val_loss improved from 0.33496 to 0.33281, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 48s 97ms/step - loss: 0.2905 - accurac
y: 0.8853 - val_loss: 0.3328 - val_accuracy: 0.8774 - lr: 5.9049e-05
Epoch 42/100
496/496 [==============================] - ETA: 0s - loss: 0.2897 - accuracy: 0.88
63
Epoch 42: val_loss improved from 0.33281 to 0.33111, saving model to /content/driv
e/MyDrive/CN/savefile2.h5

Epoch 42: ReduceLROnPlateau reducing learning rate to 5.9049008996225895e-06.
496/496 [==============================] - 48s 97ms/step - loss: 0.2897 - accurac
y: 0.8863 - val_loss: 0.3311 - val_accuracy: 0.8770 - lr: 5.9049e-06
Epoch 43/100
495/496 [============================>.] - ETA: 0s - loss: 0.2833 - accuracy: 0.88
77
Epoch 43: val_loss improved from 0.33111 to 0.33040, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 45s 91ms/step - loss: 0.2834 - accurac
y: 0.8877 - val_loss: 0.3304 - val_accuracy: 0.8785 - lr: 5.9049e-06
Epoch 44/100
495/496 [============================>.] - ETA: 0s - loss: 0.2851 - accuracy: 0.88
84
Epoch 44: val_loss improved from 0.33040 to 0.33025, saving model to /content/driv
e/MyDrive/CN/savefile2.h5
496/496 [==============================] - 49s 98ms/step - loss: 0.2853 - accurac
y: 0.8883 - val_loss: 0.3302 - val_accuracy: 0.8782 - lr: 5.9049e-06
Epoch 45/100
496/496 [==============================] - ETA: 0s - loss: 0.2830 - accuracy: 0.88
86
Epoch 45: val_loss did not improve from 0.33025

Epoch 45: ReduceLROnPlateau reducing learning rate to 5.90490071772365e-07.
496/496 [==============================] - 50s 101ms/step - loss: 0.2830 - accurac
y: 0.8886 - val_loss: 0.3308 - val_accuracy: 0.8796 - lr: 5.9049e-07
Epoch 46/100
495/496 [============================>.] - ETA: 0s - loss: 0.2849 - accuracy: 0.88
72
Epoch 46: val_loss did not improve from 0.33025
496/496 [==============================] - 56s 113ms/step - loss: 0.2849 - accurac
y: 0.8872 - val_loss: 0.3313 - val_accuracy: 0.8789 - lr: 5.9049e-07
Epoch 47/100
496/496 [==============================] - ETA: 0s - loss: 0.2856 - accuracy: 0.88
79
Epoch 47: val_loss did not improve from 0.33025
496/496 [==============================] - 51s 104ms/step - loss: 0.2856 - accurac
y: 0.8879 - val_loss: 0.3310 - val_accuracy: 0.8796 - lr: 5.9049e-07
Epoch 48/100
496/496 [==============================] - ETA: 0s - loss: 0.2807 - accuracy: 0.88
81
Epoch 48: val_loss did not improve from 0.33025

Epoch 48: ReduceLROnPlateau reducing learning rate to 5.904900604036812e-08.
496/496 [==============================] - 51s 103ms/step - loss: 0.2807 - accurac
y: 0.8881 - val_loss: 0.3304 - val_accuracy: 0.8780 - lr: 5.9049e-08
```

```
Epoch 49/100
495/496 [============================>.] - ETA: 0s - loss: 0.2818 - accuracy: 0.88
83
Epoch 49: val_loss did not improve from 0.33025
Restoring model weights from the end of the best epoch: 39.
496/496 [==============================] - 49s 98ms/step - loss: 0.2818 - accurac
y: 0.8883 - val_loss: 0.3314 - val_accuracy: 0.8796 - lr: 5.9049e-08
Epoch 49: early stopping
```

Model Summary

In [ ]:
```
print(model.summary())
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 21, 128)	512
batch_normalization (BatchNormalization)	(None, 21, 128)	512
conv1d_1 (Conv1D)	(None, 21, 128)	49280
batch_normalization_1 (BatchNormalization)	(None, 21, 128)	512
conv1d_2 (Conv1D)	(None, 21, 128)	49280
batch_normalization_2 (BatchNormalization)	(None, 21, 128)	512
conv1d_3 (Conv1D)	(None, 21, 128)	49280
batch_normalization_3 (BatchNormalization)	(None, 21, 128)	512
max_pooling1d (MaxPooling1D)	(None, 10, 128)	0
dropout (Dropout)	(None, 10, 128)	0
conv1d_4 (Conv1D)	(None, 10, 64)	24640
batch_normalization_4 (BatchNormalization)	(None, 10, 64)	256
conv1d_5 (Conv1D)	(None, 10, 64)	12352
batch_normalization_5 (BatchNormalization)	(None, 10, 64)	256
conv1d_6 (Conv1D)	(None, 10, 64)	12352
batch_normalization_6 (BatchNormalization)	(None, 10, 64)	256
conv1d_7 (Conv1D)	(None, 10, 64)	12352
batch_normalization_7 (BatchNormalization)	(None, 10, 64)	256
max_pooling1d_1 (MaxPooling1D)	(None, 5, 64)	0
dropout_1 (Dropout)	(None, 5, 64)	0
conv1d_8 (Conv1D)	(None, 5, 32)	6176
batch_normalization_8 (BatchNormalization)	(None, 5, 32)	128
conv1d_9 (Conv1D)	(None, 5, 32)	3104
batch_normalization_9 (BatchNormalization)	(None, 5, 32)	128

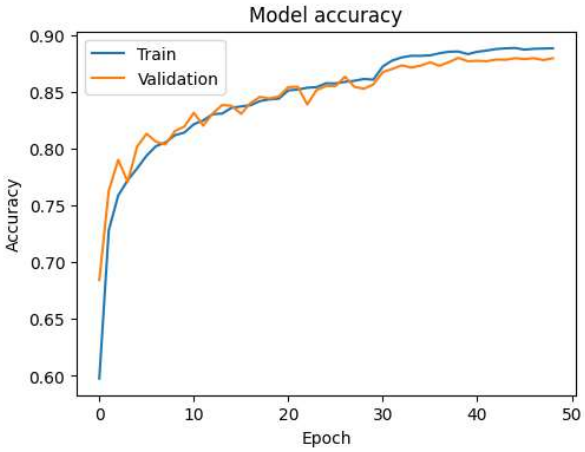
```
 conv1d_10 (Conv1D)          (None, 5, 32)             3104

 batch_normalization_10 (Ba  (None, 5, 32)             128
 tchNormalization)

 conv1d_11 (Conv1D)          (None, 5, 32)             3104

 batch_normalization_11 (Ba  (None, 5, 32)             128
 tchNormalization)

 dropout_2 (Dropout)         (None, 5, 32)             0

 lstm (LSTM)                 (None, 5, 64)             24832

 lstm_1 (LSTM)               (None, 5, 64)             33024

 lstm_2 (LSTM)               (None, 64)                33024

 batch_normalization_12 (Ba  (None, 64)                256
 tchNormalization)

 dense (Dense)               (None, 128)               8320

 batch_normalization_13 (Ba  (None, 128)               512
 tchNormalization)

 dense_1 (Dense)             (None, 128)               16512

 dense_2 (Dense)             (None, 128)               16512

 dense_3 (Dense)             (None, 5)                 645

=================================================================
Total params: 362757 (1.38 MB)
Trainable params: 360581 (1.38 MB)
Non-trainable params: 2176 (8.50 KB)
_________________________________________________________________
None
```

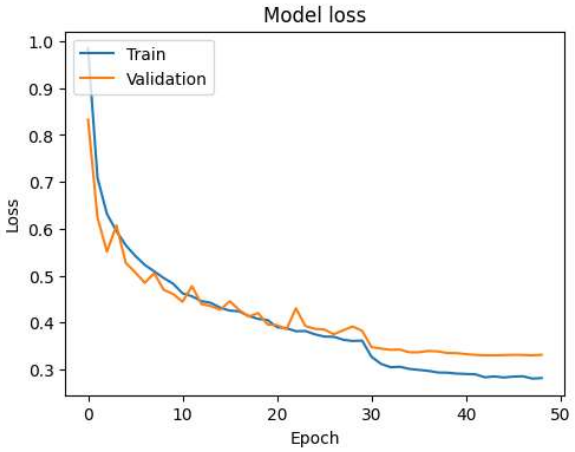
Graph Plots for the result of the training

In [ ]:

```python
import matplotlib.pyplot as plt

# Plot training & validation accuracy values
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')

# Plot training & validation loss values
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
```

```
plt.show()
```





Model Analytical prediction details.

In [ ]:
```
# Perform predictions on the testing set
y_pred = model.predict(X_test_cnn)

# Convert predictions to class labels
y_pred_labels = np.argmax(y_pred, axis=1)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred_labels)
print(f"Accuracy: {accuracy}")

# Generate a classification report
class_names = ["Benign", "Malware", "Phishing", "Spam", "Defacement"]

classification_rep = classification_report(y_test, y_pred_labels, target_names=clas
print("Classification Report:")
print(classification_rep)
```

```
219/219 [==============================] - 7s 21ms/step
Accuracy: 0.874016874016874
Classification Report:
              precision    recall  f1-score   support

      Benign       0.95      0.98      0.96      1478
     Malware       0.87      0.92      0.90      1329
    Phishing       0.81      0.75      0.78      1315
        Spam       0.90      0.92      0.91      1549
  Defacement       0.81      0.78      0.80      1322

    accuracy                           0.87      6993
   macro avg       0.87      0.87      0.87      6993
weighted avg       0.87      0.87      0.87      6993
```

Part 2- Transfer Learning. This dataset used for this, contains 4000 URLs. This is to test the efficiency of the model in a case with very less data, and checking how it performs when the retrain dataset doesn't include all the classes the original model was trained for. These results can be compared with another transfer learning model with almost similar number of URLs(5000), which involves all the classes to see if existence of all the classes proves to be of difference while retraining the model.

Data preprocessing for the Transfer Learning Model

In [ ]:
```
# Read the CSV file
data = pd.read_csv('/content/drive/MyDrive/CN/retrain2.csv')

# Preprocessing
class_mapping = {
    "Benign_list_big_final": "Benign",
    "Malware_dataset": "Malware",
    "phishing_dataset": "Phishing",
    "spam_dataset": "Spam"
}

data['File'] = data['File'].map(class_mapping)
data.drop(columns=['Unnamed: 0'], inplace=True)
data.replace(True,1,inplace = True)
data.replace(False,0,inplace = True) # Ensure data is in the correct format

# Standardize the features
scaler = StandardScaler()
data.iloc[:, 1:] = scaler.fit_transform(data.iloc[:, 1:])

# Save the scaler object
pickle.dump(scaler, open('scaler.sav', 'wb'))


# Encoding the target variable
encoder = LabelEncoder()
y = encoder.fit_transform(data["File"])
# Save the encoder object
np.save('lblenc.npy', encoder.classes_)

#Dropping the target column
X = data.drop(columns=["File"])



# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

```
<ipython-input-6-cde2c651e217>:19: DeprecationWarning: In a future version, `df.il
oc[:, i] = newvals` will attempt to set the values inplace instead of always setti
ng a new array. To retain the old behavior, use either `df[df.columns[i]] = newval
s` or, if columns are non-unique, `df.isetitem(i, newvals)`
  data.iloc[:, 1:] = scaler.fit_transform(data.iloc[:, 1:])
```

Retraining of the model

In [ ]:
```
from tensorflow.keras.models import load_model

model = load_model('/content/drive/MyDrive/CN/supermodelx2.h5')


model.compile(optimizer=Adam(learning_rate=0.001), loss='sparse_categorical_crosser


reduce_lr = LearningRateScheduler(lambda epoch, lr: lr * 0.9 if epoch % 10 == 0 els
```

```
callbacks = [
            ModelCheckpoint("/content/drive/MyDrive/CN/savefile.h5", verbose=1, sav
            ReduceLROnPlateau(monitor="val_accuracy", patience=3, factor=0.1, verbo
            EarlyStopping(monitor="val_accuracy", patience=10, verbose=1, restore_b
            reduce_lr
            ]

# Reshape inputs for CNN
X_train_cnn = X_train.values.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test_cnn = X_test.values.reshape(X_test.shape[0], X_test.shape[1], 1)

# Fit the model to the training data
history = model.fit(X_train_cnn, y_train,epochs=20,batch_size=32, validation_split=
model.save('/content/drive/MyDrive/CN/retrainmodel2.h5')
```

```
Epoch 1/20
88/88 [==============================] - ETA: 0s - loss: 0.4161 - accuracy: 0.8504
Epoch 1: val_loss improved from inf to 0.41237, saving model to /content/drive/MyD
rive/CN/savefile.h5
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWar
ning: You are saving your model as an HDF5 file via `model.save()`. This file form
at is considered legacy. We recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')`.
  saving_api.save_model(
```

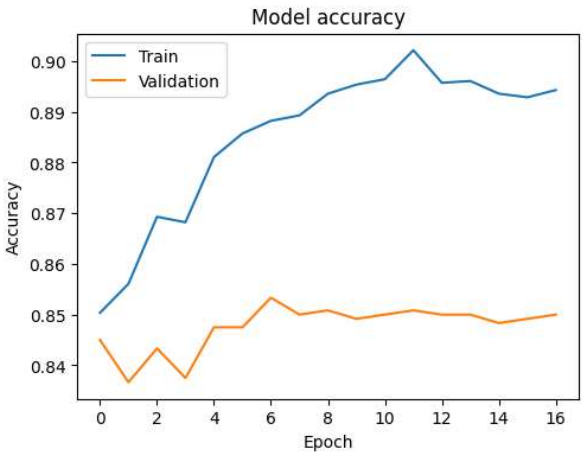
```
88/88 [==============================] - 24s 81ms/step - loss: 0.4161 - accuracy:
0.8504 - val_loss: 0.4124 - val_accuracy: 0.8450 - lr: 9.0000e-04
Epoch 2/20
88/88 [==============================] - ETA: 0s - loss: 0.3666 - accuracy: 0.8561
Epoch 2: val_loss did not improve from 0.41237
88/88 [==============================] - 6s 70ms/step - loss: 0.3666 - accuracy:
0.8561 - val_loss: 0.4200 - val_accuracy: 0.8367 - lr: 9.0000e-04
Epoch 3/20
87/88 [============================>.] - ETA: 0s - loss: 0.3301 - accuracy: 0.8693
Epoch 3: val_loss did not improve from 0.41237
88/88 [==============================] - 7s 80ms/step - loss: 0.3303 - accuracy:
0.8693 - val_loss: 0.4281 - val_accuracy: 0.8433 - lr: 9.0000e-04
Epoch 4/20
87/88 [============================>.] - ETA: 0s - loss: 0.3416 - accuracy: 0.8682
Epoch 4: val_loss did not improve from 0.41237

Epoch 4: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.
88/88 [==============================] - 5s 59ms/step - loss: 0.3412 - accuracy:
0.8682 - val_loss: 0.4581 - val_accuracy: 0.8375 - lr: 9.0000e-05
Epoch 5/20
88/88 [==============================] - ETA: 0s - loss: 0.3155 - accuracy: 0.8811
Epoch 5: val_loss did not improve from 0.41237
88/88 [==============================] - 7s 80ms/step - loss: 0.3155 - accuracy:
0.8811 - val_loss: 0.4199 - val_accuracy: 0.8475 - lr: 9.0000e-05
Epoch 6/20
87/88 [============================>.] - ETA: 0s - loss: 0.2851 - accuracy: 0.8858
Epoch 6: val_loss did not improve from 0.41237
88/88 [==============================] - 6s 68ms/step - loss: 0.2846 - accuracy:
0.8857 - val_loss: 0.4142 - val_accuracy: 0.8475 - lr: 9.0000e-05
Epoch 7/20
87/88 [============================>.] - ETA: 0s - loss: 0.2834 - accuracy: 0.8879
Epoch 7: val_loss improved from 0.41237 to 0.40915, saving model to /content/driv
e/MyDrive/CN/savefile.h5
88/88 [==============================] - 6s 63ms/step - loss: 0.2828 - accuracy:
0.8882 - val_loss: 0.4092 - val_accuracy: 0.8533 - lr: 9.0000e-05
Epoch 8/20
88/88 [==============================] - ETA: 0s - loss: 0.2761 - accuracy: 0.8893
Epoch 8: val_loss did not improve from 0.40915
88/88 [==============================] - 7s 81ms/step - loss: 0.2761 - accuracy:
0.8893 - val_loss: 0.4105 - val_accuracy: 0.8500 - lr: 9.0000e-05
Epoch 9/20
87/88 [============================>.] - ETA: 0s - loss: 0.2716 - accuracy: 0.8940
Epoch 9: val_loss did not improve from 0.40915
88/88 [==============================] - 7s 84ms/step - loss: 0.2716 - accuracy:
0.8936 - val_loss: 0.4169 - val_accuracy: 0.8508 - lr: 9.0000e-05
Epoch 10/20
88/88 [==============================] - ETA: 0s - loss: 0.2618 - accuracy: 0.8954
Epoch 10: val_loss did not improve from 0.40915

Epoch 10: ReduceLROnPlateau reducing learning rate to 9.000000136438758e-06.
88/88 [==============================] - 6s 64ms/step - loss: 0.2618 - accuracy:
0.8954 - val_loss: 0.4222 - val_accuracy: 0.8492 - lr: 9.0000e-06
Epoch 11/20
88/88 [==============================] - ETA: 0s - loss: 0.2535 - accuracy: 0.8964
Epoch 11: val_loss did not improve from 0.40915
88/88 [==============================] - 7s 80ms/step - loss: 0.2535 - accuracy:
0.8964 - val_loss: 0.4218 - val_accuracy: 0.8500 - lr: 8.1000e-06
Epoch 12/20
87/88 [============================>.] - ETA: 0s - loss: 0.2473 - accuracy: 0.9016
Epoch 12: val_loss did not improve from 0.40915
88/88 [==============================] - 5s 61ms/step - loss: 0.2465 - accuracy:
0.9021 - val_loss: 0.4201 - val_accuracy: 0.8508 - lr: 8.1000e-06
Epoch 13/20
88/88 [==============================] - ETA: 0s - loss: 0.2571 - accuracy: 0.8957
```

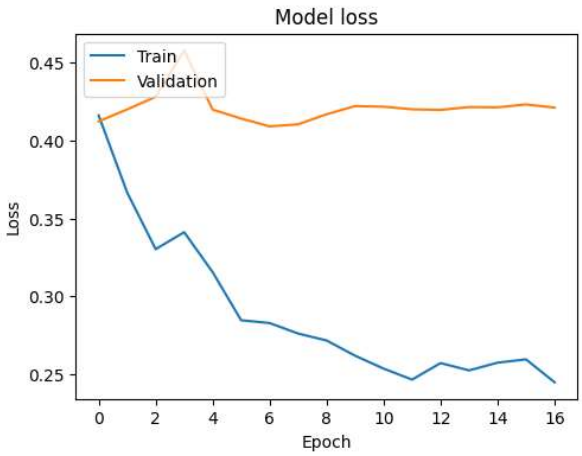
```
Epoch 13: val_loss did not improve from 0.40915

Epoch 13: ReduceLROnPlateau reducing learning rate to 8.099999831756577e-07.
88/88 [==============================] - 5s 53ms/step - loss: 0.2571 - accuracy:
0.8957 - val_loss: 0.4197 - val_accuracy: 0.8500 - lr: 8.1000e-07
Epoch 14/20
88/88 [==============================] - ETA: 0s - loss: 0.2524 - accuracy: 0.8961
Epoch 14: val_loss did not improve from 0.40915
88/88 [==============================] - 8s 92ms/step - loss: 0.2524 - accuracy:
0.8961 - val_loss: 0.4216 - val_accuracy: 0.8500 - lr: 8.1000e-07
Epoch 15/20
88/88 [==============================] - ETA: 0s - loss: 0.2573 - accuracy: 0.8936
Epoch 15: val_loss did not improve from 0.40915
88/88 [==============================] - 6s 70ms/step - loss: 0.2573 - accuracy:
0.8936 - val_loss: 0.4214 - val_accuracy: 0.8483 - lr: 8.1000e-07
Epoch 16/20
88/88 [==============================] - ETA: 0s - loss: 0.2595 - accuracy: 0.8929
Epoch 16: val_loss did not improve from 0.40915

Epoch 16: ReduceLROnPlateau reducing learning rate to 8.099999604382901e-08.
88/88 [==============================] - 5s 54ms/step - loss: 0.2595 - accuracy:
0.8929 - val_loss: 0.4232 - val_accuracy: 0.8492 - lr: 8.1000e-08
Epoch 17/20
88/88 [==============================] - ETA: 0s - loss: 0.2448 - accuracy: 0.8943
Epoch 17: val_loss did not improve from 0.40915
Restoring model weights from the end of the best epoch: 7.
88/88 [==============================] - 7s 75ms/step - loss: 0.2448 - accuracy:
0.8943 - val_loss: 0.4212 - val_accuracy: 0.8500 - lr: 8.1000e-08
Epoch 17: early stopping
```

Plots for retraining model.

In [ ]:

```python
import matplotlib.pyplot as plt

# Plot training & validation accuracy values
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')

# Plot training & validation loss values
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')

plt.show()
```





Analysis

In [ ]:
```
# Perform predictions on the testing set
y_pred = model.predict(X_test_cnn)

# Convert predictions to class labels
y_pred_labels = np.argmax(y_pred, axis=1)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred_labels)
print(f"Accuracy: {accuracy}")

# Generate a classification report
class_names = ["Benign", "Malware", "Phishing", "Spam", "Defacement"]

classification_rep = classification_report(y_test, y_pred_labels, target_names=clas
print("Classification Report:")
print(classification_rep)
```

```
32/32 [==============================] - 2s 13ms/step
Accuracy: 0.861
Classification Report:
              precision    recall  f1-score   support

      Benign       0.97      0.95      0.96       178
     Malware       0.86      0.94      0.90       192
    Phishing       0.84      0.74      0.78       208
        Spam       0.88      0.92      0.90       214
  Defacement       0.77      0.78      0.78       208

    accuracy                           0.86      1000
   macro avg       0.86      0.86      0.86      1000
weighted avg       0.86      0.86      0.86      1000
```

In [ ]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming 'y_test' and 'y_pred_labels' are your true labels and predicted labels,
conf_matrix = confusion_matrix(y_test, y_pred_labels)

# Plot the confusion matrix using a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=class_names
plt.title("Confusion Matrix for Transfer Learning Model")
plt.xlabel("Predicted")
```

```
plt.ylabel("True")
plt.show()
```