In [1]:

```python
# Team Members

# Pooja Goyal
# Roshan Sah
# Sanjib Paudel
# Sourabh Yadav
```

In [2]:

```python
# Import all libraries
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import image
import numpy as np
from imblearn.over_sampling import SMOTE

import tensorflow as tf
import keras
from keras.models import Sequential
from keras.layers import Dense,Dropout,Conv2D,Conv1D,Flatten,MaxPool2D

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import *
from sklearn.linear_model import LogisticRegression

import seaborn as sns
import joblib

from sklearn.tree import export_graphviz
import six
import pydot
from sklearn import tree
```

In [3]:

```python
# Load data
data = pd.DataFrame(pd.read_csv('datasets/mc1.csv'))
print(data.shape)
```

```
(9466, 40)
```

In [4]:

```python
data.head()
```

Out[4]:

	id	LOC_BLANK	BRANCH_COUNT	CALL_PAIRS	LOC_CODE_AND_COMMENT	LOC_COMMENTS	CONDITION_COUNT	CYCLOMATIC_COMPLEX
0	1	0	1	0	0	0	0	
1	2	0	1	0	0	0	0	
2	3	0	1	1	0	0	0	
3	4	8	17	11	32	3	12	
4	5	4	7	1	1	8	12	

5 rows × 40 columns

In [5]:

```python
# Check if any value is null
isAnyValueNull = data.isnull().values.any()
print('is Any Value Null:', isAnyValueNull)
```

```
is Any Value Null: False
```

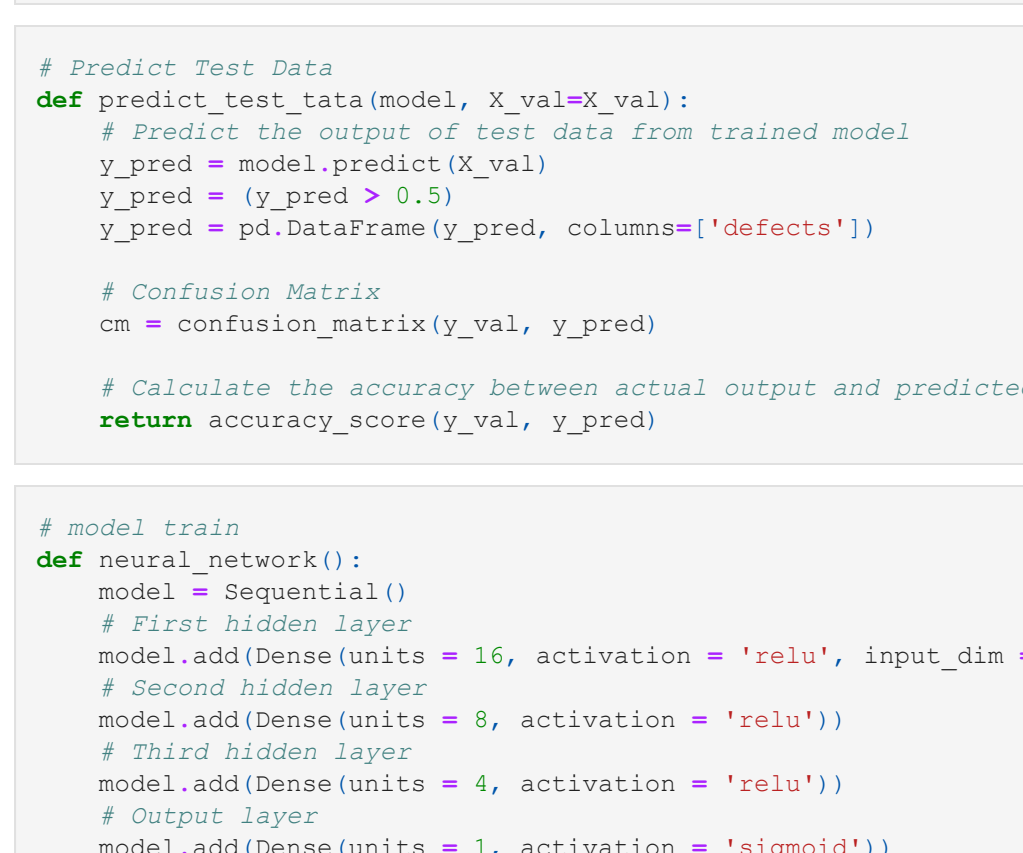
In [6]:

```python
# Pre-processing Data
if isAnyValueNull:
    data = data.dropna()
data_X = data.drop(['defects'], axis=1)
data_y = data['defects']
X_train1, X_test, y_train1, y_test = train_test_split(data_X, data_y, test_size = 0.1, random_state=1)

sm = SMOTE(random_state=1)
X_train2, y_train2 = sm.fit_sample(X_train1, y_train1)

X_train, X_val, y_train, y_val = train_test_split(X_train2, y_train2, test_size = .1, random_state=1)

combined_training_data = X_train.copy()
combined_training_data['defects'] = y_train
```

In [7]:

```python
all_data = [data, data_X, data_y, combined_training_data, X_train1, X_train2, X_train, X_test, X_val, y_train1,
# Heat map of pre processing data
corr = combined_training_data.corr()
sns.heatmap(corr, linewidths=0.1, vmin=0.5, vmax=1)
```

Out[7]: <AxesSubplot:>



In [8]:

```python
def plot_loss_accuracy_graph(history):
    # summarize history for accuracy
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('Model Accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()

    # summarize history for loss
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Model Loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()
```

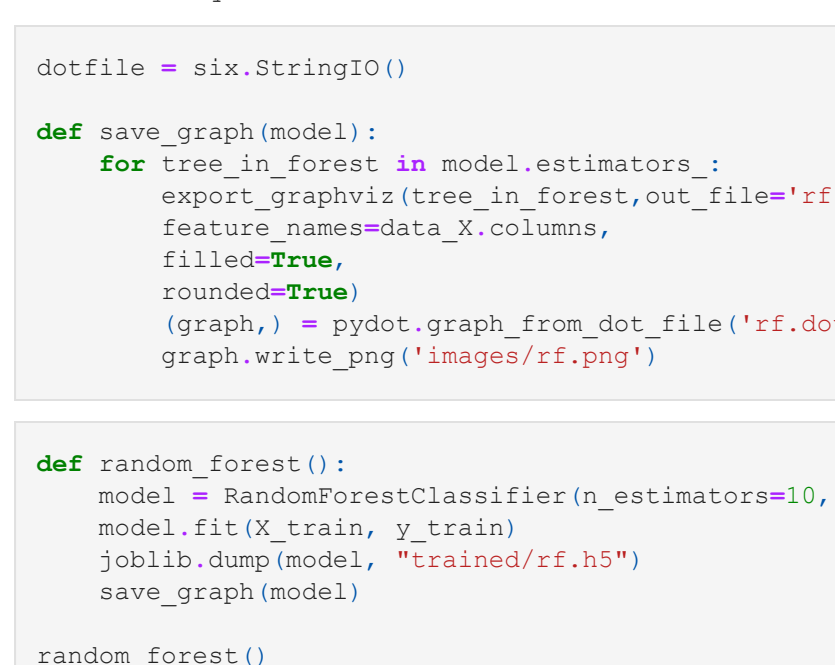
In [9]:

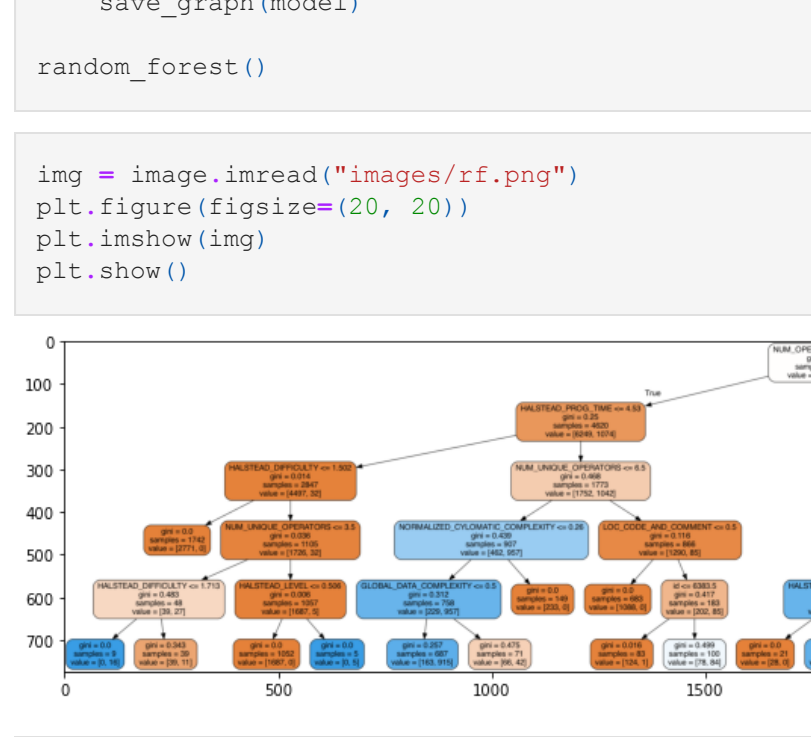
```python
# Predict Test Data
def predict_test_tata(model, X_val=X_val):
    # Predict the output of test data from trained model
    y_pred = model.predict(X_val)
    y_pred = (y_pred > 0.5)
    y_pred = pd.DataFrame(y_pred, columns=['defects'])

    # Confusion Matrix
    cm = confusion_matrix(y_val, y_pred)

    # Calculate the accuracy between actual output and predicted output
    return accuracy_score(y_val, y_pred)
```

In [10]:

```python
# model train
def neural_network():
    model = Sequential()
    # First hidden layer
    model.add(Dense(units = 16, activation = 'relu', input_dim = len(data_X.columns)))
    # Second hidden layer
    model.add(Dense(units = 8, activation = 'relu'))
    # Third hidden layer
    model.add(Dense(units = 4, activation = 'relu'))
    # Output layer
    model.add(Dense(units = 1, activation = 'sigmoid'))


    # Compiling Artificial Neural Network
    model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

    # Fitting the ANN to the Training set
    history = model.fit(X_train, y_train, batch_size = 16, epochs = 100, validation_data=(X_val, y_val))

    model.save(
        "trained/nn.h5",
        overwrite=True,
        include_optimizer=True,
        save_format=None,
        signatures=None,
        options=None,
        save_traces=True,
    )

    plot_loss_accuracy_graph(history)

neural_network()
```

```
Epoch 1/100
952/952 [==============================] - 1s 1ms/step - loss: 1.9439 - accuracy: 0.4880 - val_loss: 0.6930 - v
al_accuracy: 0.5142
Epoch 2/100
952/952 [==============================] - 1s 894us/step - loss: 0.6932 - accuracy: 0.4986 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 3/100
952/952 [==============================] - 1s 875us/step - loss: 0.6932 - accuracy: 0.4953 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 4/100
952/952 [==============================] - 1s 873us/step - loss: 0.6932 - accuracy: 0.4944 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 5/100
952/952 [==============================] - 1s 873us/step - loss: 0.6932 - accuracy: 0.4974 - val_loss: 0.6931 -
val_accuracy: 0.5142
Epoch 6/100
952/952 [==============================] - 1s 874us/step - loss: 0.6931 - accuracy: 0.5042 - val_loss: 0.6935 -
val_accuracy: 0.4858
Epoch 7/100
952/952 [==============================] - 1s 1ms/step - loss: 0.6932 - accuracy: 0.4996 - val_loss: 0.6934 - v
al_accuracy: 0.4858
Epoch 8/100
952/952 [==============================] - 1s 926us/step - loss: 0.6932 - accuracy: 0.5014 - val_loss: 0.6931 -
val_accuracy: 0.5142
Epoch 9/100
952/952 [==============================] - 1s 874us/step - loss: 0.6932 - accuracy: 0.4984 - val_loss: 0.6931 -
val_accuracy: 0.5142
Epoch 10/100
952/952 [==============================] - 1s 877us/step - loss: 0.6932 - accuracy: 0.4986 - val_loss: 0.6931 -
val_accuracy: 0.5142
Epoch 11/100
952/952 [==============================] - 1s 854us/step - loss: 0.6932 - accuracy: 0.4940 - val_loss: 0.6931 -
val_accuracy: 0.5142
Epoch 12/100
952/952 [==============================] - 1s 881us/step - loss: 0.6932 - accuracy: 0.5001 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 13/100
952/952 [==============================] - 1s 880us/step - loss: 0.6932 - accuracy: 0.4950 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 14/100
952/952 [==============================] - 1s 865us/step - loss: 0.6932 - accuracy: 0.4982 - val_loss: 0.6934 -
val_accuracy: 0.4858
Epoch 15/100
952/952 [==============================] - 1s 863us/step - loss: 0.6932 - accuracy: 0.4947 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 16/100
952/952 [==============================] - 1s 856us/step - loss: 0.6932 - accuracy: 0.4984 - val_loss: 0.6931 -
val_accuracy: 0.5142
Epoch 17/100
952/952 [==============================] - 1s 909us/step - loss: 0.6932 - accuracy: 0.4961 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 18/100
952/952 [==============================] - 1s 895us/step - loss: 0.6932 - accuracy: 0.4945 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 19/100
952/952 [==============================] - 1s 858us/step - loss: 0.6932 - accuracy: 0.4991 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 20/100
952/952 [==============================] - 1s 863us/step - loss: 0.6932 - accuracy: 0.4999 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 21/100
952/952 [==============================] - 1s 876us/step - loss: 0.6932 - accuracy: 0.4968 - val_loss: 0.6931 -
val_accuracy: 0.5142
Epoch 22/100
952/952 [==============================] - 1s 868us/step - loss: 0.6932 - accuracy: 0.5033 - val_loss: 0.6930 -
val_accuracy: 0.5142
Epoch 23/100
952/952 [==============================] - 1s 866us/step - loss: 0.6932 - accuracy: 0.4984 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 24/100
952/952 [==============================] - 1s 872us/step - loss: 0.6932 - accuracy: 0.4936 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 25/100
952/952 [==============================] - 1s 866us/step - loss: 0.6932 - accuracy: 0.4980 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 26/100
952/952 [==============================] - 1s 886us/step - loss: 0.6932 - accuracy: 0.5014 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 27/100
952/952 [==============================] - 1s 866us/step - loss: 0.6932 - accuracy: 0.4942 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 28/100
952/952 [==============================] - 1s 870us/step - loss: 0.6932 - accuracy: 0.5011 - val_loss: 0.6931 -
val_accuracy: 0.5142
Epoch 29/100
952/952 [==============================] - 1s 839us/step - loss: 0.6932 - accuracy: 0.5020 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 30/100
952/952 [==============================] - 1s 883us/step - loss: 0.6932 - accuracy: 0.4989 - val_loss: 0.6931 -
val_accuracy: 0.5142
Epoch 31/100
952/952 [==============================] - 1s 867us/step - loss: 0.6932 - accuracy: 0.4978 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 32/100
952/952 [==============================] - 1s 847us/step - loss: 0.6932 - accuracy: 0.4946 - val_loss: 0.6934 -
val_accuracy: 0.4858
Epoch 33/100
952/952 [==============================] - 1s 872us/step - loss: 0.6932 - accuracy: 0.5016 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 34/100
952/952 [==============================] - 1s 890us/step - loss: 0.6932 - accuracy: 0.4962 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 35/100
952/952 [==============================] - 1s 896us/step - loss: 0.6932 - accuracy: 0.4993 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 36/100
952/952 [==============================] - 1s 885us/step - loss: 0.6932 - accuracy: 0.4978 - val_loss: 0.6934 -
val_accuracy: 0.4858
Epoch 37/100
952/952 [==============================] - 1s 882us/step - loss: 0.6932 - accuracy: 0.5016 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 38/100
952/952 [==============================] - 1s 859us/step - loss: 0.6932 - accuracy: 0.4974 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 39/100
952/952 [==============================] - 1s 897us/step - loss: 0.6932 - accuracy: 0.4958 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 40/100
952/952 [==============================] - 1s 890us/step - loss: 0.6932 - accuracy: 0.4989 - val_loss: 0.6931 -
val_accuracy: 0.5142
Epoch 41/100
952/952 [==============================] - 1s 884us/step - loss: 0.6932 - accuracy: 0.4933 - val_loss: 0.6934 -
val_accuracy: 0.4858
Epoch 42/100
952/952 [==============================] - 1s 875us/step - loss: 0.6932 - accuracy: 0.4991 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 43/100
952/952 [==============================] - 1s 878us/step - loss: 0.6932 - accuracy: 0.4971 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 44/100
952/952 [==============================] - 1s 847us/step - loss: 0.6932 - accuracy: 0.5013 - val_loss: 0.6931 -
val_accuracy: 0.5142
Epoch 45/100
952/952 [==============================] - 1s 881us/step - loss: 0.6932 - accuracy: 0.4989 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 46/100
952/952 [==============================] - 1s 877us/step - loss: 0.6932 - accuracy: 0.4944 - val_loss: 0.6931 -
val_accuracy: 0.5142
Epoch 47/100
952/952 [==============================] - 1s 914us/step - loss: 0.6932 - accuracy: 0.4951 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 48/100
952/952 [==============================] - 1s 874us/step - loss: 0.6932 - accuracy: 0.4978 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 49/100
952/952 [==============================] - 1s 866us/step - loss: 0.6932 - accuracy: 0.4963 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 50/100
952/952 [==============================] - 1s 866us/step - loss: 0.6932 - accuracy: 0.4995 - val_loss: 0.6930 -
val_accuracy: 0.5142
Epoch 51/100
952/952 [==============================] - 1s 880us/step - loss: 0.6932 - accuracy: 0.4979 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 52/100
952/952 [==============================] - 1s 901us/step - loss: 0.6932 - accuracy: 0.4971 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 53/100
952/952 [==============================] - 1s 899us/step - loss: 0.6932 - accuracy: 0.4979 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 54/100
952/952 [==============================] - 1s 907us/step - loss: 0.6932 - accuracy: 0.4975 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 55/100
952/952 [==============================] - 1s 903us/step - loss: 0.6932 - accuracy: 0.4980 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 56/100
952/952 [==============================] - 1s 910us/step - loss: 0.6932 - accuracy: 0.4975 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 57/100
952/952 [==============================] - 1s 888us/step - loss: 0.6932 - accuracy: 0.4982 - val_loss: 0.6934 -
val_accuracy: 0.4858
Epoch 58/100
952/952 [==============================] - 1s 904us/step - loss: 0.6932 - accuracy: 0.4980 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 59/100
952/952 [==============================] - 1s 894us/step - loss: 0.6932 - accuracy: 0.4942 - val_loss: 0.6934 -
val_accuracy: 0.4858
Epoch 60/100
952/952 [==============================] - 1s 892us/step - loss: 0.6932 - accuracy: 0.4974 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 61/100
952/952 [==============================] - 1s 873us/step - loss: 0.6932 - accuracy: 0.4986 - val_loss: 0.6931 -
val_accuracy: 0.5142
Epoch 62/100
952/952 [==============================] - 1s 926us/step - loss: 0.6932 - accuracy: 0.4963 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 63/100
952/952 [==============================] - 1s 906us/step - loss: 0.6932 - accuracy: 0.4971 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 64/100
952/952 [==============================] - 1s 900us/step - loss: 0.6932 - accuracy: 0.5005 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 65/100
952/952 [==============================] - 1s 900us/step - loss: 0.6932 - accuracy: 0.4950 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 66/100
952/952 [==============================] - 1s 883us/step - loss: 0.6932 - accuracy: 0.4958 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 67/100
952/952 [==============================] - 1s 891us/step - loss: 0.6932 - accuracy: 0.5012 - val_loss: 0.6930 -
val_accuracy: 0.5142
Epoch 68/100
952/952 [==============================] - 1s 883us/step - loss: 0.6932 - accuracy: 0.4961 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 69/100
952/952 [==============================] - 1s 891us/step - loss: 0.6932 - accuracy: 0.4941 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 70/100
952/952 [==============================] - 1s 889us/step - loss: 0.6932 - accuracy: 0.5021 - val_loss: 0.6931 -
val_accuracy: 0.5142
Epoch 71/100
952/952 [==============================] - 1s 907us/step - loss: 0.6932 - accuracy: 0.4976 - val_loss: 0.6931 -
val_accuracy: 0.5142
Epoch 72/100
952/952 [==============================] - 1s 892us/step - loss: 0.6932 - accuracy: 0.4934 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 73/100
952/952 [==============================] - 1s 900us/step - loss: 0.6932 - accuracy: 0.4983 - val_loss: 0.6931 -
val_accuracy: 0.5142
Epoch 74/100
952/952 [==============================] - 1s 876us/step - loss: 0.6932 - accuracy: 0.4970 - val_loss: 0.6934 -
val_accuracy: 0.4858
Epoch 75/100
952/952 [==============================] - 1s 873us/step - loss: 0.6932 - accuracy: 0.4982 - val_loss: 0.6935 -
val_accuracy: 0.4858
Epoch 76/100
952/952 [==============================] - 1s 915us/step - loss: 0.6932 - accuracy: 0.4984 - val_loss: 0.6934 -
val_accuracy: 0.4858
Epoch 77/100
952/952 [==============================] - 1s 878us/step - loss: 0.6932 - accuracy: 0.5001 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 78/100
952/952 [==============================] - 1s 867us/step - loss: 0.6932 - accuracy: 0.4988 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 79/100
952/952 [==============================] - 1s 1ms/step - loss: 0.6932 - accuracy: 0.4991 - val_loss: 0.6933 - v
al_accuracy: 0.4858
Epoch 80/100
952/952 [==============================] - 1s 1ms/step - loss: 0.6932 - accuracy: 0.4957 - val_loss: 0.6932 - v
al_accuracy: 0.4858
Epoch 81/100
952/952 [==============================] - 1s 1ms/step - loss: 0.6932 - accuracy: 0.4938 - val_loss: 0.6933 - v
al_accuracy: 0.4858
Epoch 82/100
952/952 [==============================] - 1s 886us/step - loss: 0.6932 - accuracy: 0.4967 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 83/100
952/952 [==============================] - 1s 898us/step - loss: 0.6932 - accuracy: 0.4975 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 84/100
952/952 [==============================] - 1s 982us/step - loss: 0.6932 - accuracy: 0.4995 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 85/100
952/952 [==============================] - 1s 969us/step - loss: 0.6932 - accuracy: 0.4975 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 86/100
952/952 [==============================] - 1s 902us/step - loss: 0.6932 - accuracy: 0.4947 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 87/100
952/952 [==============================] - 1s 894us/step - loss: 0.6932 - accuracy: 0.4999 - val_loss: 0.6931 -
val_accuracy: 0.4858
Epoch 88/100
952/952 [==============================] - 1s 881us/step - loss: 0.6932 - accuracy: 0.5033 - val_loss: 0.6935 -
val_accuracy: 0.4858
Epoch 89/100
952/952 [==============================] - 1s 895us/step - loss: 0.6932 - accuracy: 0.5004 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 90/100
952/952 [==============================] - 1s 867us/step - loss: 0.6932 - accuracy: 0.4993 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 91/100
952/952 [==============================] - 1s 893us/step - loss: 0.6932 - accuracy: 0.4970 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 92/100
952/952 [==============================] - 1s 904us/step - loss: 0.6932 - accuracy: 0.5032 - val_loss: 0.6935 -
val_accuracy: 0.4858
Epoch 93/100
952/952 [==============================] - 1s 908us/step - loss: 0.6932 - accuracy: 0.4992 - val_loss: 0.6931 -
val_accuracy: 0.5142
Epoch 94/100
952/952 [==============================] - 1s 949us/step - loss: 0.6932 - accuracy: 0.5008 - val_loss: 0.6931 -
val_accuracy: 0.5142
Epoch 95/100
952/952 [==============================] - 1s 854us/step - loss: 0.6932 - accuracy: 0.4988 - val_loss: 0.6931 -
val_accuracy: 0.5142
Epoch 96/100
952/952 [==============================] - 1s 855us/step - loss: 0.6932 - accuracy: 0.5018 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 97/100
952/952 [==============================] - 1s 850us/step - loss: 0.6932 - accuracy: 0.4979 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 98/100
952/952 [==============================] - 1s 852us/step - loss: 0.6932 - accuracy: 0.4982 - val_loss: 0.6932 -
val_accuracy: 0.4858
Epoch 99/100
952/952 [==============================] - 1s 890us/step - loss: 0.6932 - accuracy: 0.4974 - val_loss: 0.6933 -
val_accuracy: 0.4858
Epoch 100/100
952/952 [==============================] - 1s 884us/step - loss: 0.6932 - accuracy: 0.4965 - val_loss: 0.6932 -
val_accuracy: 0.4858
```





In [11]:

```python
# Predict test result of neural network
# Load saved model
model = keras.models.load_model("trained/nn.h5")
print('Test Accuracy or Neural Network', predict_test_tata(model))
```

```
Test Accuracy or Neural Network 0.4858156028368794
```

In [12]:

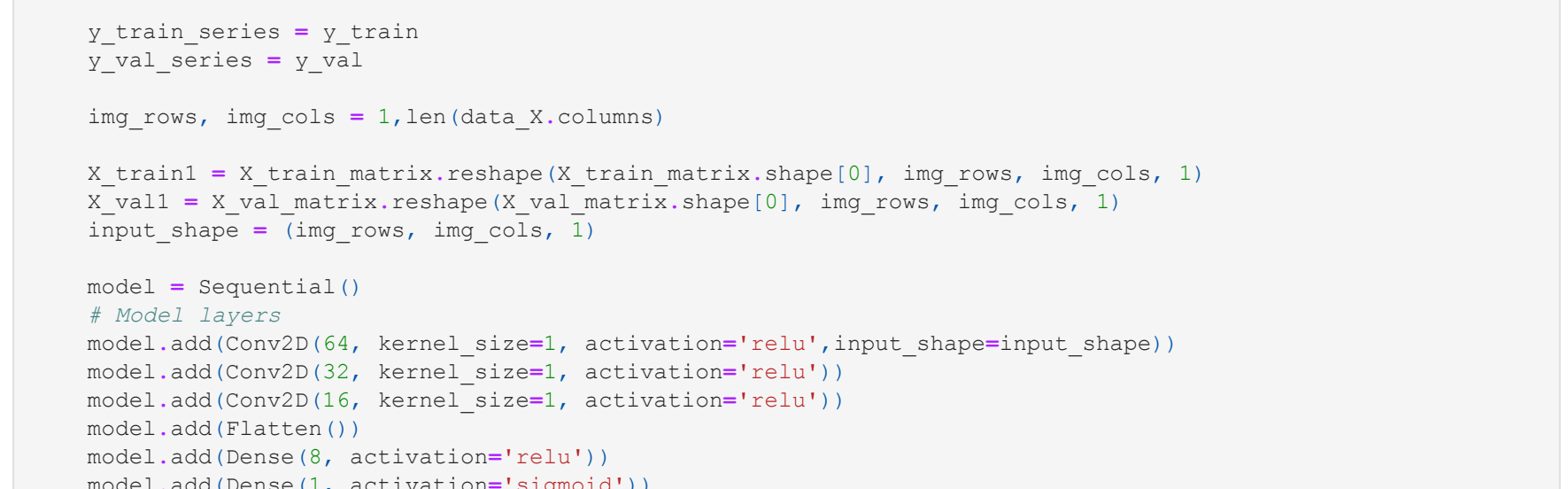
```python
dotfile = six.StringIO()

def save_graph(model):
    for tree_in_forest in model.estimators_:
        export_graphviz(tree_in_forest,out_file='rf.dot',
        feature_names=data_X.columns,
        filled=True,
        rounded=True)
        (graph,) = pydot.graph_from_dot_file('rf.dot')
        graph.write_png('images/rf.png')
```

In [13]:

```python
def random_forest():
    model = RandomForestClassifier(n_estimators=10, max_depth=5,random_state=0)
    model.fit(X_train, y_train)
    joblib.dump(model, "trained/rf.h5")
    save_graph(model)

random_forest()
```

In [14]:

```python
img = image.imread("images/rf.png")
plt.figure(figsize=(20, 20))
plt.imshow(img)
plt.show()
```



In [15]:

```python
# Predict test result of random forest
model = joblib.load("trained/rf.h5")
print('Test Accuracy of Random Forest', predict_test_tata(model))
```

```
Test Accuracy of Random Forest 0.9633569739952719
```

In [16]:

```python
def svm():
    model = SVC(gamma='auto')
    model.fit(X_train, y_train)
    joblib.dump(model, "trained/svm.h5")
svm()
```

In [17]:

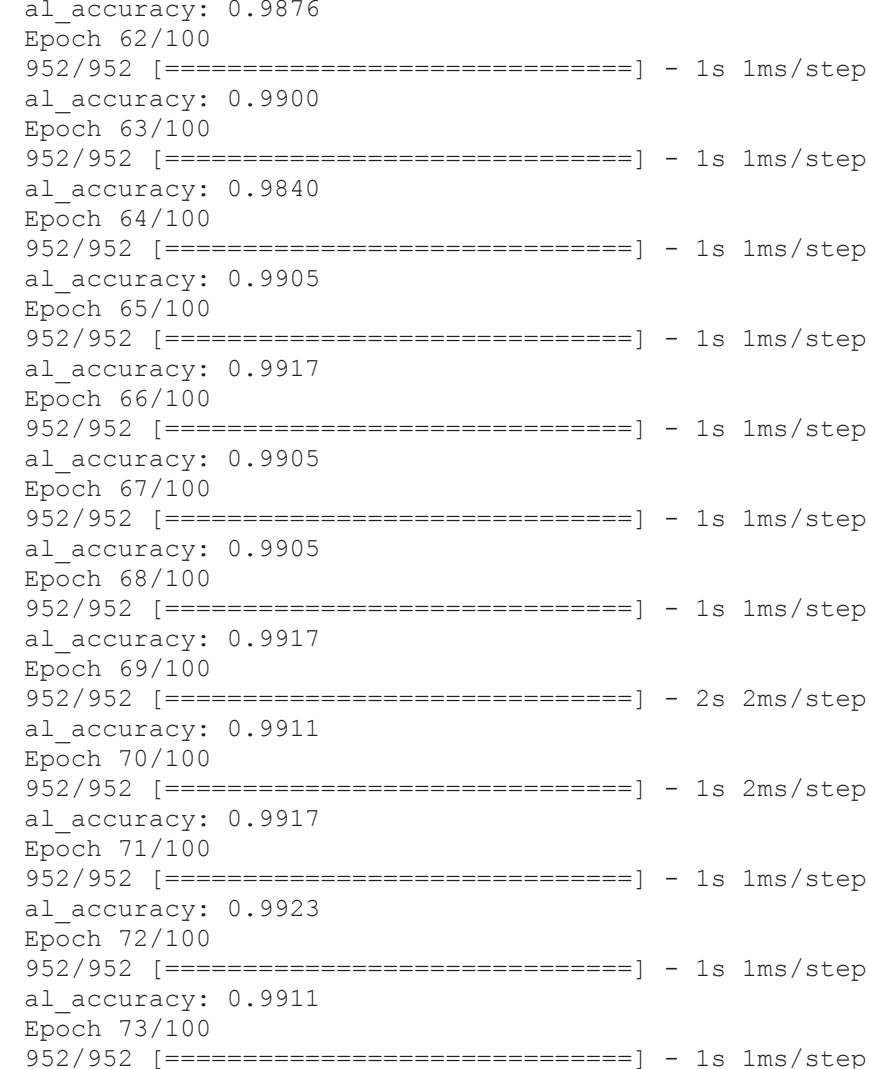
```python
# Predict test result of svc
model = joblib.load("trained/svm.h5")
print('Test Accuracy of SVC', predict_test_tata(model))
```

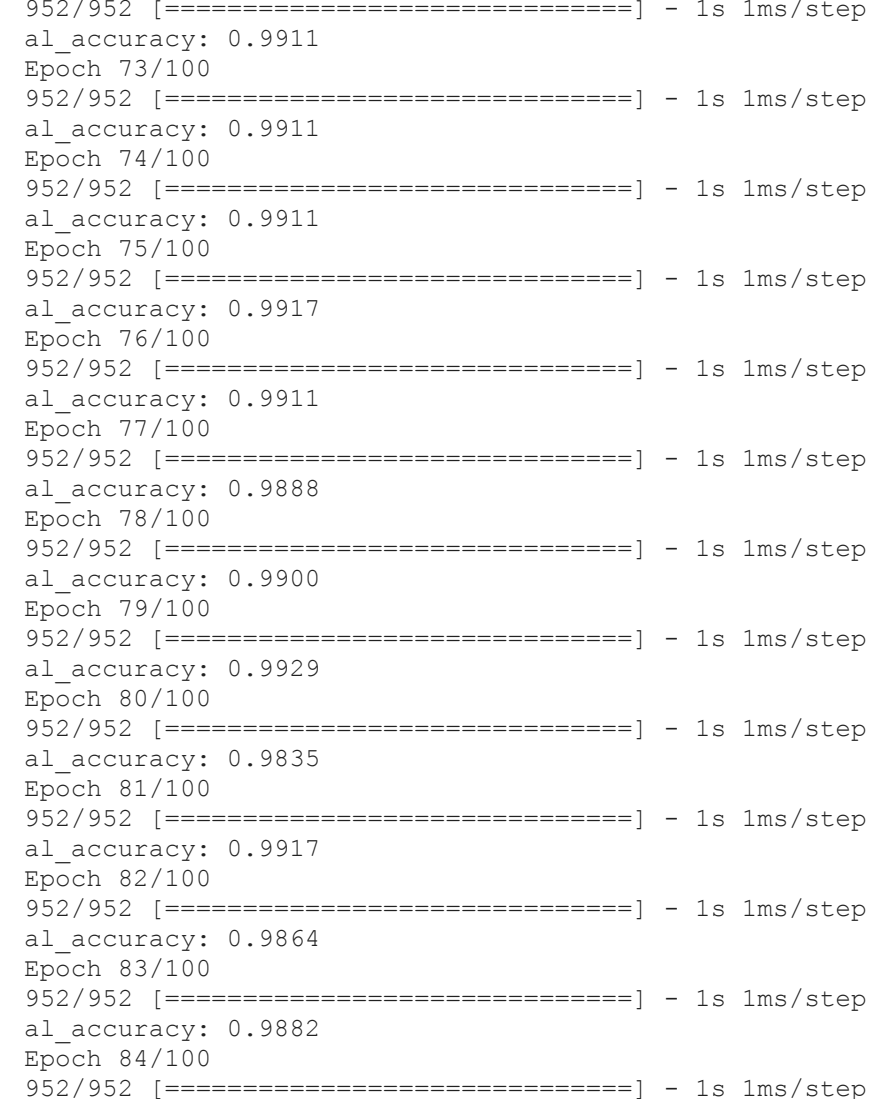
```
Test Accuracy of SVC 0.7446808510638298
```

In [18]:

```python
def cnn():
    global X_val1
    X_train_matrix = X_train.values
    X_val_matrix = X_val.values
    y_train_matrix = y_train.values
    y_val_matrix = y_val.values

    y_train_series = y_train
    y_val_series = y_val

    img_rows, img_cols = 1,len(data_X.columns)

    X_train1 = X_train_matrix.reshape(X_train_matrix.shape[0], img_rows, img_cols, 1)
    X_val1 = X_val_matrix.reshape(X_val_matrix.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

    model = Sequential()
    # Model layers
    model.add(Conv2D(64, kernel_size=1, activation='relu',input_shape=input_shape))
    model.add(Conv2D(32, kernel_size=1, activation='relu'))
    model.add(Conv2D(16, kernel_size=1, activation='relu'))
    model.add(Flatten())
    model.add(Dense(8, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))

    # Compile model using accuracy to measure model performance
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

    #train the model
    history = model.fit(X_train1, y_train_matrix, batch_size=16, epochs=100, validation_data=(X_val1, y_val_mat

    model.save(
        "trained/cnn.h5",
        overwrite=True,
        include_optimizer=True,
        save_format=None,
        signatures=None,
        options=None,
        save_traces=True,
    )

    plot_loss_accuracy_graph(history)

cnn()
```

```
Epoch 1/100
952/952 [==============================] - 2s 2ms/step - loss: 2.7318 - accuracy: 0.8478 - val_loss: 0.3513 - v
al_accuracy: 0.8741
Epoch 2/100
952/952 [==============================] - 2s 2ms/step - loss: 0.3414 - accuracy: 0.8951 - val_loss: 0.3367 - v
al_accuracy: 0.8747
Epoch 3/100
952/952 [==============================] - 2s 2ms/step - loss: 0.2722 - accuracy: 0.9088 - val_loss: 0.2254 - v
al_accuracy: 0.9267
Epoch 4/100
952/952 [==============================] - 2s 2ms/step - loss: 0.2180 - accuracy: 0.9261 - val_loss: 0.1820 - v
al_accuracy: 0.9486
Epoch 5/100
952/952 [==============================] - 1s 1ms/step - loss: 0.1877 - accuracy: 0.9369 - val_loss: 0.1633 - v
al_accuracy: 0.9539
Epoch 6/100
952/952 [==============================] - 1s 1ms/step - loss: 0.1698 - accuracy: 0.9446 - val_loss: 0.1644 - v
al_accuracy: 0.9509
Epoch 7/100
952/952 [==============================] - 1s 1ms/step - loss: 0.1590 - accuracy: 0.9484 - val_loss: 0.1470 - v
al_accuracy: 0.9563
Epoch 8/100
952/952 [==============================] - 1s 1ms/step - loss: 0.1544 - accuracy: 0.9523 - val_loss: 0.1420 - v
al_accuracy: 0.9557
Epoch 9/100
952/952 [==============================] - 1s 1ms/step - loss: 0.1380 - accuracy: 0.9580 - val_loss: 0.1401 - v
al_accuracy: 0.9598
Epoch 10/100
952/952 [==============================] - 1s 1ms/step - loss: 0.1364 - accuracy: 0.9589 - val_loss: 0.1357 - v
al_accuracy: 0.9604
Epoch 11/100
952/952 [==============================] - 1s 1ms/step - loss: 0.1294 - accuracy: 0.9603 - val_loss: 0.1063 - v
al_accuracy: 0.9657
Epoch 12/100
952/952 [==============================] - 1s 1ms/step - loss: 0.1185 - accuracy: 0.9646 - val_loss: 0.1517 - v
al_accuracy: 0.9574
Epoch 13/100
952/952 [==============================] - 1s 1ms/step - loss: 0.1123 - accuracy: 0.9674 - val_loss: 0.1027 - v
al_accuracy: 0.9699
Epoch 14/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0917 - accuracy: 0.9737 - val_loss: 0.0962 - v
al_accuracy: 0.9746
Epoch 15/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0894 - accuracy: 0.9714 - val_loss: 0.0889 - v
al_accuracy: 0.9699
Epoch 16/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0834 - accuracy: 0.9719 - val_loss: 0.0757 - v
al_accuracy: 0.9775
Epoch 17/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0655 - accuracy: 0.9784 - val_loss: 0.0670 - v
al_accuracy: 0.9793
Epoch 18/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0580 - accuracy: 0.9828 - val_loss: 0.0726 - v
al_accuracy: 0.9781
Epoch 19/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0553 - accuracy: 0.9832 - val_loss: 0.0724 - v
al_accuracy: 0.9817
Epoch 20/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0534 - accuracy: 0.9846 - val_loss: 0.0673 - v
al_accuracy: 0.9799
Epoch 21/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0509 - accuracy: 0.9859 - val_loss: 0.0615 - v
al_accuracy: 0.9846
Epoch 22/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0468 - accuracy: 0.9869 - val_loss: 0.0528 - v
al_accuracy: 0.9864
Epoch 23/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0500 - accuracy: 0.9863 - val_loss: 0.0726 - v
al_accuracy: 0.9781
Epoch 24/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0449 - accuracy: 0.9870 - val_loss: 0.0501 - v
al_accuracy: 0.9876
Epoch 25/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0430 - accuracy: 0.9873 - val_loss: 0.0524 - v
al_accuracy: 0.9858
Epoch 26/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0418 - accuracy: 0.9883 - val_loss: 0.0546 - v
al_accuracy: 0.9870
Epoch 27/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0397 - accuracy: 0.9879 - val_loss: 0.0835 - v
al_accuracy: 0.9840
Epoch 28/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0391 - accuracy: 0.9881 - val_loss: 0.0564 - v
al_accuracy: 0.9876
Epoch 29/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0377 - accuracy: 0.9894 - val_loss: 0.0507 - v
al_accuracy: 0.9882
Epoch 30/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0349 - accuracy: 0.9894 - val_loss: 0.0670 - v
al_accuracy: 0.9864
Epoch 31/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0371 - accuracy: 0.9892 - val_loss: 0.0475 - v
al_accuracy: 0.9888
Epoch 32/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0365 - accuracy: 0.9890 - val_loss: 0.0491 - v
al_accuracy: 0.9870
Epoch 33/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0338 - accuracy: 0.9893 - val_loss: 0.0499 - v
al_accuracy: 0.9888
Epoch 34/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0326 - accuracy: 0.9902 - val_loss: 0.0465 - v
al_accuracy: 0.9911
Epoch 35/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0354 - accuracy: 0.9892 - val_loss: 0.0453 - v
al_accuracy: 0.9888
Epoch 36/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0321 - accuracy: 0.9908 - val_loss: 0.0457 - v
al_accuracy: 0.9900
Epoch 37/100
952/952 [==============================] - 2s 2ms/step - loss: 0.0306 - accuracy: 0.9908 - val_loss: 0.0414 - v
al_accuracy: 0.9911
Epoch 38/100
952/952 [==============================] - 1s 2ms/step - loss: 0.0322 - accuracy: 0.9895 - val_loss: 0.0466 - v
al_accuracy: 0.9905
Epoch 39/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0304 - accuracy: 0.9913 - val_loss: 0.0449 - v
al_accuracy: 0.9900
Epoch 40/100
952/952 [==============================] - 1s 2ms/step - loss: 0.0286 - accuracy: 0.9917 - val_loss: 0.0550 - v
al_accuracy: 0.9888
Epoch 41/100
952/952 [==============================] - 1s 2ms/step - loss: 0.0316 - accuracy: 0.9903 - val_loss: 0.0464 - v
al_accuracy: 0.9900
Epoch 42/100
952/952 [==============================] - 1s 2ms/step - loss: 0.0293 - accuracy: 0.9915 - val_loss: 0.0426 - v
al_accuracy: 0.9870
Epoch 43/100
952/952 [==============================] - 2s 2ms/step - loss: 0.0296 - accuracy: 0.9910 - val_loss: 0.0443 - v
al_accuracy: 0.9900
Epoch 44/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0285 - accuracy: 0.9919 - val_loss: 0.0425 - v
al_accuracy: 0.9923
Epoch 45/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0291 - accuracy: 0.9917 - val_loss: 0.0464 - v
al_accuracy: 0.9917
Epoch 46/100
952/952 [==============================] - 1s 2ms/step - loss: 0.0273 - accuracy: 0.9920 - val_loss: 0.0415 - v
al_accuracy: 0.9905
Epoch 47/100
952/952 [==============================] - 1s 2ms/step - loss: 0.0288 - accuracy: 0.9914 - val_loss: 0.0484 - v
al_accuracy: 0.9900
Epoch 48/100
952/952 [==============================] - 1s 2ms/step - loss: 0.0270 - accuracy: 0.9925 - val_loss: 0.0542 - v
al_accuracy: 0.9835
Epoch 49/100
952/952 [==============================] - 1s 2ms/step - loss: 0.0273 - accuracy: 0.9920 - val_loss: 0.0406 - v
al_accuracy: 0.9935
Epoch 50/100
952/952 [==============================] - 1s 2ms/step - loss: 0.0257 - accuracy: 0.9928 - val_loss: 0.0430 - v
al_accuracy: 0.9905
Epoch 51/100
952/952 [==============================] - 1s 2ms/step - loss: 0.0251 - accuracy: 0.9934 - val_loss: 0.0537 - v
al_accuracy: 0.9888
Epoch 52/100
952/952 [==============================] - 1s 2ms/step - loss: 0.0255 - accuracy: 0.9925 - val_loss: 0.0463 - v
al_accuracy: 0.9917
Epoch 53/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0262 - accuracy: 0.9922 - val_loss: 0.0444 - v
al_accuracy: 0.9923
Epoch 54/100
952/952 [==============================] - 1s 2ms/step - loss: 0.0245 - accuracy: 0.9927 - val_loss: 0.0388 - v
al_accuracy: 0.9911
Epoch 55/100
952/952 [==============================] - 1s 2ms/step - loss: 0.0246 - accuracy: 0.9924 - val_loss: 0.0437 - v
al_accuracy: 0.9923
Epoch 56/100
952/952 [==============================] - 1s 2ms/step - loss: 0.0249 - accuracy: 0.9919 - val_loss: 0.0441 - v
al_accuracy: 0.9917
Epoch 57/100
952/952 [==============================] - 1s 2ms/step - loss: 0.0228 - accuracy: 0.9936 - val_loss: 0.0460 - v
al_accuracy: 0.9900
Epoch 58/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0243 - accuracy: 0.9930 - val_loss: 0.0482 - v
al_accuracy: 0.9917
Epoch 59/100
952/952 [==============================] - 1s 2ms/step - loss: 0.0266 - accuracy: 0.9923 - val_loss: 0.0430 - v
al_accuracy: 0.9923
Epoch 60/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0247 - accuracy: 0.9931 - val_loss: 0.0519 - v
al_accuracy: 0.9905
Epoch 61/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0228 - accuracy: 0.9932 - val_loss: 0.0441 - v
al_accuracy: 0.9876
Epoch 62/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0242 - accuracy: 0.9928 - val_loss: 0.0486 - v
al_accuracy: 0.9900
Epoch 63/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0249 - accuracy: 0.9923 - val_loss: 0.0507 - v
al_accuracy: 0.9840
Epoch 64/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0239 - accuracy: 0.9938 - val_loss: 0.0465 - v
al_accuracy: 0.9905
Epoch 65/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0219 - accuracy: 0.9935 - val_loss: 0.0375 - v
al_accuracy: 0.9917
Epoch 66/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0250 - accuracy: 0.9928 - val_loss: 0.0482 - v
al_accuracy: 0.9905
Epoch 67/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0224 - accuracy: 0.9930 - val_loss: 0.0412 - v
al_accuracy: 0.9905
Epoch 68/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0245 - accuracy: 0.9929 - val_loss: 0.0406 - v
al_accuracy: 0.9917
Epoch 69/100
952/952 [==============================] - 2s 2ms/step - loss: 0.0214 - accuracy: 0.9937 - val_loss: 0.0384 - v
al_accuracy: 0.9911
Epoch 70/100
952/952 [==============================] - 1s 2ms/step - loss: 0.0256 - accuracy: 0.9920 - val_loss: 0.0397 - v
al_accuracy: 0.9917
Epoch 71/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0218 - accuracy: 0.9938 - val_loss: 0.0379 - v
al_accuracy: 0.9923
Epoch 72/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0219 - accuracy: 0.9938 - val_loss: 0.0399 - v
al_accuracy: 0.9911
Epoch 73/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0207 - accuracy: 0.9941 - val_loss: 0.0489 - v
al_accuracy: 0.9911
Epoch 74/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0226 - accuracy: 0.9931 - val_loss: 0.0377 - v
al_accuracy: 0.9911
Epoch 75/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0212 - accuracy: 0.9942 - val_loss: 0.0394 - v
al_accuracy: 0.9917
Epoch 76/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0222 - accuracy: 0.9932 - val_loss: 0.0427 - v
al_accuracy: 0.9911
Epoch 77/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0256 - accuracy: 0.9940 - val_loss: 0.0398 - v
al_accuracy: 0.9888
Epoch 78/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0205 - accuracy: 0.9932 - val_loss: 0.0395 - v
al_accuracy: 0.9900
Epoch 79/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0200 - accuracy: 0.9940 - val_loss: 0.0417 - v
al_accuracy: 0.9929
Epoch 80/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0200 - accuracy: 0.9944 - val_loss: 0.0460 - v
al_accuracy: 0.9835
Epoch 81/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0220 - accuracy: 0.9933 - val_loss: 0.0484 - v
al_accuracy: 0.9917
Epoch 82/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0203 - accuracy: 0.9944 - val_loss: 0.0476 - v
al_accuracy: 0.9864
Epoch 83/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0212 - accuracy: 0.9934 - val_loss: 0.0611 - v
al_accuracy: 0.9882
Epoch 84/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0202 - accuracy: 0.9939 - val_loss: 0.0483 - v
al_accuracy: 0.9917
Epoch 85/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0191 - accuracy: 0.9944 - val_loss: 0.0395 - v
al_accuracy: 0.9935
Epoch 86/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0222 - accuracy: 0.9927 - val_loss: 0.0393 - v
al_accuracy: 0.9923
Epoch 87/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0183 - accuracy: 0.9947 - val_loss: 0.0419 - v
al_accuracy: 0.9923
Epoch 88/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0199 - accuracy: 0.9942 - val_loss: 0.0436 - v
al_accuracy: 0.9911
Epoch 89/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0188 - accuracy: 0.9941 - val_loss: 0.0444 - v
al_accuracy: 0.9905
Epoch 90/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0203 - accuracy: 0.9934 - val_loss: 0.0623 - v
al_accuracy: 0.9876
Epoch 91/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0179 - accuracy: 0.9946 - val_loss: 0.0356 - v
al_accuracy: 0.9935
Epoch 92/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0173 - accuracy: 0.9950 - val_loss: 0.0648 - v
al_accuracy: 0.9858
Epoch 93/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0214 - accuracy: 0.9934 - val_loss: 0.0332 - v
al_accuracy: 0.9923
Epoch 94/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0189 - accuracy: 0.9942 - val_loss: 0.0439 - v
al_accuracy: 0.9917
Epoch 95/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0181 - accuracy: 0.9945 - val_loss: 0.0490 - v
al_accuracy: 0.9882
Epoch 96/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0203 - accuracy: 0.9942 - val_loss: 0.0352 - v
al_accuracy: 0.9929
Epoch 97/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0188 - accuracy: 0.9942 - val_loss: 0.0425 - v
al_accuracy: 0.9923
Epoch 98/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0179 - accuracy: 0.9946 - val_loss: 0.0493 - v
al_accuracy: 0.9923
Epoch 99/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0175 - accuracy: 0.9948 - val_loss: 0.0398 - v
al_accuracy: 0.9929
Epoch 100/100
952/952 [==============================] - 1s 1ms/step - loss: 0.0179 - accuracy: 0.9947 - val_loss: 0.0345 - v
al_accuracy: 0.9923
```





In [19]:

```python
# Predict test result of convolutional neural network
# Load saved model
model = keras.models.load_model("trained/cnn.h5")
print('Test Accuracy of CNN', predict_test_tata(model, X_val1))
```

```
Test Accuracy of CNN 0.9923167848699763
```

In [20]:

```python
svm_model = joblib.load("trained/svm.h5")
rf_model = joblib.load("trained/rf.h5")
nn_model = keras.models.load_model("trained/nn.h5")
cnn_model = keras.models.load_model("trained/cnn.h5")

# print accuracy of svm, random forest, nn, cnn
def print_accuracy(model):
    if (model == nn_model):
        y_pred_on_val = model.predict(X_val)>0.5
        y_pred_on_test = model.predict(X_test)>0.5
    elif (model == cnn_model):
        X_val_matrix = X_val.values
        X_val1 = X_val_matrix.reshape(X_val_matrix.shape[0], 1, len(X_val.columns), 1)
        y_pred_on_val = model.predict(X_val1)>0.5
        X_test_matrix = X_test.values
        X_test1 = X_test_matrix.reshape(X_test_matrix.shape[0], 1, len(X_test.columns), 1)
        y_pred_on_test = model.predict(X_test1)>0.5
    else:
        y_pred_on_val = model.predict(X_val)
        y_pred_on_test = model.predict(X_test)

    print('*****************', model, '*********************')
    print('*****Validation Set*****')
    print('Accuracy:',balanced_accuracy_score(y_val,y_pred_on_val))
    print('Avg Precision:', average_precision_score(y_val,y_pred_on_val))
    print('Precision:', precision_score(y_val,y_pred_on_val))
    print('Recall:', recall_score(y_val, y_pred_on_val))
    print('************************')
    print('*****Test Set*****')
    print('Accuracy:',balanced_accuracy_score(y_test,y_pred_on_test))
    print('Avg Precision:', average_precision_score(y_test,y_pred_on_test))
    print('Precision:', precision_score(y_test,y_pred_on_test))
    print('Recall:', recall_score(y_test, y_pred_on_test))
    print('************************')
    print('')
    y_pred_on_val_df = pd.DataFrame(y_pred_on_val, columns=['defects1'])
    y_pred_on_test_df = pd.DataFrame(y_pred_on_test, columns=['defects1'])
    val_result = pd.concat([y_val.reset_index(drop=True), y_pred_on_val_df['defects1']],axis=1)
    val_result = val_result.rename(columns={'defects':'val_actual', 'defects1':'val_predict'})
    test_result = pd.concat([y_test.reset_index(drop=True),y_pred_on_test_df['defects1']],axis=1)
    test_result = test_result.rename(columns={'defects':'test_actual','defects1':'test_predict'})
    return val_result, test_result


svm_val_result, svm_test_result = print_accuracy(svm_model)
rf_val_result, rf_test_result = print_accuracy(rf_model)
nn_val_result, nn_test_result = print_accuracy(nn_model)
cnn_val_result, cnn_test_result = print_accuracy(cnn_model)


new_val_set_X = pd.concat([svm_val_result['val_predict'],rf_val_result['val_predict'],nn_val_result['val_predic
new_val_set_X_matrix = new_val_set_X.values
new_val_set_y_matrix = svm_val_result['val_actual'].values

new_test_set_X = pd.concat([svm_test_result['test_predict'],rf_test_result['test_predict'],nn_test_result['test
new_test_set_X_matrix = new_test_set_X.values
new_test_set_y_matrix = svm_test_result['test_actual'].values
```

```
***************** SVC(gamma='auto') *********************
*****Validation Set*****
Accuracy: 0.7516570182062254
Avg Precision: 0.7575957937401121
Precision: 0.995475113122172
Recall: 0.5057471264367817
************************
*****Test Set*****
Accuracy: 0.5
Avg Precision: 0.006335797254487857
Precision: 0.0
Recall: 0.0
************************

***************** RandomForestClassifier(max_depth=5, n_estimators=10, random_state=0) *********************
*****Validation Set*****
Accuracy: 0.9623542243476801
Avg Precision: 0.9343766426886825
Precision: 0.9353448275862069
Recall: 0.9977011494252873
************************
*****Test Set*****
Accuracy: 0.7902940134608571
Avg Precision: 0.03348448143777046
Precision: 0.047058823529411764
Recall: 0.6666666666666666
************************

***************** <keras.engine.sequential.Sequential object at 0x7fcb71384790> *********************
*****Validation Set*****
Accuracy: 0.5
Avg Precision: 0.5141843971631206
Precision: 0.0
Recall: 0.0
************************
*****Test Set*****
Accuracy: 0.5
Avg Precision: 0.006335797254487857
Precision: 0.0
Recall: 0.0
************************

/opt/anaconda3/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Pre
cision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` parameter to contro
l this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/anaconda3/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Pre
cision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` parameter to contro
l this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/anaconda3/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Pre
cision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` parameter to contro
l this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
***************** <keras.engine.sequential.Sequential object at 0x7fcb504a28e0> *********************
*****Validation Set*****
Accuracy: 0.9921260172833292
Avg Precision: 0.985836362093625
Precision: 0.9863791146424518
Recall: 0.9988505747126437
************************
*****Test Set*****
Accuracy: 0.6608218207580588
Avg Precision: 0.05550591611837652
Precision: 0.15384615384615385
Recall: 0.3333333333333333
************************
```

In [21]:

```python
def ensemble_logistic_regression():
    model = LogisticRegression(random_state=0)
    model.fit(new_val_set_X_matrix, new_val_set_y_matrix)
    joblib.dump((model, new_test_set_X_matrix), "trained/ensemble.h5")
ensemble_logistic_regression()
```

In [22]:

```python
ensemble = joblib.load("trained/ensemble.h5")
prediction = ensemble[0].predict(ensemble[1])
print('Ensemble Accuracy:',accuracy_score(y_test.values, prediction))
```

```
Ensemble Accuracy: 0.9841605068637803
```

In [ ]: