Appendix A. Algorithms and Coding

A.1. Common Preconditions for Construction of Models

A.1.1. Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Dropout
from keras.layers import Conv1D,MaxPooling1D
from numpy import ndarray
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from tensorflow import keras
from keras.layers.core import Activation, Dense
from keras.preprocessing import sequence
from sklearn.model_selection import train_test_split
from keras.models import load_model
from tensorflow.keras.callbacks import TensorBoard
```

A.1.2. Importing Modeling Data of this Study and Confirming the Number of Samples and Variables

```
df = pd.read_excel ("BD.xlsx")
df.shape
```

A.1.3. Data Pre-Processing

Dividing Data into Training Dataset, Test Dataset, and Validation Dataset

```
df_num = df.shape (0)
indexes = np.random.permutation (df_num)
train_indexes = indexes (:int (df_num *0.7))
test_indexes = indexes (int(df_num *0.7):)
train_df = df.loc (train_indexes)
test_df= df.loc(test_indexes)
```

Normalize Data by Converting their Values to 0 to 1

```
scaler = MinMaxScaler (feature_range= (0,1))
df= scaler.fit_transform (df)
```

Defining dependent and independent variables

```
x_train = np.array (train_df.drop ('FC', axis = 'columns'))
y_train = np.array (train_df('FC'))
x_test = np.array (test_df.drop ('FC', axis = 'columns'))
y_test = np.array (test_df ('FC'))
```

Transforming Data Dimensions to Conform to the Model Input Conditions

```
x_train = np.reshape (x_train, (x_train.shape (0), 1, x_train.shape (1)))
x_test = np.reshape (x_test, (x_test.shape (0), 1, x_test.shape (1)))
```

A.2. Modeling–Inputting when Individual Models are Constructed

A.2.1. Deep Neural Networks

```
model = Sequential ()
model.add (Flatten(input_shape = (x_train.shape (1), x_train.shape (2))))
model.add (Dense (16, activation = 'relu'))
model.add (Dropout(0.25))
model.add (Dense(16, activation = 'relu'))
model.add (Dropout (0.25))
model.add (Dense (8, activation = 'relu'))
model.add (Dropout (0.25))
model.add (Dense(1, activation = 'sigmoid'))
model.compile (loss = "binary_crossentropy", optimizer = "adam",metrics = ('accuracy'))
model.summary ()
```

A.2.2. Deep Neural Networks without Selecting Variables

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 23)                0
_________________________________________________________________
dense (Dense)                (None, 32)                768
_________________________________________________________________
dropout (Dropout)            (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 264
_________________________________________________________________
dropout_1 (Dropout)          (None, 8)                 0
_________________________________________________________________
dense_2 (Dense)              (None, 8)                 72
_________________________________________________________________
dropout_2 (Dropout)          (None, 8)                 0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 9
=================================================================
Total params: 1,113
Trainable params: 1,113
Non-trainable params: 0
_________________________________________________________________
```

A.2.3. Deep Neural Networks after Selection by CHAID

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 9)                 0
_________________________________________________________________
dense (Dense)                (None, 16)                160
_________________________________________________________________
dropout (Dropout)            (None, 16)                0
_________________________________________________________________
dense_1 (Dense)              (None, 16)                272
_________________________________________________________________
dropout_1 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_2 (Dense)              (None, 8)                 136
_________________________________________________________________
dropout_2 (Dropout)          (None, 8)                 0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 9
=================================================================
Total params: 577
Trainable params: 577
Non-trainable params: 0
_________________________________________________________________
```

A.2.4. Convolutional Neural Networks

```
model = Sequential ()
model.add (Conv1D (filters = 16,
kernel_size = (1),
padding = 'same',
input_shape = (x_train.shape (1), x_train.shape (2)),
activation = 'relu'))
model.add (MaxPooling1D (pool_size = (1)))
model.add (Dense (8, activation = 'relu'))
model.add (Dense (1, activation = 'sigmoid'))
model.compile (loss = 'binary_crossentropy',
optimizer = 'adam', metrics = ('accuracy'))
print(model.summary())
```

A.2.5. Convolutional Neural Networks without Selecting Variables

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 1, 16)             384
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 1, 16)             0
_________________________________________________________________
dense (Dense)                (None, 1, 8)              136
_________________________________________________________________
dense_1 (Dense)              (None, 1, 1)              9
=================================================================
Total params: 529
Trainable params: 529
Non-trainable params: 0
_________________________________________________________________
```

A.2.6. Convolutional Neural Networks after Selection by CHAID

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 1, 16)             160
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 1, 16)             0
_________________________________________________________________
dense (Dense)                (None, 1, 8)              136
_________________________________________________________________
dense_1 (Dense)              (None, 1, 1)              9
=================================================================
Total params: 305
Trainable params: 305
Non-trainable params: 0
_________________________________________________________________
```

A.3. Setting a Callback Function to Store the Dynamic Training Process of Models

```
TensorBoard = TensorBoard (log_dir = 'file name', histogram_freq = 2)
```

A.4. Training Models, Using the Validation Dataset for Validation and Storing the Process in the Callback Function

```
history = model.fit (x_train, y_train, epochs = 150, batch_size = 4, validation_split = 0.15, callbacks = (TensorBoard)).
```

A.5. Assessing Models Based on the Test Dataset

```
t_loss,t_acc = model.evaluate (x_test, y_test)
```

A.6. CHAID (The IBM SPSS Modeler is Utilized to Conduct CHAID in this Study, which is Very Easy to Operate and no Coding Required. The IBM SPSS Modeler is Simpler, more Convenient and Powerful for Users.)

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
groupIndependent_data = pd.read_excel("Independent_data.xlsx")
Response = pd.read_excel ("Response.xlsx")
X_train, X_test, y_train, y_test = train_test_split(Independent_data,Response,test_size = 0.3,random_state = 42)
train_data = pd.concat ((X_train,y_train), axis = 1)
test_data = pd.concat ((X_test,y_test),axis = 1)
from CHAID import Tree
tree = Tree.from_pandas_df (test_data, dict (zip (X_train.columns.tolist (), list (np.repeat('nominal', len(X_train.columns))))), y_train.columns (0), max_depth = 10, min_child_node_size = 3)
tree.print_tree ()
tree.classification_rules ()
```