Preparing the Dataset

Includes neccessary imports and preparing the 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/featurefinal1.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-1-4d48df06652f>: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/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=100, validation_split=0.2,batch_siz
model.save('/content/drive/MyDrive/CN/supermodelx1.h5')
```

```
Epoch 1/100
515/516 [============================>.] - ETA: 0s - loss: 1.0088 - accuracy: 0.59
42
Epoch 1: val_loss improved from inf to 0.80563, saving model to /content/drive/MyD
rive/CN/savefile.h5
516/516 [==============================] - 94s 141ms/step - loss: 1.0088 - accurac
y: 0.5942 - val_loss: 0.8056 - val_accuracy: 0.6841 - lr: 9.0000e-04
Epoch 2/100
516/516 [==============================] - ETA: 0s - loss: 0.7195 - accuracy: 0.72
54
Epoch 2: val_loss improved from 0.80563 to 0.56622, saving model to /content/driv
e/MyDrive/CN/savefile.h5
516/516 [==============================] - 68s 131ms/step - loss: 0.7195 - accurac
y: 0.7254 - val_loss: 0.5662 - val_accuracy: 0.7860 - lr: 9.0000e-04
Epoch 3/100
516/516 [==============================] - ETA: 0s - loss: 0.6391 - accuracy: 0.75
66
Epoch 3: val_loss did not improve from 0.56622
516/516 [==============================] - 67s 129ms/step - loss: 0.6391 - accurac
y: 0.7566 - val_loss: 0.5727 - val_accuracy: 0.7775 - lr: 9.0000e-04
Epoch 4/100
516/516 [==============================] - ETA: 0s - loss: 0.5961 - accuracy: 0.77
05
Epoch 4: val_loss improved from 0.56622 to 0.51166, saving model to /content/driv
e/MyDrive/CN/savefile.h5
516/516 [==============================] - 67s 131ms/step - loss: 0.5961 - accurac
y: 0.7705 - val_loss: 0.5117 - val_accuracy: 0.8050 - lr: 9.0000e-04
Epoch 5/100
516/516 [==============================] - ETA: 0s - loss: 0.5566 - accuracy: 0.78
58
Epoch 5: val_loss improved from 0.51166 to 0.50898, saving model to /content/driv
e/MyDrive/CN/savefile.h5
516/516 [==============================] - 68s 132ms/step - loss: 0.5566 - accurac
y: 0.7858 - val_loss: 0.5090 - val_accuracy: 0.8006 - lr: 9.0000e-04
Epoch 6/100
516/516 [==============================] - ETA: 0s - loss: 0.5381 - accuracy: 0.79
55
Epoch 6: val_loss improved from 0.50898 to 0.48907, saving model to /content/driv
e/MyDrive/CN/savefile.h5
516/516 [==============================] - 71s 137ms/step - loss: 0.5381 - accurac
y: 0.7955 - val_loss: 0.4891 - val_accuracy: 0.8177 - lr: 9.0000e-04
Epoch 7/100
516/516 [==============================] - ETA: 0s - loss: 0.5245 - accuracy: 0.80
21
Epoch 7: val_loss did not improve from 0.48907
516/516 [==============================] - 66s 128ms/step - loss: 0.5245 - accurac
y: 0.8021 - val_loss: 0.5018 - val_accuracy: 0.8048 - lr: 9.0000e-04
Epoch 8/100
516/516 [==============================] - ETA: 0s - loss: 0.5035 - accuracy: 0.80
84
Epoch 8: val_loss improved from 0.48907 to 0.46089, saving model to /content/driv
e/MyDrive/CN/savefile.h5
516/516 [==============================] - 67s 131ms/step - loss: 0.5035 - accurac
y: 0.8084 - val_loss: 0.4609 - val_accuracy: 0.8225 - lr: 9.0000e-04
Epoch 9/100
516/516 [==============================] - ETA: 0s - loss: 0.4892 - accuracy: 0.81
38
Epoch 9: val_loss improved from 0.46089 to 0.45497, saving model to /content/driv
e/MyDrive/CN/savefile.h5
516/516 [==============================] - 67s 130ms/step - loss: 0.4892 - accurac
y: 0.8138 - val_loss: 0.4550 - val_accuracy: 0.8255 - lr: 9.0000e-04
Epoch 10/100
516/516 [==============================] - ETA: 0s - loss: 0.4783 - accuracy: 0.81
72
```

```
Epoch 10: val_loss improved from 0.45497 to 0.44898, saving model to /content/driv
e/MyDrive/CN/savefile.h5
516/516 [==============================] - 68s 132ms/step - loss: 0.4783 - accurac
y: 0.8172 - val_loss: 0.4490 - val_accuracy: 0.8322 - lr: 9.0000e-04
Epoch 11/100
516/516 [==============================] - ETA: 0s - loss: 0.4589 - accuracy: 0.82
26
Epoch 11: val_loss did not improve from 0.44898
516/516 [==============================] - 67s 130ms/step - loss: 0.4589 - accurac
y: 0.8226 - val_loss: 0.4583 - val_accuracy: 0.8228 - lr: 8.1000e-04
Epoch 12/100
516/516 [==============================] - ETA: 0s - loss: 0.4540 - accuracy: 0.82
49
Epoch 12: val_loss improved from 0.44898 to 0.42693, saving model to /content/driv
e/MyDrive/CN/savefile.h5
516/516 [==============================] - 68s 131ms/step - loss: 0.4540 - accurac
y: 0.8249 - val_loss: 0.4269 - val_accuracy: 0.8431 - lr: 8.1000e-04
Epoch 13/100
516/516 [==============================] - ETA: 0s - loss: 0.4420 - accuracy: 0.83
14
Epoch 13: val_loss did not improve from 0.42693
516/516 [==============================] - 68s 132ms/step - loss: 0.4420 - accurac
y: 0.8314 - val_loss: 0.4522 - val_accuracy: 0.8326 - lr: 8.1000e-04
Epoch 14/100
516/516 [==============================] - ETA: 0s - loss: 0.4309 - accuracy: 0.83
26
Epoch 14: val_loss did not improve from 0.42693
516/516 [==============================] - 67s 130ms/step - loss: 0.4309 - accurac
y: 0.8326 - val_loss: 0.4305 - val_accuracy: 0.8331 - lr: 8.1000e-04
Epoch 15/100
516/516 [==============================] - ETA: 0s - loss: 0.4238 - accuracy: 0.83
67
Epoch 15: val_loss improved from 0.42693 to 0.41541, saving model to /content/driv
e/MyDrive/CN/savefile.h5

Epoch 15: ReduceLROnPlateau reducing learning rate to 8.100000559352338e-05.
516/516 [==============================] - 72s 139ms/step - loss: 0.4238 - accurac
y: 0.8367 - val_loss: 0.4154 - val_accuracy: 0.8363 - lr: 8.1000e-05
Epoch 16/100
516/516 [==============================] - ETA: 0s - loss: 0.3754 - accuracy: 0.85
36
Epoch 16: val_loss improved from 0.41541 to 0.37789, saving model to /content/driv
e/MyDrive/CN/savefile.h5
516/516 [==============================] - 68s 132ms/step - loss: 0.3754 - accurac
y: 0.8536 - val_loss: 0.3779 - val_accuracy: 0.8522 - lr: 8.1000e-05
Epoch 17/100
515/516 [============================>.] - ETA: 0s - loss: 0.3625 - accuracy: 0.85
72
Epoch 17: val_loss improved from 0.37789 to 0.36826, saving model to /content/driv
e/MyDrive/CN/savefile.h5
516/516 [==============================] - 79s 154ms/step - loss: 0.3624 - accurac
y: 0.8572 - val_loss: 0.3683 - val_accuracy: 0.8561 - lr: 8.1000e-05
Epoch 18/100
515/516 [============================>.] - ETA: 0s - loss: 0.3540 - accuracy: 0.85
98
Epoch 18: val_loss improved from 0.36826 to 0.36664, saving model to /content/driv
e/MyDrive/CN/savefile.h5
516/516 [==============================] - 70s 136ms/step - loss: 0.3541 - accurac
y: 0.8597 - val_loss: 0.3666 - val_accuracy: 0.8582 - lr: 8.1000e-05
Epoch 19/100
515/516 [============================>.] - ETA: 0s - loss: 0.3508 - accuracy: 0.86
19
Epoch 19: val_loss improved from 0.36664 to 0.36118, saving model to /content/driv
e/MyDrive/CN/savefile.h5
```

```
516/516 [==============================] - 71s 137ms/step - loss: 0.3508 - accurac
y: 0.8619 - val_loss: 0.3612 - val_accuracy: 0.8583 - lr: 8.1000e-05
Epoch 20/100
516/516 [==============================] - ETA: 0s - loss: 0.3486 - accuracy: 0.86
31
Epoch 20: val_loss did not improve from 0.36118
516/516 [==============================] - 67s 130ms/step - loss: 0.3486 - accurac
y: 0.8631 - val_loss: 0.3616 - val_accuracy: 0.8610 - lr: 8.1000e-05
Epoch 21/100
516/516 [==============================] - ETA: 0s - loss: 0.3427 - accuracy: 0.86
56
Epoch 21: val_loss improved from 0.36118 to 0.35891, saving model to /content/driv
e/MyDrive/CN/savefile.h5
516/516 [==============================] - 68s 132ms/step - loss: 0.3427 - accurac
y: 0.8656 - val_loss: 0.3589 - val_accuracy: 0.8618 - lr: 7.2900e-05
Epoch 22/100
516/516 [==============================] - ETA: 0s - loss: 0.3408 - accuracy: 0.86
64
Epoch 22: val_loss improved from 0.35891 to 0.35843, saving model to /content/driv
e/MyDrive/CN/savefile.h5
516/516 [==============================] - 68s 132ms/step - loss: 0.3408 - accurac
y: 0.8664 - val_loss: 0.3584 - val_accuracy: 0.8638 - lr: 7.2900e-05
Epoch 23/100
516/516 [==============================] - ETA: 0s - loss: 0.3384 - accuracy: 0.86
59
Epoch 23: val_loss improved from 0.35843 to 0.35632, saving model to /content/driv
e/MyDrive/CN/savefile.h5
516/516 [==============================] - 75s 145ms/step - loss: 0.3384 - accurac
y: 0.8659 - val_loss: 0.3563 - val_accuracy: 0.8659 - lr: 7.2900e-05
Epoch 24/100
516/516 [==============================] - ETA: 0s - loss: 0.3365 - accuracy: 0.86
85
Epoch 24: val_loss improved from 0.35632 to 0.35552, saving model to /content/driv
e/MyDrive/CN/savefile.h5
516/516 [==============================] - 69s 134ms/step - loss: 0.3365 - accurac
y: 0.8685 - val_loss: 0.3555 - val_accuracy: 0.8642 - lr: 7.2900e-05
Epoch 25/100
516/516 [==============================] - ETA: 0s - loss: 0.3330 - accuracy: 0.86
93
Epoch 25: val_loss improved from 0.35552 to 0.35257, saving model to /content/driv
e/MyDrive/CN/savefile.h5
516/516 [==============================] - 70s 135ms/step - loss: 0.3330 - accurac
y: 0.8693 - val_loss: 0.3526 - val_accuracy: 0.8650 - lr: 7.2900e-05
Epoch 26/100
515/516 [============================>.] - ETA: 0s - loss: 0.3305 - accuracy: 0.86
82
Epoch 26: val_loss improved from 0.35257 to 0.35067, saving model to /content/driv
e/MyDrive/CN/savefile.h5
516/516 [==============================] - 70s 135ms/step - loss: 0.3305 - accurac
y: 0.8682 - val_loss: 0.3507 - val_accuracy: 0.8685 - lr: 7.2900e-05
Epoch 27/100
515/516 [============================>.] - ETA: 0s - loss: 0.3287 - accuracy: 0.87
09
Epoch 27: val_loss did not improve from 0.35067
516/516 [==============================] - 67s 129ms/step - loss: 0.3288 - accurac
y: 0.8708 - val_loss: 0.3514 - val_accuracy: 0.8663 - lr: 7.2900e-05
Epoch 28/100
515/516 [============================>.] - ETA: 0s - loss: 0.3283 - accuracy: 0.87
15
Epoch 28: val_loss improved from 0.35067 to 0.34967, saving model to /content/driv
e/MyDrive/CN/savefile.h5
516/516 [==============================] - 70s 136ms/step - loss: 0.3282 - accurac
y: 0.8715 - val_loss: 0.3497 - val_accuracy: 0.8657 - lr: 7.2900e-05
Epoch 29/100
```

```
515/516 [============================>.] - ETA: 0s - loss: 0.3251 - accuracy: 0.87
09
Epoch 29: val_loss improved from 0.34967 to 0.34699, saving model to /content/driv
e/MyDrive/CN/savefile.h5

Epoch 29: ReduceLROnPlateau reducing learning rate to 7.2900002123788e-06.
516/516 [==============================] - 68s 132ms/step - loss: 0.3250 - accurac
y: 0.8710 - val_loss: 0.3470 - val_accuracy: 0.8664 - lr: 7.2900e-06
Epoch 30/100
515/516 [============================>.] - ETA: 0s - loss: 0.3210 - accuracy: 0.87
34
Epoch 30: val_loss improved from 0.34699 to 0.34644, saving model to /content/driv
e/MyDrive/CN/savefile.h5
516/516 [==============================] - 70s 136ms/step - loss: 0.3211 - accurac
y: 0.8734 - val_loss: 0.3464 - val_accuracy: 0.8687 - lr: 7.2900e-06
Epoch 31/100
515/516 [============================>.] - ETA: 0s - loss: 0.3205 - accuracy: 0.87
29
Epoch 31: val_loss did not improve from 0.34644
516/516 [==============================] - 67s 129ms/step - loss: 0.3204 - accurac
y: 0.8730 - val_loss: 0.3467 - val_accuracy: 0.8692 - lr: 6.5610e-06
Epoch 32/100
515/516 [============================>.] - ETA: 0s - loss: 0.3195 - accuracy: 0.87
33
Epoch 32: val_loss improved from 0.34644 to 0.34628, saving model to /content/driv
e/MyDrive/CN/savefile.h5
516/516 [==============================] - 70s 135ms/step - loss: 0.3195 - accurac
y: 0.8734 - val_loss: 0.3463 - val_accuracy: 0.8685 - lr: 6.5610e-06
Epoch 33/100
515/516 [============================>.] - ETA: 0s - loss: 0.3166 - accuracy: 0.87
50
Epoch 33: val_loss did not improve from 0.34628
516/516 [==============================] - 68s 132ms/step - loss: 0.3167 - accurac
y: 0.8750 - val_loss: 0.3465 - val_accuracy: 0.8687 - lr: 6.5610e-06
Epoch 34/100
515/516 [============================>.] - ETA: 0s - loss: 0.3199 - accuracy: 0.87
53
Epoch 34: val_loss did not improve from 0.34628

Epoch 34: ReduceLROnPlateau reducing learning rate to 6.561000191140921e-07.
516/516 [==============================] - 68s 131ms/step - loss: 0.3204 - accurac
y: 0.8752 - val_loss: 0.3465 - val_accuracy: 0.8686 - lr: 6.5610e-07
Epoch 35/100
516/516 [==============================] - ETA: 0s - loss: 0.3179 - accuracy: 0.87
38
Epoch 35: val_loss improved from 0.34628 to 0.34544, saving model to /content/driv
e/MyDrive/CN/savefile.h5
516/516 [==============================] - 69s 134ms/step - loss: 0.3179 - accurac
y: 0.8738 - val_loss: 0.3454 - val_accuracy: 0.8683 - lr: 6.5610e-07
Epoch 36/100
516/516 [==============================] - ETA: 0s - loss: 0.3168 - accuracy: 0.87
48
Epoch 36: val_loss did not improve from 0.34544
516/516 [==============================] - 69s 133ms/step - loss: 0.3168 - accurac
y: 0.8748 - val_loss: 0.3457 - val_accuracy: 0.8681 - lr: 6.5610e-07
Epoch 37/100
516/516 [==============================] - ETA: 0s - loss: 0.3169 - accuracy: 0.87
47
Epoch 37: val_loss did not improve from 0.34544

Epoch 37: ReduceLROnPlateau reducing learning rate to 6.561000418514596e-08.
516/516 [==============================] - 67s 131ms/step - loss: 0.3169 - accurac
y: 0.8747 - val_loss: 0.3457 - val_accuracy: 0.8685 - lr: 6.5610e-08
Epoch 38/100
```

```
516/516 [==============================] - ETA: 0s - loss: 0.3198 - accuracy: 0.87
40
Epoch 38: val_loss did not improve from 0.34544
516/516 [==============================] - 67s 130ms/step - loss: 0.3198 - accurac
y: 0.8740 - val_loss: 0.3462 - val_accuracy: 0.8686 - lr: 6.5610e-08
Epoch 39/100
516/516 [==============================] - ETA: 0s - loss: 0.3174 - accuracy: 0.87
38
Epoch 39: val_loss did not improve from 0.34544
516/516 [==============================] - 67s 130ms/step - loss: 0.3174 - accurac
y: 0.8738 - val_loss: 0.3461 - val_accuracy: 0.8683 - lr: 6.5610e-08
Epoch 40/100
516/516 [==============================] - ETA: 0s - loss: 0.3175 - accuracy: 0.87
50
Epoch 40: val_loss did not improve from 0.34544

Epoch 40: ReduceLROnPlateau reducing learning rate to 6.561000276406049e-09.
516/516 [==============================] - 68s 131ms/step - loss: 0.3175 - accurac
y: 0.8750 - val_loss: 0.3457 - val_accuracy: 0.8689 - lr: 6.5610e-09
Epoch 41/100
516/516 [==============================] - ETA: 0s - loss: 0.3152 - accuracy: 0.87
56
Epoch 41: val_loss did not improve from 0.34544
Restoring model weights from the end of the best epoch: 31.
516/516 [==============================] - 68s 132ms/step - loss: 0.3152 - accurac
y: 0.8756 - val_loss: 0.3464 - val_accuracy: 0.8687 - lr: 5.9049e-09
Epoch 41: 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

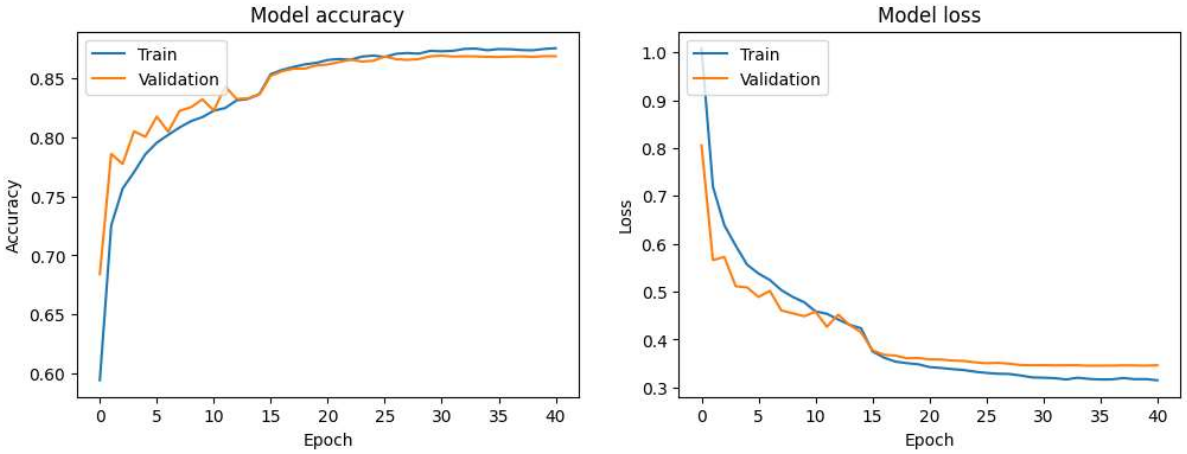
```
conv1d_10 (Conv1D)           (None, 5, 32)             3104

batch_normalization_10 (Bat  (None, 5, 32)             128
chNormalization)

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

batch_normalization_11 (Bat  (None, 5, 32)             128
chNormalization)

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 (Bat  (None, 64)                256
chNormalization)

dense (Dense)                (None, 128)               8320

batch_normalization_13 (Bat  (None, 128)               512
chNormalization)

dense_1 (Dense)              (None, 128)               16512

dense_2 (Dense)              (None, 128)               16512

dense_3 (Dense)              (None, 5)                 645

=================================================================
Total params: 362,757
Trainable params: 360,581
Non-trainable params: 2,176
_________________________________________________________________
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)
```

```
228/228 [==============================] - 7s 16ms/step
Accuracy: 0.8623676612127045
Classification Report:
              precision    recall  f1-score   support

      Benign       0.94      0.97      0.96      1469
     Malware       0.85      0.92      0.88      1378
    Phishing       0.84      0.68      0.75      1322
        Spam       0.90      0.90      0.90      1603
  Defacement       0.77      0.83      0.80      1501

    accuracy                           0.86      7273
   macro avg       0.86      0.86      0.86      7273
weighted avg       0.86      0.86      0.86      7273
```

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/retrain1.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-6-37a14c1ec82b>: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:])
```

Re-training of the model

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

model = load_model('/content/drive/MyDrive/CN/supermodelx1.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/retrainmodel1.h5')
```

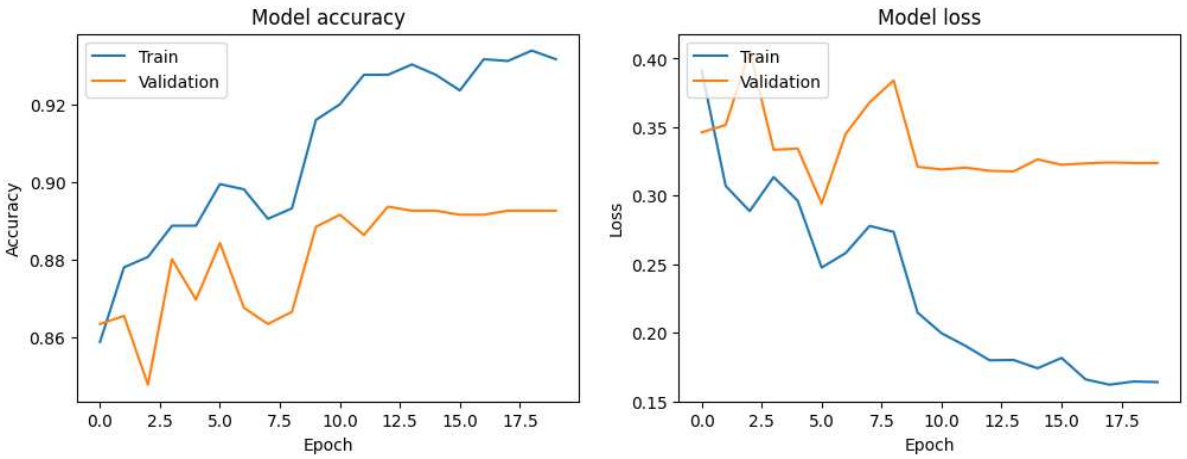
```
Epoch 1/20
70/70 [==============================] - ETA: 0s - loss: 0.3912 - accuracy: 0.8589
Epoch 1: val_loss improved from inf to 0.34620, saving model to /content/drive/MyD
rive/CN/savefile.h5
70/70 [==============================] - 27s 126ms/step - loss: 0.3912 - accuracy:
0.8589 - val_loss: 0.3462 - val_accuracy: 0.8635 - lr: 9.0000e-04
Epoch 2/20
70/70 [==============================] - ETA: 0s - loss: 0.3072 - accuracy: 0.8781
Epoch 2: val_loss did not improve from 0.34620
70/70 [==============================] - 7s 105ms/step - loss: 0.3072 - accuracy:
0.8781 - val_loss: 0.3516 - val_accuracy: 0.8656 - lr: 9.0000e-04
Epoch 3/20
70/70 [==============================] - ETA: 0s - loss: 0.2888 - accuracy: 0.8808
Epoch 3: val_loss did not improve from 0.34620
70/70 [==============================] - 5s 70ms/step - loss: 0.2888 - accuracy:
0.8808 - val_loss: 0.4059 - val_accuracy: 0.8479 - lr: 9.0000e-04
Epoch 4/20
70/70 [==============================] - ETA: 0s - loss: 0.3136 - accuracy: 0.8888
Epoch 4: val_loss improved from 0.34620 to 0.33351, saving model to /content/driv
e/MyDrive/CN/savefile.h5
70/70 [==============================] - 6s 91ms/step - loss: 0.3136 - accuracy:
0.8888 - val_loss: 0.3335 - val_accuracy: 0.8802 - lr: 9.0000e-04
Epoch 5/20
70/70 [==============================] - ETA: 0s - loss: 0.2964 - accuracy: 0.8888
Epoch 5: val_loss did not improve from 0.33351
70/70 [==============================] - 8s 107ms/step - loss: 0.2964 - accuracy:
0.8888 - val_loss: 0.3344 - val_accuracy: 0.8698 - lr: 9.0000e-04
Epoch 6/20
70/70 [==============================] - ETA: 0s - loss: 0.2476 - accuracy: 0.8996
Epoch 6: val_loss improved from 0.33351 to 0.29425, saving model to /content/driv
e/MyDrive/CN/savefile.h5
70/70 [==============================] - 6s 90ms/step - loss: 0.2476 - accuracy:
0.8996 - val_loss: 0.2942 - val_accuracy: 0.8844 - lr: 9.0000e-04
Epoch 7/20
70/70 [==============================] - ETA: 0s - loss: 0.2582 - accuracy: 0.8982
Epoch 7: val_loss did not improve from 0.29425
70/70 [==============================] - 5s 68ms/step - loss: 0.2582 - accuracy:
0.8982 - val_loss: 0.3451 - val_accuracy: 0.8677 - lr: 9.0000e-04
Epoch 8/20
70/70 [==============================] - ETA: 0s - loss: 0.2780 - accuracy: 0.8906
Epoch 8: val_loss did not improve from 0.29425
70/70 [==============================] - 7s 96ms/step - loss: 0.2780 - accuracy:
0.8906 - val_loss: 0.3681 - val_accuracy: 0.8635 - lr: 9.0000e-04
Epoch 9/20
70/70 [==============================] - ETA: 0s - loss: 0.2736 - accuracy: 0.8933
Epoch 9: val_loss did not improve from 0.29425

Epoch 9: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.
70/70 [==============================] - 8s 108ms/step - loss: 0.2736 - accuracy:
0.8933 - val_loss: 0.3841 - val_accuracy: 0.8667 - lr: 9.0000e-05
Epoch 10/20
70/70 [==============================] - ETA: 0s - loss: 0.2147 - accuracy: 0.9161
Epoch 10: val_loss did not improve from 0.29425
70/70 [==============================] - 5s 70ms/step - loss: 0.2147 - accuracy:
0.9161 - val_loss: 0.3211 - val_accuracy: 0.8885 - lr: 9.0000e-05
Epoch 11/20
70/70 [==============================] - ETA: 0s - loss: 0.1997 - accuracy: 0.9201
Epoch 11: val_loss did not improve from 0.29425
70/70 [==============================] - 5s 70ms/step - loss: 0.1997 - accuracy:
0.9201 - val_loss: 0.3192 - val_accuracy: 0.8917 - lr: 8.1000e-05
Epoch 12/20
70/70 [==============================] - ETA: 0s - loss: 0.1905 - accuracy: 0.9277
Epoch 12: val_loss did not improve from 0.29425
70/70 [==============================] - 8s 118ms/step - loss: 0.1905 - accuracy:
```

```
0.9277 - val_loss: 0.3205 - val_accuracy: 0.8865 - lr: 8.1000e-05
Epoch 13/20
70/70 [==============================] - ETA: 0s - loss: 0.1800 - accuracy: 0.9277
Epoch 13: val_loss did not improve from 0.29425
70/70 [==============================] - 7s 97ms/step - loss: 0.1800 - accuracy:
0.9277 - val_loss: 0.3182 - val_accuracy: 0.8938 - lr: 8.1000e-05
Epoch 14/20
70/70 [==============================] - ETA: 0s - loss: 0.1802 - accuracy: 0.9304
Epoch 14: val_loss did not improve from 0.29425
70/70 [==============================] - 5s 68ms/step - loss: 0.1802 - accuracy:
0.9304 - val_loss: 0.3178 - val_accuracy: 0.8927 - lr: 8.1000e-05
Epoch 15/20
70/70 [==============================] - ETA: 0s - loss: 0.1742 - accuracy: 0.9277
Epoch 15: val_loss did not improve from 0.29425
70/70 [==============================] - 6s 93ms/step - loss: 0.1742 - accuracy:
0.9277 - val_loss: 0.3265 - val_accuracy: 0.8927 - lr: 8.1000e-05
Epoch 16/20
70/70 [==============================] - ETA: 0s - loss: 0.1817 - accuracy: 0.9237
Epoch 16: val_loss did not improve from 0.29425

Epoch 16: ReduceLROnPlateau reducing learning rate to 8.099999831756577e-06.
70/70 [==============================] - 8s 109ms/step - loss: 0.1817 - accuracy:
0.9237 - val_loss: 0.3226 - val_accuracy: 0.8917 - lr: 8.1000e-06
Epoch 17/20
70/70 [==============================] - ETA: 0s - loss: 0.1661 - accuracy: 0.9317
Epoch 17: val_loss did not improve from 0.29425
70/70 [==============================] - 5s 75ms/step - loss: 0.1661 - accuracy:
0.9317 - val_loss: 0.3236 - val_accuracy: 0.8917 - lr: 8.1000e-06
Epoch 18/20
70/70 [==============================] - ETA: 0s - loss: 0.1622 - accuracy: 0.9312
Epoch 18: val_loss did not improve from 0.29425
70/70 [==============================] - 5s 67ms/step - loss: 0.1622 - accuracy:
0.9312 - val_loss: 0.3243 - val_accuracy: 0.8927 - lr: 8.1000e-06
Epoch 19/20
70/70 [==============================] - ETA: 0s - loss: 0.1645 - accuracy: 0.9339
Epoch 19: val_loss did not improve from 0.29425

Epoch 19: ReduceLROnPlateau reducing learning rate to 8.099999831756577e-07.
70/70 [==============================] - 7s 100ms/step - loss: 0.1645 - accuracy:
0.9339 - val_loss: 0.3238 - val_accuracy: 0.8927 - lr: 8.1000e-07
Epoch 20/20
70/70 [==============================] - ETA: 0s - loss: 0.1641 - accuracy: 0.9317
Epoch 20: val_loss did not improve from 0.29425
70/70 [==============================] - 7s 102ms/step - loss: 0.1641 - accuracy:
0.9317 - val_loss: 0.3239 - val_accuracy: 0.8927 - lr: 8.1000e-07
```

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()
```



In [ ]:
```
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"]

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

```
25/25 [==============================] - 3s 16ms/step
Accuracy: 0.89625
Classification Report:
              precision    recall  f1-score   support

      Benign       0.97      0.95      0.96       195
     Malware       0.93      0.91      0.92       210
    Phishing       0.82      0.86      0.84       192
        Spam       0.87      0.86      0.87       203

    accuracy                           0.90       800
   macro avg       0.90      0.90      0.90       800
weighted avg       0.90      0.90      0.90       800
```

Confusion Matrix

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()
```