Malware Detection Using Machine Learning And Deep Learning

Importing Libraires

In [26]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Exploring The Malware Data Set

In [47]:

```
malware_data=pd.read_csv("C:/Users/rajak/OneDrive/Desktop/Malware/\MalwareData.csv",
malware_data
```

Out[47]:

	Name	md5	Machine
0	memtest.exe	631ea355665f28d4707448e442fbf5b8	332
1	ose.exe	9d10f99a6712e28f8acd5641e3a7ea6b	332
2	setup.exe	4d92f518527353c0db88a70fddcfd390	332
3	DW20.EXE	a41e524f8d45f0074fd07805ff0c9b12	332
4	dwtrig20.exe	c87e561258f2f8650cef999bf643a731	332
...	...	...	...
138042	VirusShare_8e292b418568d6e7b87f2a32aee7074b	8e292b418568d6e7b87f2a32aee7074b	332
138043	VirusShare_260d9e2258aed4c8a3bbd703ec895822	260d9e2258aed4c8a3bbd703ec895822	332
138044	VirusShare_8d088a51b7d225c9f5d11d239791ec3f	8d088a51b7d225c9f5d11d239791ec3f	332
138045	VirusShare_4286dccf67ca220fe67635388229a9f3	4286dccf67ca220fe67635388229a9f3	332
138046	VirusShare_d7648eae45f09b3adb75127f43be6d11	d7648eae45f09b3adb75127f43be6d11	332

138047 rows × 57 columns

In [48]:

```
malware_data.head()
```

Out[48]:

	Name	md5	Machine	SizeOfOptionalHeader	Characteristics
0	memtest.exe	631ea355665f28d4707448e442fbf5b8	332	224	258
1	ose.exe	9d10f99a6712e28f8acd5641e3a7ea6b	332	224	3330
2	setup.exe	4d92f518527353c0db88a70fddcfd390	332	224	3330
3	DW20.EXE	a41e524f8d45f0074fd07805ff0c9b12	332	224	258
4	dwtrig20.exe	c87e561258f2f8650cef999bf643a731	332	224	258

5 rows × 57 columns

In [49]:

```
malware_data.shape
```

Out[49]:

```
(138047, 57)
```

In [50]:

```
malware_data.describe()
```

Out[50]:

	Machine	SizeOfOptionalHeader	Characteristics	MajorLinkerVersion	MinorLinkerVersion
count	138047.000000	138047.000000	138047.000000	138047.000000	138047.000000
mean	4259.069274	225.845632	4444.145994	8.619774	3.819286
std	10880.347245	5.121399	8186.782524	4.088757	11.862675
min	332.000000	224.000000	2.000000	0.000000	0.000000
25%	332.000000	224.000000	258.000000	8.000000	0.000000
50%	332.000000	224.000000	258.000000	9.000000	0.000000
75%	332.000000	224.000000	8226.000000	10.000000	0.000000
max	34404.000000	352.000000	49551.000000	255.000000	255.000000

8 rows × 55 columns

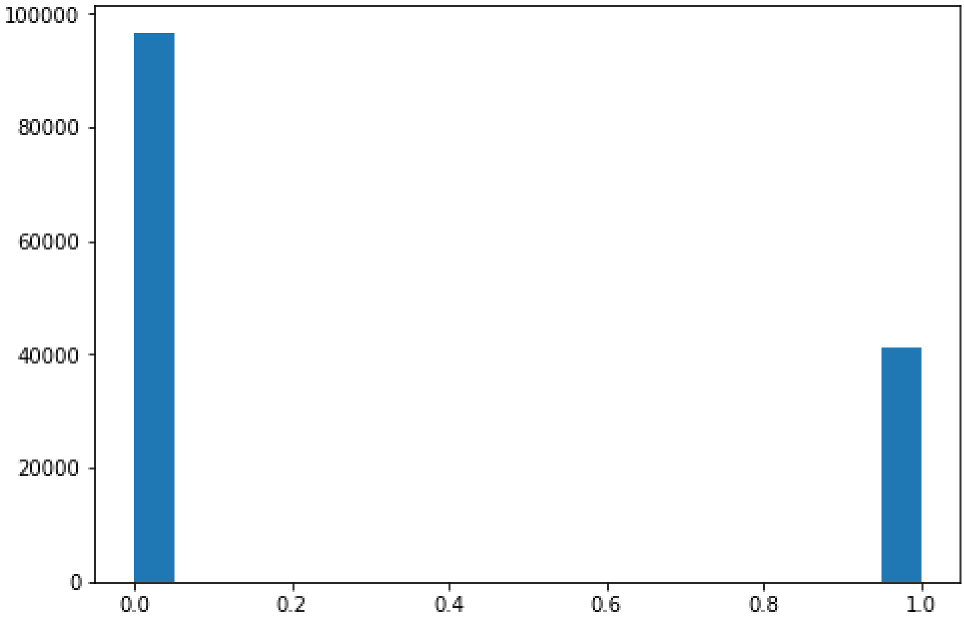
In [51]:

```
legitimate=malware_data[0:41323].drop(["legitimate"],axis=1)
malware=malware_data[41323::].drop(["legitimate"],axis=1)
print("The Shape Of Legitimate Dataset is %s Samples,%s Features"%(legitimate.shape[
print("The Shape Of malware Dataset is %s Samples,%s Features"%(malware.shape[0],mal
```

```
The Shape Of Legitimate Dataset is 41323 Samples,56 Features
The Shape Of malware Dataset is 96724 Samples,56 Features
```

In [52]:

```
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.hist(malware_data['legitimate'],20)
plt.show()
```



DATA CLEANING

In [53]:

```
y=malware_data['legitimate']
malware_data=malware_data.drop(['legitimate'],axis=1)
```

In [54]:

```
malware_data=malware_data.drop(['Name'],axis=1)
malware_data=malware_data.drop(['md5'],axis=1)
print("  The Name and md5 variables are removed successfully")
```

```
  The Name and md5 variables are removed successfully
```

In [55]:

```
malware_data
```

Out[55]:

	Machine	SizeOfOptionalHeader	Characteristics	MajorLinkerVersion	MinorLinkerVersion	Si
0	332	224	258	9	0	
1	332	224	3330	9	0	
2	332	224	3330	9	0	
3	332	224	258	9	0	
4	332	224	258	9	0	
...	...	...	...	...	...	
138042	332	224	258	11	0	
138043	332	224	33167	2	25	
138044	332	224	258	10	0	
138045	332	224	33166	2	25	
138046	332	224	258	11	0	

138047 rows × 54 columns

Spliting The Dataset Into Test and Train

In [56]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(malware_data,y,test_size=0.2,random_s
```

In [57]:

```
x_train.shape
```

Out[57]:

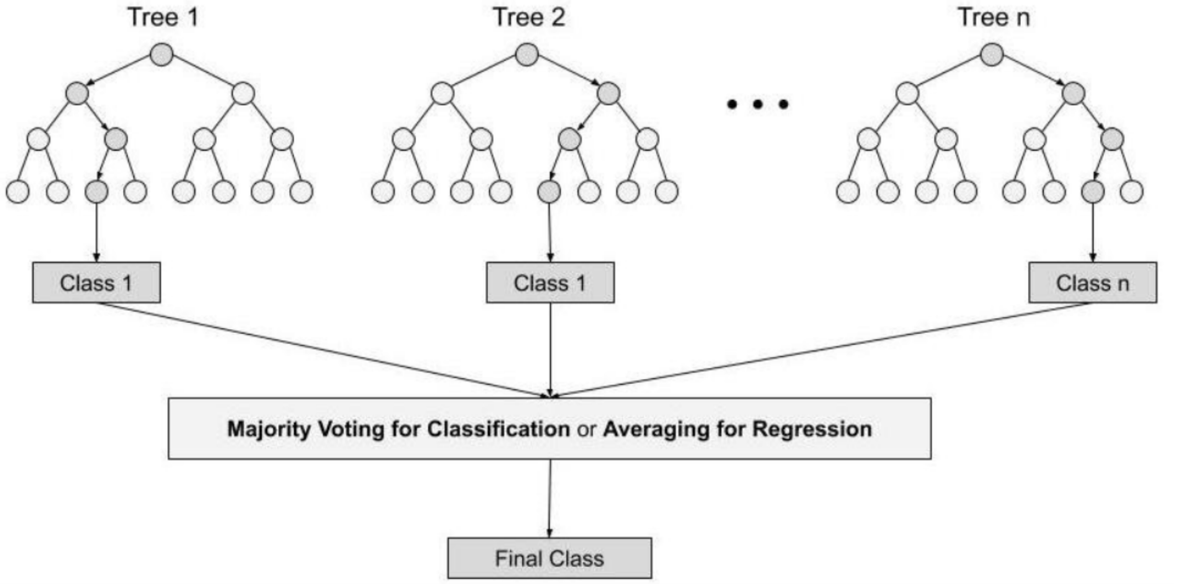
```
(110437, 54)
```

MODEL BUILDING

1- Random Forest

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results for classification problems.


Tree 1
Tree 2
Tree n
Class 1
Class 1
Class n
Majority Voting for Classification or Averaging for Regression
Final Class

In [58]:

```
y_train
```

Out[58]:

```
125264    0
51953     0
40505     1
53059     0
45729     0
         ..
110268    0
119879    0
103694    0
131932    0
```

```
121958    0
Name: legitimate, Length: 110437, dtype: int64
```

In [59]:

```
x_train
```

Out[59]:

	Machine	SizeOfOptionalHeader	Characteristics	MajorLinkerVersion	MinorLinkerVersion	Si
125264	332	224	258	12	0	
51953	332	224	783	2	56	
40505	332	224	8450	10	10	
53059	332	224	258	10	0	
45729	332	224	258	11	0	
...	...	...	...	...	...	
110268	332	224	258	10	0	
119879	332	224	258	12	0	
103694	332	224	783	2	56	
131932	332	224	258	11	0	
121958	332	224	258	10	0	

110437 rows × 54 columns

In [60]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
clf=RandomForestClassifier(max_depth=2,random_state=0)
randomModel=clf.fit(x_train,y_train)
```

Random Forest Evaluation On Test Data

In [61]:

```
from sklearn.metrics import f1_score,accuracy_score,plot_confusion_matrix,auc,confus
```

In [62]:

```
# Accuracy on the train dataset
train_pred=randomModel.predict(x_train)
accuracy_score(y_train,train_pred)
```

Out[62]:

```
0.9828318407780001
```

In [63]:

```
# Accuracy on the test dataset
prediction=randomModel.predict(x_test)
accuracy_score(y_train,train_pred)
```

Out[63]:

```
0.9828318407780001
```

In [64]:

```
f1_score(y_test,prediction)
```

Out[64]:

```
0.9730933606212002
```

Confusion Matrix

A confusion matrix is a summary of prediction results on a classification problem.

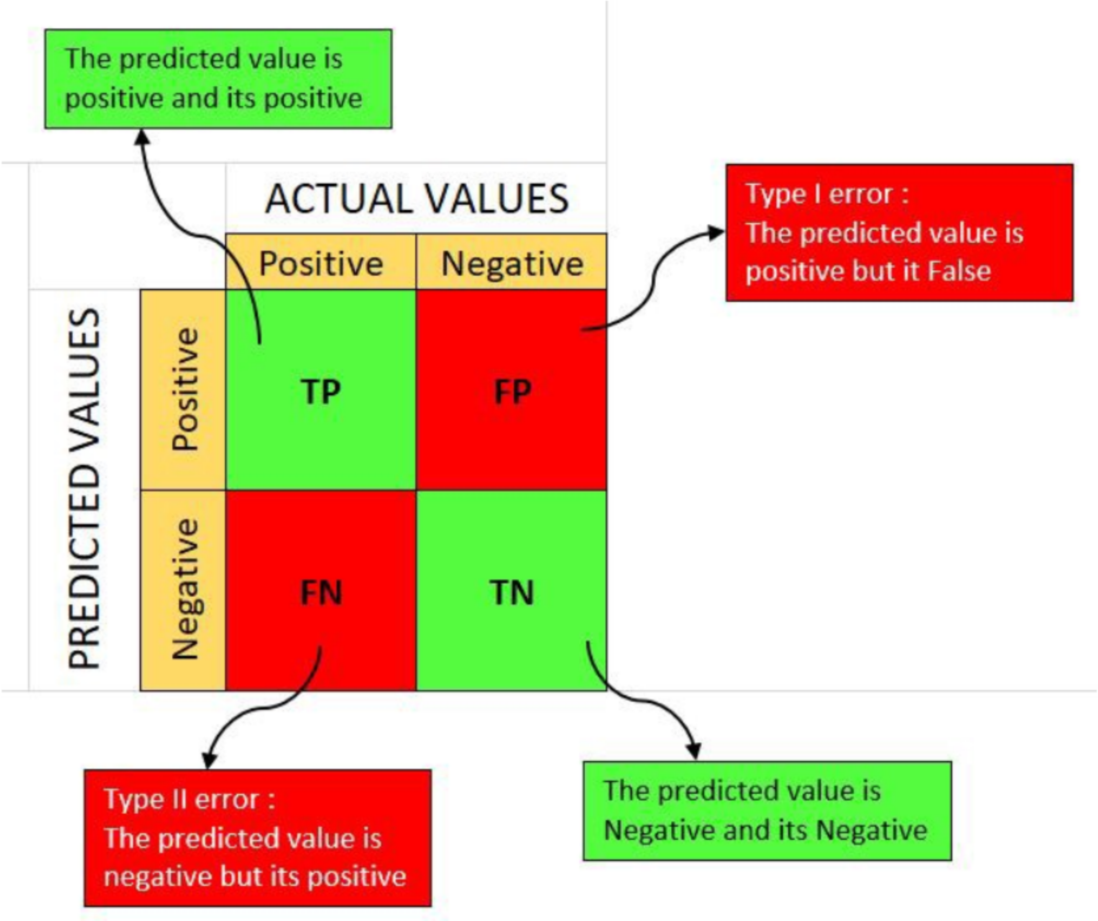
The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix.

The confusion matrix shows the ways in which your classification model

is confused when it makes predictions.

It gives you insight not only into the errors being made by your classifier but more importantly the types of errors that are being made.

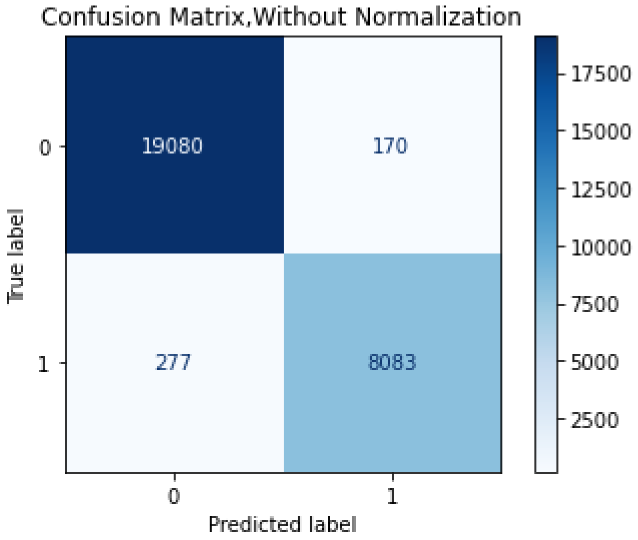
It is this breakdown that overcomes the limitation of using classification accuracy alone.


The predicted value is positive and its positive
ACTUAL VALUES
Positive
Negative
PREDICTED VALUES
Positive
Negative
TP
FP
FN
TN
Type I error : The predicted value is positive but it False
Type II error : The predicted value is negative but its positive
The predicted value is Negative and its Negative

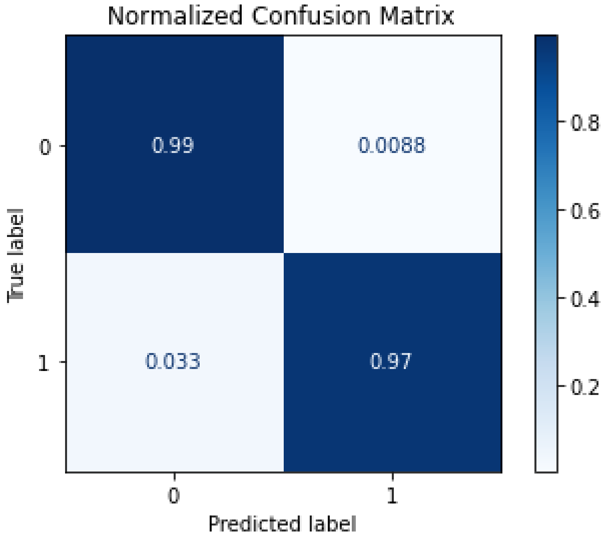
In [107…

```
titles_options=[("Confusion Matrix,Without Normalization",None),
                ("Normalized Confusion Matrix",'true')]
for title,normalize in titles_options:
    disp=plot_confusion_matrix(randomModel,x_test,y_test,

                                cmap=plt.cm.Blues,
                                      normalize=normalize)
    disp.ax_.set_title(title)

    print(title)
    print(disp.confusion_matrix)

    plt.show()
```

```
Confusion Matrix,Without Normalization
[[19080   170]
 [  277  8083]]
```



```
Normalized Confusion Matrix
[[0.99116883 0.00883117]
 [0.03313397 0.96686603]]
```



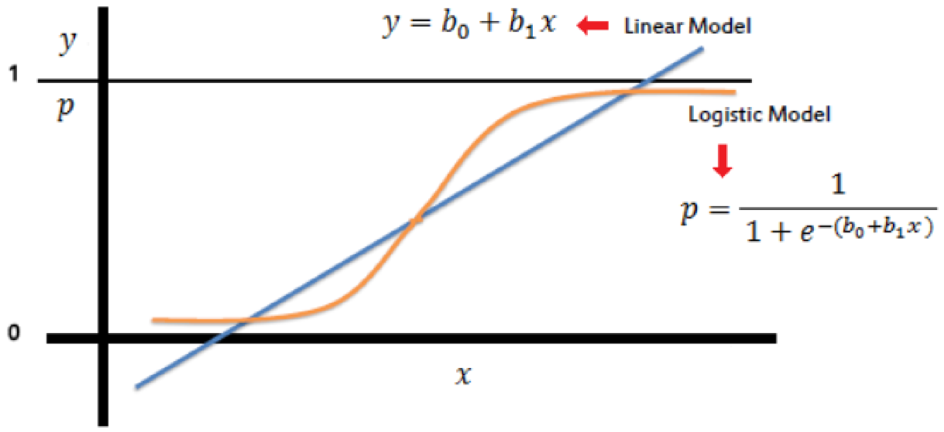
2- Logistic Regression

Logistic regression predicts the probability of an outcome that can only have two values (i.e. a dichotomy). The prediction is based on the use of one or several predictors (numerical and categorical). A linear regression is not appropriate for predicting the value of a binary variable for two reasons:

A linear regression will predict values outside the acceptable range (e.g. predicting probabilities

outside the range 0 to 1)

Since the dichotomous experiments can only have one of two possible values for each experiment, the residuals will not be normally distributed about the predicted line.



In the logistic regression the constant (b_0) moves the curve left and right and the slope (b_1) defines the steepness of the curve. By simple transformation, the logistic regression equation can be written in terms of an odds ratio.

$$\frac{p}{1-p} = \exp\left(b_0 + b_1 x\right)$$

Finally, taking the natural log of both sides, we can write the equation in terms of log-odds (logit) which is a linear function of the predictors. The coefficient (b_1) is the amount the logit (log-odds) changes with a one unit change in x.

$$ln\left(\frac{p}{1-p}\right) = b_0 + b_1 x$$

As mentioned before, logistic regression can handle any number of numerical and/or categorical variables.

$$p = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p)}}$$

There are several analogies between linear regression and logistic regression. Just as ordinary least square regression is the method used to estimate coefficients for the best fit line in linear regression, logistic regression uses maximum likelihood estimation (MLE) to obtain the model coefficients that relate predictors to the target. After this initial function is estimated, the process is repeated until LL (Log Likelihood) does not change significantly.

$$\boldsymbol{\beta}^1 = \boldsymbol{\beta}^0 + [\boldsymbol{X}^T \boldsymbol{W} \boldsymbol{X}]^{-1}.\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{\mu})$$

$\boldsymbol{\beta}$ is a vector of the logistic regression coefficients.

$\boldsymbol{W}$ is a square matrix of order N with elements $n_i \pi_i (1 - \pi_i)$ on the diagonal and zeros everywhere else.

$\boldsymbol{\mu}$ is a vector of length N with elements $\mu_i = n_i \pi_i$.

In [78]:

```
from sklearn.linear_model import LogisticRegression

clf =LogisticRegression (random_state=0)
logModel=clf.fit(x_train, y_train)
```

```
C:\Users\rajak\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: Co
nvergenceWarning: lbfgs failed to converge (status=2):
ABNORMAL_TERMINATION_IN_LNSRCH.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Model Evaluation

In [79]:

```
# Accuracy on the train dataset
train_log=logModel.predict(x_train)
accuracy_score(y_train,train_log)
```

Out[79]:

```
0.7015221347917817
```

In [80]:

```
# Accuracy on the test dataset
pred=logModel.predict(x_test)
accuracy_score(y_test,pred)
```

Out[80]:

```
0.6972111553784861
```

In [81]:

```
f1_score(y_test,pred)
```

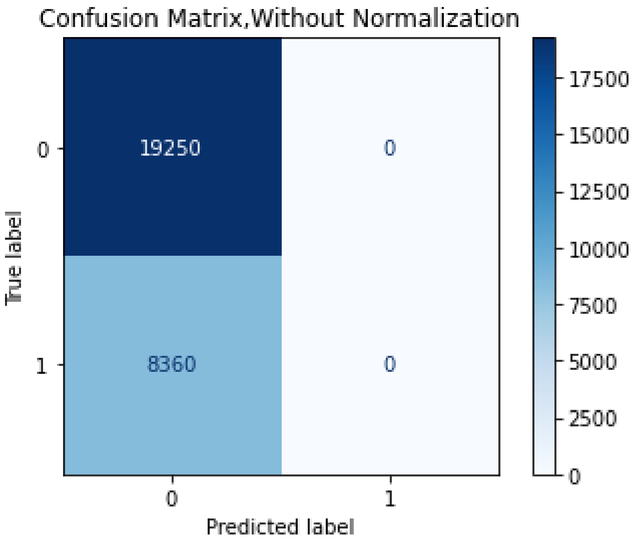
Out[81]:

```
0.0
```

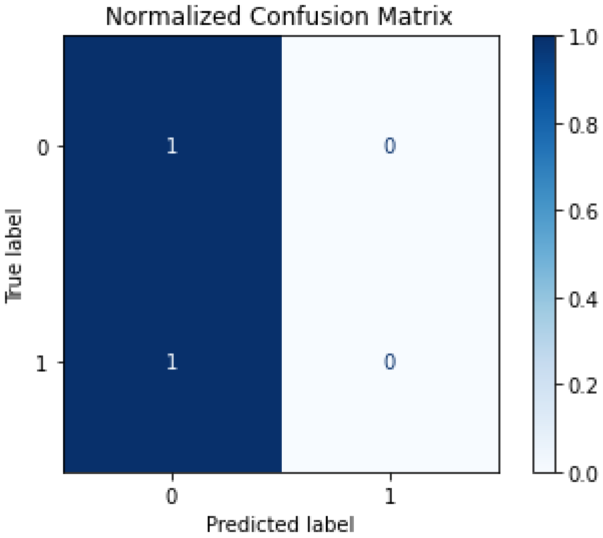
In [85]:

```
titles_options=[("Confusion Matrix,Without Normalization",None),
               ("Normalized Confusion Matrix",'true')]
for title,normalize in titles_options:
    disp=plot_confusion_matrix(logModel,x_test,y_test,
                                                       cmap=plt.cm.Blues,
                               normalize=normalize)
    disp.ax_.set_title(title)

    print(title)
    print(disp.confusion_matrix)

    plt.show()
```

```
Confusion Matrix,Without Normalization
[[19250     0]
 [ 8360     0]]
```



```
Normalized Confusion Matrix
[[1. 0.]
 [1. 0.]]
```



3- NEURAL NETWORK

A neural network is a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain. It is a type of machine learning process, called deep learning, that uses interconnected nodes or neurons in a layered structure that resembles the human brain.

Why are neural networks important?

Neural networks can help computers make intelligent decisions with limited human assistance. This is because they can learn and model the relationships between input and output data that are nonlinear and complex. For instance, they can do the following tasks.

In [88]:

```
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in c:\users\rajak\anaconda3\lib\site-packa
ges (2.10.0)
Requirement already satisfied: numpy>=1.20 in c:\users\rajak\anaconda3\lib\site-pack
ages (from tensorflow) (1.20.1)
Requirement already satisfied: setuptools in c:\users\rajak\anaconda3\lib\site-packa
ges (from tensorflow) (52.0.0.post20210125)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\rajak\anaconda3
\lib\site-packages (from tensorflow) (3.7.4.3)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\raja
k\anaconda3\lib\site-packages (from tensorflow) (0.27.0)
Requirement already satisfied: h5py>=2.9.0 in c:\users\rajak\anaconda3\lib\site-pack
ages (from tensorflow) (2.10.0)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in c:\users\rajak\anaconda3\lib\s
ite-packages (from tensorflow) (0.4.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in c:\users\rajak\anaconda
3\lib\site-packages (from tensorflow) (1.1.2)
Requirement already satisfied: keras<2.11,>=2.10.0 in c:\users\rajak\anaconda3\lib\s
ite-packages (from tensorflow) (2.10.0)
Requirement already satisfied: six>=1.12.0 in c:\users\rajak\anaconda3\lib\site-pack
ages (from tensorflow) (1.15.0)
Requirement already satisfied: flatbuffers>=2.0 in c:\users\rajak\anaconda3\lib\site
-packages (from tensorflow) (22.10.26)
Requirement already satisfied: tensorboard<2.11,>=2.10 in c:\users\rajak\anaconda3\l
ib\site-packages (from tensorflow) (2.10.1)
```

```
Requirement already satisfied: protobuf<3.20,>=3.9.2 in c:\users\rajak\anaconda3\lib
\site-packages (from tensorflow) (3.19.6)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\rajak\anaconda3\lib\site-pa
ckages (from tensorflow) (1.12.1)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\rajak\anaconda3\lib\site-p
ackages (from tensorflow) (1.3.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\rajak\anaconda3\lib\site
-packages (from tensorflow) (2.1.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\rajak\anaconda3\lib\s
ite-packages (from tensorflow) (1.50.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\rajak\anaconda3\lib\site
-packages (from tensorflow) (14.0.6)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\rajak\anaconda3\lib\s
ite-packages (from tensorflow) (0.2.0)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\rajak\anaconda3\lib\sit
e-packages (from tensorflow) (3.3.0)
Requirement already satisfied: tensorflow-estimator<2.11,>=2.10.0 in c:\users\rajak
\anaconda3\lib\site-packages (from tensorflow) (2.10.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\rajak\anaconda3\lib\sit
e-packages (from tensorflow) (1.6.3)
Requirement already satisfied: packaging in c:\users\rajak\anaconda3\lib\site-packag
es (from tensorflow) (20.9)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\rajak\anaconda3\lib\si
te-packages (from astunparse>=1.6.0->tensorflow) (0.36.2)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\rajak\anaconda3\lib\s
ite-packages (from tensorboard<2.11,>=2.10->tensorflow) (2.25.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in c:\users\rajak\anaco
nda3\lib\site-packages (from tensorboard<2.11,>=2.10->tensorflow) (1.8.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in c:\users\rajak\an
aconda3\lib\site-packages (from tensorboard<2.11,>=2.10->tensorflow) (0.4.6)
Requirement already satisfied: google-auth<3,>=1.6.3 in c:\users\rajak\anaconda3\lib
\site-packages (from tensorboard<2.11,>=2.10->tensorflow) (2.14.0)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in c:\users\raj
ak\anaconda3\lib\site-packages (from tensorboard<2.11,>=2.10->tensorflow) (0.6.1)
Requirement already satisfied: markdown>=2.6.8 in c:\users\rajak\anaconda3\lib\site-
packages (from tensorboard<2.11,>=2.10->tensorflow) (3.4.1)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\rajak\anaconda3\lib\site-
packages (from tensorboard<2.11,>=2.10->tensorflow) (1.0.1)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in c:\users\rajak\anaconda3\li
b\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.11,>=2.10->tensorflow)
(5.2.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\rajak\anaconda3\lib
\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.11,>=2.10->tensorflow) (0.
2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\rajak\anaconda3\lib\site-pa
ckages (from google-auth<3,>=1.6.3->tensorboard<2.11,>=2.10->tensorflow) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\rajak\anaconda3
\lib\site-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.11,>=2.10->
tensorflow) (1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in c:\users\rajak\anaconda3\l
ib\site-packages (from markdown>=2.6.8->tensorboard<2.11,>=2.10->tensorflow) (5.0.0)
Requirement already satisfied: zipp>=0.5 in c:\users\rajak\anaconda3\lib\site-packag
es (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard<2.11,>=2.10->tensorfl
ow) (3.4.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\rajak\anaconda3\lib
\site-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.11,
>=2.10->tensorflow) (0.4.8)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\rajak\anaconda3\lib\si
te-packages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tensorflow) (2020.1
2.5)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\users\rajak\anaconda3\lib\sit
e-packages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tensorflow) (4.0.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\rajak\anaconda3\lib
\site-packages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tensorflow) (1.2
6.4)
Requirement already satisfied: idna<3,>=2.5 in c:\users\rajak\anaconda3\lib\site-pac
kages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tensorflow) (2.10)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\rajak\anaconda3\lib\site-
packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboa
rd<2.11,>=2.10->tensorflow) (3.2.2)
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\rajak\anaconda3\lib\site
-packages (from packaging->tensorflow) (2.4.7)
```

TENSORFLOW:

TensorFlow, which competes with frameworks such as PyTorch and Apache MXNet, can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and PDE (partial differential equation)-based simulations. Best of all, TensorFlow supports production prediction at scale, with the same models used for training.

In [89]:

```python
import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense
```

In [93]:

```python
# Define model

model=Sequential()

model.add(Dense (16, input_dim=54, activation ="relu"))
model.add(Dense (8, activation= "relu"))

model.add(Dense (4, activation= "relu"))

model.add(Dense (1, activation ='sigmoid'))

model.summary() #Print model Summary
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 16)                880

 dense_4 (Dense)             (None, 8)                 136

 dense_5 (Dense)             (None, 4)                 36

 dense_6 (Dense)             (None, 1)                 5

=================================================================
Total params: 1,057
Trainable params: 1,057
Non-trainable params: 0
_________________________________________________________________
```

In [95]:

```python
#Compile Model
model.compile(loss ="binary_crossentropy", optimizer="rmsprop", metrics=["accuracy"]
```

In [98]:

```python
#Fit Model
model.fit(x_train, y_train, epochs =5, batch_size=32)
```

```
Epoch 1/5
3452/3452 [==============================] - 9s 2ms/step - loss: 15481796.0000 - acc
uracy: 0.9278
Epoch 2/5
3452/3452 [==============================] - 6s 2ms/step - loss: 2715272.2500 - accu
racy: 0.9278
Epoch 3/5
3452/3452 [==============================] - 6s 2ms/step - loss: 81287.3203 - accura
cy: 0.8480
Epoch 4/5
3452/3452 [==============================] - 5s 2ms/step - loss: 5092.9443 - accurac
y: 0.9617
Epoch 5/5
3452/3452 [==============================] - 6s 2ms/step - loss: 2173.2666 - accurac
y: 0.9617
```

Out[98]:

```
<keras.callbacks.History at 0x1d78e78c2b0>
```

Model Evaluation

In [99]:

```
# Accuracy on the training dataset
trainPred=model.predict(x_train)
trainPred=[1 if y>=0.5 else 0 for y in trainPred]
accuracy_score(y_train,trainPred)
```

```
3452/3452 [==============================] - 7s 2ms/step
```

Out[99]:

```
0.9620054872913969
```

In [103…

```
# Accuracy on the test dataset
y_prediction=model.predict(x_test)
y_prediction=[1 if y>=0.5 else 0   for y in y_prediction]
accuracy_score(y_test,y_prediction)
```

```
863/863 [==============================] - 1s 1ms/step
```

Out[103…

```
0.9633828323071351
```

In [104…

```
confusion_matrix(y_test,y_prediction)
```

Out[104…

```
array([[18955,   295],
       [  716,  7644]], dtype=int64)
```

In [105…

```
f1_score(y_test,y_prediction)
```

Out[105…

```
0.9379716547027425
```

In [ ]: