In [5]:
```
#imports
%matplotlib inline
import pandas as pd
import plotly.express as px

# 2 lines below for html export
import plotly.io as pio
pio.renderers.default ='notebook'

# 2 lines below for PDF export
!pip install Pyppeteer
!pyppeteer-install
import random
import tensorflow as tf
import numpy as np
import pandas as pd
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense, Flatten, Activation, Conv1D, MaxPooling1D, Dropout, Lambda, LeakyReLU
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten, Dense
```

```
Requirement already satisfied: Pyppeteer in /home/mahinur/miniconda3/lib/python3.10/site-packages (1.0.2)
Requirement already satisfied: appdirs<2.0.0,>=1.4.3 in /home/mahinur/miniconda3/lib/python3.10/site-packages (from Pyppeteer) (1.4.4)
Requirement already satisfied: certifi>=2021 in /home/mahinur/.local/lib/python3.10/site-packages (from Pyppeteer) (2022.12.7)
Requirement already satisfied: importlib-metadata>=1.4 in /home/mahinur/miniconda3/lib/python3.10/site-packages (from Pyppeteer) (6.8.0)
Requirement already satisfied: pyee<9.0.0,>=8.1.0 in /home/mahinur/miniconda3/lib/python3.10/site-packages (from Pyppeteer) (8.2.2)
Requirement already satisfied: tqdm<5.0.0,>=4.42.1 in /home/mahinur/.local/lib/python3.10/site-packages (from Pyppeteer) (4.65.0)
Requirement already satisfied: urllib3<2.0.0,>=1.25.8 in /home/mahinur/miniconda3/lib/python3.10/site-packages (from Pyppeteer) (1.26.15)
Requirement already satisfied: websockets<11.0,>=10.0 in /home/mahinur/miniconda3/lib/python3.10/site-packages (from Pyppeteer) (10.4)
Requirement already satisfied: zipp>=0.5 in /home/mahinur/miniconda3/lib/python3.10/site-packages (from importlib-metadata>=1.4->Pyppeteer) (3.16.1)
chromium is already installed.
```

In [6]:
```
data1 = pd.read_csv('/home/mahinur/Desktop/CSV_1.csv')
data2 = pd.read_csv('/home/mahinur/Desktop/CSV_2.csv')
```

In [7]:
```
merged_data = pd.merge(data1, data2, on='sid')

numeric_columns = merged_data.select_dtypes(include=[float, int]).columns
merged_data = merged_data[numeric_columns]

# Normalize the merged data using Min-Max scaling
scaler = MinMaxScaler()
normalized_data = pd.DataFrame(scaler.fit_transform(merged_data), columns=merged_data.columns)

# Save the normalized data to a new CSV file
normalized_data.to_csv('/home/mahinur/Desktop/normalized_data.csv', index=False)
```

In [8]:
```
# Load the normalized data from the CSV file
normalized_data = pd.read_csv('/home/mahinur/Desktop/normalized_data.csv')

# Extract the features (X) and target (y) columns
X = normalized_data.drop('output1', axis=1).values
y = normalized_data['output1'].values

# Reshape X to match the expected input shape of the 1D CNN
X = np.reshape(X, (X.shape[0], X.shape[1], 1))

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a 1D CNN model
model = Sequential()
model.add(Conv1D(filters=32, kernel_size=3, activation='relu', input_shape=(X.shape[1], 1)))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

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

# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=16, validation_data=(X_test, y_test))

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Loss: {loss:.4f}")
print(f"Test Accuracy: {accuracy:.4f}")
```

```
Epoch 1/10
374/374 [==============================] - 1s 2ms/step - loss: 0.2771 - accuracy: 0.9101 - val_loss: 0.2603 - val_accuracy: 0.9177
Epoch 2/10
374/374 [==============================] - 1s 2ms/step - loss: 0.2733 - accuracy: 0.9121 - val_loss: 0.2531 - val_accuracy: 0.9177
Epoch 3/10
374/374 [==============================] - 1s 2ms/step - loss: 0.2724 - accuracy: 0.9121 - val_loss: 0.2560 - val_accuracy: 0.9177
Epoch 4/10
374/374 [==============================] - 1s 2ms/step - loss: 0.2726 - accuracy: 0.9121 - val_loss: 0.2529 - val_accuracy: 0.9177
Epoch 5/10
374/374 [==============================] - 1s 2ms/step - loss: 0.2700 - accuracy: 0.9121 - val_loss: 0.2526 - val_accuracy: 0.9177
Epoch 6/10
374/374 [==============================] - 1s 2ms/step - loss: 0.2713 - accuracy: 0.9121 - val_loss: 0.2638 - val_accuracy: 0.9177
Epoch 7/10
374/374 [==============================] - 1s 2ms/step - loss: 0.2702 - accuracy: 0.9121 - val_loss: 0.2523 - val_accuracy: 0.9177
Epoch 8/10
374/374 [==============================] - 1s 2ms/step - loss: 0.2692 - accuracy: 0.9121 - val_loss: 0.2553 - val_accuracy: 0.9177
Epoch 9/10
374/374 [==============================] - 1s 2ms/step - loss: 0.2680 - accuracy: 0.9121 - val_loss: 0.2520 - val_accuracy: 0.9157
Epoch 10/10
374/374 [==============================] - 1s 2ms/step - loss: 0.2689 - accuracy: 0.9125 - val_loss: 0.2536 - val_accuracy: 0.9177
47/47 [==============================] - 0s 1ms/step - loss: 0.2536 - accuracy: 0.9177
```

```
Test Loss: 0.2536
Test Accuracy: 0.9177
```

In [ ]: