***Feed-Forward Neural Networks*** are a type of artificial neural network architecture that has been extensively researched and utilized in different fields. The name implies that the data flows in one direction, from the input layer through one or more hidden layers, and finally to the output layer. These networks are defined by the unidirectional flow of information without any loops or feedback connections.

- ***The input*** layer acts as the entry point for raw data from outside the system. Various forms of data can be used, including numerical values, images, text, or any other relevant representation. The input layer is responsible for distributing this data to the subsequent hidden layers.
- The input data is processed by ***hidden layers***, which can be one or multiple. Weighted inputs are received by every neuron in a hidden layer, and they calculate a weighted sum before applying a non-linear activation function to it. The activation function allows the network to model complex, non-linear relationships within the data by introducing non-linearity into it. Feed-forward networks use sigmoid function, rectified linear unit (ReLU), and hyperbolic tangent (tanh) as common activation functions.
- Transformation data is mapped to the desired output format in the ***output layer***, which is the final stage of the network. In the case of classification tasks, such as distinguishing between malicious and benign websites, the output layer would typically have one neuron per class, with the highest output value indicating the predicted class. The output layer may have just one neuron for regression tasks that aim to predict a continuous value.

Training is a process that allows feed-forward neural networks to learn from data, which is one of their key strengths. Optimization algorithms, such as gradient descent or its variants, are used to iteratively adjust the weights of the network's connections between neurons during training. The goal is to minimize the disparity between the network's output and the desired output, as per the training data. Backpropagation is a process in which an error is propagated backward through the network, and the weights are updated accordingly.

Classification, regression, clustering, and pattern recognition are just some of the tasks that feed-forward neural networks can be used for. They are apt for problems where traditional linear models may struggle due to their versatility and ability to automatically extract relevant features from the input data. It's important to note that feed-forward neural networks can recognize patterns powerfully, but they also have limitations. It is possible for them to be computationally expensive, particularly for large datasets or deep networks with many layers. Furthermore, they are susceptible to overfitting, in which the network learns the training data too well and fails to generalize to new, unseen data. Regularization, dropout, and early stopping are techniques that can help mitigate this issue.

Fee-forward neural networks can be trained on a dataset that contains features extracted from both types of websites in order to analyze malicious and benign websites. There are many features that could include URL patterns, website content, network traffic patterns, or any other relevant indicators. By learning from this data, the neural network can develop an understanding of the patterns and characteristics that distinguish malicious websites from benign ones, enabling accurate classification during the operational phase.

**Application: Malicious Website Detection**

Feed-forward neural networks can be used to detect malicious websites by training the model on features extracted from website URLs and content, such as the presence of certain keywords, domain age, IP reputation, and more.

**Sample Data**

For this example, let's assume we have a dataset containing features extracted from websites, such as the domain age, the number of special characters in the URL, the presence of certain keywords, and whether the website is malicious or not. The dataset might look something like this:

**Python code:**

This code does:

1. Import and review data set;
2. Remove individual and redundant features;
3. Encode categorical data;
4. Handle missing values - change "none" with data.median;
5. Normalize data;
6. Split dataset in training and test datasets;
7. Define model architecture;
8. Compile the model;
9. Train the model on training dataset;
10. Use model to validate test dataset;
11. Evaluate model;
12. Analyze test dataset using ROC/AUC, confusion matrix, predicting probability for test set, classification report;

```
import pandas as pd
import seaborn as sns
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, roc_curve
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential

data = pd.read_csv("dataset.csv")
print(data.head())

features_to_remove = ['URL', 'WHOIS_STATEPRO', 'WHOIS_REGDATE', 'WHOIS_UPDATED_DATE']
data.drop(features_to_remove, axis=1, inplace=True)

data = pd.get_dummies(data, columns=['CHARSET', 'SERVER', 'WHOIS_COUNTRY'])

data.fillna(data.median(), inplace=True)

from sklearn.preprocessing import MinMaxScaler
```

```
data[data.columns[:-1]] = scaler.fit_transform(data[data.columns[:-1]])

from sklearn.model_selection import train_test_split

X = data.drop('Type', axis=1)
y = data['Type']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential([
    Dense(32, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(16, activation='relu'),
    Dense(1, activation='sigmoid')
])

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

history = model.fit(X_train, y_train, epochs=32, batch_size=32, validation_split=0.2)

y_pred = model.predict(X_test)

loss, accuracy = model.evaluate(X_test, y_test)
print(f'Test Loss: {loss}, Test Accuracy: {accuracy}')

roc_auc = roc_auc_score(y_test, y_pred)
print(f'ROC AUC: {roc_auc}')

conf_matrix = confusion_matrix(y_test, y_pred.round())
print('Confusion Matrix:')
print(conf_matrix)

class_report = classification_report(y_test, y_pred.round())
print('Classification Report:')
print(class_report)

plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.show()

fpr, tpr, _ = roc_curve(y_test, y_pred)
plt.plot(fpr, tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.show()
```

| URL | URL_LENG | NUMBER_ | CHARSET | SERVER | CONTENT | WHOIS_C | WHOIS_ST | WHOIS_REGDATE | WHOIS_UPDATED_DATE | TCP_CON | DIST_REM | REMOTE_I | APP_BYTE | SOURCE_A | REMOTE_ | SOURCE_A | REMOTE_ | APP_PACI | DNS_QUE | Type |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| B0_23 | 23 | 6 | None | None | NA | US | TX | 30/07/1996 0:00 | 4/7/2016 0:00 | 19 | 7 | 6 | 2404 | 23 | 20 | 6179 | 2684 | 23 | 4 | 0 |
| B0_241 | 23 | 6 | UTF-8 | None | NA | GB | None | 9/5/2008 0:00 | 27/11/2015 0:00 | 19 | 14 | 6 | 1980 | 23 | 25 | 5737 | 2276 | 23 | 4 | 0 |
| B0_285 | 23 | 6 | UTF-8 | None | NA | US | WI | 23/04/1999 0:00 | 25/07/2016 0:00 | 4 | 0 | 1 | 519 | 8 | 8 | 1138 | 823 | 8 | 4 | 0 |
| B0_465 | 23 | 6 | UTF-8 | cloudflare | NA | US | Oregon | 4/2/1997 0:00 | 8/11/2014 0:00 | 10 | 0 | 5 | 1186 | 14 | 15 | 1900 | 1476 | 14 | 4 | 0 |
| B0_599 | 23 | 6 | ISO-8859- | nginx/1.1 | NA | IN | Andhra Pr | 13/02/2003 0:00 | 14/02/2017 0:00 | 43 | 4 | 4 | 5738 | 47 | 50 | 56926 | 6046 | 47 | 4 | 0 |
| B0_614 | 23 | 6 | iso-8859-1 | Apache | 240 | US | CA | 17/05/2008 0:00 | 18/05/2016 0:00 | 8 | 0 | 2 | 723 | 10 | 11 | 1837 | 871 | 10 | 2 | 0 |
| B0_622 | 23 | 6 | ISO-8859- | Apache | 3985 | CA | AB | 30/05/2002 0:00 | 31/05/2016 0:00 | 40 | 17 | 16 | 3285 | 46 | 45 | 12003 | 3729 | 46 | 6 | 0 |
| B0_790 | 23 | 6 | us-ascii | Microsoft | 324 | None | None | 20/10/2005 0:00 | 30/09/2016 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| M0_175 | 24 | 8 | ISO-8859- | nginx/1.6. | 6173 | None | None | None | None | 2 | 0 | 1 | 132 | 2 | 5 | 318 | 132 | 2 | 0 | 1 |
| M0_89 | 24 | 9 | iso-8859-1 | Apache | 319 | IN | Tamil Nad | None | None | 7 | 0 | 2 | 720 | 9 | 10 | 1269 | 868 | 9 | 2 | 1 |
| B0_108 | 24 | 6 | utf-8 | openresty | NA | LU | None | 7/1/2006 0:00 | 14/12/2016 0:00 | 84 | 54 | 5 | 10490 | 96 | 106 | 106925 | 11482 | 96 | 12 | 0 |
| B0_126 | 24 | 6 | ISO-8859- | None | 34 | US | VA | 5/3/1996 5:00 | 27/05/2016 14:46 | 65 | 3 | 5 | 6286 | 69 | 75 | 90508 | 6596 | 69 | 4 | 0 |
| B0_171 | 24 | 7 | utf-8 | nginx | NA | US | NY | 23/03/1995 0:00 | 3/3/2017 0:00 | 14 | 6 | 6 | 1907 | 18 | 18 | 5601 | 2171 | 18 | 4 | 0 |
| B0_2301 | 24 | 6 | utf-8 | Heptu we | 1778 | None | None | 10/1/1998 0:00 | 11/11/2012 0:00 | 7 | 7 | 3 | 618 | 7 | 9 | 562 | 618 | 7 | 0 | 0 |
| B0_2313 | 24 | 6 | utf-8 | Pepyaka/ | NA | None | None | 27/04/2016 0:00 | 29/04/2017 0:00 | 6 | 3 | 2 | 528 | 6 | 4 | 244 | 528 | 6 | 0 | 0 |
| B0_2315 | 24 | 6 | us-ascii | Microsoft | 324 | CA | quebec | 7/4/2011 0:00 | 8/4/2017 0:00 | 6 | 4 | 4 | 564 | 6 | 8 | 508 | 564 | 6 | 0 | 0 |
| B0_282 | 24 | 8 | iso-8859-1 | Apache/2. | 224 | US | MA | 26/02/2009 0:00 | 6/1/2017 0:00 | 19 | 2 | 4 | 2131 | 23 | 23 | 9471 | 2441 | 23 | 4 | 0 |
| B0_321 | 24 | 6 | utf-8 | nginx/1.8. | NA | CA | ON | 3/7/2002 0:00 | 2/7/2016 0:00 | 12 | 2 | 7 | 984 | 16 | 16 | 1839 | 1292 | 16 | 4 | 0 |
| B0_326 | 24 | 7 | iso-8859-1 | Apache | 318 | None | None | None | None | 21 | 1 | 8 | 1606 | 25 | 13 | 1276 | 1914 | 25 | 4 | 0 |
| B0_483 | 24 | 7 | utf-8 | nginx/1.1( | NA | None | None | 21/02/1995 0:00 | 19/02/2017 0:00 | 16 | 6 | 8 | 2279 | 22 | 22 | 2562 | 2729 | 22 | 6 | 0 |