ds-project

May 15, 2024

#

Bank Marketing

0.0.1 Team Members :

1. Ahmed Ashraf Ahmed Alsayed (Leader)
2. Ahmed Elsayed Ahmed Hassan ( )
.3 Ahmed Elsayed Mahmoud Arafa
4. Radwa Mohamed Said Abd El- Shafi (Meme Lord)
5. Abdelrahman Abdo Hassan Ahmed
6. Nada Adel Ismael Hussainen Shahin
7. Hani Yasser ElMetwaly Sorour Ahmed

0.0.2 Table of contents :

- Introduction
- Import Libraries
- Read Data
- EDA
- PreProcessing
- ML Models
- DL Models

** #

Introduction

Tabel of Contents

`Dataset`

1. Title: Bank Marketing.
2. Sources Created by: Sérgio Moro (ISCTE-IUL), Paulo Cortez (Univ. Minho) and Paulo Rita (ISCTE-IUL).
3. Past Usage: The full dataset (bank-additional-full.csv) was described and analyzed in: S. Moro, P. Cortez, and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing.
4. Relevant Information: This dataset is based on “Bank Marketing” UCI dataset (check the description at: http://archive.ics.uci.edu/ml/datasets/Bank+Marketing.
5. Number of Instances: 41188 for bank-additional-full.csv
6. Number of Attributes: 20 + output attribute.

7. Attribute information:

Attribute	Information
age	numeric
job	categorical: "admin","blue-collar" ,"entrepreneur" , "housemaid","management","retired","selfemployed", "services","student","technician","unemployed", "unknown")
marital	categorical: "divorced","married", "single","unknown")
education	categorical:"basic.4y","basic.6y","basic.9y","high.school","illiterate","professional.course","university.degree","unknown")
default	has credit in default? (categorical: "no","yes","unknown")
housing	has housing loan? (categorical: "no","yes","unknown")
loan	has personal loan? (categorical: "no","yes" ,"unknown")
contact	categorical: "cellular","telephone")
month	last contact month of year (categorical: "jan", "feb", "mar", ..., "nov", "dec")
day_of_week	last contact day of the week (categorical: "mon" ,"tue" ,"wed","thu","fri")
duration	last contact duration, in seconds (numeric).
campaign	number of contacts performed during this campaign and for this client (numeric, includes last contact)
pdays	number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
previous	number of contacts performed before this campaign and for this client (numeric)
poutcome	outcome of the previous marketing campaign (categorical: "failure","nonexistent","success")
emp.var.rate	employment variation rate - quarterly indicator (numeric)
cons.price.idx	consumer price index - monthly indicator (numeric)
cons.conf.idx	consumer confidence index - monthly indicator (numeric)
euribor3m	euribor 3 month rate - daily indicator (numeric)
nr.employed	number of employees - quarterly indicator (numeric)
y	has the client subscribed a term deposit? (binary: "yes","no")

8. Missing Attribute Values: There are several missing values in some categorical attributes, all coded with the "unknown" label. These missing values can be treated as a possible class label or using deletion or imputation techniques.

** #

Import Libraries

Tabel of Contents

[1]:
```
pip install yellowbrick
```

```
Requirement already satisfied: yellowbrick in /opt/conda/lib/python3.10/site-
packages (1.5)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.2 in
/opt/conda/lib/python3.10/site-packages (from yellowbrick) (3.7.5)
Requirement already satisfied: scipy>=1.0.0 in /opt/conda/lib/python3.10/site-
packages (from yellowbrick) (1.11.4)
Requirement already satisfied: scikit-learn>=1.0.0 in
/opt/conda/lib/python3.10/site-packages (from yellowbrick) (1.2.2)
Requirement already satisfied: numpy>=1.16.0 in /opt/conda/lib/python3.10/site-
packages (from yellowbrick) (1.26.4)
Requirement already satisfied: cycler>=0.10.0 in /opt/conda/lib/python3.10/site-
packages (from yellowbrick) (0.12.1)
Requirement already satisfied: contourpy>=1.0.1 in
/opt/conda/lib/python3.10/site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (1.2.0)
Requirement already satisfied: fonttools>=4.22.0 in
/opt/conda/lib/python3.10/site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (4.47.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/conda/lib/python3.10/site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
/opt/conda/lib/python3.10/site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (21.3)
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.10/site-
packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (9.5.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/opt/conda/lib/python3.10/site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/conda/lib/python3.10/site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (2.9.0.post0)
Requirement already satisfied: joblib>=1.1.1 in /opt/conda/lib/python3.10/site-
packages (from scikit-learn>=1.0.0->yellowbrick) (1.4.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/opt/conda/lib/python3.10/site-packages (from scikit-learn>=1.0.0->yellowbrick)
(3.2.0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.10/site-
packages (from python-dateutil>=2.7->matplotlib!=3.0.0,>=2.0.2->yellowbrick)
(1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

[2]:
```python
import pandas as pd
import numpy as np
import plotly.express as px
from plotly.offline import init_notebook_mode
import plotly.graph_objs as go
```

```python
import cufflinks as cf
from plotly.subplots import make_subplots
import plotly.figure_factory as ff
init_notebook_mode(connected=True)
cf.go_offline()
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay
from sklearn.metrics import classification_report
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
import warnings
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
from sklearn.preprocessing import MinMaxScaler,Normalizer
from sklearn.feature_selection import SelectFromModel
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import KFold
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import median_absolute_error
from sklearn.metrics import f1_score,accuracy_score,r2_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import roc_curve,RocCurveDisplay
from sklearn.metrics import auc
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
import scipy.cluster.hierarchy as sch
from sklearn.decomposition import PCA
from yellowbrick.cluster import KElbowVisualizer
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import Ridge
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import SGDRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_validate
from sklearn.model_selection import GridSearchCV
#from sklearn.preprocessing import PolynomialFeature
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
```

```
from sklearn.linear_model import SGDClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import GradientBoostingClassifier
import keras
warnings.filterwarnings('ignore')
px.defaults.template = 'plotly_dark'
```

```
2024-05-14 17:50:26.422786: E
external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable to register
cuDNN factory: Attempting to register factory for plugin cuDNN when one has
already been registered
2024-05-14 17:50:26.422884: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register
cuFFT factory: Attempting to register factory for plugin cuFFT when one has
already been registered
2024-05-14 17:50:26.518596: E
external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to
register cuBLAS factory: Attempting to register factory for plugin cuBLAS when
one has already been registered
```

** #

Read Data

Tabel of Contents

Read in the csv file as a dataframe called data

The tabulated data is meticulously arranged, featuring distinct columns housing a range of dive

[3]:
```
data=pd.read_csv('/kaggle/input/bank-marketing/bank-additional-full.csv',sep=';
↪')
```

Check the head of data

[4]:
```
data.head()
```

[4]:
```
   age        job  marital    education  default housing loan    contact  \
0   56  housemaid  married     basic.4y       no      no   no  telephone
1   57   services  married  high.school  unknown      no   no  telephone
2   37   services  married  high.school       no     yes   no  telephone
3   40     admin.  married     basic.6y       no      no   no  telephone
4   56   services  married  high.school       no      no  yes  telephone

  month day_of_week  …  campaign  pdays  previous     poutcome emp.var.rate  \
0   may         mon  …         1    999         0  nonexistent          1.1
1   may         mon  …         1    999         0  nonexistent          1.1
2   may         mon  …         1    999         0  nonexistent          1.1
3   may         mon  …         1    999         0  nonexistent          1.1
4   may         mon  …         1    999         0  nonexistent          1.1
```

```
   cons.price.idx  cons.conf.idx  euribor3m  nr.employed   y
0          93.994          -36.4      4.857       5191.0  no
1          93.994          -36.4      4.857       5191.0  no
2          93.994          -36.4      4.857       5191.0  no
3          93.994          -36.4      4.857       5191.0  no
4          93.994          -36.4      4.857       5191.0  no

[5 rows x 21 columns]
```

Check the shape of data

[5]:
```
data.shape
```

[5]:
```
(41188, 21)
```

The dataset comprises 41,188 rows and 21 columns

Check the info of data

[6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  int64
 1   job             41188 non-null  object
 2   marital         41188 non-null  object
 3   education       41188 non-null  object
 4   default         41188 non-null  object
 5   housing         41188 non-null  object
 6   loan            41188 non-null  object
 7   contact         41188 non-null  object
 8   month           41188 non-null  object
 9   day_of_week     41188 non-null  object
 10  duration        41188 non-null  int64
 11  campaign        41188 non-null  int64
 12  pdays           41188 non-null  int64
 13  previous        41188 non-null  int64
 14  poutcome        41188 non-null  object
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  object
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

The dataset includes 5 columns of floating-point values, 5 columns of integers, and 11 columns

Description of data

If the DataFrame contains numerical data, the description contains these information for each c

count - The number of not-empty values.
mean - The average (mean) value.
std - The standard deviation.
min - the minimum value.
25% - The 25% percentile*.
50% - The 50% percentile*.
75% - The 75% percentile*.
max - the maximum value.

[7]:
```
data.describe().transpose()
```

[7]:

	count	mean	std	min	25%
age	41188.0	40.024060	10.421250	17.000	32.000
duration	41188.0	258.285010	259.279249	0.000	102.000
campaign	41188.0	2.567593	2.770014	1.000	1.000
pdays	41188.0	962.475454	186.910907	0.000	999.000
previous	41188.0	0.172963	0.494901	0.000	0.000
emp.var.rate	41188.0	0.081886	1.570960	-3.400	-1.800
cons.price.idx	41188.0	93.575664	0.578840	92.201	93.075
cons.conf.idx	41188.0	-40.502600	4.628198	-50.800	-42.700
euribor3m	41188.0	3.621291	1.734447	0.634	1.344
nr.employed	41188.0	5167.035911	72.251528	4963.600	5099.100

	50%	75%	max
age	38.000	47.000	98.000
duration	180.000	319.000	4918.000
campaign	2.000	3.000	56.000
pdays	999.000	999.000	999.000
previous	0.000	0.000	7.000
emp.var.rate	1.100	1.400	1.400
cons.price.idx	93.749	93.994	94.767
cons.conf.idx	-41.800	-36.400	-26.900
euribor3m	4.857	4.961	5.045
nr.employed	5191.000	5228.100	5228.100

For object data types, the describe method typically includes:

Count: The number of non-empty values.
Unique: The number of unique values.
Top: The most frequently occurring value.
Freq: The frequency of the top value.

[8]:
```
data.describe(include='O').transpose()
```

[8]:

	count	unique	top	freq
job	41188	12	admin.	10422
marital	41188	4	married	24928
education	41188	8	university.degree	12168
default	41188	3	no	32588
housing	41188	3	yes	21576
loan	41188	3	no	33950
contact	41188	2	cellular	26144
month	41188	10	may	13769
day_of_week	41188	5	thu	8623
poutcome	41188	3	nonexistent	35563
y	41188	2	no	36548

check for null values in the data

```
Missing Attribute Values: There are several missing values in some categorical attributes, all
```

[9]:
```python
## Since there are no missing values, this step is not applicable in this case
data.replace("unknown",np.nan,inplace=True)
```

[10]:
```python
is_null,precentage = data.isnull().sum(),(data.isnull().sum()/data.shape[0])*100
df = pd.DataFrame()
df['Count'],df['Precentage%']=is_null,precentage
df
```

[10]:

	Count	Precentage%
age	0	0.000000
job	330	0.801204
marital	80	0.194231
education	1731	4.202680
default	8597	20.872584
housing	990	2.403613
loan	990	2.403613
contact	0	0.000000
month	0	0.000000
day_of_week	0	0.000000
duration	0	0.000000
campaign	0	0.000000
pdays	0	0.000000
previous	0	0.000000
poutcome	0	0.000000
emp.var.rate	0	0.000000
cons.price.idx	0	0.000000
cons.conf.idx	0	0.000000
euribor3m	0	0.000000
nr.employed	0	0.000000
y	0	0.000000

[11]:
```
fig = go.Figure()
distribution = df['Count']
bar_trace = go.Bar(x=distribution.index, y=distribution.values, name="Missing␣
↪Values",text=distribution.values, textposition='inside')
fig.add_trace(bar_trace)
fig.update_layout(
    title_text='Missing Values',
    title_x=0.5,
    title_font=dict(size=20),
    xaxis_title="Columns",
    yaxis_title='Count',
    font=dict(size=15),
    width=1000,
    height=700,
    xaxis=dict(tickangle=-90),
    template='plotly_dark'
)
fig.update_annotations(font=dict(size=20))
fig.show()
```

```
Based on the provided data frame:

 The "age" column has 0 missing values, accounting for 0% of the total.
 The "job" column has 330 missing values, making up about 0.80% of the total.
 The "marital" column has 80 missing values, representing approximately 0.19% of the total.
 The "education" column has 1731 missing values, comprising around 4.20% of the total.
 The "default" column has 8597 missing values, accounting for about 20.87% of the total.
 The "housing" and "loan" columns each have 990 missing values, accounting for about 2.40% of t
 The columns "contact", "month", "day_of_week", "duration", "campaign", "pdays", "previous", "p
```

Handle null values

```
To handle null values in your dataset, you can use various methods depending on the type of dat

For Numerical Data:
1. Mean/Median Imputation: Replace missing values with the mean or median of the column.
2. Random Imputation: Replace missing values with randomly sampled values from the distributio
3. Predictive Imputation: Use a predictive model to predict missing values based on other varia

For Categorical Data:
1. Most Frequent Imputation: Replace missing values with the most frequent value in the column
2. Constant Imputation: Replace missing values with a specific constant value.
3. Predictive Imputation: You can also use a predictive model tailored for categorical data to

Best Practices:
all null columns object In this case, using Most Frequent Imputation for handling missing value
```

[12]:
```
data.mode().iloc[0]
```

[12]:
```
age                              31.0
job                            admin.
marital                       married
education           university.degree
default                            no
housing                           yes
loan                               no
contact                      cellular
month                             may
day_of_week                       thu
duration                           85
campaign                          1.0
pdays                           999.0
previous                          0.0
poutcome                  nonexistent
emp.var.rate                      1.4
cons.price.idx                 93.994
cons.conf.idx                   -36.4
euribor3m                       4.857
nr.employed                    5228.1
y                                  no
Name: 0, dtype: object
```

[13]:
```
#data.fillna(data.mode().iloc[0],inplace=True)
key = data.keys()
imp = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
data = imp.fit_transform(data)
data = pd.DataFrame(data,columns=key)
```

[14]:
```
is_null,precentage = data.isnull().sum(),(data.isnull().sum()/data.shape[0])*100
df = pd.DataFrame()
df['Count'],df['Precentage%']=is_null,precentage
df
```

[14]:

	Count	Precentage%
age	0	0.0
job	0	0.0
marital	0	0.0
education	0	0.0
default	0	0.0
housing	0	0.0
loan	0	0.0
contact	0	0.0
month	0	0.0
day_of_week	0	0.0
duration	0	0.0
campaign	0	0.0

```
pdays                 0          0.0
previous              0          0.0
poutcome              0          0.0
emp.var.rate          0          0.0
cons.price.idx        0          0.0
cons.conf.idx         0          0.0
euribor3m             0          0.0
nr.employed           0          0.0
y                     0          0.0
```

check duplicate data

data[data.duplicated(keep=False)]
returns all rows in the DataFrame that are duplicates, including both the original rows and the

[15]:
```python
data[data.duplicated(keep=False)]
```

[15]:
```
       age          job   marital            education default housing loan  \
236     56  blue-collar   married             basic.4y      no      no   no
1265    39  blue-collar   married             basic.6y      no      no   no
1266    39  blue-collar   married             basic.6y      no      no   no
5664    56  blue-collar   married             basic.4y      no      no   no
12260   36      retired   married    university.degree      no      no   no
12261   36      retired   married    university.degree      no      no   no
14155   27   technician    single  professional.course      no      no   no
14234   27   technician    single  professional.course      no      no   no
16819   47   technician  divorced          high.school      no     yes   no
16956   47   technician  divorced          high.school      no     yes   no
18464   32   technician    single  professional.course      no     yes   no
18465   32   technician    single  professional.course      no     yes   no
19451   33       admin.   married    university.degree      no     yes   no
19608   33       admin.   married    university.degree      no     yes   no
20072   55     services   married          high.school      no      no   no
20216   55     services   married          high.school      no      no   no
20531   41   technician   married  professional.course      no     yes   no
20534   41   technician   married  professional.course      no     yes   no
25183   39       admin.   married    university.degree      no      no   no
25217   39       admin.   married    university.degree      no      no   no
28476   24     services    single          high.school      no     yes   no
28477   24     services    single          high.school      no     yes   no
32505   35       admin.   married    university.degree      no     yes   no
32516   35       admin.   married    university.degree      no     yes   no
36950   45       admin.   married    university.degree      no      no   no
36951   45       admin.   married    university.degree      no      no   no
38255   71      retired    single    university.degree      no      no   no
38281   71      retired    single    university.degree      no      no   no

         contact month day_of_week  … campaign pdays previous     poutcome  \
```

236	telephone	may	mon	…	1	999	0	nonexistent
1265	telephone	may	thu	…	1	999	0	nonexistent
1266	telephone	may	thu	…	1	999	0	nonexistent
5664	telephone	may	mon	…	1	999	0	nonexistent
12260	telephone	jul	thu	…	1	999	0	nonexistent
12261	telephone	jul	thu	…	1	999	0	nonexistent
14155	cellular	jul	mon	…	2	999	0	nonexistent
14234	cellular	jul	mon	…	2	999	0	nonexistent
16819	cellular	jul	thu	…	3	999	0	nonexistent
16956	cellular	jul	thu	…	3	999	0	nonexistent
18464	cellular	jul	thu	…	1	999	0	nonexistent
18465	cellular	jul	thu	…	1	999	0	nonexistent
19451	cellular	aug	thu	…	1	999	0	nonexistent
19608	cellular	aug	thu	…	1	999	0	nonexistent
20072	cellular	aug	mon	…	1	999	0	nonexistent
20216	cellular	aug	mon	…	1	999	0	nonexistent
20531	cellular	aug	tue	…	1	999	0	nonexistent
20534	cellular	aug	tue	…	1	999	0	nonexistent
25183	cellular	nov	tue	…	2	999	0	nonexistent
25217	cellular	nov	tue	…	2	999	0	nonexistent
28476	cellular	apr	tue	…	1	999	0	nonexistent
28477	cellular	apr	tue	…	1	999	0	nonexistent
32505	cellular	may	fri	…	4	999	0	nonexistent
32516	cellular	may	fri	…	4	999	0	nonexistent
36950	cellular	jul	thu	…	1	999	0	nonexistent
36951	cellular	jul	thu	…	1	999	0	nonexistent
38255	telephone	oct	tue	…	1	999	0	nonexistent
38281	telephone	oct	tue	…	1	999	0	nonexistent

	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
236	1.1	93.994	-36.4	4.857	5191.0	no
1265	1.1	93.994	-36.4	4.855	5191.0	no
1266	1.1	93.994	-36.4	4.855	5191.0	no
5664	1.1	93.994	-36.4	4.857	5191.0	no
12260	1.4	93.918	-42.7	4.966	5228.1	no
12261	1.4	93.918	-42.7	4.966	5228.1	no
14155	1.4	93.918	-42.7	4.962	5228.1	no
14234	1.4	93.918	-42.7	4.962	5228.1	no
16819	1.4	93.918	-42.7	4.962	5228.1	no
16956	1.4	93.918	-42.7	4.962	5228.1	no
18464	1.4	93.918	-42.7	4.968	5228.1	no
18465	1.4	93.918	-42.7	4.968	5228.1	no
19451	1.4	93.444	-36.1	4.968	5228.1	no
19608	1.4	93.444	-36.1	4.968	5228.1	no
20072	1.4	93.444	-36.1	4.965	5228.1	no
20216	1.4	93.444	-36.1	4.965	5228.1	no
20531	1.4	93.444	-36.1	4.966	5228.1	no

```
20534          1.4         93.444         -36.1     4.966       5228.1    no
25183         -0.1           93.2         -42.0     4.153       5195.8    no
25217         -0.1           93.2         -42.0     4.153       5195.8    no
28476         -1.8         93.075         -47.1     1.423       5099.1    no
28477         -1.8         93.075         -47.1     1.423       5099.1    no
32505         -1.8         92.893         -46.2     1.313       5099.1    no
32516         -1.8         92.893         -46.2     1.313       5099.1    no
36950         -2.9         92.469         -33.6     1.072       5076.2   yes
36951         -2.9         92.469         -33.6     1.072       5076.2   yes
38255         -3.4         92.431         -26.9     0.742       5017.5    no
38281         -3.4         92.431         -26.9     0.742       5017.5    no

[28 rows x 21 columns]
```

keep='first': When you use data.duplicated(keep='first')
it identifies and marks duplicates in the DataFrame, keeping only the first occurrence of each

[16]:
```python
data[data.duplicated(keep='first')]
```

[16]:
```
       age          job   marital            education default housing loan  \
1266    39  blue-collar   married             basic.6y      no      no   no
5664    56  blue-collar   married             basic.4y      no      no   no
12261   36      retired   married    university.degree      no      no   no
14234   27   technician    single  professional.course      no      no   no
16956   47   technician  divorced          high.school      no     yes   no
18465   32   technician    single  professional.course      no     yes   no
19608   33       admin.   married    university.degree      no     yes   no
20216   55     services   married          high.school      no      no   no
20534   41   technician   married  professional.course      no     yes   no
25217   39       admin.   married    university.degree      no      no   no
28477   24     services    single          high.school      no     yes   no
32516   35       admin.   married    university.degree      no     yes   no
36951   45       admin.   married    university.degree      no      no   no
38281   71      retired    single    university.degree      no      no   no

         contact month day_of_week  … campaign pdays previous     poutcome  \
1266   telephone   may         thu  …        1   999        0  nonexistent
5664   telephone   may         mon  …        1   999        0  nonexistent
12261  telephone   jul         thu  …        1   999        0  nonexistent
14234   cellular   jul         mon  …        2   999        0  nonexistent
16956   cellular   jul         thu  …        3   999        0  nonexistent
18465   cellular   jul         thu  …        1   999        0  nonexistent
19608   cellular   aug         thu  …        1   999        0  nonexistent
20216   cellular   aug         mon  …        1   999        0  nonexistent
20534   cellular   aug         tue  …        1   999        0  nonexistent
25217   cellular   nov         tue  …        2   999        0  nonexistent
28477   cellular   apr         tue  …        1   999        0  nonexistent
32516   cellular   may         fri  …        4   999        0  nonexistent
```

```
36951   cellular   jul         thu  …        1   999        0  nonexistent
38281  telephone   oct         tue  …        1   999        0  nonexistent

      emp.var.rate cons.price.idx cons.conf.idx euribor3m nr.employed    y
1266           1.1         93.994         -36.4     4.855      5191.0   no
5664           1.1         93.994         -36.4     4.857      5191.0   no
12261          1.4         93.918         -42.7     4.966      5228.1   no
14234          1.4         93.918         -42.7     4.962      5228.1   no
16956          1.4         93.918         -42.7     4.962      5228.1   no
18465          1.4         93.918         -42.7     4.968      5228.1   no
19608          1.4         93.444         -36.1     4.968      5228.1   no
20216          1.4         93.444         -36.1     4.965      5228.1   no
20534          1.4         93.444         -36.1     4.966      5228.1   no
25217         -0.1           93.2         -42.0     4.153      5195.8   no
28477         -1.8         93.075         -47.1     1.423      5099.1   no
32516         -1.8         92.893         -46.2     1.313      5099.1   no
36951         -2.9         92.469         -33.6     1.072      5076.2  yes
38281         -3.4         92.431         -26.9     0.742      5017.5   no

[14 rows x 21 columns]
```

keep='last': Conversely, when you use data.duplicated(keep='last')
it also identifies and marks duplicates in the DataFrame. However, it keeps only the last occu

[17]:
```python
data[data.duplicated(keep='last')]
```

[17]:
```
      age          job   marital            education default housing loan  \
236    56  blue-collar   married             basic.4y      no      no   no
1265   39  blue-collar   married             basic.6y      no      no   no
12260  36      retired   married    university.degree      no      no   no
14155  27   technician    single  professional.course      no      no   no
16819  47   technician  divorced          high.school      no     yes   no
18464  32   technician    single  professional.course      no     yes   no
19451  33       admin.   married    university.degree      no     yes   no
20072  55     services   married          high.school      no      no   no
20531  41   technician   married  professional.course      no     yes   no
25183  39       admin.   married    university.degree      no      no   no
28476  24     services    single          high.school      no     yes   no
32505  35       admin.   married    university.degree      no     yes   no
36950  45       admin.   married    university.degree      no      no   no
38255  71      retired    single    university.degree      no      no   no

         contact month day_of_week  … campaign pdays previous     poutcome  \
236    telephone   may         mon  …        1   999        0  nonexistent
1265   telephone   may         thu  …        1   999        0  nonexistent
12260  telephone   jul         thu  …        1   999        0  nonexistent
14155   cellular   jul         mon  …        2   999        0  nonexistent
16819   cellular   jul         thu  …        3   999        0  nonexistent
```

```
18464   cellular   jul        thu  ...       1  999          0  nonexistent
19451   cellular   aug        thu  ...       1  999          0  nonexistent
20072   cellular   aug        mon  ...       1  999          0  nonexistent
20531   cellular   aug        tue  ...       1  999          0  nonexistent
25183   cellular   nov        tue  ...       2  999          0  nonexistent
28476   cellular   apr        tue  ...       1  999          0  nonexistent
32505   cellular   may        fri  ...       4  999          0  nonexistent
36950   cellular   jul        thu  ...       1  999          0  nonexistent
38255  telephone   oct        tue  ...       1  999          0  nonexistent

      emp.var.rate cons.price.idx cons.conf.idx euribor3m nr.employed    y
236            1.1         93.994         -36.4     4.857      5191.0   no
1265           1.1         93.994         -36.4     4.855      5191.0   no
12260          1.4         93.918         -42.7     4.966      5228.1   no
14155          1.4         93.918         -42.7     4.962      5228.1   no
16819          1.4         93.918         -42.7     4.962      5228.1   no
18464          1.4         93.918         -42.7     4.968      5228.1   no
19451          1.4         93.444         -36.1     4.968      5228.1   no
20072          1.4         93.444         -36.1     4.965      5228.1   no
20531          1.4         93.444         -36.1     4.966      5228.1   no
25183         -0.1           93.2         -42.0     4.153      5195.8   no
28476         -1.8         93.075         -47.1     1.423      5099.1   no
32505         -1.8         92.893         -46.2     1.313      5099.1   no
36950         -2.9         92.469         -33.6     1.072      5076.2  yes
38255         -3.4         92.431         -26.9     0.742      5017.5   no

[14 rows x 21 columns]
```

Remove duplicates

Remove duplicates keeping the first occurrence

[18]:
```
data.drop_duplicates(keep='first', inplace=True)
```

[19]:
```
data[data.duplicated()]
```

[19]:
```
Empty DataFrame
Columns: [age, job, marital, education, default, housing, loan, contact, month,
day_of_week, duration, campaign, pdays, previous, poutcome, emp.var.rate,
cons.price.idx, cons.conf.idx, euribor3m, nr.employed, y]
Index: []

[0 rows x 21 columns]
```

** #

EDA

Tabel of Contents

Exploratory Data Analysis (EDA) is a crucial step in data analysis where you explore and summa

Helper Functions

[20]:
```
def hist_hue(feature,hue,title_f,title_h,title):
    num_bins=20
    total_hist, _ = np.histogram(data[feature], bins=num_bins)
    fig = make_subplots(rows=1, cols=2, subplot_titles=(title_f, f"{title_f} VS␣
↪{title_h}"))
    histogram_trace_total = go.Bar(x=np.arange(num_bins), y=total_hist,␣
↪name=title_f, text=total_hist, textposition='inside')
    fig.add_trace(histogram_trace_total, row=1, col=1)
    for category in data[hue].unique():
        category_data = data[data[hue] == category][feature]
        category_hist, _ = np.histogram(category_data, bins=num_bins)
        histogram_trace_by_hue = go.Bar(x=np.arange(num_bins), y=category_hist,␣
↪name=f'{title_f} VS {title_h} ({category})', text=category_hist,␣
↪textposition='inside')
        fig.add_trace(histogram_trace_by_hue, row=1, col=2)
    fig.update_layout(
       title_text=title,
       title_x=0.5,
       title_font=dict(size=20),
       font=dict(size=15),
       width=1000,
       height=700,
       barmode='stack',
       template='plotly_dark',
        xaxis_title=title_f,
        yaxis_title='Count',
        xaxis2_title=title_f,
    )
    fig.update_annotations(font=dict(size=20))
    fig.show()
```

[21]:
```
def Bar_hue(feature,hue,title_f,title_h,title):
    fig = make_subplots(rows=1, cols=2, subplot_titles=(title_f, f"{title_f} VS␣
↪{title_h}"))
    distribution = data[feature].value_counts()
    bar_trace = go.Bar(x=distribution.index, y=distribution.values,␣
↪name=title_f,text=distribution.values, textposition='inside')
    fig.add_trace(bar_trace, row=1, col=1)
    for category in data[hue].unique():
        category_data = data[data[hue] == category][feature].value_counts()
        bar_trace_by_hue = go.Bar(x=category_data.index, y=category_data.
↪values, name=f'{title_f} VS {title_h} ({category})', text=category_data.
↪values, textposition='inside')
```

```
        fig.add_trace(bar_trace_by_hue, row=1, col=2)
    fig.update_layout(
        title_text=title,
        title_x=0.5,
        title_font=dict(size=20),
        xaxis_title=title_f,
        yaxis_title='Count',
        xaxis2_title=title_f,
        font=dict(size=15),
        barmode='stack',
        width=1000,
        height=700,
        xaxis=dict(tickangle=-90),
        xaxis2=dict(tickangle=-90),
        template='plotly_dark'
    )
    fig.update_annotations(font=dict(size=20))
    fig.show()
```

[22]:

```
def␣
↪Bar_2hue(feature,hue1,title_f,title_h1,title='',make_subplot=True,hue2='',title_h2=''):
↪
    if make_subplot:
        fig = make_subplots(rows=1, cols=2, subplot_titles=(f"{title_f} VS␣
↪{title_h1}",f"{title_f} VS {title_h2}"))
        for category in data[hue1].unique():
            category_data = data[data[hue1] == category][feature].value_counts()
            bar_trace_by_hue = go.Bar(x=category_data.index, y=category_data.
↪values, name=f'{title_f} VS {title_h1} ({category})',text=category_data.
↪values, textposition='inside')
            fig.add_trace(bar_trace_by_hue, row=1, col=1)
        for category in data[hue2].unique():
            category_data = data[data[hue2] == category][feature].value_counts()
            bar_trace_by_hue = go.Bar(x=category_data.index, y=category_data.
↪values, name=f'{title_f} VS {title_h2} ({category})',text=category_data.
↪values, textposition='inside')
            fig.add_trace(bar_trace_by_hue, row=1, col=2)
        fig.update_layout(
            title_text=title,
            title_x=0.5,
            title_font=dict(size=20),
            xaxis_title=title_f,
            yaxis_title='Count',
            xaxis2_title=title_f,
            font=dict(
                size=15,
            ),
```

```python
            barmode='stack',
            width=1000,
            height=700,
            xaxis=dict(tickangle=-90),
            xaxis1=dict(tickangle=-90),
            xaxis2=dict(tickangle=-90),
            template='plotly_dark'
        )
        fig.update_annotations(font=dict(size=20))
        fig.show()
    else:
        fig = go.Figure()
        for category in data[hue1].unique():
            category_data = data[data[hue1] == category][feature].value_counts()
            bar_trace_by_hue = go.Bar(x=category_data.index, y=category_data.
↪values, name=f'{title_f} VS {title_h1} ({category})',text=category_data.
↪values, textposition='inside')
            fig.add_trace(bar_trace_by_hue)
        fig.update_layout(
            title_text=f'{title_f} VS {title_h1}',
            title_x=0.5,
            title_font=dict(size=20),
            xaxis_title=title_f,
            yaxis_title='Count',
            font=dict(
                size=15,
            ),
            barmode="stack",
            width=800,
            height=700,
            xaxis=dict(tickangle=-90),
            template='plotly_dark'
        )
        fig.update_annotations(font=dict(size=20))
        fig.show()
```

[23]:
```python
def Boxplot_outlier(feature,title):
    fig = make_subplots(rows=1, cols=2, subplot_titles=("Box Plot", "Violin␣
↪Plot"))
    fig.add_trace(
        go.Box(y=data[feature], name='BoxPlot'),
        row=1, col=1
    )
    fig.add_trace(
        go.Violin(y=data[feature], name='ViolinPlot'),
        row=1, col=2
    )
```

```
    fig.update_layout(
        title_text=title,
        title_x=0.5,
        title_font=dict(size=20),
        font=dict(size=15),
        width=1000,
        height=500,
        template='plotly_dark'
    )
    fig.update_annotations(font=dict(size=20))
    fig.show()
```

[24]:
```
def Pie(feature,title_f,title):
    distribution = data[feature].value_counts()
    pie_trace = go.Pie(labels=distribution.index, values=distribution.values,␣
 ↪name=title_f)
    pie_layout = go.Layout(
        title=title,
        title_font=dict(size=20),
        width=600,
        height=500,
        title_x=0.5,
        template='plotly_dark'
    )
    fig_pie = go.Figure(data=[pie_trace], layout=pie_layout)
    fig_pie.show()
```

[25]:
```
def Heatmap(pivot1, title, feature, feature_h1, make_subplot=True,␣
 ↪feature_h2='', pivot2='', color='inferno'):
    fig_heatmap = None
    if make_subplot:
        fig = make_subplots(rows=1, cols=2, subplot_titles=(f"{feature} VS␣
 ↪{feature_h1}", f"{feature} VS {feature_h2}"))
        heat1 = go.Heatmap(
            z=pivot1.values,
            x=pivot1.columns,
            y=pivot1.index,
            colorscale=color,
            colorbar=dict(title='Count'),
            colorbar_x=0.45,
            colorbar_len=0.8
        )
        fig.add_trace(heat1, row=1, col=1)
        heat2 = go.Heatmap(
            z=pivot2.values,
            x=pivot2.columns,
            y=pivot2.index,
```

```
            colorscale=color,
            colorbar=dict(title='Count'),
            colorbar_x=1,
            colorbar_len=0.8
        )
        fig.add_trace(heat2, row=1, col=2)
        fig.update_layout(
            title=title,
            title_x=0.5,
            title_font=dict(size=20),
            width=1100,
            height=500,
            xaxis=dict(title=feature_h1, tickangle=-90),
            xaxis2=dict(title=feature_h2, tickangle=-90),
            yaxis=dict(title=feature, tickangle=-90),
            yaxis2=dict(tickangle=-90),
            font=dict(size=15),
            template='plotly_dark'
        )
        fig_heatmap = fig
    else:
        fig_heatmap = go.Figure(data=go.Heatmap(
            z=pivot1.values,
            x=pivot1.columns,
            y=pivot1.index,
            colorscale=color,
            colorbar=dict(title='Count')
        ))
        fig_heatmap.update_layout(
            title=title,
            title_x=0.5,
            title_font=dict(size=20),
            xaxis=dict(title=feature_h1),
            yaxis=dict(title=feature),
            font=dict(size=15),
            width=800,
            height=500,
            template='plotly_dark'
        )
    fig_heatmap.update_annotations(font=dict(size=20))
    fig_heatmap.show()
```

[26]:
```
def mean_plot(pivot_table,feature,hue,feature_t,hue_t):
    fig = go.Figure()
    for i in data[hue].unique():
        cate = pivot_table[pivot_table.index==i]
```

```
        bar_trace = go.Bar(x=cate.index, y=cate[feature],␣
↪text=round(cate[feature],2), textposition='inside', name=i)
        fig.add_trace(bar_trace)
    fig.update_layout(
        title_text=f'Average {feature_t}',
        title_x=0.5,
        title_font=dict(size=20),
        xaxis_title=hue_t,
        yaxis_title='Average',
        font=dict(size=15),
        barmode='stack',
        width=800,
        height=700,
        xaxis=dict(tickangle=-90),
        template='plotly_dark'
    )

    fig.update_annotations(font=dict(size=20))
    fig.show()
```

[27]:
```
def pivot(values_f,index_f,mean=True):
    if mean:
        return pd.pivot_table(data, values=values_f, index=index_f,␣
↪aggfunc='mean')
    else:
        return pd.pivot_table(data, index=values_f, columns=index_f,␣
↪aggfunc='size', fill_value=0)
def cross_t(index,columns):
    return pd.crosstab(index=data[index], columns=data[columns])
```

What is age distribution?

Find the minimum age

[28]:
```
data.age.min()
```

[28]: 17

Find the maximum age

[29]:
```
data.age.max()
```

[29]: 98

Find the top 5 most frequent ages

[30]:
```
data['age'].value_counts().head(5)
```

[30]:
```
age
31    1947
32    1845
33    1832
36    1779
35    1758
Name: count, dtype: int64
```

Based on the output, it seems that the age group 31 to 36 has the highest counts of observatio

Age 31: 1947 observations
Age 32: 1845 observations
Age 33: 1832 observations
Age 35: 1758 observations
Age 36: 1779 observations

calculate the mean age for each category in the y column

[31]:
```
pivot_table = pivot('age','y')
pivot_table
```

[31]:
```
           age
y
no   39.910743
yes  40.912266
```

Observation: The pivot table reveals that the mean age of individuals who subscribed to the se

Visualization

[32]:
```
hist_hue('age','y','Age','Y','Age Distribution')
```

[33]:
```
Boxplot_outlier('age','Age Distribution')
```

Observation: Based on the figure, it appears that the age column contains some outliers.

What is Job distribution?

calculate the value counts for the job column

[34]:
```
data.job.value_counts().to_frame()
```

[34]:
```
               count
job
admin.         10748
blue-collar     9252
technician      6739
services        3967
management      2924
retired         1718
entrepreneur    1456
```

```
self-employed   1421
housemaid       1060
unemployed      1014
student          875
```

Observation: The value counts for the "job" column indicate the frequency of each job category

The most common job category is "admin." with 10,748 occurrences.
Following "admin.", the next most frequent categories are "blue-collar" (9,252 occurrences) and
Some job categories have relatively fewer occurrences, such as "student" (875 occurrences) and

count the occurrences of each combination of job and y

[35]:
```
pivot_table = pivot('job','y',False)
pivot_table
```

[35]:

y	no	yes
job		
admin.	9360	1388
blue-collar	8614	638
entrepreneur	1332	124
housemaid	954	106
management	2596	328
retired	1284	434
self-employed	1272	149
services	3644	323
student	600	275
technician	6009	730
unemployed	870	144

Visualization

[36]:
```
Pie('job','Job','Job Distribution')
```

[37]:
```
Bar_hue('job','y','Job','Y','Job Distribution')
```

[38]:
```
Heatmap(pivot_table,'Job Vs Y Categories','Job','Y',make_subplot=False)
```

observation based on figure: show frequency between Job and Y

What is marital distribution?

calculate the value counts for the "marital" column

[39]:
```
data.marital.value_counts().to_frame()
```

[39]:

	count
marital	
married	24999
single	11564

```
divorced   4611
```

shows the count of each category in the marital status data. For example, there are 24999 marr

count the occurrences of each combination of marital and y

[40]:
```
pivot_table =pivot('marital','y',False)
pivot_table
```

[40]:
```
y            no   yes
marital
divorced   4135   476
married   22456  2543
single     9944  1620
```

count the occurrences of each combination of marital and job

[41]:
```
pivot_table1 = pivot('marital','job',False)
pivot_table1
```

[41]:
```
job       admin.  blue-collar  entrepreneur  housemaid  management  retired  \
marital
divorced    1293          728           179        161         331      348
married     5506         6699          1074        780        2092     1278
single      3949         1825           203        119         501       92

job       self-employed  services  student  technician  unemployed
marital
divorced            133       532        9         773         124
married             909      2299       42        3681         639
single              379      1136      824        2285         251
```

count the occurrences of each combination of marital , job and y

[42]:
```
data.groupby(['y','job','marital'])['marital'].count().to_frame()
```

[42]:
```
                          marital
y   job         marital
no  admin.      divorced     1158
                married      4834
                single       3368
    blue-collar divorced      675
                married      6275
…                              …
yes technician  married       386
                single        279
    unemployed  divorced       10
                married        86
                single         48
```

```
[66 rows x 1 columns]
```

Visalization

[43]:
```
Pie('marital','Marital','Marital Distribution')
```

[44]:
```
Bar_hue('marital','y','Marital','Y','Marital Distribution')
```

[45]:
```
Bar_2hue('marital','job','Marital','Job',make_subplot=False)
```

[46]:
```
Heatmap(pivot_table,'Marital␣
↪Distribution','Marital','Y',make_subplot=True,feature_h2='Job',pivot2=pivot_table1)
```

What is education distribution?

calculate the value counts for the education column

[47]:
```
data.education.value_counts().to_frame()
```

[47]:
```
                     count
education
university.degree    13893
high.school           9512
basic.9y              6045
professional.course   5240
basic.4y              4175
basic.6y              2291
illiterate              18
```

The observation for the education data shows the count of individuals in each category. For ins

count the occurrences of each combination of education and y

[48]:
```
pivot_table = pivot('education','y',False)
pivot_table
```

[48]:
```
y                       no   yes
education
basic.4y              3747   428
basic.6y              2103   188
basic.9y              5572   473
high.school           8481  1031
illiterate              14     4
professional.course   4645   595
university.degree    11973  1920
```

count the occurrences of each combination of education and job

[49]:
```
pivot_table1 = pivot('education','job',False)
pivot_table1
```

[49]:

```
job                 admin.  blue-collar  entrepreneur  housemaid  management  \
education
basic.4y               129         2317           137        474         100
basic.6y               173         1425            71         77          85
basic.9y               530         3623           210         94         166
high.school           3366          878           234        174         298
illiterate               1            8             2          1           0
professional.course    375          453           135         59          89
university.degree     6174          548           667        181        2186

job                 retired  self-employed  services  student  technician  \
education
basic.4y                597             93       132       26          58
basic.6y                 75             25       226       13          87
basic.9y                145            220       388       99         384
high.school             276            118      2680      357         872
illiterate                3              3         0        0           0
professional.course     241            168       218       43        3317
university.degree       381            794       323      337        2021

job                 unemployed
education
basic.4y                   112
basic.6y                    34
basic.9y                   186
high.school                259
illiterate                   0
professional.course        142
university.degree          281
```

count the occurrences of each combination of education and marital

[50]:

```python
pivot_table2 = pivot('education','marital',False)
pivot_table2
```

[50]:

```
marital             divorced  married  single
education
basic.4y                 489     3233     453
basic.6y                 182     1772     337
basic.9y                 565     4164    1316
high.school             1192     5171    3149
illiterate                 2       15       1
professional.course      657     3161    1422
university.degree       1524     7483    4886
```

count the occurrences of each combination of education , job , marital and y

[51]:

```python
data.groupby(['y','job','marital','education'])['education'].count().to_frame()
```

[51]:
```
                                         education
y   job        marital  education
no  admin.     divorced basic.4y                 5
                        basic.6y                16
                        basic.9y                72
                        high.school            400
                        professional.course     43
…                                              …
yes unemployed single   basic.4y                 3
                        basic.9y                 6
                        high.school             12
                        professional.course      2
                        university.degree       25

[370 rows x 1 columns]
```

Visualization

[52]:
```
Pie('education','Education','Education Distribution')
```

[53]:
```
Bar_hue('education','y','Education','Y','Education Distribution')
```

[54]:
```
Bar_2hue('education','job','Education','Job',title='Education Distribution',hue2='marital',title_h2='Marital')
```

[55]:
```
Heatmap(pivot_table,'Education Vs Y Categories','Education','Y',make_subplot=False)
```

[56]:
```
Heatmap(pivot_table1,'Education Vs Job Categories','Education','Job',make_subplot=False)
```

[57]:
```
Heatmap(pivot_table2,'Education Vs Marital Categories','Education','Marital',make_subplot=False)
```

What is default distribution

calculate the value counts for the default column

[58]:
```
data.default.value_counts().to_frame()
```

[58]:
```
         count
default
no       41171
yes          3
```

based on statistic most people don't have credit
no : 41171/41174 = 99.99271384854521 %
yes : 3/41174 = 0.007286151454801573 %

count the occurrences of each combination of default and y

[59]:
```
cross = cross_t('default','y')
cross
```

[59]:
```
y            no   yes
default
no        36532  4639
yes           3     0
```

There are 36,532 observations where 'default' is 'no' and 'y' is 'no'.
There are 4,639 observations where 'default' is 'no' and 'y' is 'yes'.
There are 3 observations where 'default' is 'yes' and 'y' is 'no'.
There are 0 observations where 'default' is 'yes' and 'y' is 'yes'.

count the occurrences of each combination of default and job

[60]:
```
cross1 = cross_t('default','job')
cross1
```

[60]:
```
job      admin.  blue-collar  entrepreneur  housemaid  management  retired  \
default
no        10748         9252          1456       1060        2924     1718
yes           0            0             0          0           0        0

job      self-employed  services  student  technician  unemployed
default
no                1421      3967      875        6737        1013
yes                  0         0        0           2           1
```

There are 10,748 observations where 'default' is 'no' and the job is 'admin.'.
There are 9,252 observations where 'default' is 'no' and the job is 'blue-collar'.
There are 1,456 observations where 'default' is 'no' and the job is 'entrepreneur'.
And so on...

count the occurrences of each combination of default and marital

[61]:
```
cross2 = cross_t('default','marital')
cross2
```

[61]:
```
marital  divorced  married  single
default
no           4611    24996   11564
yes             0        3       0
```

There are 4,611 observations where 'default' is 'no' and the marital status is 'divorced'.
There are 24,996 observations where 'default' is 'no' and the marital status is 'married'.
There are 11,564 observations where 'default' is 'no' and the marital status is 'single'.

count the occurrences of each combination of default and education

[62]:
```
cross3 = cross_t('default','education')
cross3
```

[62]:
```
education  basic.4y  basic.6y  basic.9y  high.school  illiterate  \
default
no             4175      2291      6045         9511          18
yes               0         0         0            1           0

education  professional.course  university.degree
default
no                        5238              13893
yes                          2                  0
```

```
There are 4,175 observations where 'default' is 'no' and the education level is 'basic.4y'.
There are 2,291 observations where 'default' is 'no' and the education level is 'basic.6y'.
There are 6,045 observations where 'default' is 'no' and the education level is 'basic.9y'.
There are 9,511 observations where 'default' is 'no' and the education level is 'high.school'.
There is 1 observation where 'default' is 'yes' and the education level is 'high.school'.
There are 18 observations where 'default' is 'no' and the education level is 'illiterate'.
There are 5,238 observations where 'default' is 'no' and the education level is 'professional.c
There are 13,893 observations where 'default' is 'no' and the education level is 'university.de
There are 2 observations where 'default' is 'yes' and the education level is 'professional.cou
```

count the occurrences of each combination of default , education , job , marital and y

[63]:
```
data.groupby(['y','job','marital','education','default'])['default'].count().
↪to_frame()
```

[63]:
```
                                                      default
y   job        marital  education           default
no  admin.     divorced basic.4y            no              5
                        basic.6y            no             16
                        basic.9y            no             72
                        high.school         no            400
                        professional.course no             43
…                                                          …
yes unemployed single   basic.4y            no              3
                        basic.9y            no              6
                        high.school         no             12
                        professional.course no              2
                        university.degree   no             25

[372 rows x 1 columns]
```

Visualization

[64]:
```
Pie('default','Default','Default Distribution')
```

[65]:
```
Bar_hue('default','y','Default','Y','Default Distribution')
```

[66]:
```
Bar_2hue('default','job','Default','Job',title='Default␣
↪Distribution',hue2='marital',title_h2='Marital')
```

[67]:
```
Bar_2hue('default','education','Default','Education',make_subplot=False)
```

[68]:
```
Heatmap(cross,'Default␣
↪Distribution','Default','Y',make_subplot=True,feature_h2='Job',pivot2=cross1)
```

[69]:
```
Heatmap(cross2,'Default␣
↪Distribution','Default','Marital',make_subplot=True,feature_h2='Education',pivot2=cross3)
```

What is housing distribution

calculate the value counts for the housing column

[70]:
```
data.housing.value_counts().to_frame()
```

[70]:

housing	count
yes	22560
no	18614

There are 22,560 observations where 'housing' is 'yes'.
There are 18,614 observations where 'housing' is 'no'.

count the occurrences of each combination of housing and y

[71]:
```
cross = cross_t('housing','y')
cross
```

[71]:

housing \ y	no	yes
no	16589	2025
yes	19946	2614

There are 16,589 observations where 'housing' is 'no' and 'y' is 'no'.
There are 2,025 observations where 'housing' is 'no' and 'y' is 'yes'.
There are 19,946 observations where 'housing' is 'yes' and 'y' is 'no'.
There are 2,614 observations where 'housing' is 'yes' and 'y' is 'yes'.

count the occurrences of each combination of housing and job

[72]:
```
cross1 = cross_t('housing','job')
cross1
```

[72]:

housing \ job	admin.	blue-collar	entrepreneur	housemaid	management	retired
no	4787	4302	641	491	1363	782
yes	5961	4950	815	569	1561	936

housing \ job	self-employed	services	student	technician	unemployed
no	641	1817	381	2979	430
yes	780	2150	494	3760	584

```
There are 4,787 observations where 'housing' is 'no' and the job is 'admin.'.
There are 4,302 observations where 'housing' is 'no' and the job is 'blue-collar'.
There are 641 observations where 'housing' is 'no' and the job is 'entrepreneur'.
There are 491 observations where 'housing' is 'no' and the job is 'housemaid'.
There are 1,363 observations where 'housing' is 'no' and the job is 'management'.
And so on...
```

count the occurrences of each combination of housing and marital

[73]:
```python
cross2 = cross_t('housing','marital')
cross2
```

[73]:
```
marital  divorced  married  single
housing
no           2092    11427    5095
yes          2519    13572    6469
```

```
There are 2,092 observations where 'housing' is 'no' and the marital status is 'divorced'.
There are 11,427 observations where 'housing' is 'no' and the marital status is 'married'.
There are 5,095 observations where 'housing' is 'no' and the marital status is 'single'.
There are 2,519 observations where 'housing' is 'yes' and the marital status is 'divorced'.
There are 13,572 observations where 'housing' is 'yes' and the marital status is 'married'.
There are 6,469 observations where 'housing' is 'yes' and the marital status is 'single'.
```

count the occurrences of each combination of housing and education

[74]:
```python
cross3 = cross_t('housing','education')
cross3
```

[74]:
```
education  basic.4y  basic.6y  basic.9y  high.school  illiterate  \
housing
no             1954      1069      2743         4362           8
yes            2221      1222      3302         5150          10

education  professional.course  university.degree
housing
no                        2279               6199
yes                       2961               7694
```

```
There are 1,954 observations where 'housing' is 'no' and the education level is 'basic.4y'.
There are 1,069 observations where 'housing' is 'no' and the education level is 'basic.6y'.
There are 2,743 observations where 'housing' is 'no' and the education level is 'basic.9y'.
There are 4,362 observations where 'housing' is 'no' and the education level is 'high.school'.
There are 8 observations where 'housing' is 'no' and the education level is 'illiterate'.
There are 2,279 observations where 'housing' is 'no' and the education level is 'professional.c
There are 6,199 observations where 'housing' is 'no' and the education level is 'university.deg
There are similar counts for each education level when 'housing' is 'yes'.
```

count the occurrences of each combination of housing and default

[75]:
```python
cross4 = cross_t('housing','default')
cross4
```

[75]:
```
default     no  yes
housing
no       18612    2
yes      22559    1
```

There are 18,612 observations where 'housing' is 'no' and 'default' is 'no'.
There are 2 observations where 'housing' is 'no' and 'default' is 'yes'.
There are 22,559 observations where 'housing' is 'yes' and 'default' is 'no'.
There is 1 observation where 'housing' is 'yes' and 'default' is 'yes'.

count the occurrences of each combination of housing , default , education , job , marital and

[76]:
```python
data.groupby(['y','job','marital','education','default','housing'])['housing'].
↪count().to_frame()
```

[76]:
```
                                                          housing
y   job        marital  education           default housing
no  admin.     divorced basic.4y            no      no              3
                                                    yes             2
                        basic.6y            no      no             11
                                                    yes             5
                        basic.9y            no      no             29
…                                                                 …
yes unemployed single   high.school         no      yes             9
                        professional.course no      no              1
                                                    yes             1
                        university.degree   no      no              5
                                                    yes            20

[696 rows x 1 columns]
```

Visualization

[77]:
```python
Pie('housing','Housing','Housing Distribution')
```

[78]:
```python
Bar_hue('housing','y','Housing','Y','Housing Distribution')
```

[79]:
```python
Bar_2hue('housing','job','Housing','Job',title='Housing␣
↪Distribution',hue2='marital',title_h2='Marital')
```

[80]:
```python
Bar_2hue('housing','education','Housing','Education',title='Housing␣
↪Distribution',hue2='default',title_h2='Default')
```

[81]:
```python
Heatmap(cross,'Housing␣
↪Distribution','Housing','Y',make_subplot=True,feature_h2='Job',pivot2=cross1)
```

[82]:
```
Heatmap(cross2,'Housing␣
↪Distribution','Housing','Marital',make_subplot=True,feature_h2='Education',pivot2=cross3)
```

[83]:
```
Heatmap(cross4,'Housing VS  Default␣
↪Categories','Housing','Default',make_subplot=False)
```

What is loan distribution

calculate the value counts for the loan column

[84]:
```
data.loan.value_counts().to_frame()
```

[84]:
```
       count
loan
no     34926
yes     6248
```

There are 34,926 observations where 'loan' is 'no'.
There are 6,248 observations where 'loan' is 'yes'.

count the occurrences of each combination of loan and y

[85]:
```
cross = cross_t('loan','y')
cross
```

[85]:
```
y         no   yes
loan
no     30970  3956
yes     5565   683
```

There are 30,970 observations where 'loan' is 'no' and 'y' is 'no'.
There are 3,956 observations where 'loan' is 'no' and 'y' is 'yes'.
There are 5,565 observations where 'loan' is 'yes' and 'y' is 'no'.
There are 683 observations where 'loan' is 'yes' and 'y' is 'yes'.

count the occurrences of each combination of loan and job

[86]:
```
cross1 = cross_t('loan','job')
cross1
```

[86]:
```
job   admin.  blue-collar  entrepreneur  housemaid  management  retired  \
loan
no      8981         7886          1250        906        2485     1478
yes     1767         1366           206        154         439      240

job   self-employed  services  student  technician  unemployed
loan
no             1226      3366      733        5750         865
yes             195       601      142         989         149
```

There are 8,981 observations where 'loan' is 'no' and the job is 'admin.'.

```
There are 7,886 observations where 'loan' is 'no' and the job is 'blue-collar'.
There are 1,250 observations where 'loan' is 'no' and the job is 'entrepreneur'.
There are 906 observations where 'loan' is 'no' and the job is 'housemaid'.
There are 2,485 observations where 'loan' is 'no' and the job is 'management'.
And so on...
```

count the occurrences of each combination of loan and marital

[87]:
```python
cross2 = cross_t('loan','marital')
cross2
```

[87]:

marital	divorced	married	single
loan			
no	3936	21214	9776
yes	675	3785	1788

```
There are 3,936 observations where 'loan' is 'no' and the marital status is 'divorced'.
There are 21,214 observations where 'loan' is 'no' and the marital status is 'married'.
There are 9,776 observations where 'loan' is 'no' and the marital status is 'single'.
There are 675 observations where 'loan' is 'yes' and the marital status is 'divorced'.
There are 3,785 observations where 'loan' is 'yes' and the marital status is 'married'.
There are 1,788 observations where 'loan' is 'yes' and the marital status is 'single'.
```

count the occurrences of each combination of loan and education

[88]:
```python
cross3 = cross_t('loan','education')
cross3
```

[88]:

education	basic.4y	basic.6y	basic.9y	high.school	illiterate	\
loan						
no	3551	1961	5162	8069	15	
yes	624	330	883	1443	3	

education	professional.course	university.degree
loan		
no	4447	11721
yes	793	2172

```
There are 3,551 observations where 'loan' is 'no' and the education level is 'basic.4y'.
There are 1,961 observations where 'loan' is 'no' and the education level is 'basic.6y'.
There are 5,162 observations where 'loan' is 'no' and the education level is 'basic.9y'.
There are 8,069 observations where 'loan' is 'no' and the education level is 'high.school'.
There are 15 observations where 'loan' is 'no' and the education level is 'illiterate'.
There are 4,447 observations where 'loan' is 'no' and the education level is 'professional.cou
There are 11,721 observations where 'loan' is 'no' and the education level is 'university.degr
There are similar counts for each education level when 'loan' is 'yes'.
```

count the occurrences of each combination of loan and default

[89]:
```
cross4 = cross_t('loan','default')
cross4
```

[89]:
```
default     no  yes
loan
no       34923    3
yes       6248    0
```

There are 34,923 observations where 'loan' is 'no' and 'default' is 'no'.
There are 3 observations where 'loan' is 'no' and 'default' is 'yes'.
There are 6,248 observations where 'loan' is 'yes' and 'default' is 'no'.
There are 0 observations where 'loan' is 'yes' and 'default' is 'yes'.

count the occurrences of each combination of loan and housing

[90]:
```
cross5 = cross_t('loan','housing')
cross5
```

[90]:
```
housing     no    yes
loan
no       16057  18869
yes       2557   3691
```

There are 16,057 observations where 'loan' is 'no' and 'housing' is 'no'.
There are 18,869 observations where 'loan' is 'no' and 'housing' is 'yes'.
There are 2,557 observations where 'loan' is 'yes' and 'housing' is 'no'.
There are 3,691 observations where 'loan' is 'yes' and 'housing' is 'yes'.

count the occurrences of each combination of loan , housing , default , education , job , mar

[91]:
```
data.
↪groupby(['y','job','marital','education','default','housing','loan'])['loan'].
↪count().to_frame()
```

[91]:
```
                                                                loan
y   job        marital  education           default housing loan
no  admin.     divorced basic.4y            no      no      no     3
                                                    yes     no     2
                        basic.6y            no      no      no     7
                                                            yes    4
                                                    yes     no     5
…                                                                …
yes unemployed single   professional.course no      no      no     1
                                                    yes     yes    1
                        university.degree   no      no      no     5
                                                    yes     no    18
                                                            yes    2

[1170 rows x 1 columns]
```

Visualization

[92]:
```python
Pie('loan','Loan','Loan Distribution')
```

[93]:
```python
Bar_hue('loan','y','Loan','Y','Loan Distribution')
```

[94]:
```python
Bar_2hue('loan','job','Loan','Job',title='Loan Distribution',hue2='marital',title_h2='Marital')
```

[95]:
```python
Bar_2hue('loan','education','Loan','Education',title='Loan Distribution',hue2='default',title_h2='Default')
```

[96]:
```python
Bar_2hue('loan','housing','Loan','Housing',make_subplot=False)
```

[97]:
```python
Heatmap(cross,'Loan Distribution','Loan','Y',make_subplot=True,feature_h2='Job',pivot2=cross1)
```

[98]:
```python
Heatmap(cross2,'Loan Distribution','Loan','Marital',make_subplot=True,feature_h2='Education',pivot2=cross3)
```

[99]:
```python
Heatmap(cross4,'Loan Distribution','Loan','Default',make_subplot=True,feature_h2='Housing',pivot2=cross5)
```

What is contact distribution

calculate the value counts for the contact column

[100]:
```python
data.contact.value_counts().to_frame()
```

[100]:

	count
contact	
cellular	26134
telephone	15040

```
There are 26,134 observations where the contact method is 'cellular'.
There are 15,040 observations where the contact method is 'telephone'.
```

count the occurrences of each combination of contact and y

[101]:
```python
cross = cross_t('contact','y')
cross
```

[101]:

y	no	yes
contact		
cellular	22282	3852
telephone	14253	787

```
There are 22,282 observations where the contact method is 'cellular' and the outcome 'y' is 'n
There are 3,852 observations where the contact method is 'cellular' and the outcome 'y' is 'yes
There are 14,253 observations where the contact method is 'telephone' and the outcome 'y' is '
There are 787 observations where the contact method is 'telephone' and the outcome 'y' is 'yes
```

count the occurrences of each combination of contact and job

[102]:
```python
cross1 = cross_t('contact','job')
cross1
```

[102]:
```
job        admin.  blue-collar  entrepreneur  housemaid  management  retired  \
contact
cellular     7290         5090           855        640        1902     1231
telephone    3458         4162           601        420        1022      487

job        self-employed  services  student  technician  unemployed
contact
cellular             893      2309      671        4633         620
telephone            528      1658      204        2106         394
```

There are 7,290 observations where the contact method is 'cellular' and the job is 'admin.'.
There are 5,090 observations where the contact method is 'cellular' and the job is 'blue-colla
There are 855 observations where the contact method is 'cellular' and the job is 'entrepreneur
There are 640 observations where the contact method is 'cellular' and the job is 'housemaid'.
There are 1,902 observations where the contact method is 'cellular' and the job is 'management
There are similar counts for each occupation when the contact method is 'cellular', and simila

count the occurrences of each combination of contact and marital

[103]:
```python
cross2 = cross_t('contact','marital')
cross2
```

[103]:
```
marital    divorced  married  single
contact
cellular       2907    15253    7974
telephone      1704     9746    3590
```

There are 2,907 observations where the contact method is 'cellular' and the marital status is
There are 15,253 observations where the contact method is 'cellular' and the marital status is
There are 7,974 observations where the contact method is 'cellular' and the marital status is
There are 1,704 observations where the contact method is 'telephone' and the marital status is
There are 9,746 observations where the contact method is 'telephone' and the marital status is
There are 3,590 observations where the contact method is 'telephone' and the marital status is

count the occurrences of each combination of contact and education

[104]:
```python
cross3 = cross_t('contact','education')
cross3
```

[104]:
```
education  basic.4y  basic.6y  basic.9y  high.school  illiterate  \
contact
cellular       2350      1247      3452         5925          15
telephone      1825      1044      2593         3587           3

education  professional.course  university.degree
```

```
contact
cellular                   3475              9670
telephone                  1765              4223
```

```
There are 2,350 observations where the contact method is 'cellular' and the education level is
There are 1,247 observations where the contact method is 'cellular' and the education level is
There are 3,452 observations where the contact method is 'cellular' and the education level is
There are 5,925 observations where the contact method is 'cellular' and the education level is
There are 15 observations where the contact method is 'cellular' and the education level is 'il
There are 3,475 observations where the contact method is 'cellular' and the education level is
There are 9,670 observations where the contact method is 'cellular' and the education level is
```

count the occurrences of each combination of contact and default

[105]:
```
cross4 = cross_t('contact','default')
cross4
```

[105]:
```
default       no  yes
contact
cellular   26131    3
telephone  15040    0
```

```
There are 26,131 observations where 'contact' is 'cellular' and 'default' is 'no'.
There are 3 observations where 'contact' is 'cellular' and 'default' is 'yes'.
There are 15,040 observations where 'contact' is 'telephone' and 'default' is 'no'.
There are 0 observations where 'contact' is 'telephone' and 'default' is 'yes'.
```

count the occurrences of each combination of contact and housing

[106]:
```
cross5 = cross_t('contact','housing')
cross5
```

[106]:
```
housing       no    yes
contact
cellular   11047  15087
telephone   7567   7473
```

```
There are 11,047 observations where 'housing' is 'no' and 'contact' is 'cellular'.
There are 15,087 observations where 'housing' is 'yes' and 'contact' is 'cellular'.
There are 7,567 observations where 'housing' is 'no' and 'contact' is 'telephone'.
There are 7,473 observations where 'housing' is 'yes' and 'contact' is 'telephone'.
```

count the occurrences of each combination of contact and loan

[107]:
```
cross6 = cross_t('contact','loan')
cross6
```

[107]:
```
loan          no   yes
contact
cellular   22073  4061
```

```
telephone  12853  2187
```

```
There are 22,073 observations where 'loan' is 'no' and 'contact' is 'cellular'.
There are 4,061 observations where 'loan' is 'yes' and 'contact' is 'cellular'.
There are 12,853 observations where 'loan' is 'no' and 'contact' is 'telephone'.
There are 2,187 observations where 'loan' is 'yes' and 'contact' is 'telephone'.
```

count the occurrences of each combination of contact , loan , housing , default , education , j

[108]:
```
data.
  ↪groupby(['y','job','marital','education','default','housing','loan','contact'])['contact'].
  ↪count().to_frame()
```

[108]:
```
contact
y   job        marital  education           default housing loan contact
no  admin.     divorced basic.4y            no      no      no   cellular
2
                                                                 telephone
1
                                                    yes     no   cellular
2
                        basic.6y            no      no      no   cellular
5
                                                                 telephone
2
…
…
yes unemployed single   professional.course no      yes     yes  cellular
1
                        university.degree   no      no      no   cellular
5
                                                    yes     no   cellular
16
                                                                 telephone
2
                                                            yes  cellular
2

[1937 rows x 1 columns]
```

Visualization

[109]:
```
Pie('contact','Contact','Contact Distribution')
```

[110]:
```
Bar_hue('contact','y','Contact','Y','Contact Distribution')
```

[111]:
```
Bar_2hue('contact','job','Contact','Job',title='Contact␣
  ↪Distribution',hue2='marital',title_h2='Marital')
```

[112]:
```
Bar_2hue('contact','education','Contact','Education',title='Contact Distribution',hue2='default',title_h2='Default')
```

[113]:
```
Bar_2hue('contact','housing','Contact','Housing',title='Contact Distribution',hue2='loan',title_h2='Loan')
```

[114]:
```
Heatmap(cross,'Contcat Distribution','Contcat','Y',make_subplot=True,feature_h2='Job',pivot2=cross1)
```

[115]:
```
Heatmap(cross2,'Contcat Distribution','Contcat','Marital',make_subplot=True,feature_h2='Education',pivot2=cross3)
```

[116]:
```
Heatmap(cross4,'Contcat Distribution','Contcat','Default',make_subplot=True,feature_h2='Housing',pivot2=cross5)
```

[117]:
```
Heatmap(cross6,'Contact VS  Loan Categories','Contact','Loan',make_subplot=False)
```

What is month distribution

calculate the value counts for the month column

[118]:
```
data.month.value_counts().to_frame()
```

[118]:

	count
month	
may	13766
jul	7169
aug	6175
jun	5318
nov	4100
apr	2631
oct	717
sep	570
mar	546
dec	182

```
There are 13,766 observations in the month of May.
There are 7,169 observations in the month of July.
There are 6,175 observations in the month of August.
There are 5,318 observations in the month of June.
There are 4,100 observations in the month of November.
There are 2,631 observations in the month of April.
There are 717 observations in the month of October.
There are 570 observations in the month of September.
There are 546 observations in the month of March.
There are 182 observations in the month of December.
```

count the occurrences of each combination of month and y

[119]:
```
cross = cross_t('month','y')
cross
```

[119]:
```
y         no  yes
month
apr     2092  539
aug     5520  655
dec       93   89
jul     6521  648
jun     4759  559
mar      270  276
may    12880  886
nov     3684  416
oct      402  315
sep      314  256
```

In April, there are 2,092 observations where the outcome 'y' is 'no', and 539 observations whe
In August, there are 5,520 observations where the outcome 'y' is 'no', and 655 observations whe
In December, there are 93 observations where the outcome 'y' is 'no', and 89 observations where
Similar counts are provided for each month and each outcome category.

count the occurrences of each combination of month and contact

[120]:
```
cross1 = cross_t('month','contact')
cross1
```

[120]:
```
contact  cellular  telephone
month
apr          2444        187
aug          5906        269
dec           149         33
jul          6092       1077
jun           820       4498
mar           486         60
may          5517       8249
nov          3675        425
oct           563        154
sep           482         88
```

In April, there are 2,444 observations where the contact method is 'cellular', and 187 observa
In August, there are 5,906 observations where the contact method is 'cellular', and 269 observa
In December, there are 149 observations where the contact method is 'cellular', and 33 observa
Similar counts are provided for each month and each contact method.

Visualization

[121]:
```
Pie('month','Month','Month Distribution')
```

[122]:
```
Bar_hue('month','y','Month','Y','Month Distribution')
```

[123]:
```python
Bar_2hue('month','contact','Month','Contact',make_subplot=False)
```

[124]:
```python
Heatmap(cross,'Month VS Y Categories','Month','Y',make_subplot=False)
```

[125]:
```python
Heatmap(cross1,'Month VS Contact␣
↪Categories','Month','Contact',make_subplot=False)
```

[126]:
```python
monthly_duration_by_contact = data.groupby(['month', 'contact'])['duration'].
↪sum().reset_index()
custom_colors = {
    'cellular': 'rgb(255, 127, 14)',
    'telephone': 'rgb(255, 0, 0)'
}
fig = px.area(monthly_duration_by_contact, x='month', y='duration',␣
↪color='contact',
              color_discrete_map=custom_colors)
fig.update_xaxes(title='Month')
fig.update_yaxes(title='Total Duration')
fig.update_layout(title_text="Monthly Duration by Contact Type ", title_x=0.5,
                  title_font=dict(size=20),template='plotly_dark')

fig.show()
```

[127]:
```python
fig = go.Figure()
for contact_type in monthly_duration_by_contact['contact'].unique():
    data_subset =␣
↪monthly_duration_by_contact[monthly_duration_by_contact['contact'] ==␣
↪contact_type]
    fig.add_trace(go.Scatter(x=data_subset['month'], y=data_subset['duration'],
                             mode='lines',
                             name=contact_type,
                             stackgroup='one',
                             line=dict(color=custom_colors[contact_type])))
fig.update_layout(title='Monthly Duration by Contact Type',title_x=.
↪5,title_font=dict(size=20),
                  xaxis_title='Month',
                  yaxis_title='Total Duration',
                  template='plotly_dark')

fig.show()
```

What is day_of_week distribution

`calculate the value counts for the day_of_week column`

[128]:
```python
data.day_of_week.value_counts().to_frame()
```

[128]:

day_of_week	count
thu	8617
mon	8511
wed	8134
tue	8086
fri	7826

There are 8,617 observations that occurred on Thursday ('thu').
There are 8,511 observations that occurred on Monday ('mon').
There are 8,134 observations that occurred on Wednesday ('wed').
There are 8,086 observations that occurred on Tuesday ('tue').
There are 7,826 observations that occurred on Friday ('fri').

count the occurrences of each combination of day_of_week and y

[129]:
```python
cross = cross_t('day_of_week','y')
cross
```

[129]:

y day_of_week	no	yes
fri	6980	846
mon	7664	847
thu	7573	1044
tue	7133	953
wed	7185	949

On Fridays, there are 6,980 observations where the outcome 'y' is 'no', and 846 observations wh
On Mondays, there are 7,664 observations where the outcome 'y' is 'no', and 847 observations wh
On Thursdays, there are 7,573 observations where the outcome 'y' is 'no', and 1,044 observatio
Similar counts are provided for each day of the week and each outcome category.

count the occurrences of each combination of day_of_week and month

[130]:
```python
cross1 = cross_t('day_of_week','month')
cross1
```

[130]:

month day_of_week	apr	aug	dec	jul	jun	mar	may	nov	oct	sep
fri	610	1070	24	1012	1147	94	2857	755	142	115
mon	702	1221	53	1515	1251	143	2641	766	129	90
thu	768	1346	45	1668	967	99	2536	903	163	122
tue	251	1295	25	1517	970	140	2809	813	148	118
wed	300	1243	35	1457	983	70	2923	863	135	125

There are 610 observations that occurred on Fridays in April ('apr').
There are 1,070 observations that occurred on Fridays in August ('aug').
There are 24 observations that occurred on Fridays in December ('dec').
There are similar counts for each combination of day of the week and month.

count the occurrences of each combination of day_of_week and contact

[131]:
```
cross2 = cross_t('day_of_week','contact')
cross2
```

[131]:
```
contact      cellular  telephone
day_of_week
fri              4644       3182
mon              5533       2978
thu              5801       2816
tue              5104       2982
wed              5052       3082
```

```
On Fridays, there are 4,644 observations where the contact method is 'cellular', and 3,182 obse
On Mondays, there are 5,533 observations where the contact method is 'cellular', and 2,978 obse
On Thursdays, there are 5,801 observations where the contact method is 'cellular', and 2,816 of
Similar counts are provided for each day of the week and each contact method.
```

```
Visualization
```

[132]:
```
Pie('day_of_week','Day','Day Distribution')
```

[133]:
```
Bar_hue('day_of_week','y','Day','Y','Day Distribution')
```

[134]:
```
Bar_2hue('day_of_week','month','Day','Month',title='Default␣
↪Distribution',hue2='contact',title_h2='Contact')
```

[135]:
```
Heatmap(cross,'Day␣
↪Distribution','Day','Y',make_subplot=True,feature_h2='Month',pivot2=cross1)
```

[136]:
```
Heatmap(cross2,'Day VS  Contact Categories','Day','Contact',make_subplot=False)
```

[137]:
```
day_duration_by_contact = data.groupby(['day_of_week', 'contact'])['duration'].
↪sum().reset_index()
fig = px.area(day_duration_by_contact, x='day_of_week', y='duration',␣
↪color='contact',
              color_discrete_map=custom_colors)
fig.update_xaxes(title='Day')
fig.update_yaxes(title='Total Duration')
fig.update_layout(title_text="Days Duration by Contact Type ", title_x=0.5,
                  title_font=dict(size=20),template='plotly_dark')

fig.show()
```

[138]:
```
fig = go.Figure()
for contact_type in day_duration_by_contact['contact'].unique():
    data_subset = day_duration_by_contact[day_duration_by_contact['contact'] ==␣
↪contact_type]
    fig.add_trace(go.Scatter(x=data_subset['day_of_week'],␣
↪y=data_subset['duration'],
```

```
                                mode='lines',
                                name=contact_type,
                                stackgroup='one',
                                line=dict(color=custom_colors[contact_type]))
fig.update_layout(title='Days Duration by Contact Type',title_x=.
 ↪5,title_font=dict(size=20),
                  xaxis_title='Day',
                  yaxis_title='Total Duration',
                  template='plotly_dark')

fig.show()
```

What is duration distribution?

Find the minimum duration

[139]:
```
data.duration.min()
```

[139]: 0

Find the maximum duration

[140]:
```
data.duration.max()
```

[140]: 4918

Find the top 5 most frequent duration

[141]:
```
data.duration.value_counts().to_frame().head()
```

[141]:

duration	count
85	170
90	170
136	167
73	167
124	163

calculate the mean duration for each category in the contact column

[142]:
```
pivot_table = pivot('duration','contact')
pivot_table
```

[142]:

contact	duration
cellular	263.569067
telephone	249.208976

Visualization

[143]:
```
Boxplot_outlier('duration','Duration Distribution')
```

Observation: Based on the figure, it appears that the duration column contains some outliers.

[144]:
```
mean_plot(pivot_table,'duration','contact','Duration','Contact')
```

What is campaign distribution?

Find the minimum campaign

[145]:
```
data.campaign.min()
```

[145]: 1

Find the maximum campaign

[146]:
```
data.campaign.max()
```

[146]: 56

Find the top 5 most frequent duration

[147]:
```
data.campaign.value_counts().to_frame().head()
```

[147]:

campaign	count
1	17632
2	10568
3	5340
4	2650
5	1599

count the occurrences of each combination of campaign and contact

[148]:
```
cross = cross_t('campaign','contact')
cross
```

[148]:

contact campaign	cellular	telephone
1	11753	5879
2	6675	3893
3	3303	2037
4	1583	1067
5	998	601
6	577	402
7	359	270
8	219	181
9	148	135
10	116	109
11	101	76
12	54	71
13	49	43
14	31	38

```
15              22          29
16              17          34
17              30          28
18              11          22
19              11          15
20              16          14
21               6          18
22               7          10
23               6          10
24               8           7
25               3           5
26               2           6
27               5           6
28               2           6
29               5           5
30               5           2
31               2           5
32               0           4
33               4           0
34               2           1
35               2           3
37               0           1
39               0           1
40               1           1
41               0           1
42               0           2
43               1           1
56               0           1
```

Average between campaign and duration

[149]:
```
pivot_table = pivot('duration','campaign')
pivot_table
```

[149]:
```
            duration
campaign
1         256.804049
2         279.706945
3         270.044569
4          251.46566
5         227.759225
6         225.955056
7         223.330684
8            189.525
9         211.526502
10        208.706667
11        207.723164
12           185.288
```

```
13        175.282609
14        134.594203
15             152.0
16        117.352941
17        199.258621
18         85.424242
19        164.692308
20         62.233333
21         82.583333
22        113.529412
23          129.1875
24        111.466667
25            45.875
26           305.625
27        100.909091
28            118.25
29             118.0
30              69.0
31         33.571429
32             30.25
33              37.5
34              37.0
35              49.6
37              17.0
39              44.0
40              15.5
41              25.0
42             135.5
43              40.5
56             261.0
```

Visualization

[150]:
```
Bar_hue('campaign','contact','Campaign','Contact','Cmpaign Distribution')
```

[151]:
```
Boxplot_outlier('campaign','Campaign Distribution')
```

[152]:
```
fig = go.Figure()
scatter_trace = go.Scatter(
    x=pivot_table.index,
    y=pivot_table['duration'],
    mode='lines+markers',
    marker=dict(
        size=10,
        color='blue',
        symbol='circle',
        opacity=0.8
    ),
```

```
    line=dict(
        color='red',
        width=2
    )
)
fig.add_trace(scatter_trace)
fig.update_layout(
    title_text='Average between Campaign and Duration',
    title_x=0.5,
    title_font=dict(size=20),
    xaxis_title='Campaign',
    yaxis_title='Average Duration',
    font=dict(size=15),
    width=800,
    height=700,
    xaxis=dict(tickangle=-90),
    template='plotly_dark'
)
fig.update_annotations(font=dict(size=20))
fig.show()
```

[153]:

```
if not pd.api.types.is_numeric_dtype(data['duration']):
    data['duration'] = pd.to_numeric(data['duration'], errors='coerce')

grouped_data = data.groupby(['y', 'campaign'])['duration'].mean().reset_index()
fig = go.Figure()
for category, group in grouped_data.groupby('y'):
    fig.add_trace(go.Scatter(
        x=group['campaign'],
        y=group['duration'],
        mode='markers',
        marker=dict(
            size=group['duration'],
            sizemode='area',
            sizeref=2. * max(group['duration']) / (40. ** 2),
            color=group['campaign'],
            opacity=0.7,
            line=dict(width=0.5, color='DarkSlateGrey')
        ),
        name=category
    ))
fig.update_layout(
    title='Mean Duration VS. Campaign',
    title_x=.5,
    title_font=dict(size=20),
    xaxis_title='Campaign',
    yaxis_title='Mean Duration',
```

```
    template='plotly_dark'
)
fig.show()
```

[154]:
```
grouped_data = data.groupby(['contact', 'campaign'])['duration'].mean().
 ↪reset_index()
fig = go.Figure()
for category, group in grouped_data.groupby('contact'):
    fig.add_trace(go.Scatter(
        x=group['campaign'],
        y=group['duration'],
        mode='markers',
        marker=dict(size=group['duration'], sizemode='area', sizeref=2.
 ↪*max(group['duration'])/(40.**2),
                    color=group['campaign'],
                    opacity=0.7,
                    line=dict(width=0.5, color='DarkSlateGrey')),
        name=category
    ))
fig.update_layout(title='Mean Duration VS. Campaign',title_x=.
 ↪5,title_font=dict(size=20),
                  xaxis_title='Campaign',
                  yaxis_title='Mean Duration',
                 template='plotly_dark')
fig.show()
```

Observation: Based on the figure, it appears that the campaign column contains some outliers.

What is pdays distribution?

Find the minimum pdays

[155]:
```
data.pdays.min()
```

[155]: 0

Find the maximum pdays

[156]:
```
data.pdays.max()
```

[156]: 999

Find the top 5 most frequent pdays

[157]:
```
data.pdays.value_counts().to_frame().head()
```

[157]:

pdays	count
999	39659
3	439

```
6        412
4        118
9         64
```

calculate the mean pdays for each category in the contact column

[158]:
```python
pivot_table = pivot('pdays','contact')
pivot_table
```

[158]:
```
                pdays
contact
cellular   945.728859
telephone  991.540891
```

Visualization

[159]:
```python
mean_plot(pivot_table,'pdays','contact','Pdays','Contact')
```

What is previous distribution?

Find the minimum previous

[160]:
```python
data.previous.min()
```

[160]: 0

Find the maximum previous

[161]:
```python
data.previous.max()
```

[161]: 7

Find the top 5 most frequent previous

[162]:
```python
data.previous.value_counts().to_frame().head()
```

[162]:
```
          count
previous
0         35549
1          4561
2           754
3           216
4            70
```

count the occurrences of each combination of previous and contact

[163]:
```python
cross = cross_t('previous','contact')
cross
```

[163]:
```
contact   cellular  telephone
previous
```

```
0             20912      14637
1              4240        321
2               691         63
3               205         11
4                63          7
5                17          1
6                 5          0
7                 1          0
```

Visualization

[164]:
```python
Bar_hue('previous','contact','Previous','Contact','Previous Distribution')
```

[165]:
```python
Heatmap(cross,'Previous VS Contact Categories','Previous','Contact',make_subplot=False)
```

What is poutcome distribution?

calculate the value counts for the poutcome column

[166]:
```python
data.poutcome.value_counts().to_frame()
```

[166]:
```
             count
poutcome
nonexistent  35549
failure       4252
success       1373
```

count the occurrences of each combination of poutcome and y

[167]:
```python
cross = cross_t('poutcome','y')
cross
```

[167]:
```
y               no   yes
poutcome
failure       3647   605
nonexistent  32409  3140
success        479   894
```

count the occurrences of each combination of poutcome and contact

[168]:
```python
cross1 = cross_t('poutcome','contact')
cross1
```

[168]:
```
contact      cellular  telephone
poutcome
failure          3952        300
nonexistent     20912      14637
success          1270        103
```

Visualization

[169]:
```
Pie('poutcome','Poutcome','Poutcome Distribution')
```

[170]:
```
Bar_hue('poutcome','y','Poutcome','Y','Poutcome Distribution')
```

[171]:
```
Bar_2hue('poutcome','contact','Poutcome','Contact',make_subplot=False)
```

[172]:
```
Heatmap(cross,'Poutcome␣
↪Distribution','Poutcome','Y',make_subplot=True,feature_h2='Contact',pivot2=cross1)
```

What is emp.var.rate distribution?

Find the minimum emp.var.rate

[173]:
```
data['emp.var.rate'].min()
```

[173]: -3.4

Find the maximum emp.var.rate

[174]:
```
data['emp.var.rate'].max()
```

[174]: 1.4

Visualization

[175]:
```
Boxplot_outlier('emp.var.rate','Emp.Var.Rate Distribution')
```

What is cons.price.idx distribution?

Find the minimum cons.price.idx

[176]:
```
data['cons.price.idx'].min()
```

[176]: 92.201

Find the maximum cons.price.idx

[177]:
```
data['cons.price.idx'].max()
```

[177]: 94.767

Visualization

[178]:
```
Boxplot_outlier('cons.price.idx','Cons.Price.Idx Distribution')
```

What is cons.conf.idx distribution?

Find the minimum cons.conf.idx

[179]:
```
data['cons.conf.idx'].min()
```

[179]: -50.8

Find the maximum cons.conf.idx

[180]:
```python
data['cons.conf.idx'].max()
```

[180]: -26.9

Visualization

[181]:
```python
Boxplot_outlier('cons.conf.idx','Cons.Conf.Idx Distribution')
```

What is euribor3m distribution?

Find the minimum euribor3m

[182]:
```python
data.euribor3m.min()
```

[182]: 0.634

Find the maximum euribor3m

[183]:
```python
data.euribor3m.max()
```

[183]: 5.045

Visualization

[184]:
```python
Boxplot_outlier('euribor3m','Euribor3m Distribution')
```

What is nr.employed distribution?

Find the minimum nr.employed

[185]:
```python
data['nr.employed'].min()
```

[185]: 4963.6

Find the maximum nr.employed

[186]:
```python
data['nr.employed'].max()
```

[186]: 5228.1

Visualization

[187]:
```python
Boxplot_outlier('nr.employed','Nr.Employed Distribution')
```

What is y distribution?

calculate the value counts for the y column

[188]:
```python
data.y.value_counts().to_frame()
```

[188]:

y	count
no	36535
yes	4639

Visualization

[189]:
```
Pie('y','Traget','Traget Distribution')
```

Observation based on figure the dataset is imbanlanced

Remove Outliers

applying outlier removal techniques using the interquartile range (IQR) method to the specified columns ('age', 'duration', 'campaign', 'cons.conf.idx')

[190]:
```
cols = ['age', 'duration', 'campaign', 'cons.conf.idx']

for col in cols:
    q1 = data[col].quantile(0.25)
    q3 = data[col].quantile(0.75)
    iqr = q3 - q1
    upper = q3 + (1.5 * iqr)
    lower = q1 - (1.5 * iqr)

    data.loc[data[col] > upper, col] = upper
    data.loc[data[col] < lower, col] = lower

    print(f'For {col} :\n', q1, q3, iqr, upper, lower)
```

```
For age :
 32.0 47.0 15.0 69.5 9.5
For duration :
 102.0 319.0 217.0 644.5 -223.5
For campaign :
 1.0 3.0 2.0 6.0 -2.0
For cons.conf.idx :
 -42.7 -36.4 6.300000000000004 -26.949999999999992 -52.150000000000006
```

Observations:

- For the 'age' column, the first quartile (Q1) is approximately 32.0, the third quartile (Q3) is approximately 47.0, and the interquartile range (IQR) is 15.0. The upper bound for outlier detection is 69.5, and the lower bound is 9.5.
- For the 'duration' column, Q1 is approximately 102.0, Q3 is approximately 319.0, and the IQR is 217.0. The upper bound for outlier detection is 644.5, and the lower bound is -223.5.
- For the 'campaign' column, Q1 is 1.0, Q3 is 3.0, and the IQR is 2.0. The upper bound for outlier detection is 6.0, and the lower bound is -2.0.
- For the 'cons.conf.idx' column, Q1 is approximately -42.7, Q3 is approximately -36.4, and the IQR is approximately 6.3. The upper bound for outlier detection is approximately -26.95, and the lower bound is approximately -52.15.

[191]:
```
fig = make_subplots(rows=2, cols=2, subplot_titles=cols)
for i, col in enumerate(cols, start=1):
    q1 = data[col].quantile(0.25)
```

```
    q3 = data[col].quantile(0.75)
    iqr = q3 - q1
    upper = q3 + (1.5 * iqr)
    lower = q1 - (1.5 * iqr)
    data[col][data[col]>upper] = upper
    data[col][data[col]<lower] = lower
    trace = go.Box(y=data[col], name=col)
    fig.add_trace(trace, row=(i - 1) // 2 + 1, col=(i - 1) % 2 + 1)
fig.update_layout(title_text='Box Plot of Columns without Outliers',title_x=0.
 ↪5, title_y=0.95,
                  height=800, width=1000, template='plotly_dark')
fig.show()
```

** #

PreProcessing

Tabel of Contents

[192]:
```
#create new features or transform existing features to improve the performance␣
 ↪of your data science model
#data['duration']=data['duration']/60
```

[193]:
```
ct = ColumnTransformer(transformers=[('encoder',␣
 ↪OneHotEncoder(),['education'])])
data_ = ct.fit_transform(data[['education']])
```

[194]:
```
pd.DataFrame(data_.toarray(),columns=data['education'].unique())
```

[194]:
```
       basic.4y  high.school  basic.6y  basic.9y  professional.course  \
0           1.0          0.0       0.0       0.0                  0.0
1           0.0          0.0       0.0       1.0                  0.0
2           0.0          0.0       0.0       1.0                  0.0
3           0.0          1.0       0.0       0.0                  0.0
4           0.0          0.0       0.0       1.0                  0.0
…            …            …         …         …                    …
41169       0.0          0.0       0.0       0.0                  0.0
41170       0.0          0.0       0.0       0.0                  0.0
41171       0.0          0.0       0.0       0.0                  0.0
41172       0.0          0.0       0.0       0.0                  0.0
41173       0.0          0.0       0.0       0.0                  0.0

       university.degree  illiterate
0                    0.0         0.0
1                    0.0         0.0
2                    0.0         0.0
3                    0.0         0.0
4                    0.0         0.0
```

```
…                     …         …
41169                 1.0       0.0
41170                 1.0       0.0
41171                 0.0       1.0
41172                 1.0       0.0
41173                 1.0       0.0

[41174 rows x 7 columns]
```

Transform Object Columns

[195]:
```python
data2=data.copy()
object=data2.select_dtypes(include='object').columns
label=LabelEncoder()
for col in object:
    data2[col] = label.fit_transform(data2[col])
data2.head()
```

[195]:
```
   age  job  marital  education  default  housing  loan  contact  month  \
0   39    3        1          0        0        0     0        1      6
1   40    7        1          3        0        0     0        1      6
2   20    7        1          3        0        1     0        1      6
3   23    0        1          1        0        0     0        1      6
4   39    7        1          3        0        0     1        1      6

   day_of_week  …  campaign  pdays  previous  poutcome  emp.var.rate  \
0            1  …         0     26         0         1             8
1            1  …         0     26         0         1             8
2            1  …         0     26         0         1             8
3            1  …         0     26         0         1             8
4            1  …         0     26         0         1             8

   cons.price.idx  cons.conf.idx  euribor3m  nr.employed  y
0              18             16        287            8  0
1              18             16        287            8  0
2              18             16        287            8  0
3              18             16        287            8  0
4              18             16        287            8  0

[5 rows x 21 columns]
```

Show Correlation

[196]:
```python
data2.corr()
```

[196]:
```
                     age       job   marital  education   default   housing  \
age             1.000000 -0.014713 -0.397301  -0.124721  0.002010 -0.002133
job            -0.014713  1.000000  0.025377   0.131910  0.013701  0.007435
```

```
marital        -0.397301  0.025377  1.000000   0.111375 -0.002388  0.011345
education      -0.124721  0.131910  0.111375   1.000000  0.002577  0.016452
default         0.002010  0.013701 -0.002388   0.002577  1.000000 -0.003680
housing        -0.002133  0.007435  0.011345   0.016452 -0.003680  1.000000
loan           -0.007670 -0.011802  0.006495   0.009289 -0.003610  0.036398
contact         0.011662 -0.031847 -0.054634  -0.110425 -0.006476 -0.077803
month          -0.027123 -0.033017 -0.008822  -0.084502 -0.004530 -0.016868
day_of_week    -0.019192 -0.004149  0.002440  -0.016863  0.006079  0.003329
duration        0.002060 -0.002335  0.007733  -0.019020 -0.006338 -0.010737
campaign        0.003200 -0.007181 -0.011543   0.002299 -0.005187 -0.011019
pdays          -0.029975 -0.024770 -0.035479  -0.045991  0.001638 -0.011442
previous        0.016304  0.022185  0.037718   0.037724  0.002765  0.021653
poutcome        0.018395  0.006647  0.002458   0.016768 -0.006195 -0.012577
emp.var.rate    0.013960 -0.007612 -0.081283  -0.028145  0.005324 -0.055101
cons.price.idx -0.000150 -0.022616 -0.055310  -0.085634 -0.002861 -0.075450
cons.conf.idx   0.124852  0.048777 -0.028161   0.084596  0.004757 -0.026991
euribor3m      -0.032588 -0.027161 -0.078541  -0.056048  0.004853 -0.040914
nr.employed    -0.011279 -0.022999 -0.079942  -0.034934  0.006332 -0.036220
y               0.021529  0.025596  0.045892   0.057237 -0.003042  0.011144

                    loan   contact     month  day_of_week  …  campaign  \
age            -0.007670  0.011662 -0.027123    -0.019192  …  0.003200
job            -0.011802 -0.031847 -0.033017    -0.004149  … -0.007181
marital         0.006495 -0.054634 -0.008822     0.002440  … -0.011543
education       0.009289 -0.110425 -0.084502    -0.016863  …  0.002299
default        -0.003610 -0.006476 -0.004530     0.006079  … -0.005187
housing         0.036398 -0.077803 -0.016868     0.003329  … -0.011019
loan            1.000000 -0.013393 -0.007111    -0.009492  …  0.012112
contact        -0.013393  1.000000  0.276465    -0.009591  …  0.071659
month          -0.007111  0.276465  1.000000     0.027697  … -0.063819
day_of_week    -0.009492 -0.009591  0.027697     1.000000  … -0.051029
duration       -0.006608 -0.036197  0.008218     0.031255  … -0.080191
campaign        0.012112  0.071659 -0.063819    -0.051029  …  1.000000
pdays          -0.001016  0.116138 -0.047412    -0.010465  …  0.059798
previous       -0.002194 -0.212905  0.103149    -0.004109  … -0.083856
poutcome       -0.000209  0.118773 -0.065009     0.018737  …  0.030048
emp.var.rate    0.000827  0.350374 -0.188202     0.035965  …  0.142389
cons.price.idx -0.005576  0.584651 -0.006331     0.002217  …  0.112596
cons.conf.idx  -0.013157  0.243189 -0.018811     0.035204  … -0.024704
euribor3m       0.005097  0.274110 -0.197034     0.023543  …  0.134282
nr.employed     0.006289  0.176080 -0.266913     0.023306  …  0.142881
y              -0.004486 -0.144774 -0.006057     0.015964  … -0.069413

                   pdays  previous  poutcome  emp.var.rate  cons.price.idx  \
age            -0.029975  0.016304  0.018395      0.013960       -0.000150
job            -0.024770  0.022185  0.006647     -0.007612       -0.022616
marital        -0.035479  0.037718  0.002458     -0.081283       -0.055310
```

```
education       -0.045991  0.037724  0.016768     -0.028145       -0.085634
default          0.001638  0.002765 -0.006195      0.005324       -0.002861
housing         -0.011442  0.021653 -0.012577     -0.055101       -0.075450
loan            -0.001016 -0.002194 -0.000209      0.000827       -0.005576
contact          0.116138 -0.212905  0.118773      0.350374        0.584651
month           -0.047412  0.103149 -0.065009     -0.188202       -0.006331
day_of_week     -0.010465 -0.004109  0.018737      0.035965        0.002217
duration        -0.062397  0.037364  0.038345     -0.048517       -0.000610
campaign         0.059798 -0.083856  0.030048      0.142389        0.112596
pdays            1.000000 -0.579460 -0.486940      0.257257        0.090841
previous        -0.579460  1.000000 -0.313096     -0.405913       -0.197490
poutcome        -0.486940 -0.313096  1.000000      0.192381        0.198958
emp.var.rate     0.257257 -0.405913  0.192381      1.000000        0.750857
cons.price.idx   0.090841 -0.197490  0.198958      0.750857        1.000000
cons.conf.idx   -0.108991 -0.020104  0.166272      0.122006       -0.024101
euribor3m        0.384726 -0.489973  0.089883      0.868708        0.546774
nr.employed      0.375595 -0.499543  0.087034      0.845379        0.409424
y               -0.320975  0.230197  0.129814     -0.286795       -0.140511

                 cons.conf.idx  euribor3m  nr.employed         y
age                   0.124852  -0.032588    -0.011279  0.021529
job                   0.048777  -0.027161    -0.022999  0.025596
marital              -0.028161  -0.078541    -0.079942  0.045892
education             0.084596  -0.056048    -0.034934  0.057237
default               0.004757   0.004853     0.006332 -0.003042
housing              -0.026991  -0.040914    -0.036220  0.011144
loan                 -0.013157   0.005097     0.006289 -0.004486
contact               0.243189   0.274110     0.176080 -0.144774
month                -0.018811  -0.197034    -0.266913 -0.006057
day_of_week           0.035204   0.023543     0.023306  0.015964
duration             -0.004162  -0.062160    -0.074227  0.401301
campaign             -0.024704   0.134282     0.142881 -0.069413
pdays                -0.108991   0.384726     0.375595 -0.320975
previous             -0.020104  -0.489973    -0.499543  0.230197
poutcome              0.166272   0.089883     0.087034  0.129814
emp.var.rate          0.122006   0.868708     0.845379 -0.286795
cons.price.idx       -0.024101   0.546774     0.409424 -0.140511
cons.conf.idx         1.000000  -0.123080    -0.064467  0.069911
euribor3m            -0.123080   1.000000     0.912388 -0.368182
nr.employed          -0.064467   0.912388     1.000000 -0.355120
y                     0.069911  -0.368182    -0.355120  1.000000

[21 rows x 21 columns]
```

[197]:
```
corr = data2.corr()
corr=corr.round(2)
fig = ff.create_annotated_heatmap(z=corr.values,
```

```
                                  x=corr.columns.tolist(),
                                  y=corr.columns.tolist(),
                                  colorscale='RdBu',
                                  hoverinfo='none',
                                  showscale=True,
                                  ygap=1,
                                  xgap=1
                                 )
fig.update_xaxes(side='bottom')
fig.update_layout(
    title_text='Heatmap',
    title_x=0.5,
    width=1000,
    height=1000,
    xaxis=dict(showgrid=True),
    yaxis=dict(showgrid=True, autorange='reversed'),
    template='plotly_dark'
)
fig.show()
```

[198]:

```
mask = np.triu(np.ones_like(corr, dtype=bool))
df_mask = corr.mask(mask)
df_mask_rounded = df_mask.round(2)
fig = ff.create_annotated_heatmap(z=df_mask_rounded.values,
                                  x=df_mask_rounded.columns.tolist(),
                                  y=df_mask_rounded.columns.tolist(),
                                  colorscale='RdBu',
                                  hoverinfo='none',
                                  showscale=True,
                                  ygap=1,
                                  xgap=1
                                 )
fig.update_xaxes(side='bottom')
fig.update_layout(
    title_text='Heatmap',
    title_x=0.5,
    width=1000,
    height=1000,
    xaxis=dict(showgrid=True),
    yaxis=dict(showgrid=True, autorange='reversed'),
    template='plotly_dark'
)
for annotation in fig.layout.annotations:
    if annotation.text == 'nan':
        annotation.text = ""

fig.show()
```

Classification

[199]:
```
X_classification = data2.iloc[:,:-1]
y_classification = data2.iloc[:,-1]
key = X_classification.keys()
X_classification.head()
```

[199]:
```
   age  job  marital  education  default  housing  loan  contact  month  \
0   39    3        1          0        0        0     0        1      6
1   40    7        1          3        0        0     0        1      6
2   20    7        1          3        0        1     0        1      6
3   23    0        1          1        0        0     0        1      6
4   39    7        1          3        0        0     1        1      6

   day_of_week  duration  campaign  pdays  previous  poutcome  emp.var.rate  \
0            1     261.0         0     26         0         1             8
1            1     149.0         0     26         0         1             8
2            1     226.0         0     26         0         1             8
3            1     151.0         0     26         0         1             8
4            1     307.0         0     26         0         1             8

   cons.price.idx  cons.conf.idx  euribor3m  nr.employed
0              18             16        287            8
1              18             16        287            8
2              18             16        287            8
3              18             16        287            8
4              18             16        287            8
```

[200]:
```
y_classification.head()
```

[200]:
```
0    0
1    0
2    0
3    0
4    0
Name: y, dtype: int64
```

Clustering

[201]:
```
X_cluster = data2.copy()
X_cluster.head()
```

[201]:
```
   age  job  marital  education  default  housing  loan  contact  month  \
0   39    3        1          0        0        0     0        1      6
1   40    7        1          3        0        0     0        1      6
2   20    7        1          3        0        1     0        1      6
3   23    0        1          1        0        0     0        1      6
4   39    7        1          3        0        0     1        1      6
```

```
   day_of_week  …  campaign  pdays  previous  poutcome  emp.var.rate  \
0            1  …         0     26         0         1             8
1            1  …         0     26         0         1             8
2            1  …         0     26         0         1             8
3            1  …         0     26         0         1             8
4            1  …         0     26         0         1             8

   cons.price.idx  cons.conf.idx  euribor3m  nr.employed  y
0              18             16        287            8  0
1              18             16        287            8  0
2              18             16        287            8  0
3              18             16        287            8  0
4              18             16        287            8  0

[5 rows x 21 columns]
```

Regression

[202]:
```
X_regression = data2.drop('duration',axis=1)
y_regression = data2['duration']
key = X_regression.keys()
X_regression.head()
```

[202]:
```
   age  job  marital  education  default  housing  loan  contact  month  \
0   39    3        1          0        0        0     0        1      6
1   40    7        1          3        0        0     0        1      6
2   20    7        1          3        0        1     0        1      6
3   23    0        1          1        0        0     0        1      6
4   39    7        1          3        0        0     1        1      6

   day_of_week  campaign  pdays  previous  poutcome  emp.var.rate  \
0            1         0     26         0         1             8
1            1         0     26         0         1             8
2            1         0     26         0         1             8
3            1         0     26         0         1             8
4            1         0     26         0         1             8

   cons.price.idx  cons.conf.idx  euribor3m  nr.employed  y
0              18             16        287            8  0
1              18             16        287            8  0
2              18             16        287            8  0
3              18             16        287            8  0
4              18             16        287            8  0
```

[203]:
```
y_regression=y_regression/y_regression.max()
y_regression.head()
```

[203]:
```
0    0.404965
1    0.231187
2    0.350659
3    0.234290
4    0.476338
Name: duration, dtype: float64
```

Banlanced Data

[204]:
```
over = RandomOverSampler(sampling_strategy='minority')
X_classification_over,y_classification_over=over.
 ↪fit_resample(X_classification,y_classification)
```

[205]:
```
under = RandomUnderSampler()
X_classification_under,y_classification_under=under.
 ↪fit_resample(X_classification,y_classification)
```

** #

ML Models

Tabel of Contents

Classification Models

`RandomForestClassifier`

[206]:
```
def Split(X, y='', classification=1):
    if classification == 1:
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
 ↪1, random_state=44, shuffle=True, stratify=y)
    elif classification == 2:
        X_train, X_test = train_test_split(X, test_size=0.1, random_state=44,␣
 ↪shuffle=True)
        print('X_train shape is ', X_train.shape)
        print('X_test shape is ', X_test.shape)
        return X_train, X_test
    else:
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
 ↪1, random_state=44, shuffle=True)
    print('X_train shape is ', X_train.shape)
    print('X_test shape is ', X_test.shape)
    print('y_train shape is ', y_train.shape)
    print('y_test shape is ', y_test.shape)
    return X_train, y_train, X_test, y_test

def SelectFeature(model, X_train, y_train):
    FeatureSelection = SelectFromModel(estimator=model)
    FeatureSelection.fit(X_train, y_train)
    return X_train.iloc[:, FeatureSelection.get_support()].columns
```

```python
def Search(model, parameters, X_train, y_train):
    GridSearchModel = GridSearchCV(model, parameters, cv=5,␣
 ↪return_train_score=True)
    GridSearchModel.fit(X_train, y_train)
    return GridSearchModel.best_estimator_

def cross_validation(model, X_train, y_train):
    CrossValidateValues1 = cross_validate(model, X_train, y_train, cv=5,␣
 ↪return_train_score=True)
    print('Train Score Value : ', CrossValidateValues1['train_score'], "\t␣
 ↪Mean", CrossValidateValues1['train_score'].mean())
    print('Test Score Value : ', CrossValidateValues1['test_score'], "\t Mean",␣
 ↪CrossValidateValues1['test_score'].mean())

def PipeLine(model, X_train, y_train, flage=0):
    if flage == 0:
        steps = [('model', model)]
    elif flage == 1:
        steps = [('scaling', MinMaxScaler()), ('model', model)]
    elif flage == 2:
        steps = [('scaling', Normalizer()), ('model', model)]
    elif flage == 3:
        steps = [('pca', PCA()), ('model', model)]
    elif flage == 4:
        steps = [('scaling', MinMaxScaler()), ('pca', PCA()), ('model', model)]
    else:
        steps = [('scaling', Normalizer()), ('pca', PCA()), ('model', model)]
    return Pipeline(steps).fit(X_train, y_train)

def Area(fprValue2, tprValue2, AUCValue):
    fig = go.Figure()
    fig.add_trace(go.Scatter(x=fprValue2, y=tprValue2,
                    mode='lines',
                    name='ROC curve (AUC = {:.2f})'.format(AUCValue),␣
 ↪line=dict(color='red')))
    fig.add_shape(type='line',
        x0=0, y0=0, x1=1, y1=1,
        line=dict(color='orange', width=2, dash='dash'),
        name='Random Guessing')
    fig.update_layout(
                    title='Receiver Operating Characteristic (ROC) Curve',
                    title_x=.5,
                    xaxis_title='False Positive Rate',
                    yaxis_title='True Positive Rate',
                    xaxis=dict(range=[0, 1], constrain='domain'),
                    yaxis=dict(range=[0, 1]),
```

```
                      legend=dict(x=0.01, y=0.99),
                      showlegend=True,
                      template='plotly_dark'
                     )
    fig.update_annotations(font=dict(size=20))
    fig.show()

def Check(model='', X_train='', y_train='', X_test='', y_test='',␣
 ↪cluster=0,y_train2='',y_train_pred='',y_test2='',y_pred=''):
    if cluster:
        train = accuracy_score(y_train2, y_train_pred)
        test = accuracy_score(y_test2, y_pred)
        y_pred = y_pred
        y_test = y_test2

    else:
        y_pred = model.predict(X_test)
        train = accuracy_score(y_train, model.predict(X_train))
        test = accuracy_score(y_test, y_pred)
    print('Model Train Score is : ', train)
    print('Model Test Score is : ', test)
    F1Score = f1_score(y_test, y_pred)
    print('F1 Score is : ', F1Score)
    RecallScore = recall_score(y_test, y_pred)
    print('Recall Score is : ', RecallScore)
    PrecisionScore = precision_score(y_test, y_pred)
    print('Precision Score is : ', PrecisionScore)
    fprValue2, tprValue2, thresholdsValue2 = roc_curve(y_test, y_pred)
    AUCValue = auc(fprValue2, tprValue2)
    print('AUC Value  : ', AUCValue)
    Area(fprValue2, tprValue2, AUCValue)
    ClassificationReport = classification_report(y_test, y_pred)
    print('Classification Report is : ', ClassificationReport)
    CM = confusion_matrix(y_test, y_pred)
    print('Confusion Matrix is : \n', CM)
    disp = ConfusionMatrixDisplay(confusion_matrix=CM, display_labels=[0, 1])
    disp.plot(cmap='Blues')
    values = [train, test, F1Score, RecallScore, PrecisionScore, AUCValue]
    return values

def Models(models, X_train, y_train, X_test, y_test):
    print('Apply Model With Normal Data : \n')
    model = PipeLine(models, X_train, y_train)
    value1 =␣
 ↪Check(model=model,X_train=X_train,y_train=y_train,X_test=X_test,y_test=y_test)
    print("\n\n Apply Model With Feature Selection :\n")
    try:
```

```
        feature = SelectFeature(model, X_train, y_train)
    except:
        feature = SelectFeature(RandomForestClassifier(max_depth=20), X_train,␣
↪y_train)
    X_train1 = X_train.loc[:, feature]
    X_test1 = X_test.loc[:, feature]
    model = PipeLine(models, X_train1, y_train, flage=1)
    value2 =␣
↪Check(model=model,X_train=X_train1,y_train=y_train,X_test=X_test1,y_test=y_test)
    print("\n\n Apply Model With Normal Data With Scaling :\n")
    model = PipeLine(models, X_train, y_train, flage=1)
    value3 =␣
↪Check(model=model,X_train=X_train,y_train=y_train,X_test=X_test,y_test=y_test)
    print("\n\n Apply Model With Normal Data With Normalize :\n")
    model = PipeLine(models, X_train, y_train, flage=2)
    value4 =␣
↪Check(model=model,X_train=X_train,y_train=y_train,X_test=X_test,y_test=y_test)
    print("\n\n Apply Model With Normal Data With PCA :\n")
    model = PipeLine(models, X_train, y_train, flage=3)
    value5 =␣
↪Check(model=model,X_train=X_train,y_train=y_train,X_test=X_test,y_test=y_test)
    print("\n\n Apply Model With Normal Data With PCA and Scaling :\n")
    model = PipeLine(models, X_train, y_train, flage=4)
    value6 =␣
↪Check(model=model,X_train=X_train,y_train=y_train,X_test=X_test,y_test=y_test)
    print("\n\n Apply Model With Normal Data With PCA and Normalize :\n")
    model = PipeLine(models, X_train, y_train, flage=5)
    value7 =␣
↪Check(model=model,X_train=X_train,y_train=y_train,X_test=X_test,y_test=y_test)
    return [value1, value2, value3, value4, value5, value6, value7]
def models_draw(df):
    figure = go.Figure()
    for column in df.columns:
        trace = go.Bar(
            x=df.index,
            y=df[column],
            name=column,
            text=df[column].values.round(2),
            textposition='inside'
        )
        figure.add_trace(trace)
    figure.update_layout(
        barmode='group',
        title='Performance Metrics Comparison',
        title_x=.5,
        xaxis=dict(title='Models'),
```

```
        yaxis=dict(title='Score'),
        template='plotly_dark',
        width=1100,
        height=700
    )
    figure.show()
```

[207]:
```
X_train,y_train,X_test,y_test=Split(X_classification,y_classification)
```

```
X_train shape is  (37056, 20)
X_test shape is  (4118, 20)
y_train shape is  (37056,)
y_test shape is  (4118,)
```

[208]:
```
Search(RandomForestClassifier(max_depth=20),{'max_depth':
 ↪[5,10,15,20,25,30,35,40]},X_train,y_train)
```

[208]:
```
RandomForestClassifier(max_depth=10)
```

[209]:
```
cross_validation(RandomForestClassifier(max_depth=10),X_train,y_train)
```

```
Train Score Value :  [0.94126973 0.93860685 0.94066453 0.94005735 0.93813459]
Mean 0.9397466100167595
Test Score Value :  [0.91230437 0.91229254 0.91121306 0.90878424 0.91701525]
Mean 0.9123218916645179
```

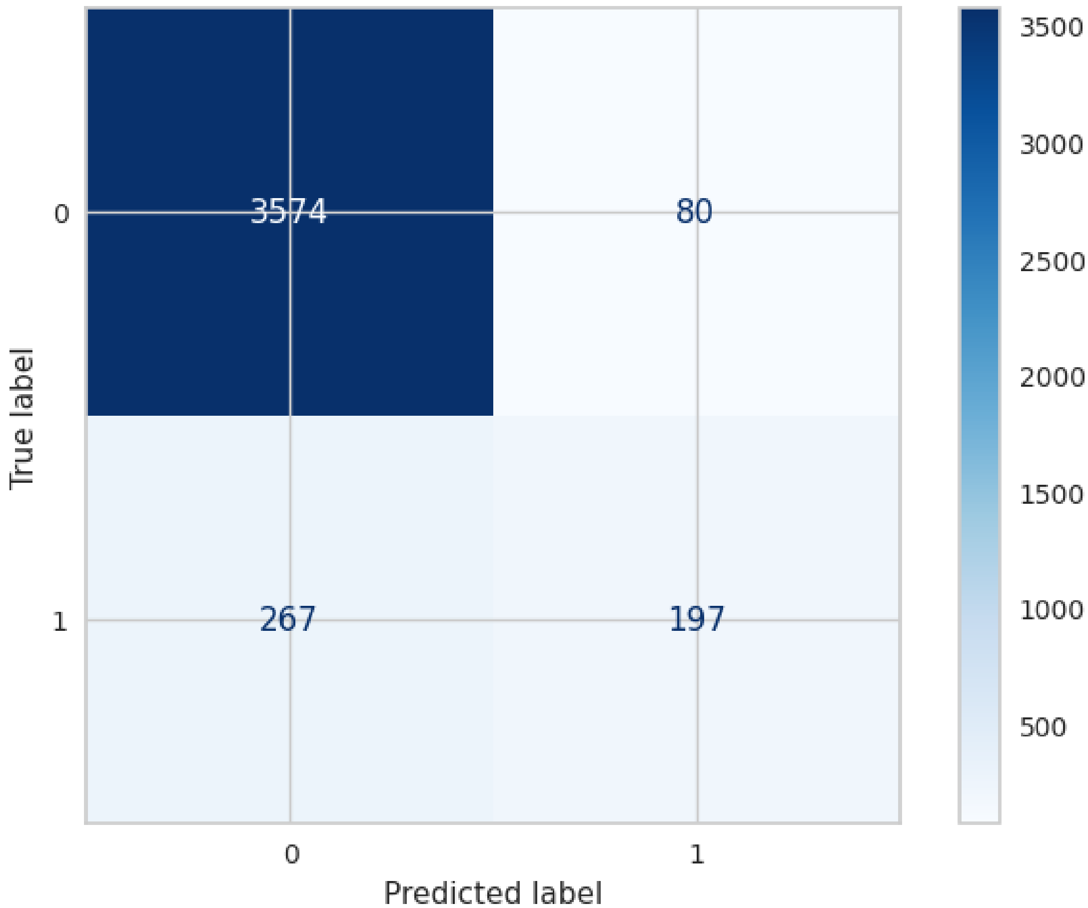
[210]:
```
Values =␣
 ↪Models(RandomForestClassifier(max_depth=10),X_train,y_train,X_test,y_test)
```

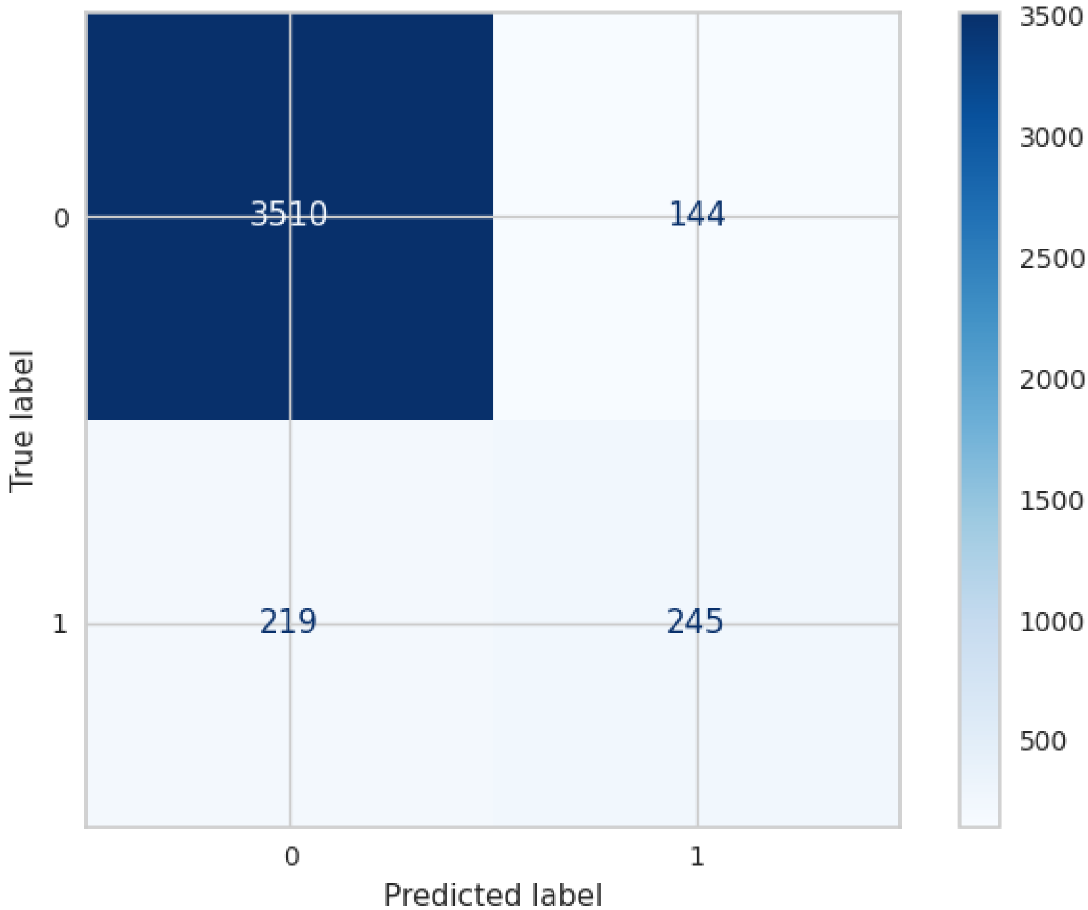
```
Apply Model With Normal Data :

Model Train Score is :  0.9350442573402418
Model Test Score is :  0.9157357940747936
F1 Score is :  0.5317139001349527
Recall Score is :  0.4245689655172414
Precision Score is :  0.7111913357400722
AUC Value  :  0.701337575259989

Classification Report is :                 precision    recall  f1-score
support

           0       0.93      0.98      0.95      3654
           1       0.71      0.42      0.53       464

    accuracy                           0.92      4118
   macro avg       0.82      0.70      0.74      4118
weighted avg       0.91      0.92      0.91      4118


Confusion Matrix is :
```

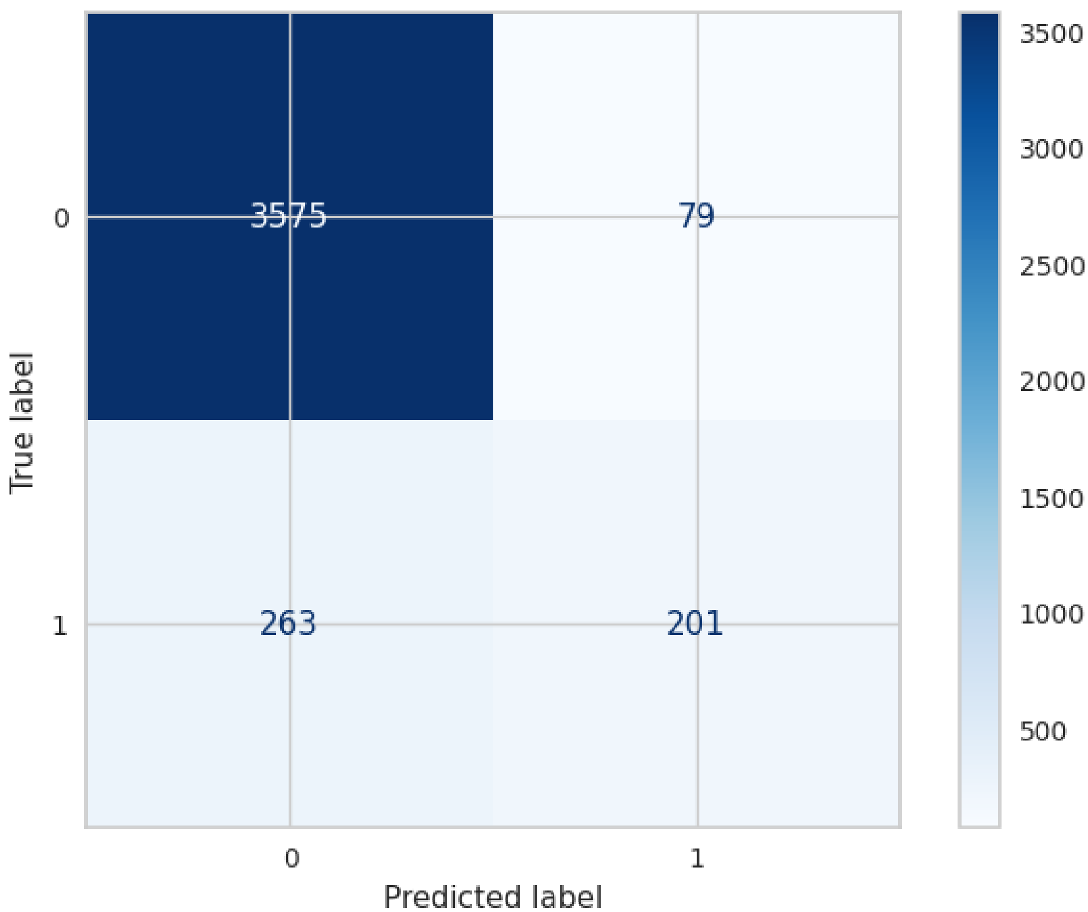
```
 [[3574   80]
 [ 267  197]]


 Apply Model With Feature Selection :

Model Train Score is :  0.9345045336787565
Model Test Score is :  0.9118504128217582
F1 Score is :  0.5744431418522861
Recall Score is :  0.5280172413793104
Precision Score is :  0.6298200514138818
AUC Value  :  0.7443041871921183
Classification Report is :                precision    recall  f1-score
support

           0       0.94      0.96      0.95      3654
           1       0.63      0.53      0.57       464

    accuracy                           0.91      4118
   macro avg       0.79      0.74      0.76      4118
weighted avg       0.91      0.91      0.91      4118

Confusion Matrix is :
 [[3510  144]
 [ 219  245]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.9346934369602763
Model Test Score is :  0.9169499757163672
F1 Score is :  0.5403225806451613
Recall Score is :  0.4331896551724138
Precision Score is :  0.7178571428571429
AUC Value  :  0.7057847564313081
Classification Report is :                precision    recall  f1-score
support

           0       0.93      0.98      0.95      3654
           1       0.72      0.43      0.54       464

    accuracy                           0.92      4118
   macro avg       0.82      0.71      0.75      4118
weighted avg       0.91      0.92      0.91      4118

Confusion Matrix is :
 [[3575   79]
```

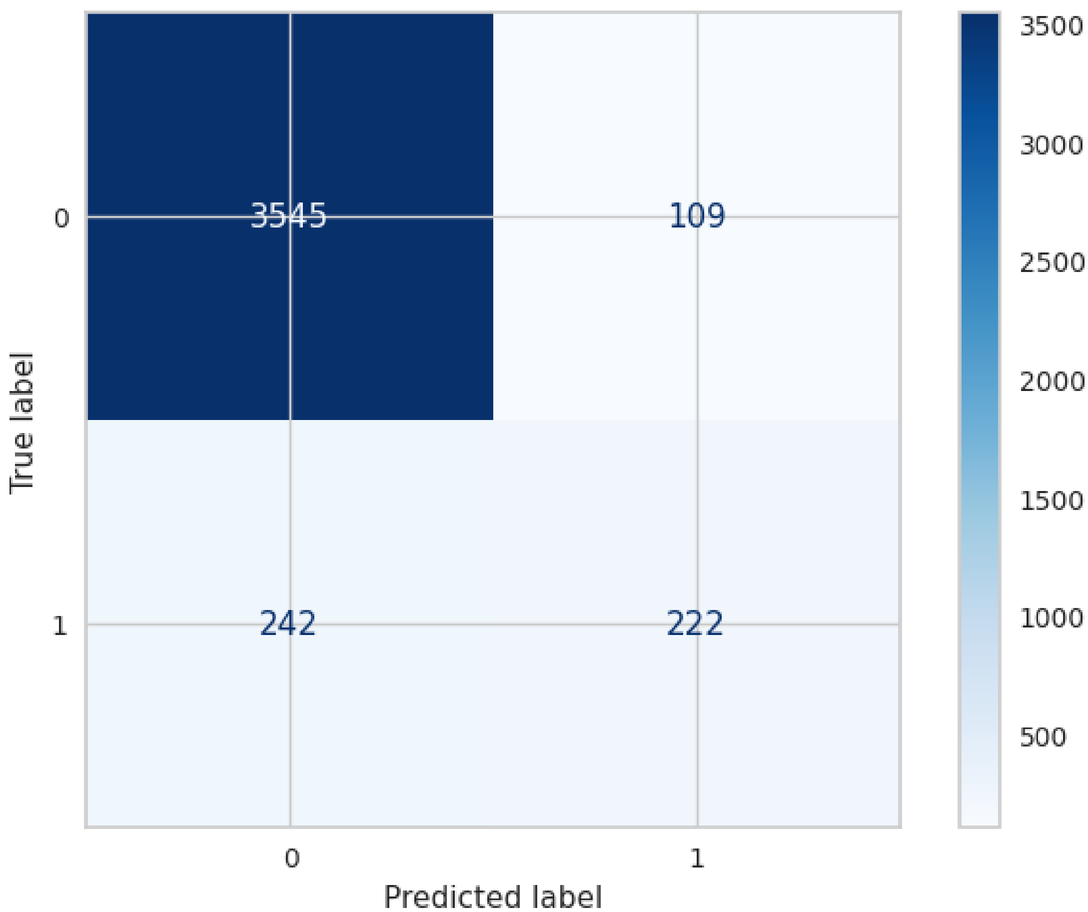
```
 [ 263  201]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.9424384715025906
Model Test Score is :  0.9147644487615347
F1 Score is :  0.5584905660377358
Recall Score is :  0.47844827586206895
Precision Score is :  0.6706948640483383
AUC Value  :  0.7243089764641488

Classification Report is :                precision    recall  f1-score
support

           0       0.94      0.97      0.95      3654
           1       0.67      0.48      0.56       464

    accuracy                           0.91      4118
   macro avg       0.80      0.72      0.76      4118
weighted avg       0.91      0.91      0.91      4118

Confusion Matrix is :
 [[3545  109]
 [ 242  222]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.9458927029360967
Model Test Score is :  0.9154929577464789
F1 Score is :  0.5347593582887701
Recall Score is :  0.43103448275862066
Precision Score is :  0.704225352112676
AUC Value  :  0.7040229885057471

Classification Report is :                precision    recall  f1-score
support

           0       0.93      0.98      0.95      3654
           1       0.70      0.43      0.53       464

    accuracy                           0.92      4118
   macro avg       0.82      0.70      0.74      4118
weighted avg       0.91      0.92      0.91      4118

Confusion Matrix is :
 [[3570   84]
 [ 264  200]]
```

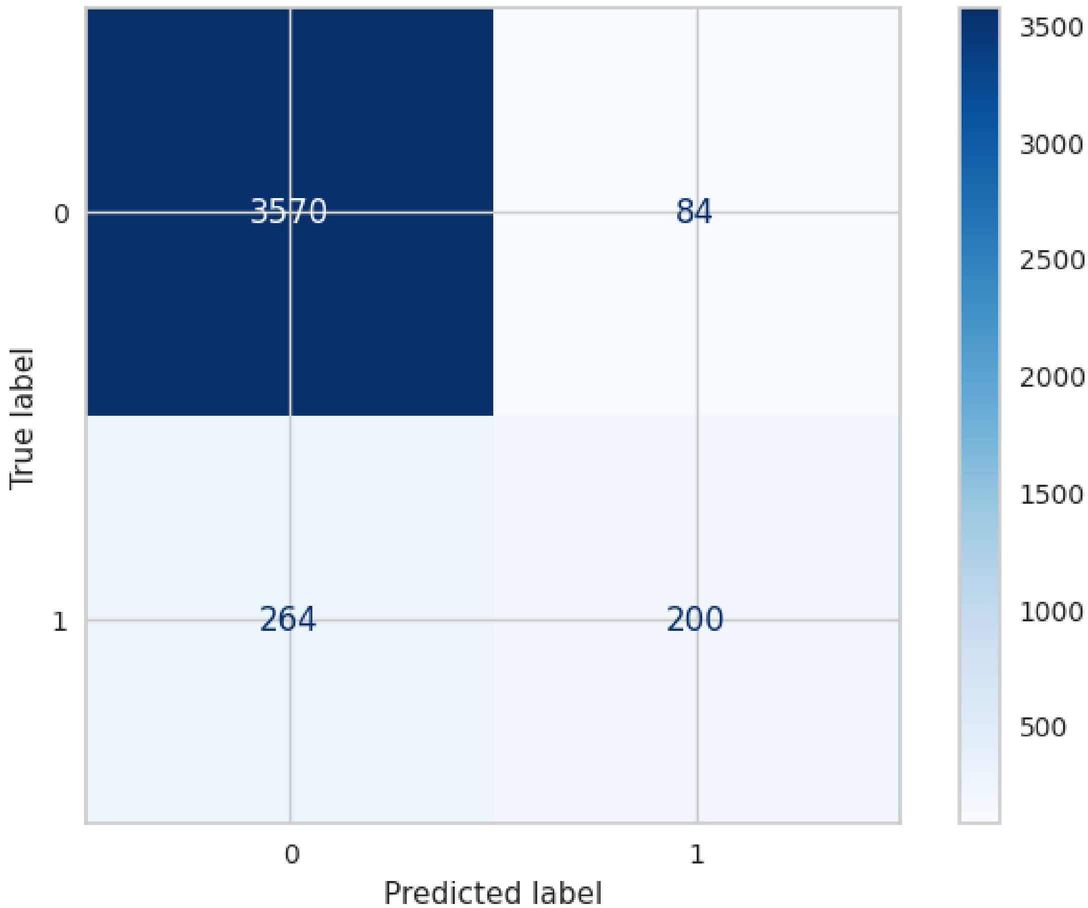
```
 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  0.9398477979274611
Model Test Score is :  0.9154929577464789
F1 Score is :  0.4985590778097983
Recall Score is :  0.3728448275862069
Precision Score is :  0.7521739130434782
AUC Value  :  0.6786227422003284

Classification Report is :                precision    recall  f1-score
support

           0       0.93      0.98      0.95      3654
           1       0.75      0.37      0.50       464

    accuracy                           0.92      4118
   macro avg       0.84      0.68      0.73      4118
weighted avg       0.91      0.92      0.90      4118

Confusion Matrix is :
 [[3597   57]
 [ 291  173]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  0.9365284974093264
Model Test Score is :  0.91452161243322
F1 Score is :  0.5404699738903394
Recall Score is :  0.44612068965517243
Precision Score is :  0.6854304635761589
AUC Value  :  0.7100608921729611

Classification Report is :                precision    recall  f1-score
support

           0       0.93      0.97      0.95      3654
           1       0.69      0.45      0.54       464

    accuracy                           0.91      4118
   macro avg       0.81      0.71      0.75      4118
weighted avg       0.90      0.91      0.91      4118

Confusion Matrix is :
 [[3559   95]
 [ 257  207]]
```

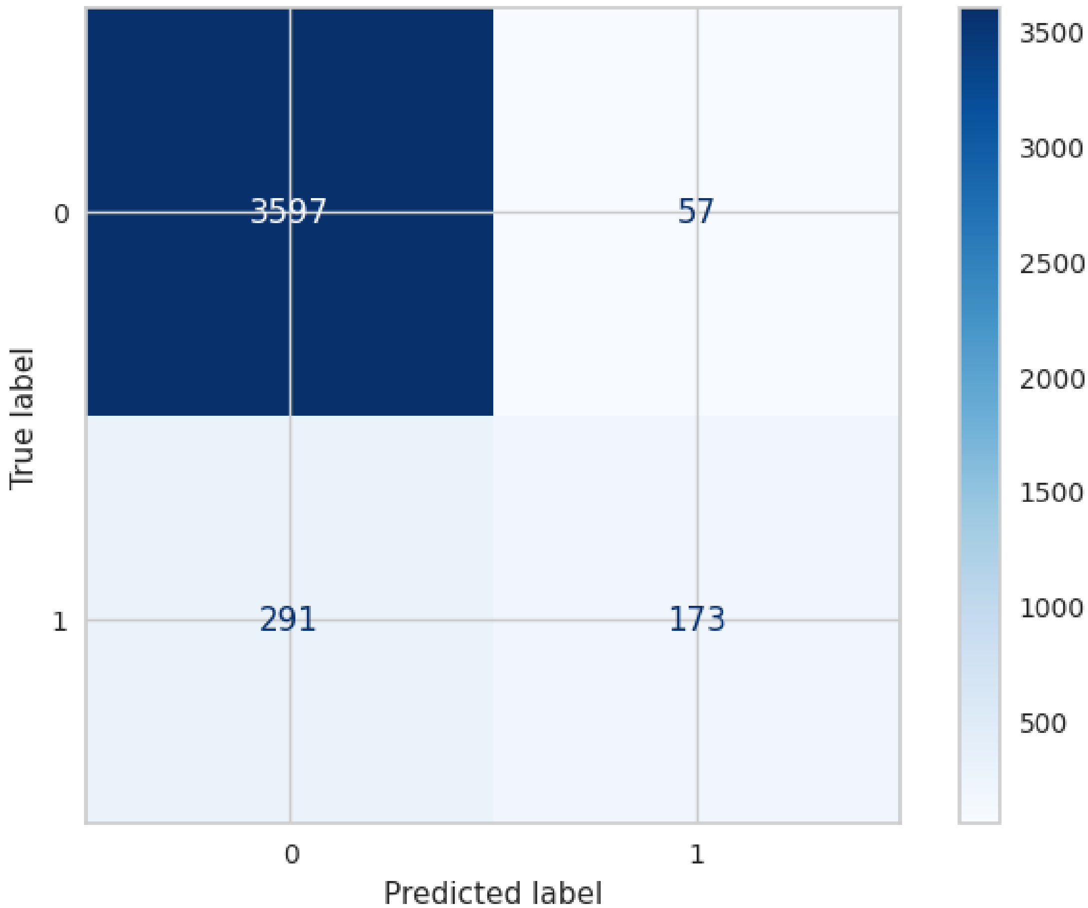

3574
80
267
197
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500

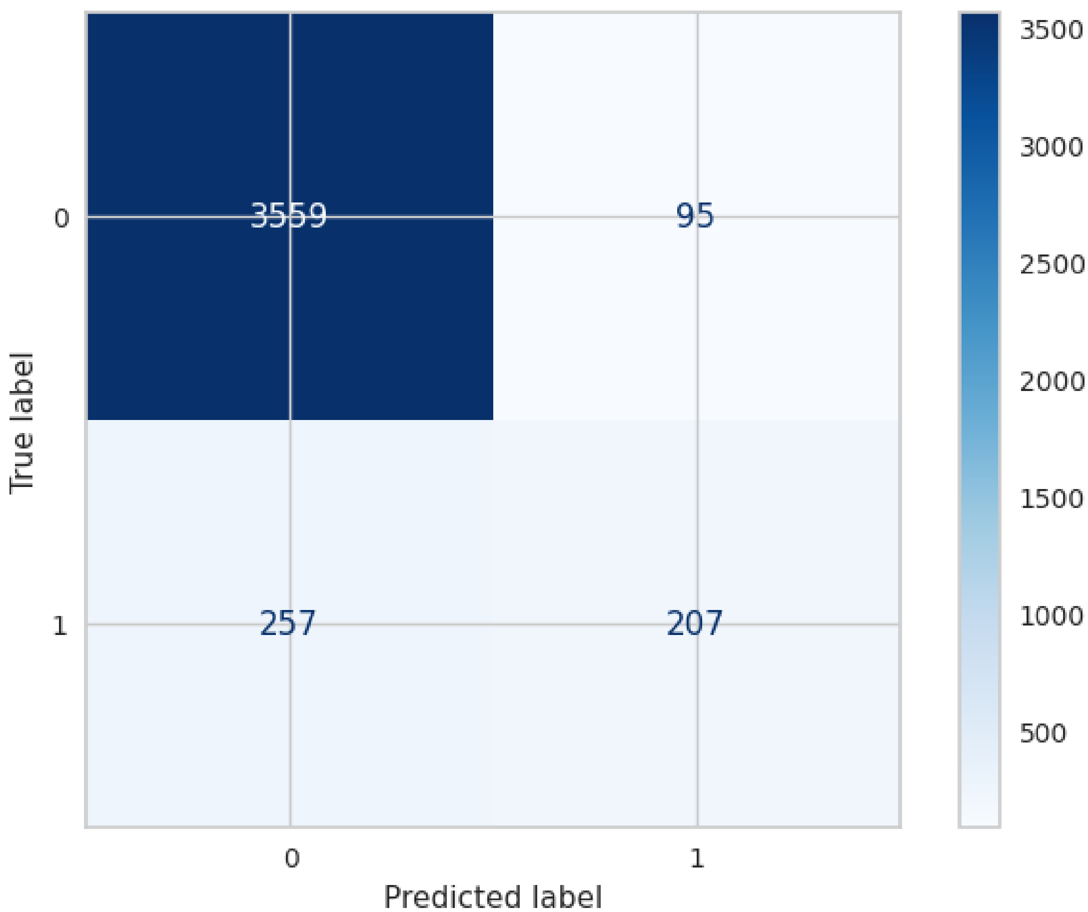

3510
144
219
245
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500


3575
79
263
201
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500


3545
109
242
222
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500


3570
84
264
200
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500


3597
57
291
173
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500



[211]:
```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['Forest','Forest With Feature','Forest Scaling','Forest With␣
↪Normalize','Forest With PCA','Forest With PCA and Scaling',
                 'Forest With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[211]:

Models	Train Accuracy	Test Accuracy	Test F1 \
Forest	0.935044	0.915736	0.531714
Forest With Feature	0.934505	0.911850	0.574443
Forest Scaling	0.934693	0.916950	0.540323
Forest With Normalize	0.942438	0.914764	0.558491
Forest With PCA	0.945893	0.915493	0.534759
Forest With PCA and Scaling	0.939848	0.915493	0.498559
Forest With PCA and Normalize	0.936528	0.914522	0.540470

```
                              Test Recall  Test Precision       AUC
Models
Forest                           0.424569        0.711191  0.701338
Forest With Feature              0.528017        0.629820  0.744304
Forest Scaling                   0.433190        0.717857  0.705785
Forest With Normalize            0.478448        0.670695  0.724309
Forest With PCA                  0.431034        0.704225  0.704023
Forest With PCA and Scaling      0.372845        0.752174  0.678623
Forest With PCA and Normalize    0.446121        0.685430  0.710061
```

[212]:
```
models_draw(df)
```

RandomOverSampler

[213]:
```
X_train,y_train,X_test,y_test=Split(X_classification_over,y_classification_over)
```

```
X_train shape is  (65763, 20)
X_test shape is  (7307, 20)
y_train shape is  (65763,)
y_test shape is  (7307,)
```

[214]:
```
Search(RandomForestClassifier(max_depth=20),{'max_depth':
↪[20,25,30,35,40]},X_train,y_train)
```

[214]:
```
RandomForestClassifier(max_depth=35)
```

[215]:
```
cross_validation(RandomForestClassifier(max_depth=40),X_train,y_train)
```

```
Train Score Value :  [0.99992397 0.99990496 0.99992397 0.99990496 0.99996199]
Mean 0.9999239693330242
Test Score Value :  [0.96259408 0.96441876 0.96228997 0.96449209 0.96190693]
Mean 0.9631403694826546
```

[216]:
```
Values =␣
↪Models(RandomForestClassifier(max_depth=40),X_train,y_train,X_test,y_test)
```

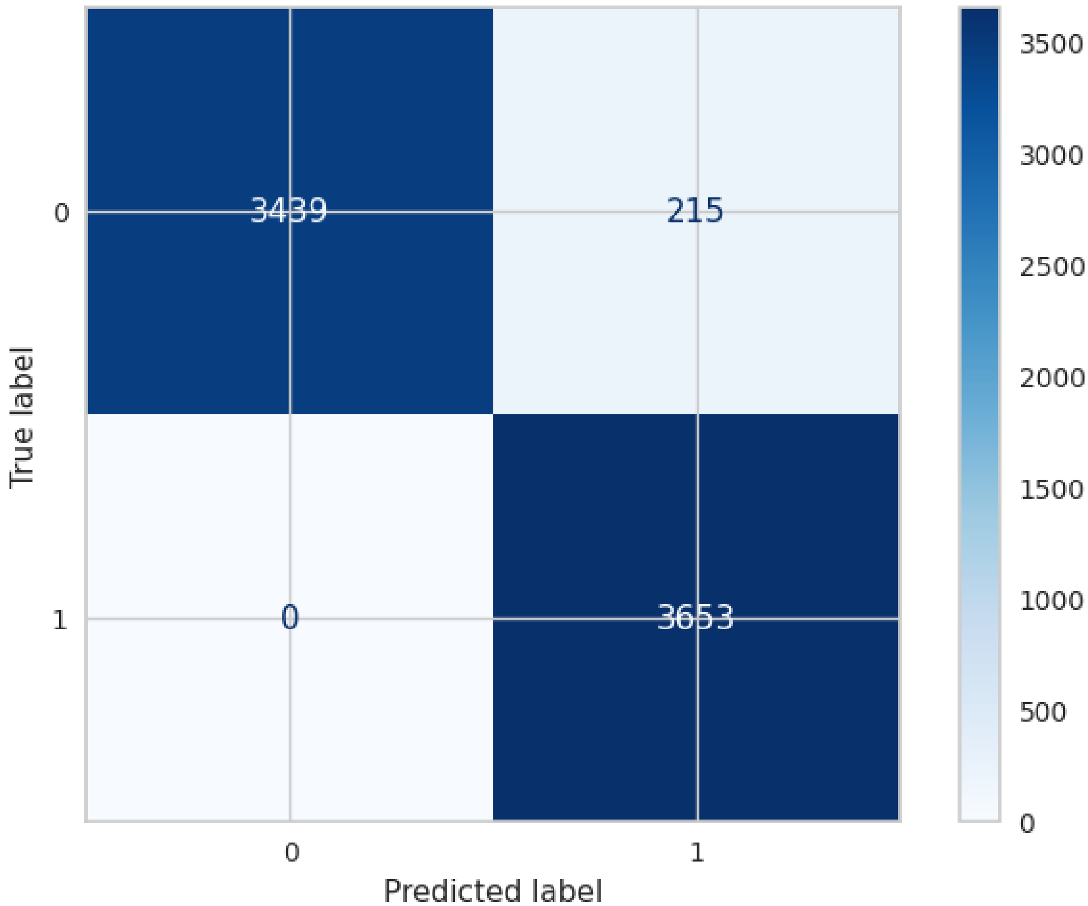
```
Apply Model With Normal Data :

Model Train Score is :  0.9999239694052887
Model Test Score is :  0.9705761598467223
F1 Score is :  0.971413375880867
Recall Score is :  1.0
Precision Score is :  0.9444157187176836
AUC Value  :  0.9705801860974275

Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.94      0.97      3654
           1       0.94      1.00      0.97      3653
```

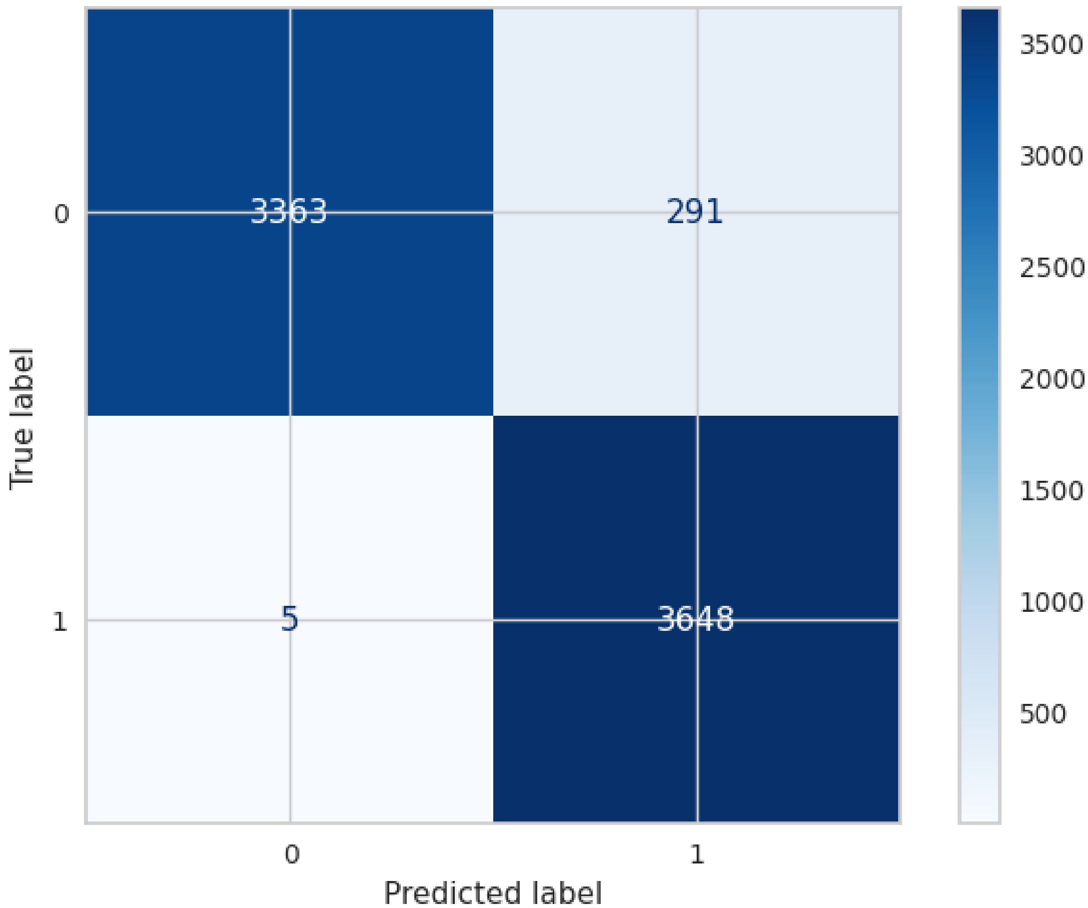
```
   accuracy                           0.97      7307
  macro avg       0.97      0.97      0.97      7307
weighted avg       0.97      0.97      0.97      7307

Confusion Matrix is :
 [[3439  215]
 [   0 3653]]


 Apply Model With Feature Selection :

Model Train Score is :  0.9883673190091693
Model Test Score is :  0.959490899137813
F1 Score is :  0.9610115911485775
Recall Score is :  0.9986312619764577
Precision Score is :  0.9261233815689262
AUC Value  :  0.9594962549619563
Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.92      0.96      3654
           1       0.93      1.00      0.96      3653

   accuracy                           0.96      7307
  macro avg       0.96      0.96      0.96      7307
weighted avg       0.96      0.96      0.96      7307

Confusion Matrix is :
 [[3363  291]
 [   5 3648]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.9999239694052887
Model Test Score is :  0.9722184206924867
F1 Score is :  0.9729657744040484
Recall Score is :  1.0
Precision Score is :  0.9473547717842323
AUC Value  :  0.9722222222222222
Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.94      0.97      3654
           1       0.95      1.00      0.97      3653
```

```
    accuracy                           0.97      7307
   macro avg       0.97      0.97      0.97      7307
weighted avg       0.97      0.97      0.97      7307

Confusion Matrix is :
 [[3451  203]
 [   0 3653]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.9999239694052887
Model Test Score is :  0.9700287395648008
F1 Score is :  0.9708970099667774
Recall Score is :  1.0
Precision Score is :  0.9434400826446281
AUC Value  :  0.970032840722496
Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.94      0.97      3654
           1       0.94      1.00      0.97      3653

    accuracy                           0.97      7307
   macro avg       0.97      0.97      0.97      7307
weighted avg       0.97      0.97      0.97      7307

Confusion Matrix is :
 [[3435  219]
 [   0 3653]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.9998783510484619
Model Test Score is :  0.9720815656220063
F1 Score is :  0.9728362183754994
Recall Score is :  1.0
Precision Score is :  0.9471091521908219
AUC Value  :  0.9720853858784892
Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.94      0.97      3654
           1       0.95      1.00      0.97      3653

    accuracy                           0.97      7307
```

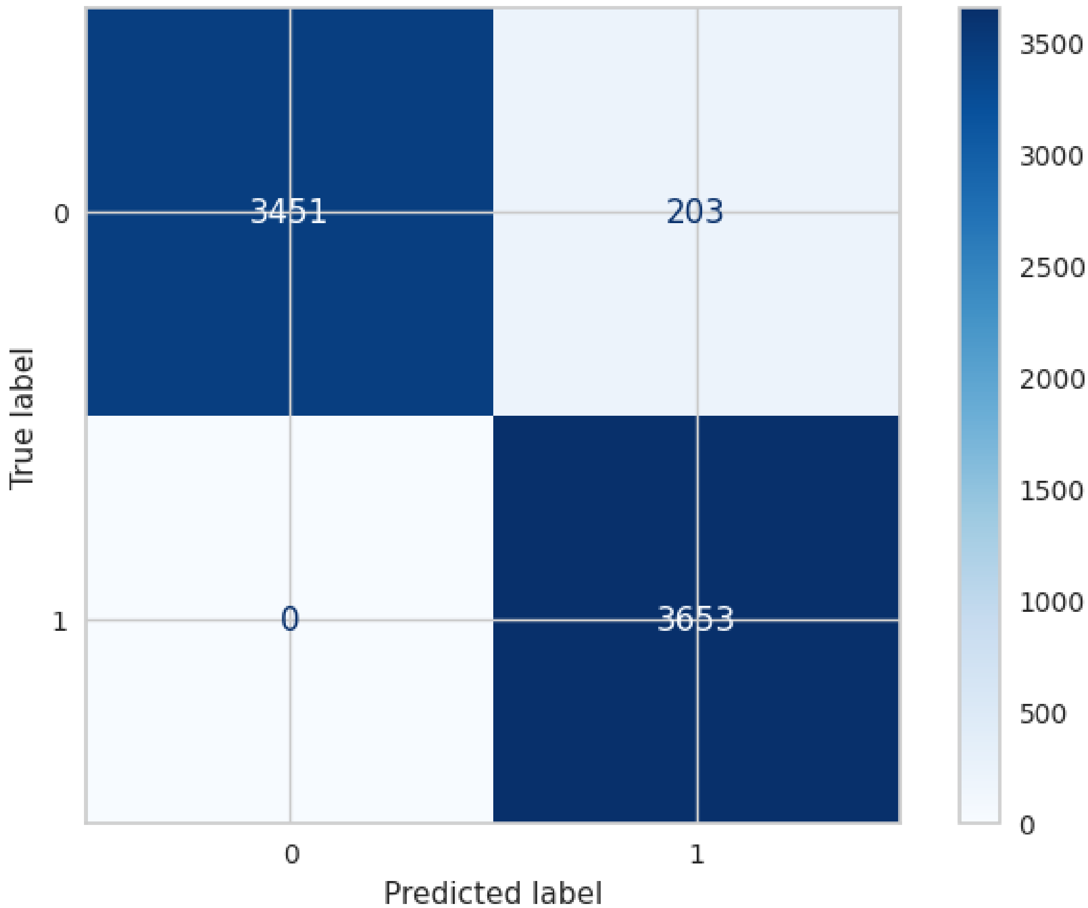
```
   macro avg       0.97      0.97      0.97      7307
weighted avg       0.97      0.97      0.97      7307

Confusion Matrix is :
 [[3450  204]
 [   0 3653]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  0.9999239694052887
Model Test Score is :  0.9746818119611331
F1 Score is :  0.9753036977706581
Recall Score is :  1.0
Precision Score is :  0.9517978113600833
AUC Value  :  0.9746852764094143

Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.95      0.97      3654
           1       0.95      1.00      0.98      3653

    accuracy                           0.97      7307
   macro avg       0.98      0.97      0.97      7307
weighted avg       0.98      0.97      0.97      7307

Confusion Matrix is :
 [[3469  185]
 [   0 3653]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  0.9999087632863465
Model Test Score is :  0.9716710004105652
F1 Score is :  0.9724477572208173
Recall Score is :  1.0
Precision Score is :  0.9463730569948187
AUC Value  :  0.9716748768472907

Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.94      0.97      3654
           1       0.95      1.00      0.97      3653

    accuracy                           0.97      7307
   macro avg       0.97      0.97      0.97      7307
```

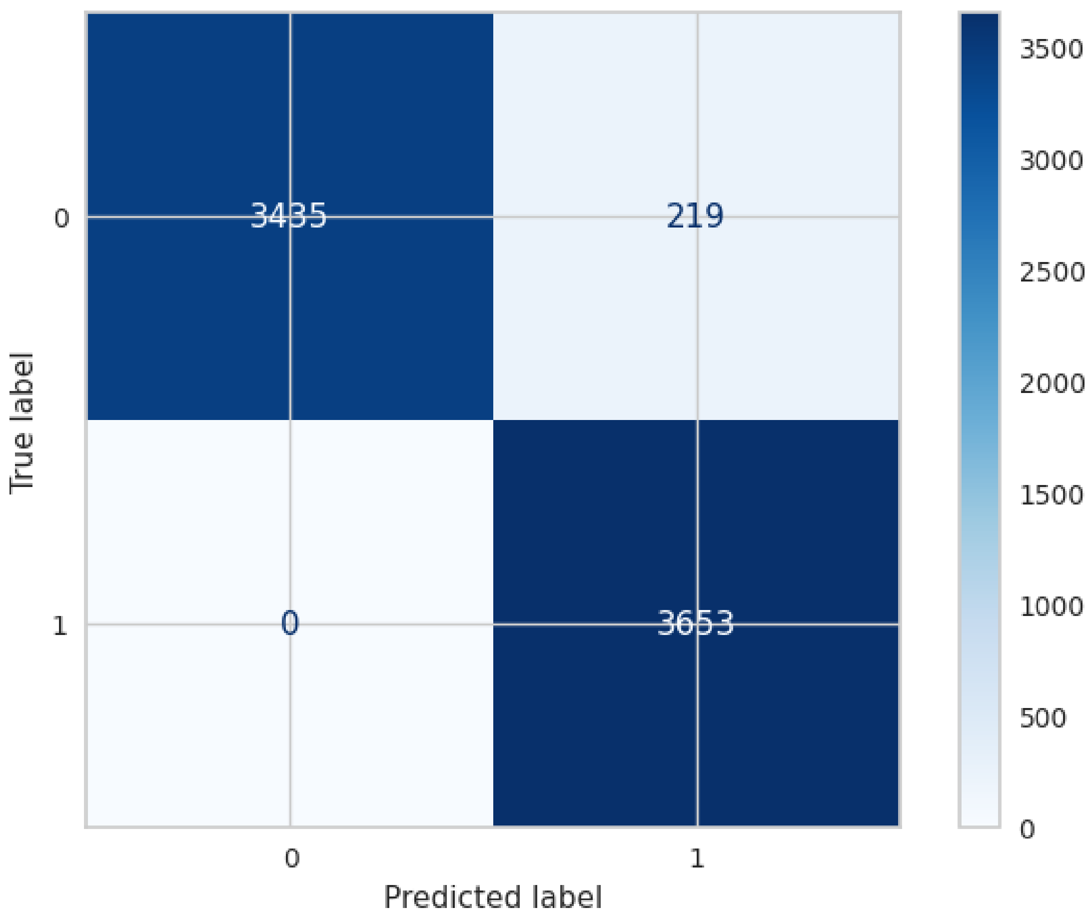
```
weighted avg       0.97      0.97      0.97      7307

Confusion Matrix is :
 [[3447  207]
 [   0 3653]]
```

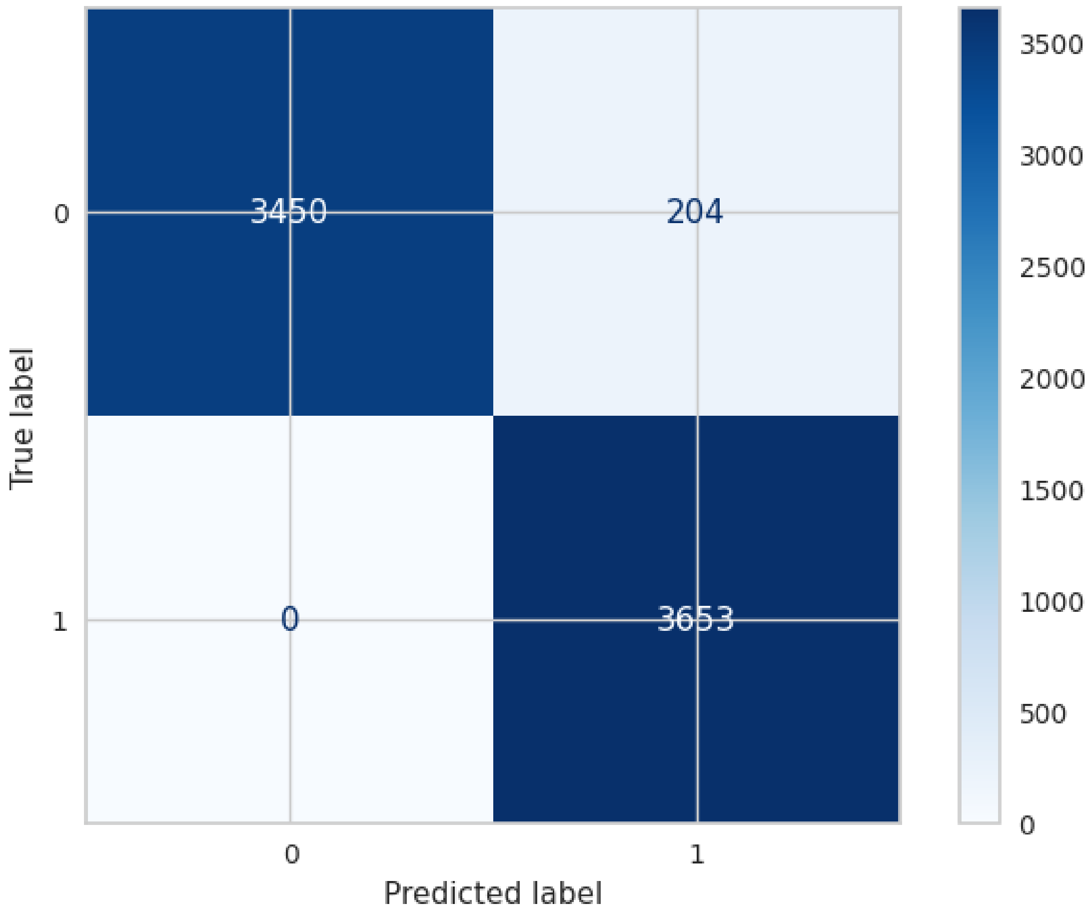


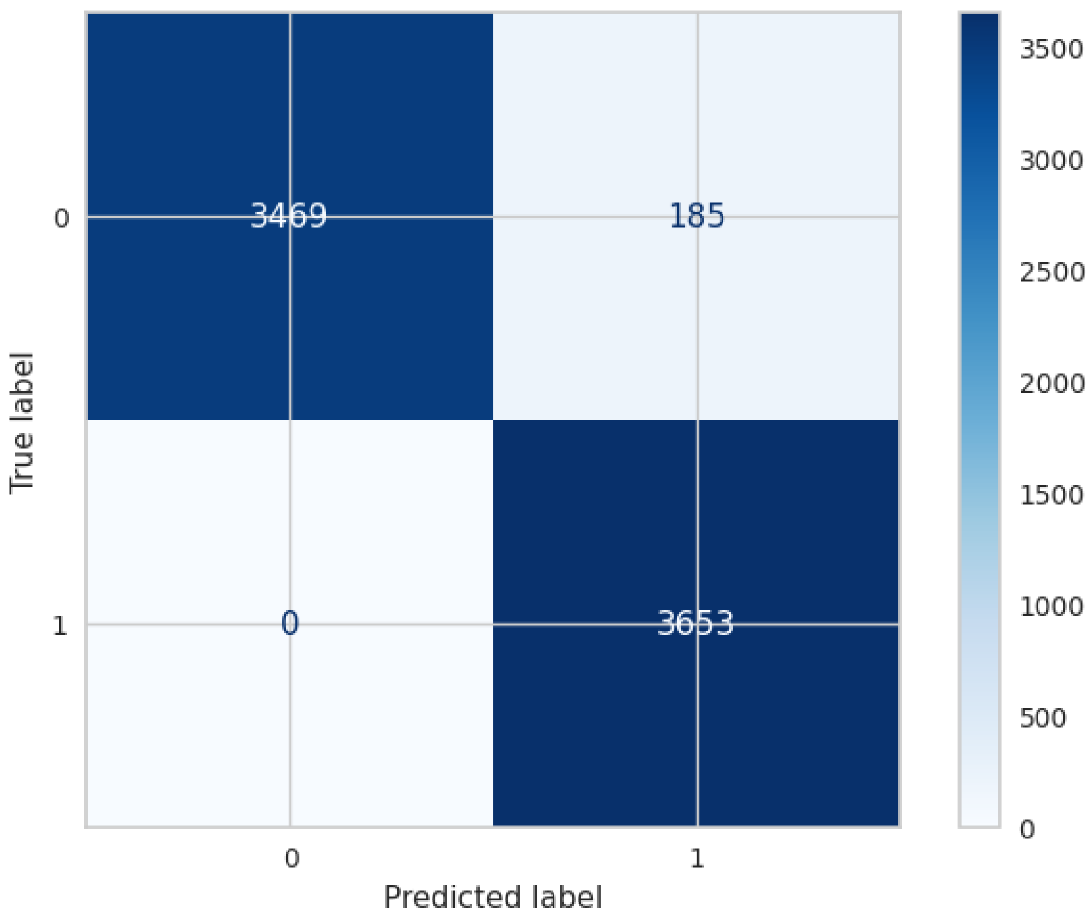


3363
291
5
3648
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500

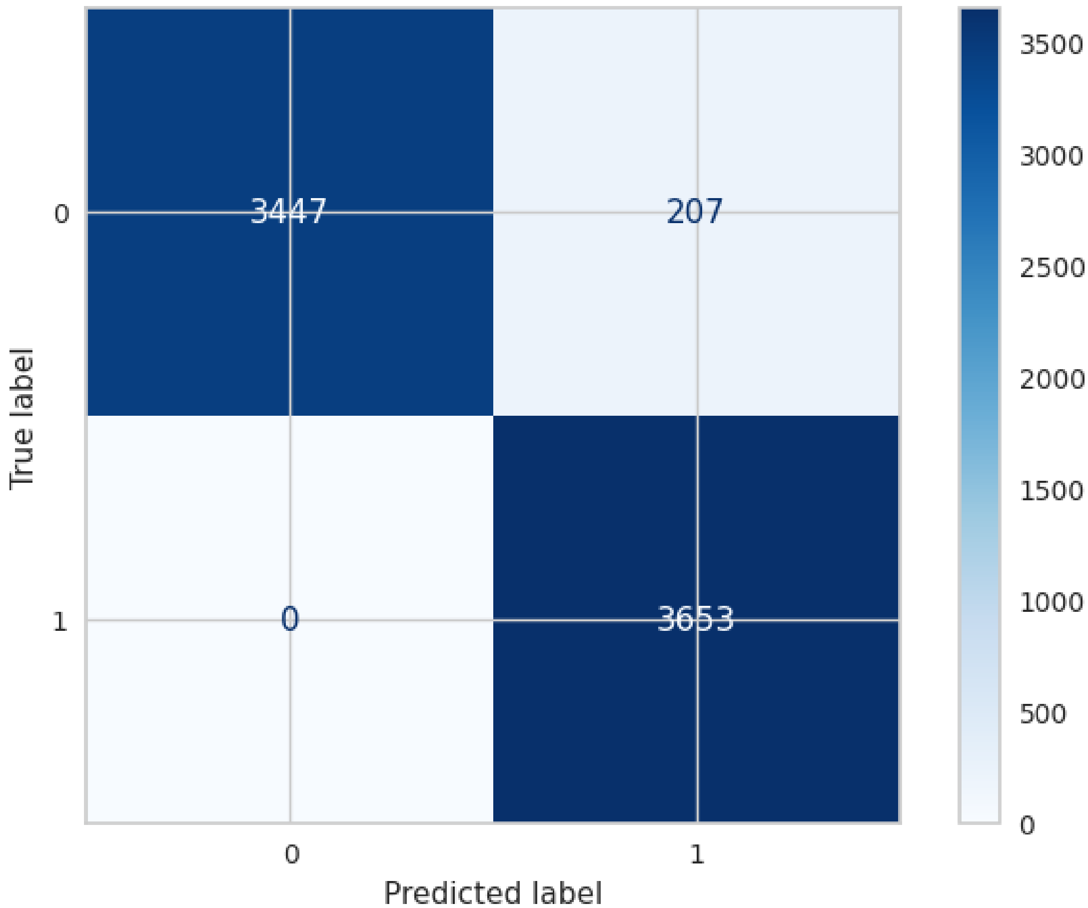

3451
203
0
3653
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500
0


3435
219
0
3653
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500
0


0
1
True label
3450
204
0
3653
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500
0



3469
185
0
3653
True label
Predicted label
0
1
0
1
3500
3000
2500
2000
1500
1000
500
0



[217]:
```python
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['Forest Over','Forest Over With Feature','Forest Over␣
↪Scaling','Foresr Over With Normalize','Forest Over With PCA'
                ,'Forest Over With PCA and Scaling',
                'Forest Over With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[217]:

```
                                    Train Accuracy  Test Accuracy   Test F1  \
Models
Forest Over                               0.999924       0.970576  0.971413
Forest Over With Feature                  0.988367       0.959491  0.961012
Forest Over Scaling                       0.999924       0.972218  0.972966
Foresr Over With Normalize                0.999924       0.970029  0.970897
Forest Over With PCA                      0.999878       0.972082  0.972836
Forest Over With PCA and Scaling          0.999924       0.974682  0.975304
Forest Over With PCA and Normalize        0.999909       0.971671  0.972448
```

```
                                   Test Recall  Test Precision       AUC
Models
Forest Over                           1.000000        0.944416  0.970580
Forest Over With Feature              0.998631        0.926123  0.959496
Forest Over Scaling                   1.000000        0.947355  0.972222
Foresr Over With Normalize            1.000000        0.943440  0.970033
Forest Over With PCA                  1.000000        0.947109  0.972085
Forest Over With PCA and Scaling      1.000000        0.951798  0.974685
Forest Over With PCA and Normalize    1.000000        0.946373  0.971675
```

[218]:
```
models_draw(df)
```

RandomUnderSampler

[219]:
```
X_train,y_train,X_test,y_test=Split(X_classification_under,y_classification_under)
```

```
X_train shape is  (8350, 20)
X_test shape is  (928, 20)
y_train shape is  (8350,)
y_test shape is  (928,)
```

[220]:
```
Search(RandomForestClassifier(max_depth=20),{'max_depth':
 ↪[20,25,30,35,40]},X_train,y_train)
```

[220]: RandomForestClassifier(max_depth=20)

[221]:
```
cross_validation(RandomForestClassifier(max_depth=35),X_train,y_train)
```

```
Train Score Value :  [1.         1.         1.         1.         0.9998503]
Mean 0.9999700598802395
Test Score Value :  [0.8988024  0.88502994 0.88443114 0.88323353 0.8994012 ]
Mean 0.8901796407185628
```

[222]:
```
Values =␣
 ↪Models(RandomForestClassifier(max_depth=35),X_train,y_train,X_test,y_test)
```

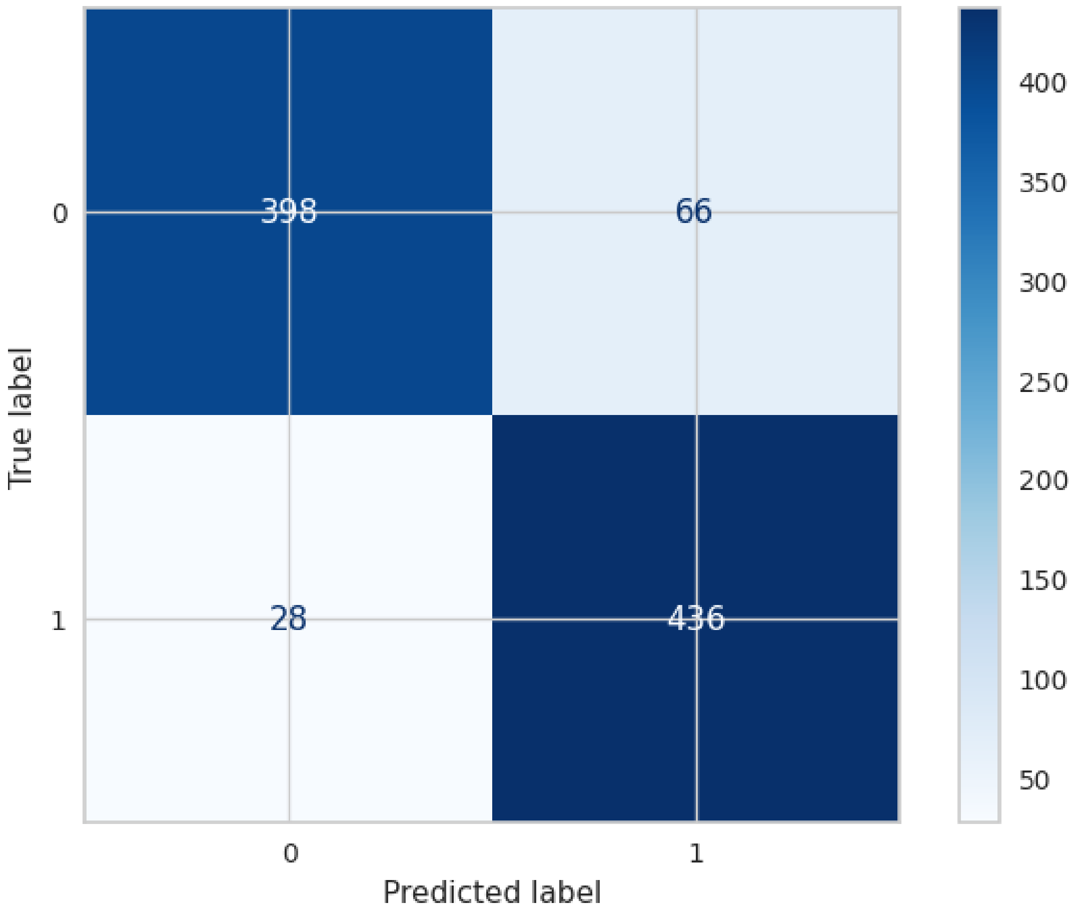
```
Apply Model With Normal Data :

Model Train Score is :  1.0
Model Test Score is :  0.8987068965517241
F1 Score is :  0.9026915113871635
Recall Score is :  0.9396551724137931
Precision Score is :  0.8685258964143426
AUC Value  :  0.8987068965517242

Classification Report is :                precision    recall  f1-score
support

           0       0.93      0.86      0.89       464
```

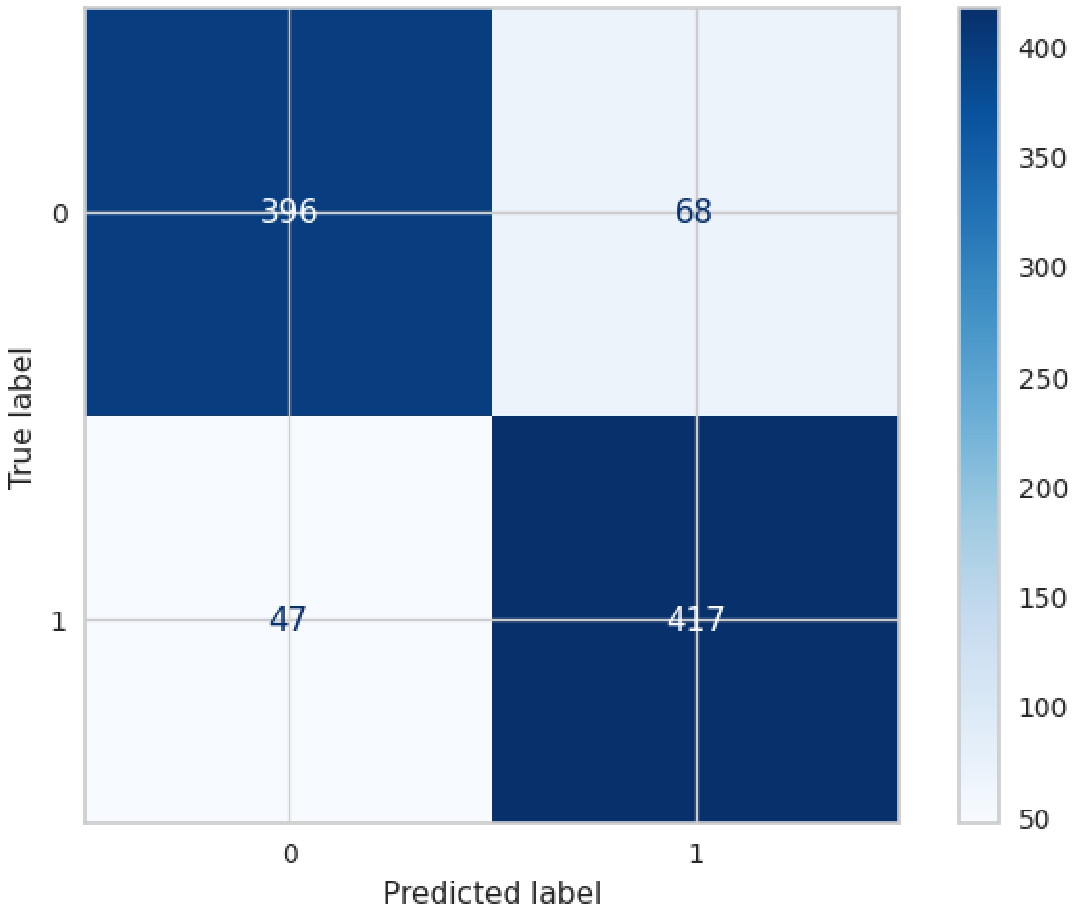
```
           1       0.87      0.94      0.90       464

    accuracy                           0.90       928
   macro avg       0.90      0.90      0.90       928
weighted avg       0.90      0.90      0.90       928

Confusion Matrix is :
 [[398  66]
 [ 28 436]]


 Apply Model With Feature Selection :

Model Train Score is :  0.9901796407185629
Model Test Score is :  0.8760775862068966
F1 Score is :  0.8788198103266596
Recall Score is :  0.8987068965517241
Precision Score is :  0.8597938144329897
AUC Value  :  0.8760775862068966
Classification Report is :                precision    recall  f1-score
support

           0       0.89      0.85      0.87       464
           1       0.86      0.90      0.88       464

    accuracy                           0.88       928
   macro avg       0.88      0.88      0.88       928
weighted avg       0.88      0.88      0.88       928

Confusion Matrix is :
 [[396  68]
 [ 47 417]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  1.0
Model Test Score is :  0.8943965517241379
F1 Score is :  0.898550724637681
Recall Score is :  0.9353448275862069
Precision Score is :  0.8645418326693227
AUC Value  :  0.8943965517241379
Classification Report is :                precision    recall  f1-score
support

           0       0.93      0.85      0.89       464
           1       0.86      0.94      0.90       464
```

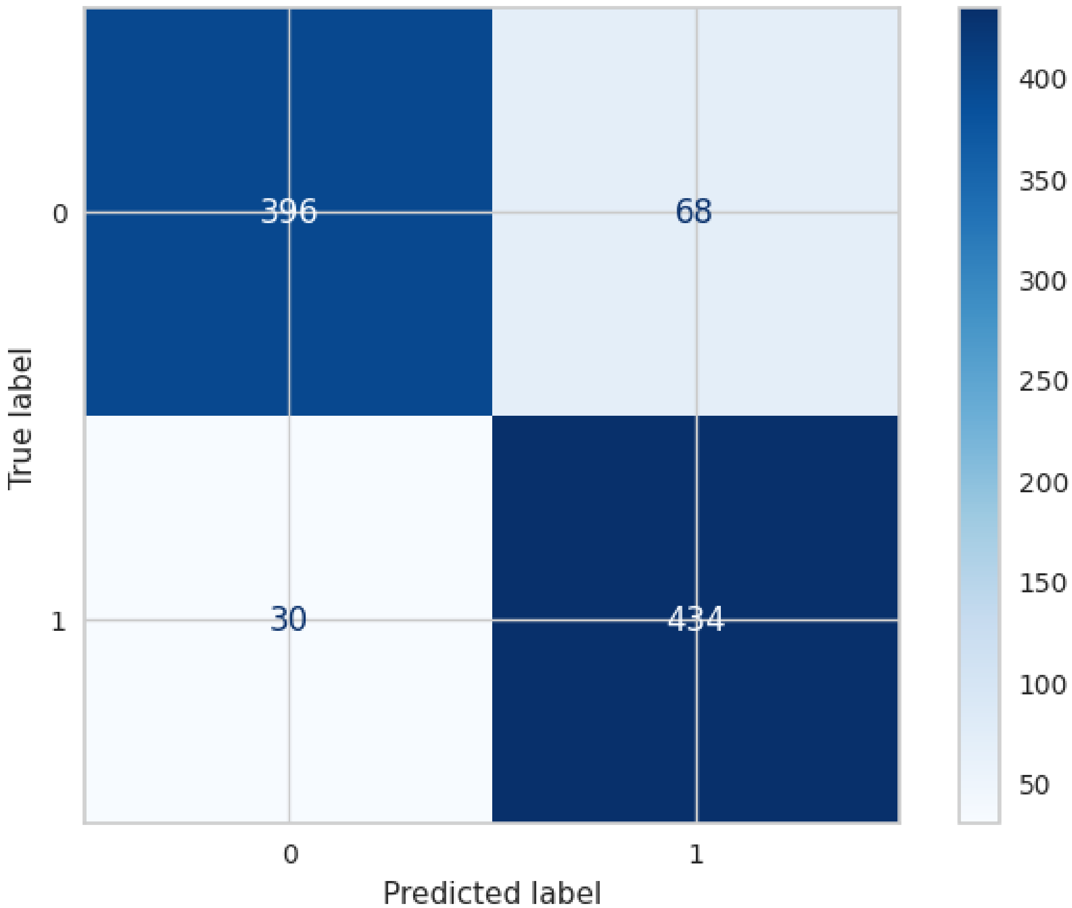
```
    accuracy                           0.89       928
   macro avg       0.90      0.89      0.89       928
weighted avg       0.90      0.89      0.89       928

Confusion Matrix is :
 [[396  68]
 [ 30 434]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  1.0
Model Test Score is :  0.8857758620689655
F1 Score is :  0.8898128898128898
Recall Score is :  0.9224137931034483
Precision Score is :  0.8594377510040161
AUC Value  :  0.8857758620689656
Classification Report is :                precision    recall  f1-score
support

           0       0.92      0.85      0.88       464
           1       0.86      0.92      0.89       464

    accuracy                           0.89       928
   macro avg       0.89      0.89      0.89       928
weighted avg       0.89      0.89      0.89       928

Confusion Matrix is :
 [[394  70]
 [ 36 428]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  1.0
Model Test Score is :  0.8997844827586207
F1 Score is :  0.9036269430051814
Recall Score is :  0.9396551724137931
Precision Score is :  0.8702594810379242
AUC Value  :  0.8997844827586208
Classification Report is :                precision    recall  f1-score
support

           0       0.93      0.86      0.90       464
           1       0.87      0.94      0.90       464
```

```
   accuracy                           0.90       928
  macro avg       0.90      0.90      0.90       928
weighted avg       0.90      0.90      0.90       928

Confusion Matrix is :
 [[399  65]
 [ 28 436]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  1.0
Model Test Score is :  0.884698275862069
F1 Score is :  0.8898043254376932
Recall Score is :  0.9310344827586207
Precision Score is :  0.8520710059171598
AUC Value  :  0.884698275862069

Classification Report is :                precision    recall  f1-score
support

           0       0.92      0.84      0.88       464
           1       0.85      0.93      0.89       464

   accuracy                           0.88       928
  macro avg       0.89      0.88      0.88       928
weighted avg       0.89      0.88      0.88       928

Confusion Matrix is :
 [[389  75]
 [ 32 432]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  1.0
Model Test Score is :  0.8911637931034483
F1 Score is :  0.8944618599791014
Recall Score is :  0.9224137931034483
Precision Score is :  0.8681541582150102
AUC Value  :  0.8911637931034484

Classification Report is :                precision    recall  f1-score
support

           0       0.92      0.86      0.89       464
           1       0.87      0.92      0.89       464

   accuracy                           0.89       928
```

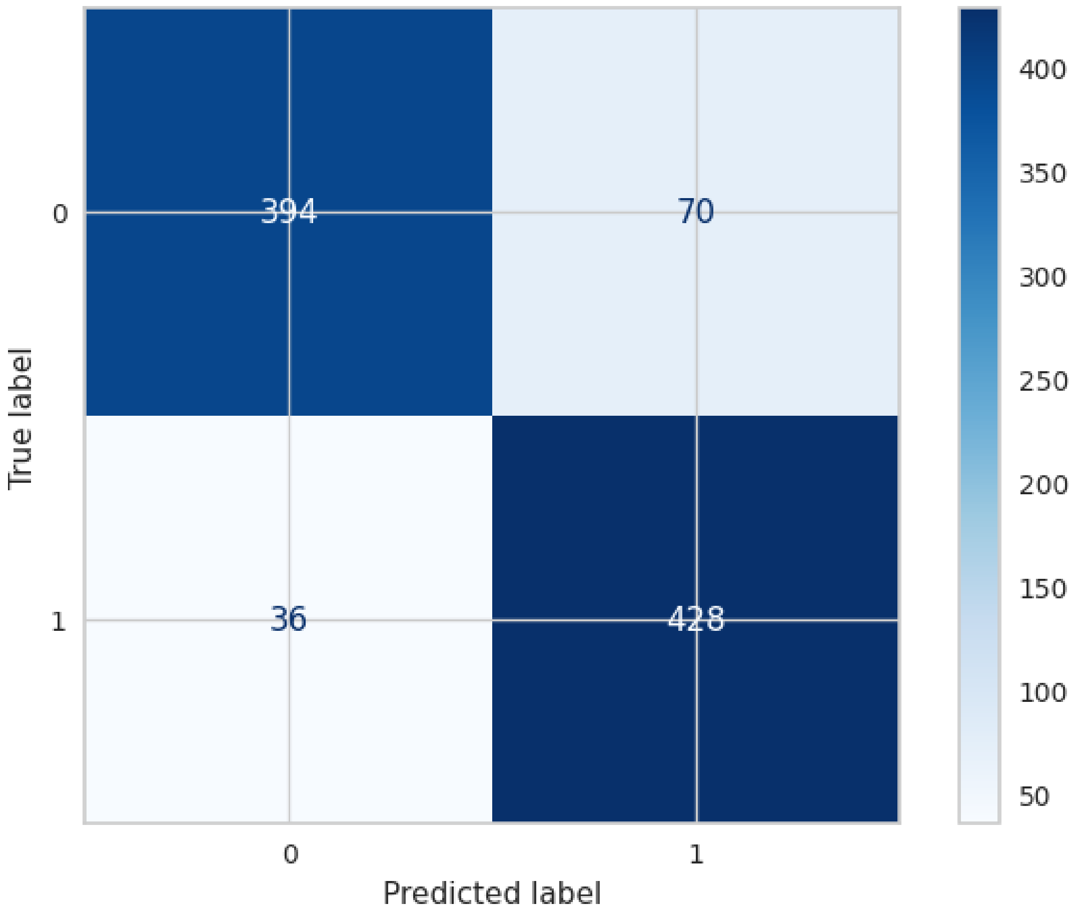
```
   macro avg       0.89      0.89      0.89       928
weighted avg       0.89      0.89      0.89       928

Confusion Matrix is :
 [[399  65]
 [ 36 428]]
```

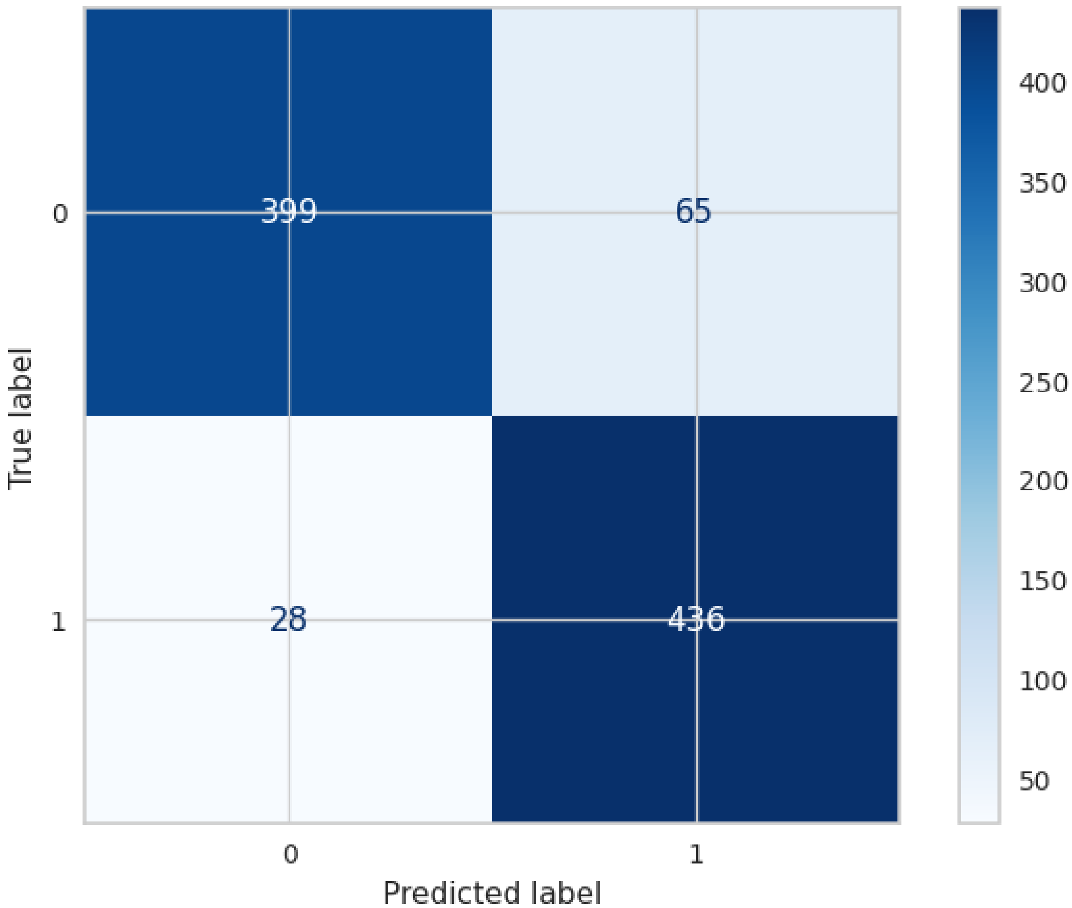


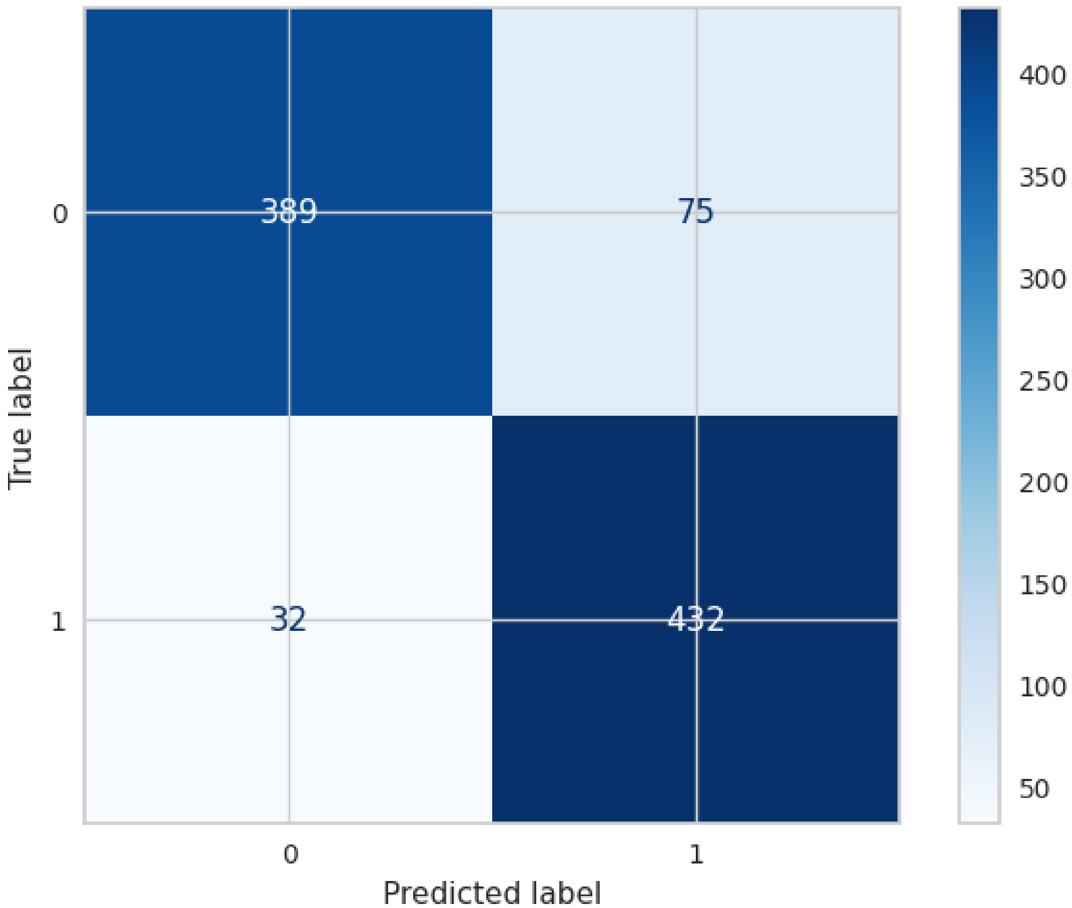

396
68
47
417
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50



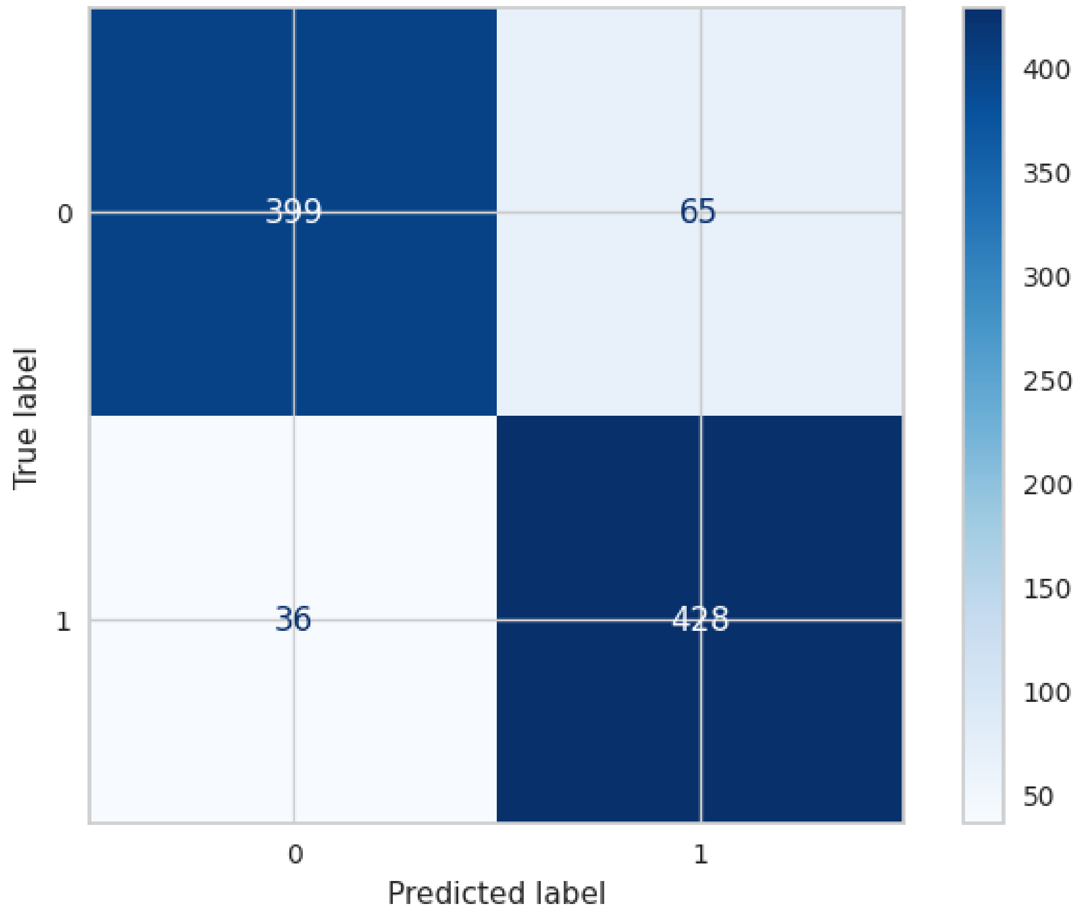
396
68
30
434
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50


394
70
36
428
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50


399
65
28
436
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50



389
75
32
432
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50



[223]:

```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['Forest Under','Forest Under With Feature','Forest Under␣
↪Scaling','Foresr Under With Normalize','Forest Under With PCA'
               ,'Forest Under With PCA and Scaling',
               'Forest Under With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[223]:

```
                                    Train Accuracy  Test Accuracy   Test F1  \
Models
Forest Under                               1.00000       0.898707  0.902692
Forest Under With Feature                  0.99018       0.876078  0.878820
Forest Under Scaling                       1.00000       0.894397  0.898551
Foresr Under With Normalize                1.00000       0.885776  0.889813
Forest Under With PCA                      1.00000       0.899784  0.903627
Forest Under With PCA and Scaling          1.00000       0.884698  0.889804
Forest Under With PCA and Normalize        1.00000       0.891164  0.894462
```

```
                                      Test Recall  Test Precision       AUC
Models
Forest Under                              0.939655        0.868526  0.898707
Forest Under With Feature                 0.898707        0.859794  0.876078
Forest Under Scaling                      0.935345        0.864542  0.894397
Foresr Under With Normalize               0.922414        0.859438  0.885776
Forest Under With PCA                     0.939655        0.870259  0.899784
Forest Under With PCA and Scaling         0.931034        0.852071  0.884698
Forest Under With PCA and Normalize       0.922414        0.868154  0.891164
```

[224]:
```
models_draw(df)
```

```
DecisionTreeClassifier
```

[225]:
```
X_train,y_train,X_test,y_test=Split(X_classification,y_classification)
```

```
X_train shape is  (37056, 20)
X_test shape is  (4118, 20)
y_train shape is  (37056,)
y_test shape is  (4118,)
```

[226]:
```
Search(DecisionTreeClassifier(max_depth=20),{'max_depth':
 ↪[5,10,15,20,25,30,35,40]},X_train,y_train)
```

[226]:
```
DecisionTreeClassifier(max_depth=5)
```

[227]:
```
cross_validation(DecisionTreeClassifier(max_depth=5),X_train,y_train)
```

```
Train Score Value :  [0.91387802 0.91506156 0.9149941  0.91361106 0.91121606]
Mean 0.9137521597437622
Test Score Value :  [0.90879655 0.91256241 0.90500607 0.91188773 0.91593577]
Mean 0.9108377062057444
```

[228]:
```
Values =␣
 ↪Models(DecisionTreeClassifier(max_depth=5),X_train,y_train,X_test,y_test)
```

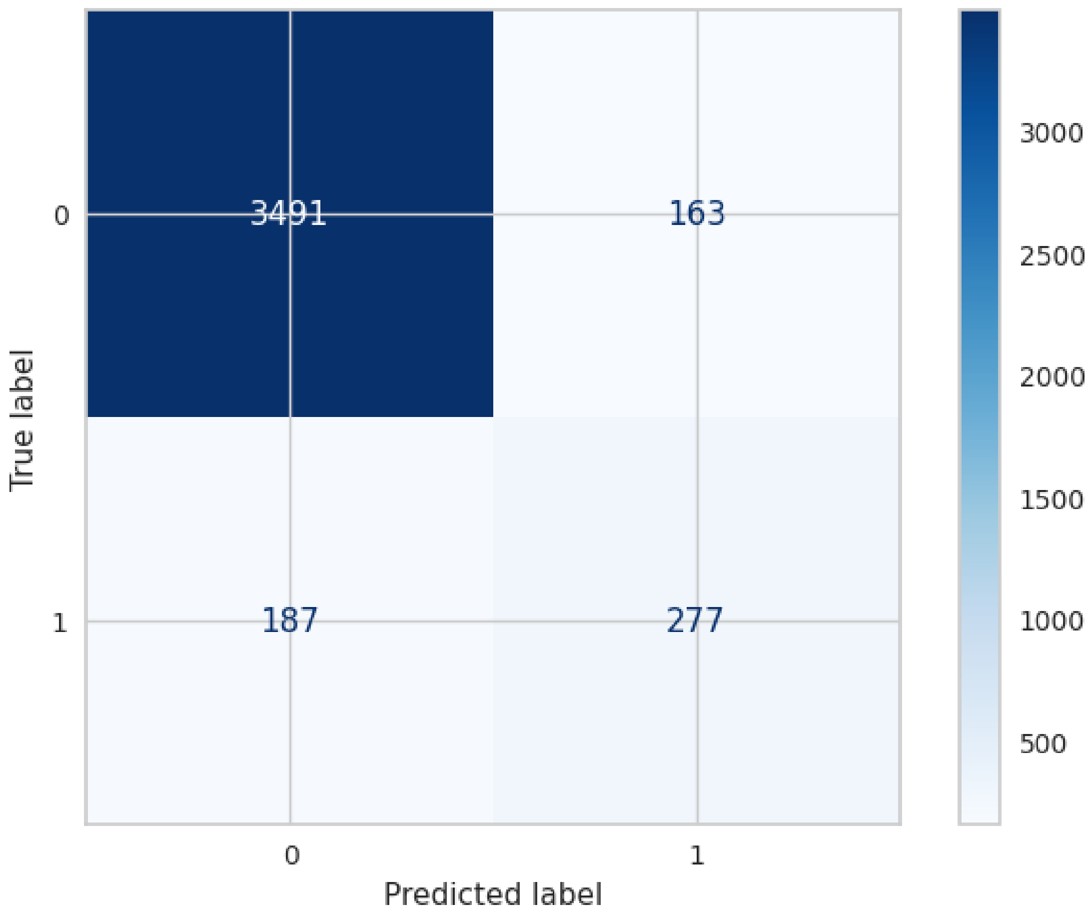
```
Apply Model With Normal Data :

Model Train Score is :  0.9132933937823834
Model Test Score is :  0.9150072850898494
F1 Score is :  0.6128318584070797
Recall Score is :  0.5969827586206896
Precision Score is :  0.6295454545454545
AUC Value  :  0.7761870552818828

Classification Report is :                precision    recall  f1-score
support

           0       0.95      0.96      0.95      3654
```

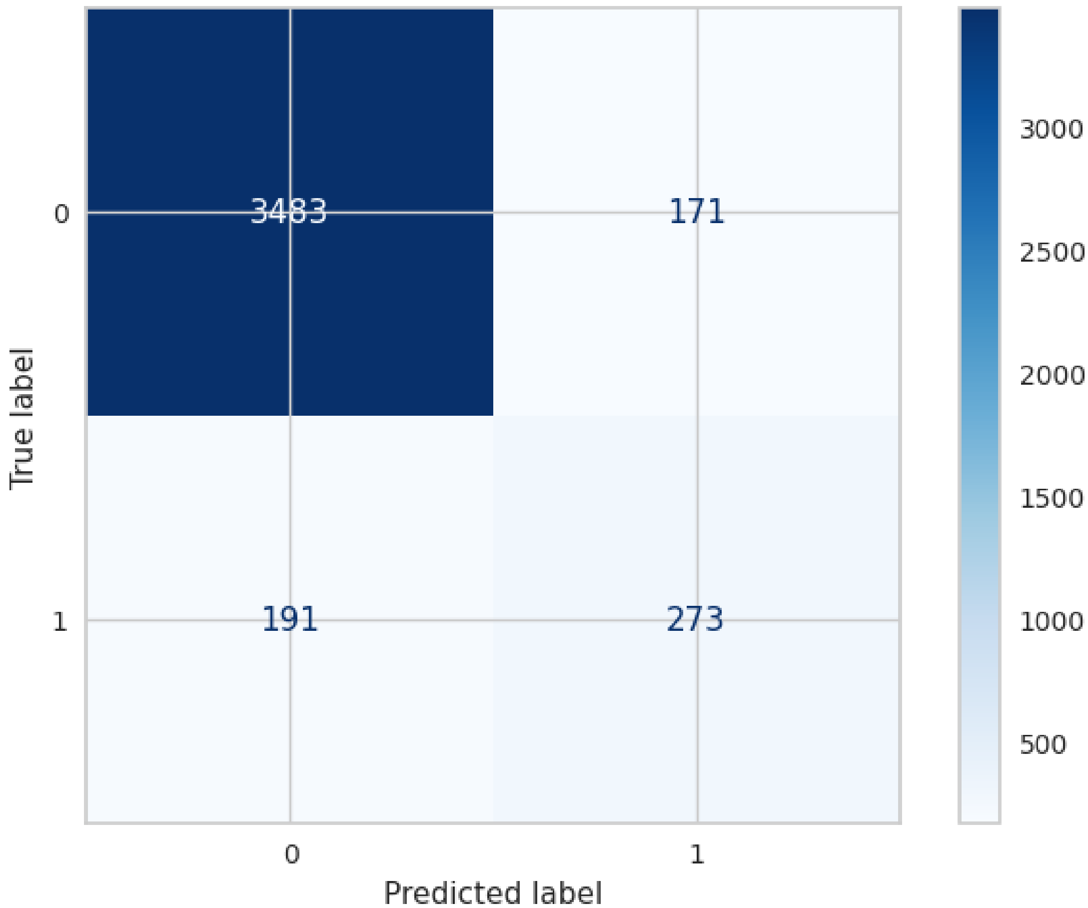
```
          1       0.63      0.60      0.61       464

    accuracy                          0.92      4118
   macro avg       0.79      0.78      0.78      4118
weighted avg       0.91      0.92      0.91      4118

Confusion Matrix is :
 [[3491  163]
 [ 187  277]]


 Apply Model With Feature Selection :

Model Train Score is :  0.9093534110535406
Model Test Score is :  0.9120932491500728
F1 Score is :  0.6013215859030837
Recall Score is :  0.5883620689655172
Precision Score is :  0.6148648648648649
AUC Value  :  0.7707820197044335
Classification Report is :                precision    recall  f1-score
support

           0       0.95      0.95      0.95      3654
           1       0.61      0.59      0.60       464

    accuracy                          0.91      4118
   macro avg       0.78      0.77      0.78      4118
weighted avg       0.91      0.91      0.91      4118

Confusion Matrix is :
 [[3483  171]
 [ 191  273]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.9132933937823834
Model Test Score is :  0.9150072850898494
F1 Score is :  0.6128318584070797
Recall Score is :  0.5969827586206896
Precision Score is :  0.6295454545454545
AUC Value  :  0.7761870552818828
Classification Report is :                precision    recall  f1-score
support

           0       0.95      0.96      0.95      3654
           1       0.63      0.60      0.61       464
```

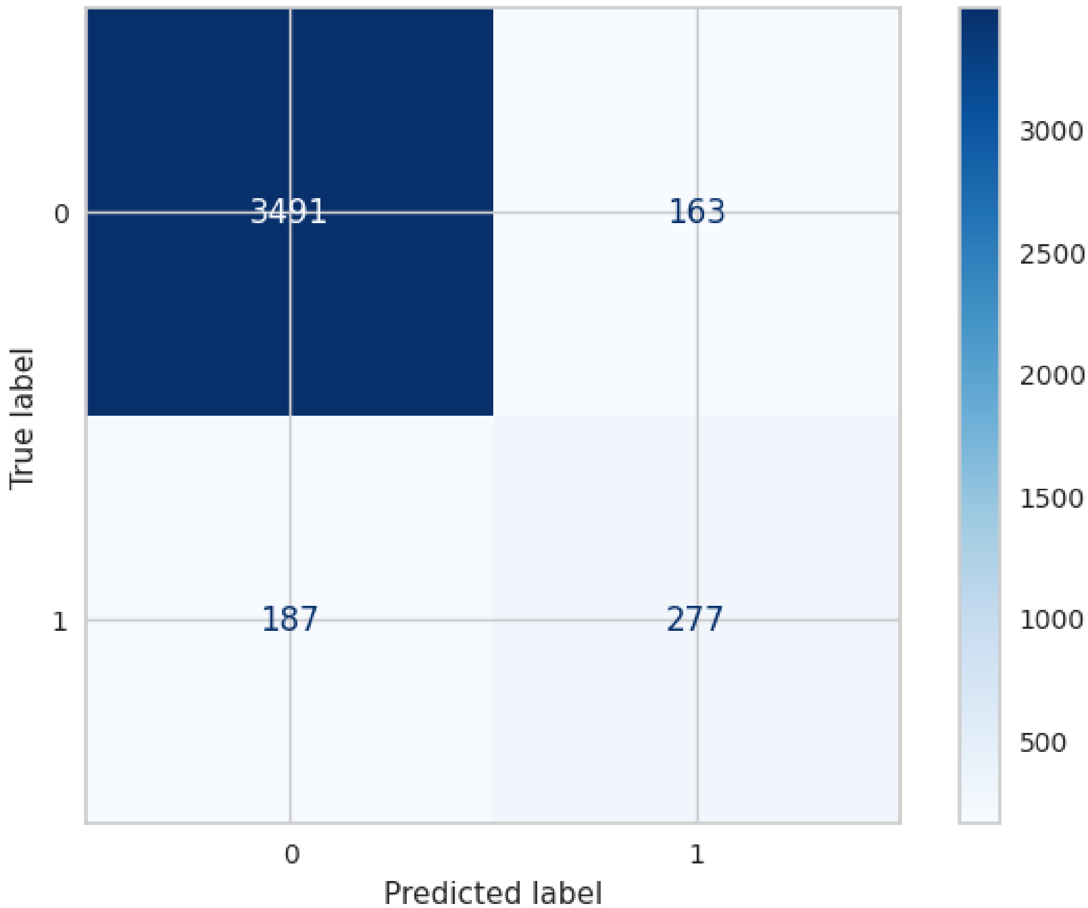
```
   accuracy                           0.92      4118
  macro avg       0.79      0.78      0.78      4118
weighted avg       0.91      0.92      0.91      4118

Confusion Matrix is :
 [[3491  163]
 [ 187  277]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.9134283246977547
Model Test Score is :  0.912821758135017
F1 Score is :  0.5282522996057819
Recall Score is :  0.4331896551724138
Precision Score is :  0.6767676767676768
AUC Value  :  0.7034585385878489

Classification Report is :                precision    recall  f1-score
support

           0       0.93      0.97      0.95      3654
           1       0.68      0.43      0.53       464

    accuracy                           0.91      4118
   macro avg       0.80      0.70      0.74      4118
weighted avg       0.90      0.91      0.90      4118

Confusion Matrix is :
 [[3558   96]
 [ 263  201]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.9149395509499136
Model Test Score is :  0.9133074307916464
F1 Score is :  0.5993265993265994
Recall Score is :  0.5754310344827587
Precision Score is :  0.6252927400468384
AUC Value  :  0.765821702244116

Classification Report is :                precision    recall  f1-score
support

           0       0.95      0.96      0.95      3654
           1       0.63      0.58      0.60       464
```

```
   accuracy                           0.91      4118
  macro avg       0.79      0.77      0.78      4118
weighted avg       0.91      0.91      0.91      4118

Confusion Matrix is :
 [[3494  160]
 [ 197  267]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  0.9082199913644214
Model Test Score is :  0.9084507042253521
F1 Score is :  0.4800000000000001
Recall Score is :  0.375
Precision Score is :  0.6666666666666666
AUC Value  :  0.675595238095238

Classification Report is :                precision    recall  f1-score
support

           0       0.92      0.98      0.95      3654
           1       0.67      0.38      0.48       464

    accuracy                           0.91      4118
   macro avg       0.80      0.68      0.71      4118
weighted avg       0.90      0.91      0.90      4118

Confusion Matrix is :
 [[3567   87]
 [ 290  174]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  0.9111614853195165
Model Test Score is :  0.9113647401651287
F1 Score is :  0.5326504481434058
Recall Score is :  0.4482758620689655
Precision Score is :  0.6561514195583596
AUC Value  :  0.7092227695675971

Classification Report is :                precision    recall  f1-score
support

           0       0.93      0.97      0.95      3654
           1       0.66      0.45      0.53       464

    accuracy                           0.91      4118
```

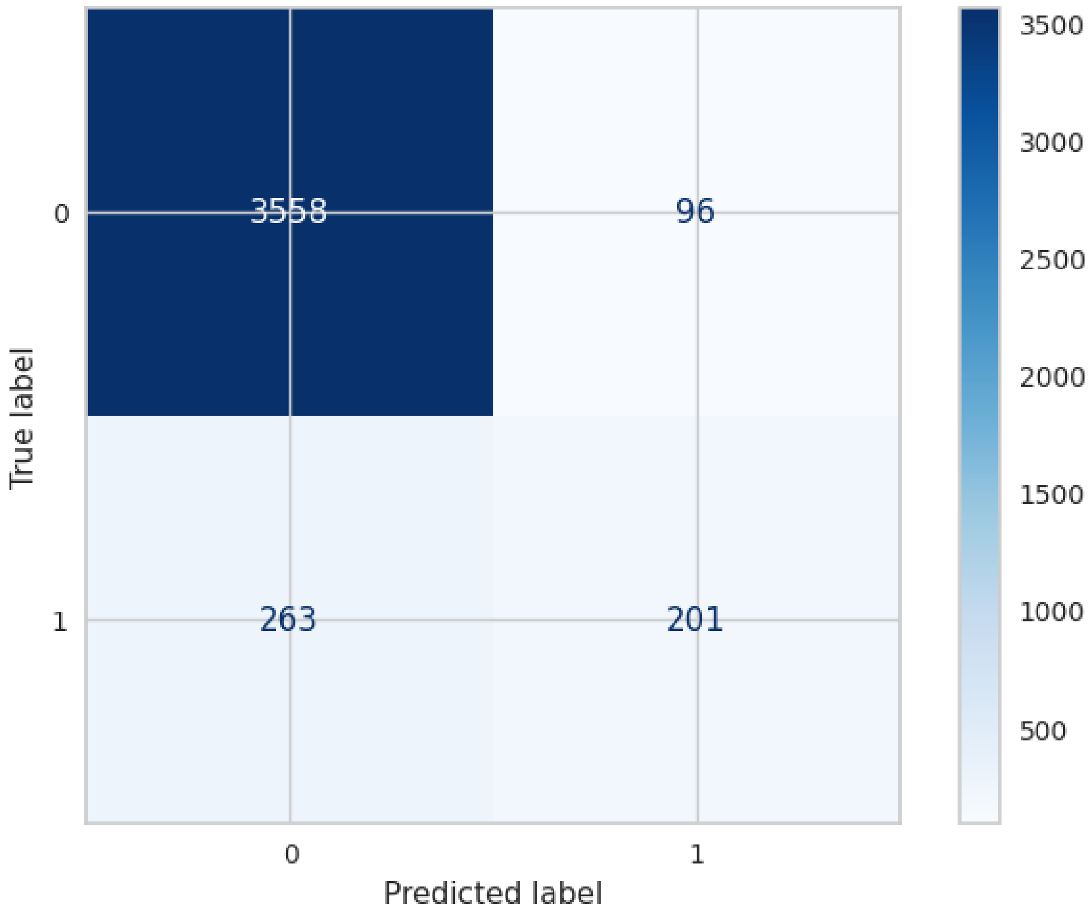
```
   macro avg       0.79      0.71      0.74      4118
weighted avg       0.90      0.91      0.90      4118

Confusion Matrix is :
 [[3545  109]
 [ 256  208]]
```

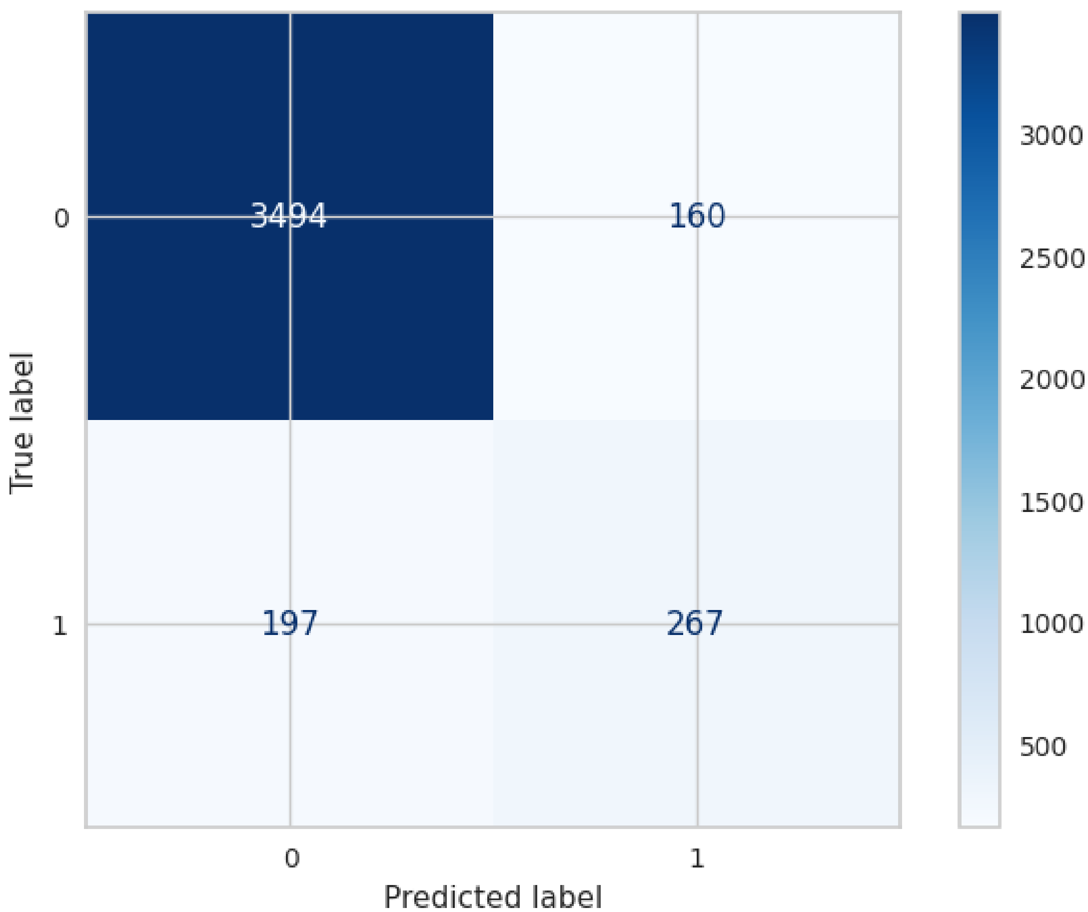



3483
171
191
273
0
1
True label
0
1
Predicted label
3000
2500
2000
1500
1000
500

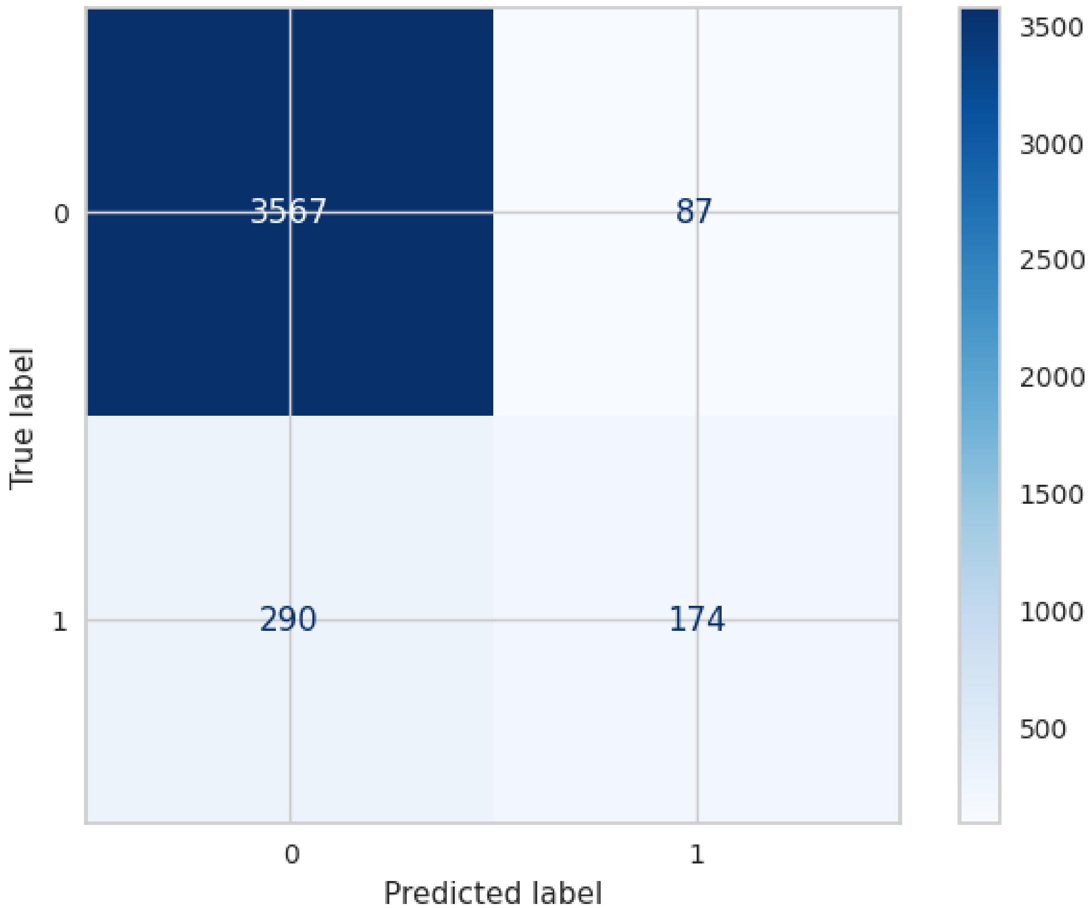


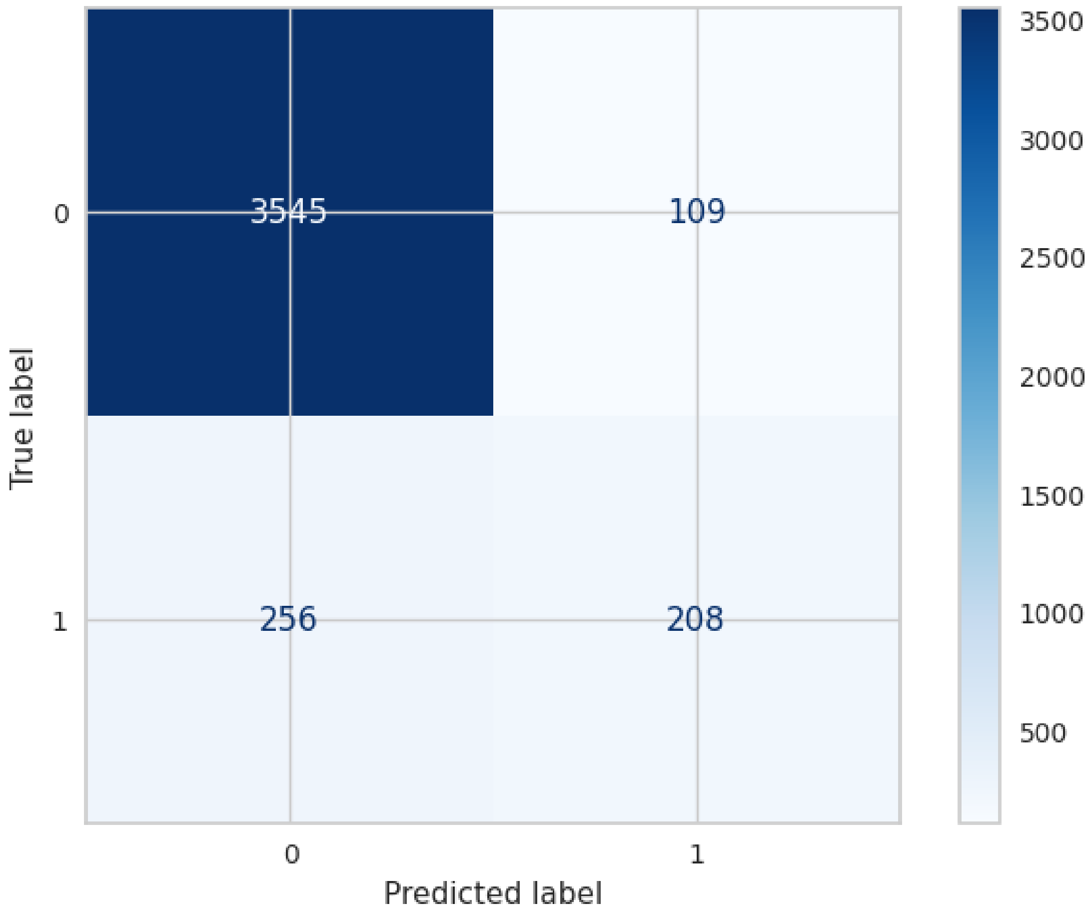
3491
163
187
277
0
1
True label
0
1
Predicted label
3000
2500
2000
1500
1000
500


3558
96
263
201
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500



3494
160
197
267
0
1
0
1
True label
Predicted label
3000
2500
2000
1500
1000
500


3567
87
290
174
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500



[229]:
```python
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['Decision','Decision With Feature','Decision Scaling','Decision␣
↪With Normalize','Decision With PCA','Decision With PCA and Scaling',
                'Decision With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[229]:
```
                                 Train Accuracy  Test Accuracy   Test F1  \
Models
Decision                               0.913293       0.915007  0.612832
Decision With Feature                  0.909353       0.912093  0.601322
Decision Scaling                       0.913293       0.915007  0.612832
Decision With Normalize                0.913428       0.912822  0.528252
Decision With PCA                      0.914940       0.913307  0.599327
Decision With PCA and Scaling          0.908220       0.908451  0.480000
Decision With PCA and Normalize        0.911161       0.911365  0.532650
```

```
                               Test Recall  Test Precision       AUC
Models
Decision                          0.596983        0.629545  0.776187
Decision With Feature             0.588362        0.614865  0.770782
Decision Scaling                  0.596983        0.629545  0.776187
Decision With Normalize           0.433190        0.676768  0.703459
Decision With PCA                 0.575431        0.625293  0.765822
Decision With PCA and Scaling     0.375000        0.666667  0.675595
Decision With PCA and Normalize   0.448276        0.656151  0.709223
```

[230]:
```
models_draw(df)
```

RandomOverSampler

[231]:
```
X_train,y_train,X_test,y_test=Split(X_classification_over,y_classification_over)
```

```
X_train shape is  (65763, 20)
X_test shape is  (7307, 20)
y_train shape is  (65763,)
y_test shape is  (7307,)
```

[232]:
```
Search(DecisionTreeClassifier(max_depth=20),{'max_depth':
 ↪[20,25,30,35,40]},X_train,y_train)
```

[232]:
```
DecisionTreeClassifier(max_depth=35)
```

[233]:
```
cross_validation(DecisionTreeClassifier(max_depth=35),X_train,y_train)
```

```
Train Score Value :  [0.99992397 0.99990496 0.99992397 0.99990496 0.99996199]
Mean 0.9999239693330242
Test Score Value :  [0.96312628 0.96411465 0.96662358 0.96722932 0.96570864]
Mean 0.9653604947609853
```

[234]:
```
Values =␣
 ↪Models(DecisionTreeClassifier(max_depth=35),X_train,y_train,X_test,y_test)
```

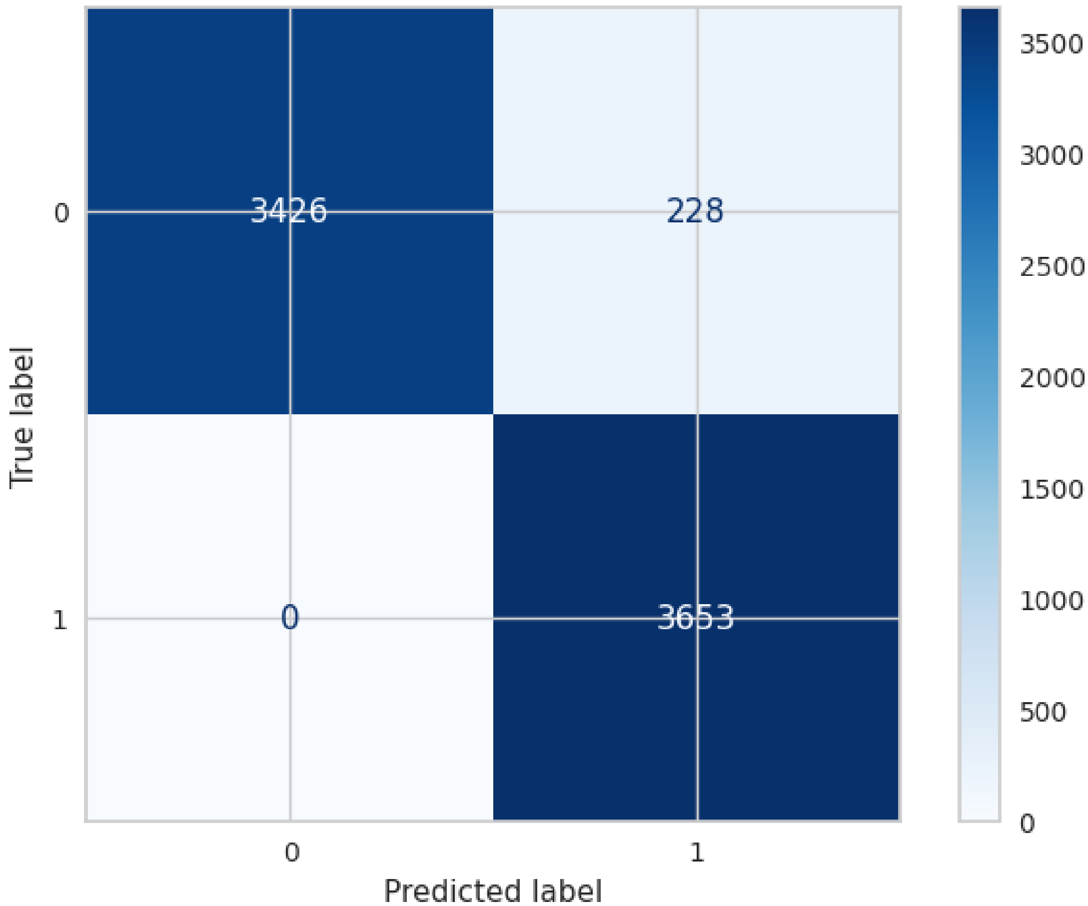
```
Apply Model With Normal Data :

Model Train Score is :  0.9999239694052887
Model Test Score is :  0.9687970439304776
F1 Score is :  0.9697371913989913
Recall Score is :  1.0
Precision Score is :  0.9412522545735635
AUC Value  :  0.9688013136288998

Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.94      0.97      3654
           1       0.94      1.00      0.97      3653
```

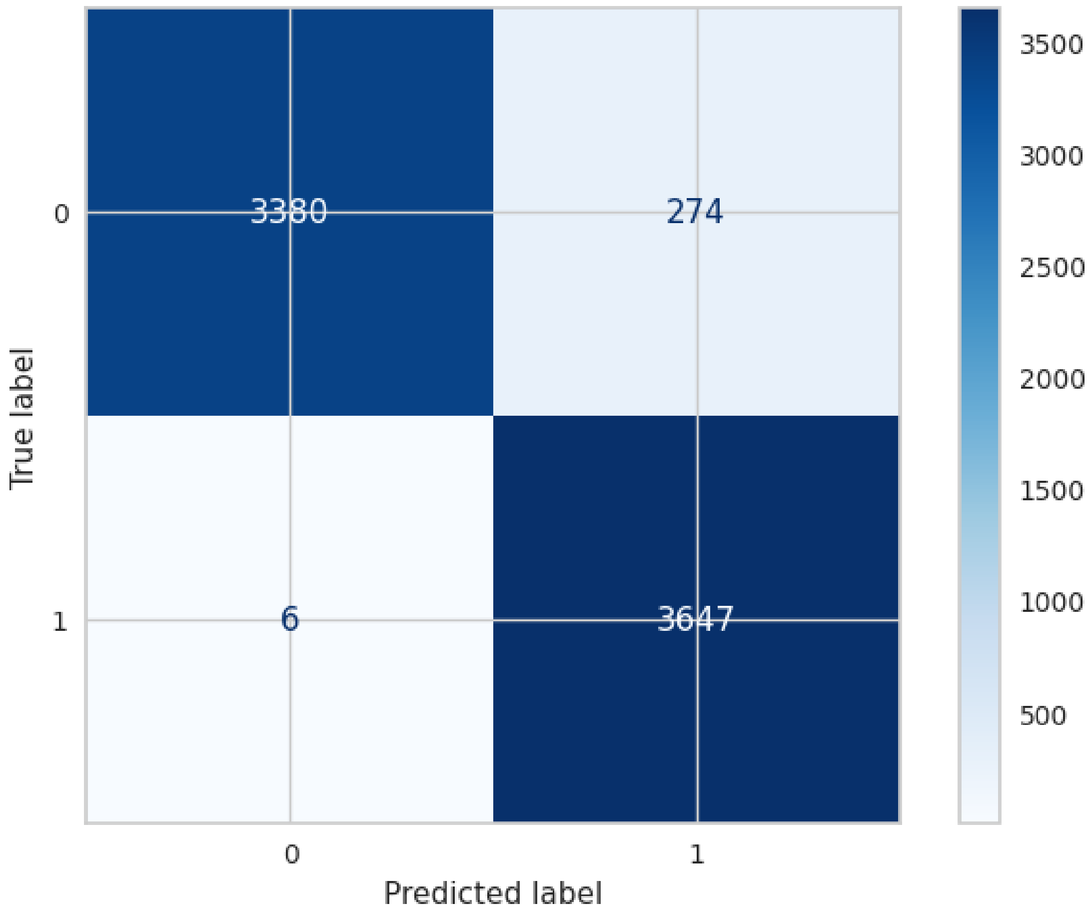
```
   accuracy                           0.97      7307
  macro avg       0.97      0.97      0.97      7307
weighted avg       0.97      0.97      0.97      7307

Confusion Matrix is :
 [[3426  228]
 [   0 3653]]


 Apply Model With Feature Selection :

Model Train Score is :  0.9883825251281115
Model Test Score is :  0.9616805802654989
F1 Score is :  0.9630314232902034
Recall Score is :  0.9983575143717492
Precision Score is :  0.9301198673807702
AUC Value  :  0.9616855990030613
Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.93      0.96      3654
           1       0.93      1.00      0.96      3653

    accuracy                           0.96      7307
   macro avg       0.96      0.96      0.96      7307
weighted avg       0.96      0.96      0.96      7307

Confusion Matrix is :
 [[3380  274]
 [   6 3647]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.9999239694052887
Model Test Score is :  0.9700287395648008
F1 Score is :  0.9708970099667774
Recall Score is :  1.0
Precision Score is :  0.9434400826446281
AUC Value  :  0.970032840722496
Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.94      0.97      3654
           1       0.94      1.00      0.97      3653
```

```
   accuracy                           0.97      7307
  macro avg       0.97      0.97      0.97      7307
weighted avg       0.97      0.97      0.97      7307

Confusion Matrix is :
 [[3435  219]
 [   0 3653]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.9945105910618433
Model Test Score is :  0.9663336526618311
F1 Score is :  0.967425847457627
Recall Score is :  1.0
Precision Score is :  0.9369068992049243
AUC Value  :  0.9663382594417077
Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.93      0.97      3654
           1       0.94      1.00      0.97      3653

    accuracy                           0.97      7307
   macro avg       0.97      0.97      0.97      7307
weighted avg       0.97      0.97      0.97      7307

Confusion Matrix is :
 [[3408  246]
 [   0 3653]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.9915301917491599
Model Test Score is :  0.9635965512522239
F1 Score is :  0.9648612945838837
Recall Score is :  0.9997262523952916
Precision Score is :  0.9323461833035487
AUC Value  :  0.9636014951084285
Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.93      0.96      3654
           1       0.93      1.00      0.96      3653

    accuracy                           0.96      7307
```

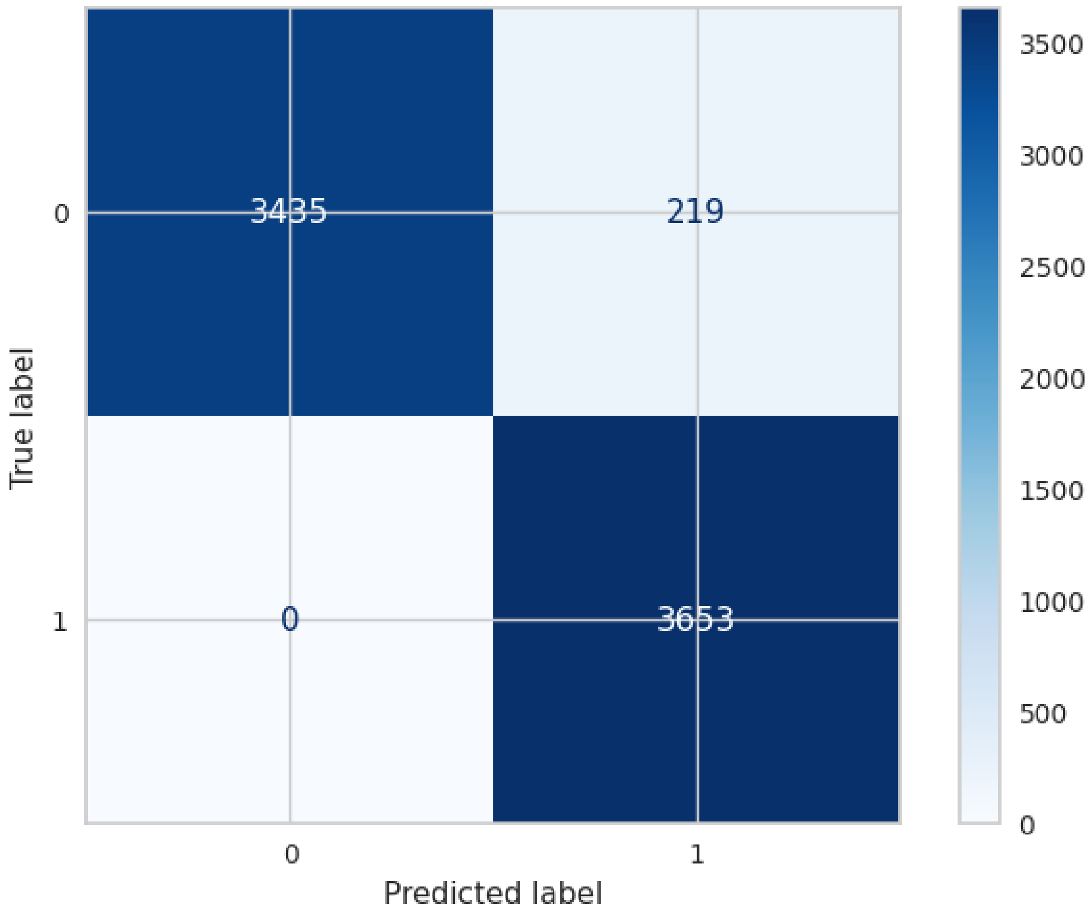
```
   macro avg       0.97      0.96      0.96      7307
weighted avg       0.97      0.96      0.96      7307

Confusion Matrix is :
 [[3389  265]
 [   1 3652]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  0.9981752657269285
Model Test Score is :  0.9667442178732722
F1 Score is :  0.9678103060007948
Recall Score is :  1.0
Precision Score is :  0.9376283367556468
AUC Value  :  0.9667487684729065
Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.93      0.97      3654
           1       0.94      1.00      0.97      3653

    accuracy                           0.97      7307
   macro avg       0.97      0.97      0.97      7307
weighted avg       0.97      0.97      0.97      7307

Confusion Matrix is :
 [[3411  243]
 [   0 3653]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  0.995483782674148
Model Test Score is :  0.9642808266046257
F1 Score is :  0.9655081273952688
Recall Score is :  1.0
Precision Score is :  0.9333163004598876
AUC Value  :  0.9642857142857143
Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.93      0.96      3654
           1       0.93      1.00      0.97      3653

    accuracy                           0.96      7307
   macro avg       0.97      0.96      0.96      7307
```

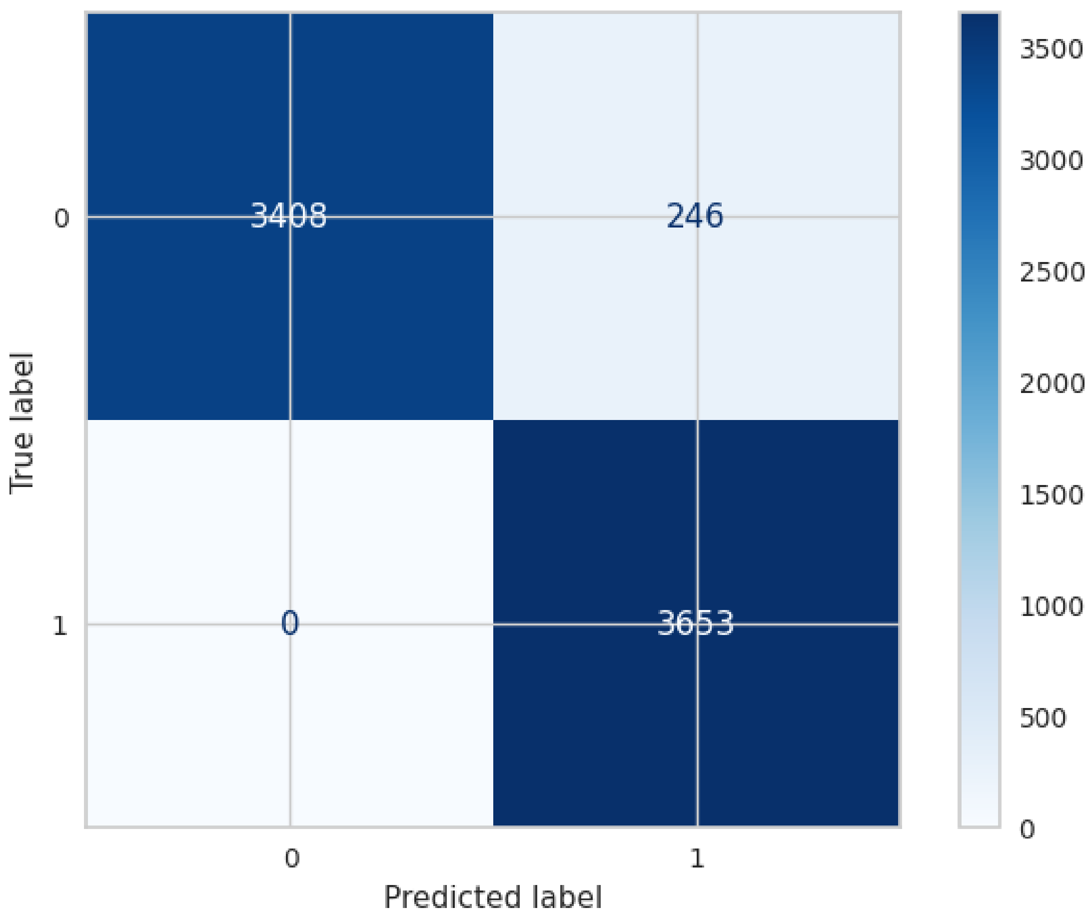
```
weighted avg       0.97      0.96      0.96      7307

Confusion Matrix is :
 [[3393  261]
 [   0 3653]]
```

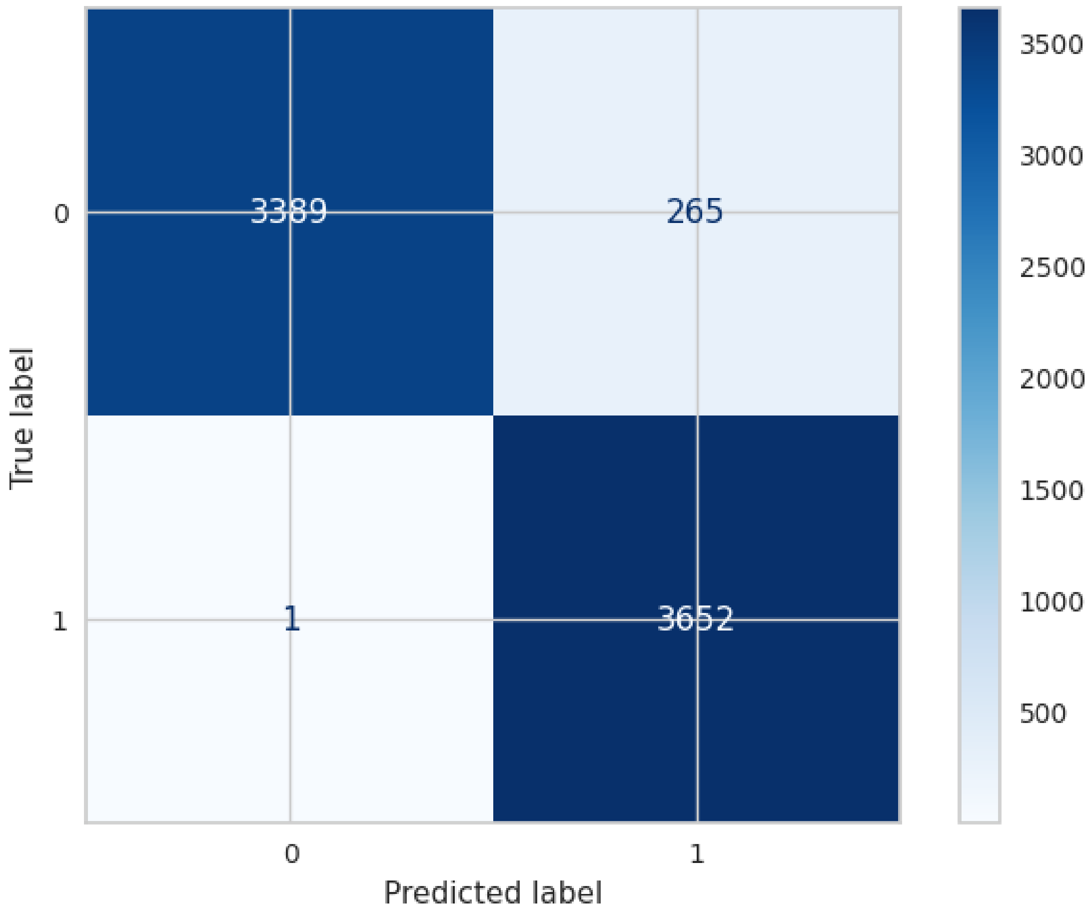


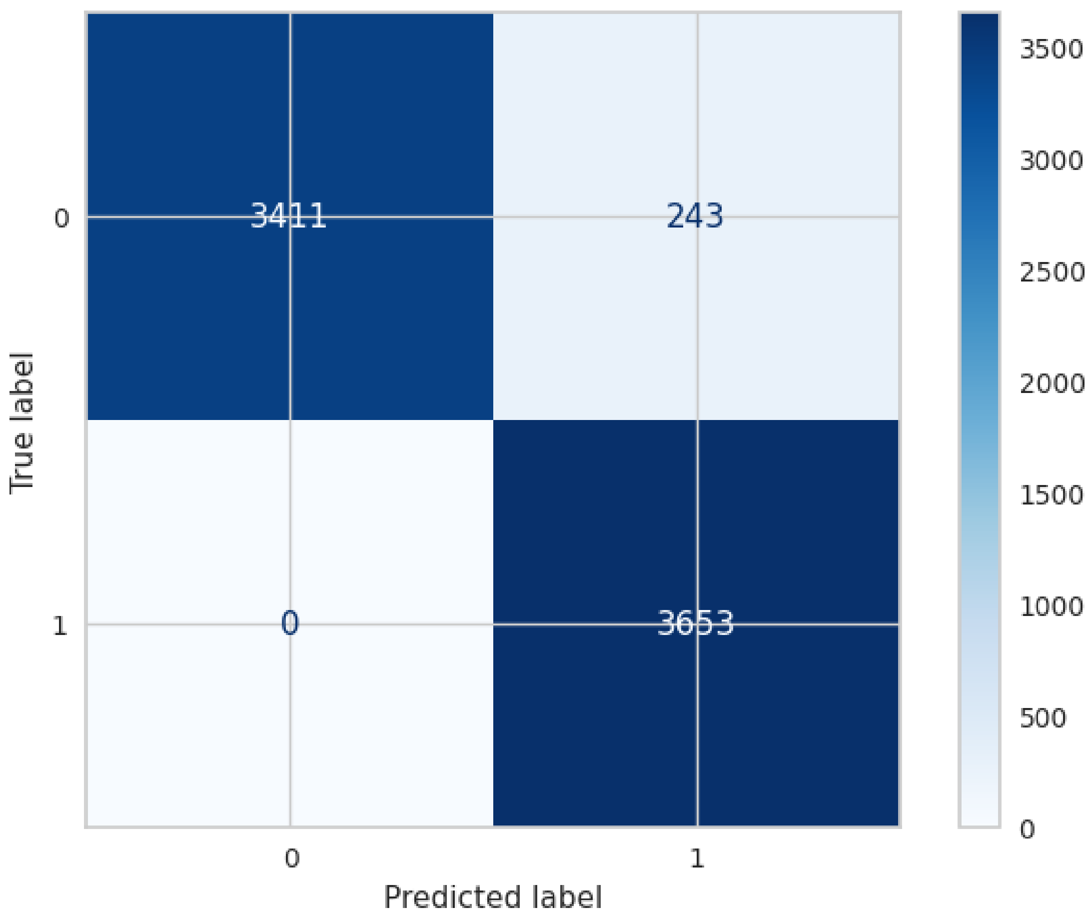


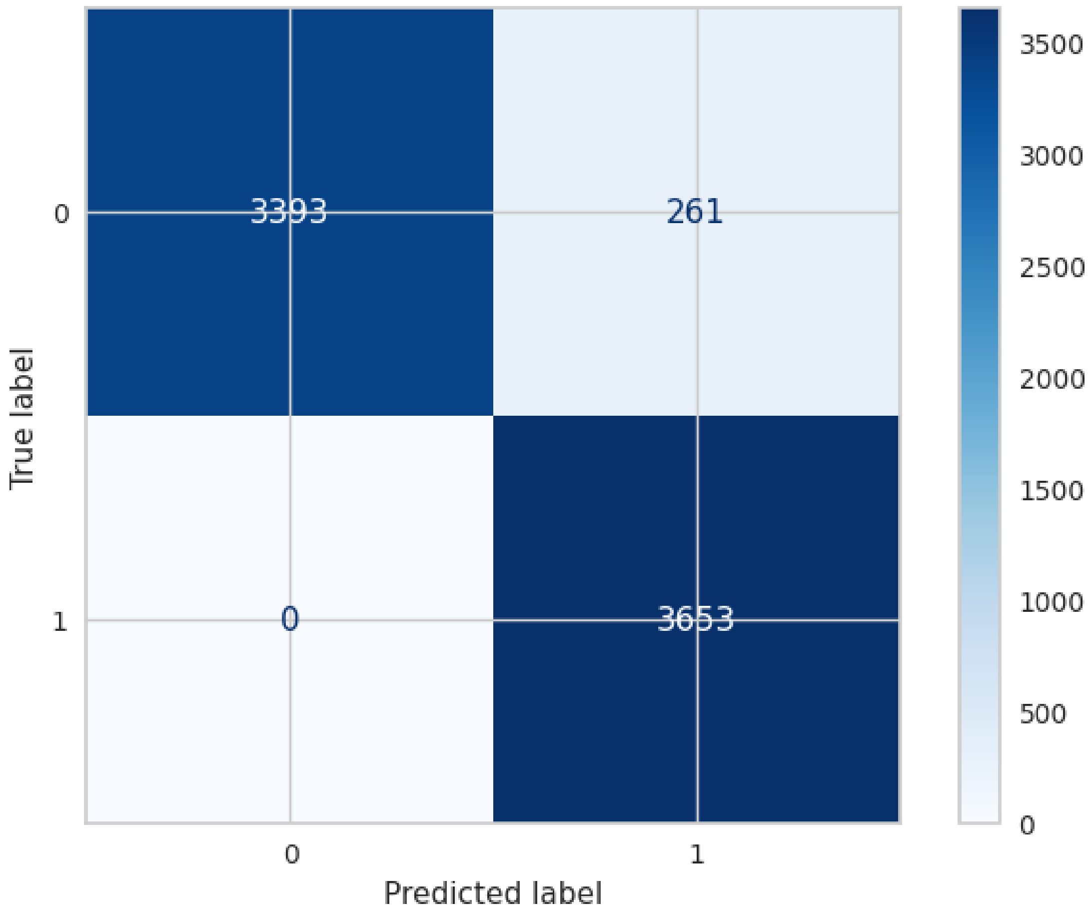
3380
274
6
3647
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500


3435
219
0
3653
True label
Predicted label
0
1
0
1
3500
3000
2500
2000
1500
1000
500
0


3408
246
0
3653
True label
0
1
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500
0


0
1
True label
3389
265
1
3652
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500


3411
243
0
3653
True label
Predicted label
0
1
0
1
3500
3000
2500
2000
1500
1000
500
0



[235]:
```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['Decision Over','Decision Over With Feature','Decision Over␣
↪Scaling','Decision Over With Normalize','Decision Over With PCA'
                ,'Decision Over With PCA and Scaling',
                'Decision Over With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[235]:
```
                                    Train Accuracy  Test Accuracy   Test F1  \
Models
Decision Over                             0.999924       0.968797  0.969737
Decision Over With Feature                0.988383       0.961681  0.963031
Decision Over Scaling                     0.999924       0.970029  0.970897
Decision Over With Normalize              0.994511       0.966334  0.967426
Decision Over With PCA                    0.991530       0.963597  0.964861
Decision Over With PCA and Scaling        0.998175       0.966744  0.967810
Decision Over With PCA and Normalize      0.995484       0.964281  0.965508
```

```
                                       Test Recall  Test Precision       AUC
Models
Decision Over                             1.000000        0.941252  0.968801
Decision Over With Feature                0.998358        0.930120  0.961686
Decision Over Scaling                     1.000000        0.943440  0.970033
Decision Over With Normalize              1.000000        0.936907  0.966338
Decision Over With PCA                    0.999726        0.932346  0.963601
Decision Over With PCA and Scaling        1.000000        0.937628  0.966749
Decision Over With PCA and Normalize      1.000000        0.933316  0.964286
```

[236]:
```
models_draw(df)
```

RandomUnderSampler

[237]:
```
X_train,y_train,X_test,y_test=Split(X_classification_under,y_classification_under)
```

```
X_train shape is  (8350, 20)
X_test shape is  (928, 20)
y_train shape is  (8350,)
y_test shape is  (928,)
```

[238]:
```
Search(DecisionTreeClassifier(max_depth=20),{'max_depth':
 ↪[20,25,30,35,40]},X_train,y_train)
```

[238]:
```
DecisionTreeClassifier(max_depth=25)
```

[239]:
```
cross_validation(DecisionTreeClassifier(max_depth=35),X_train,y_train)
```

```
Train Score Value :  [1. 1. 1. 1. 1.]    Mean 1.0
Test Score Value :  [0.85688623 0.83173653 0.82814371 0.84311377 0.84850299]
Mean 0.8416766467065868
```

[240]:
```
Values =␣
 ↪Models(DecisionTreeClassifier(max_depth=35),X_train,y_train,X_test,y_test)
```

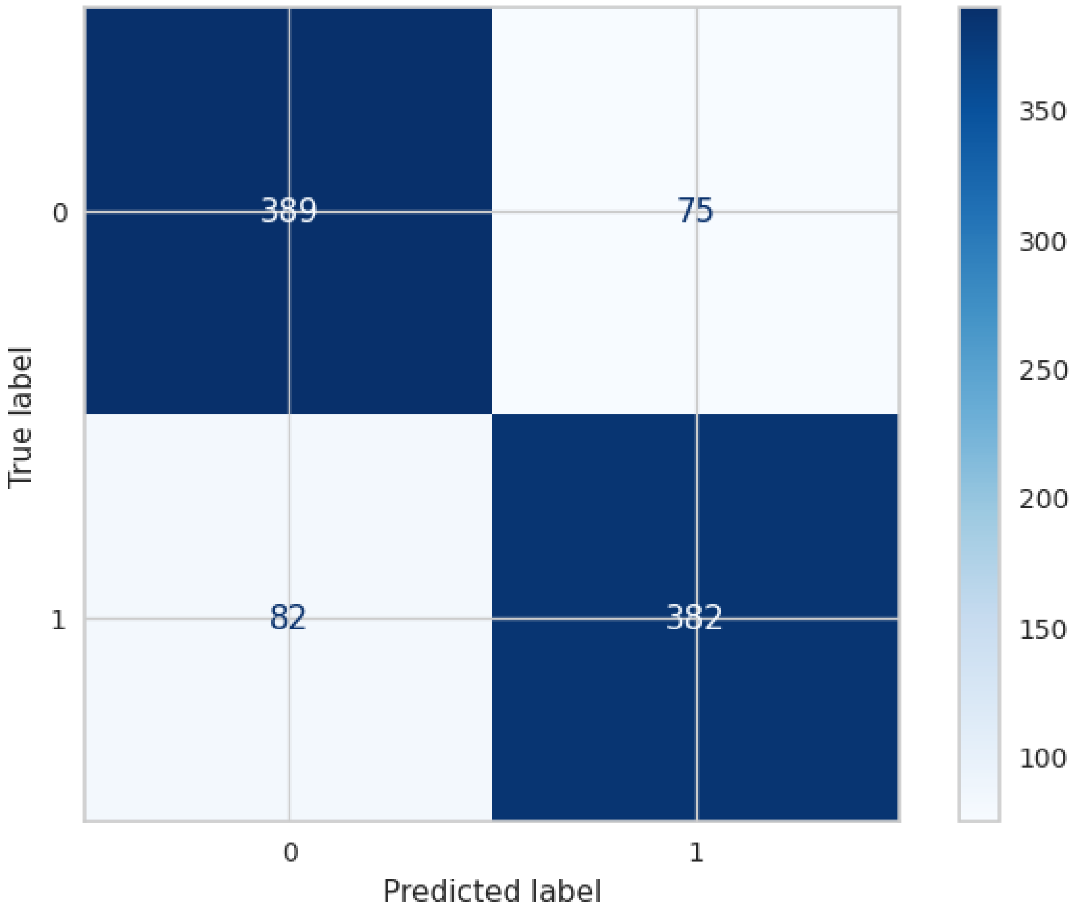
```
Apply Model With Normal Data :

Model Train Score is :  1.0
Model Test Score is :  0.8308189655172413
F1 Score is :  0.8295331161780674
Recall Score is :  0.8232758620689655
Precision Score is :  0.8358862144420132
AUC Value  :  0.8308189655172413

Classification Report is :                precision    recall  f1-score
support

           0       0.83      0.84      0.83       464
           1       0.84      0.82      0.83       464
```

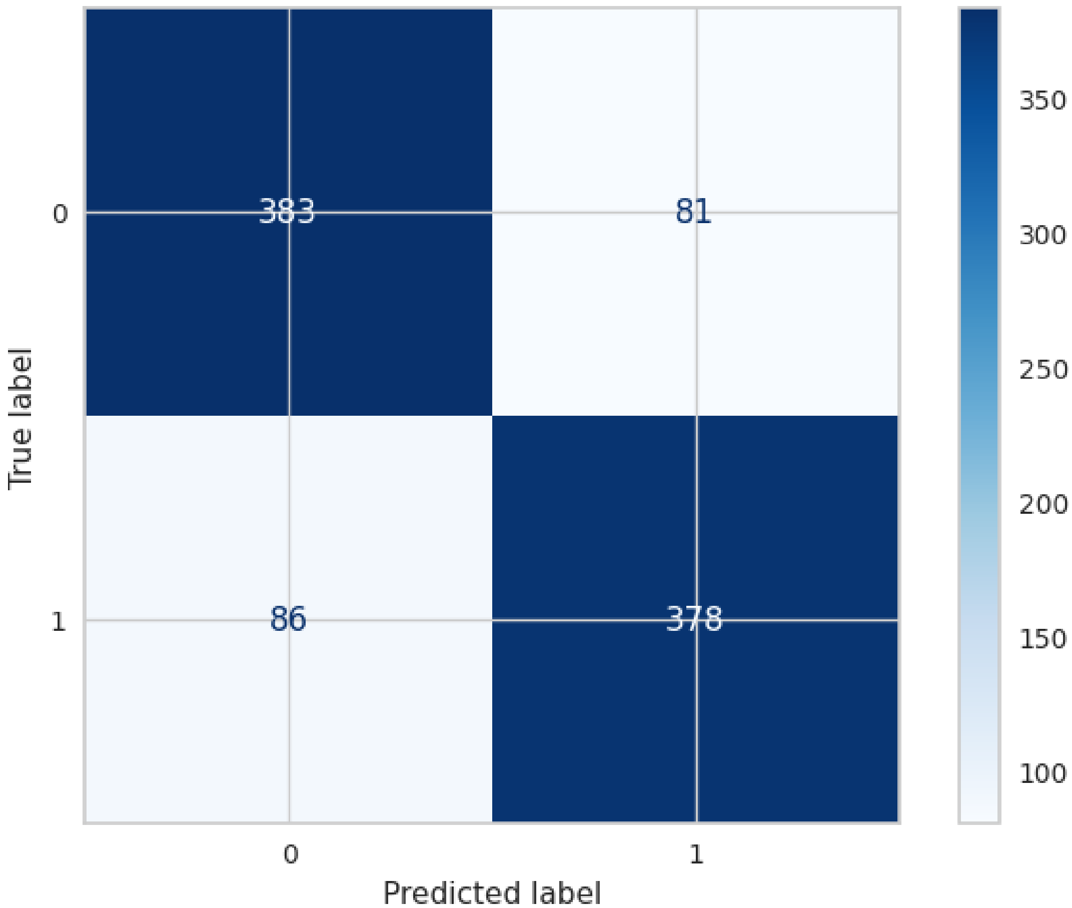
```
   accuracy                           0.83       928
  macro avg       0.83      0.83      0.83       928
weighted avg       0.83      0.83      0.83       928

Confusion Matrix is :
 [[389  75]
 [ 82 382]]


 Apply Model With Feature Selection :

Model Train Score is :  0.9905389221556886
Model Test Score is :  0.8200431034482759
F1 Score is :  0.8190682556879739
Recall Score is :  0.8146551724137931
Precision Score is :  0.8235294117647058
AUC Value  :  0.820043103448276

Classification Report is :                precision    recall  f1-score
support

           0       0.82      0.83      0.82       464
           1       0.82      0.81      0.82       464

    accuracy                           0.82       928
   macro avg       0.82      0.82      0.82       928
weighted avg       0.82      0.82      0.82       928

Confusion Matrix is :
 [[383  81]
 [ 86 378]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  1.0
Model Test Score is :  0.834051724137931
F1 Score is :  0.8326086956521739
Recall Score is :  0.8254310344827587
Precision Score is :  0.8399122807017544
AUC Value  :  0.8340517241379309

Classification Report is :                precision    recall  f1-score
support

           0       0.83      0.84      0.84       464
           1       0.84      0.83      0.83       464
```

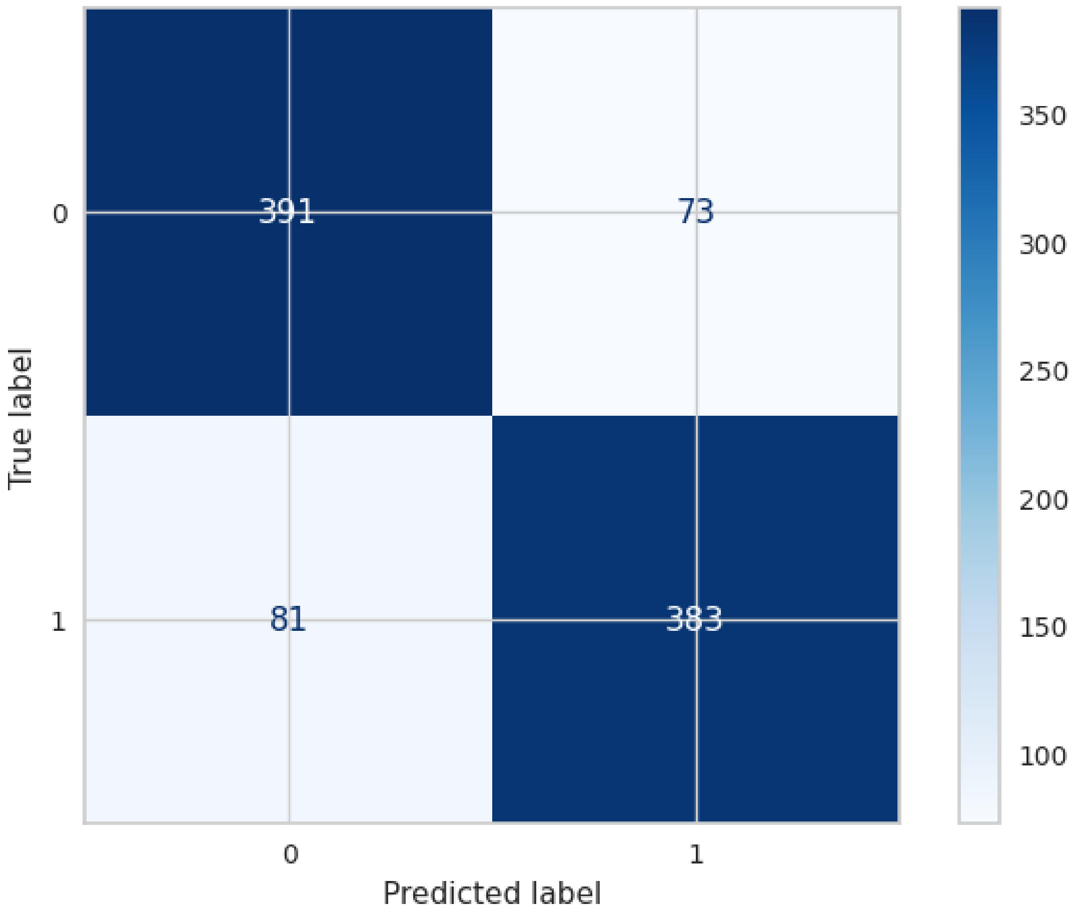
```
    accuracy                           0.83       928
   macro avg       0.83      0.83      0.83       928
weighted avg       0.83      0.83      0.83       928

Confusion Matrix is :
 [[391  73]
 [ 81 383]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  1.0
Model Test Score is :  0.8200431034482759
F1 Score is :  0.8146503884572697
Recall Score is :  0.790948275862069
Precision Score is :  0.8398169336384439
AUC Value  :  0.8200431034482759

Classification Report is :                precision    recall  f1-score
support

           0       0.80      0.85      0.83       464
           1       0.84      0.79      0.81       464

    accuracy                           0.82       928
   macro avg       0.82      0.82      0.82       928
weighted avg       0.82      0.82      0.82       928

Confusion Matrix is :
 [[394  70]
 [ 97 367]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  1.0
Model Test Score is :  0.8362068965517241
F1 Score is :  0.8354978354978355
Recall Score is :  0.8318965517241379
Precision Score is :  0.8391304347826087
AUC Value  :  0.836206896551724

Classification Report is :                precision    recall  f1-score
support

           0       0.83      0.84      0.84       464
           1       0.84      0.83      0.84       464

    accuracy                           0.84       928
```

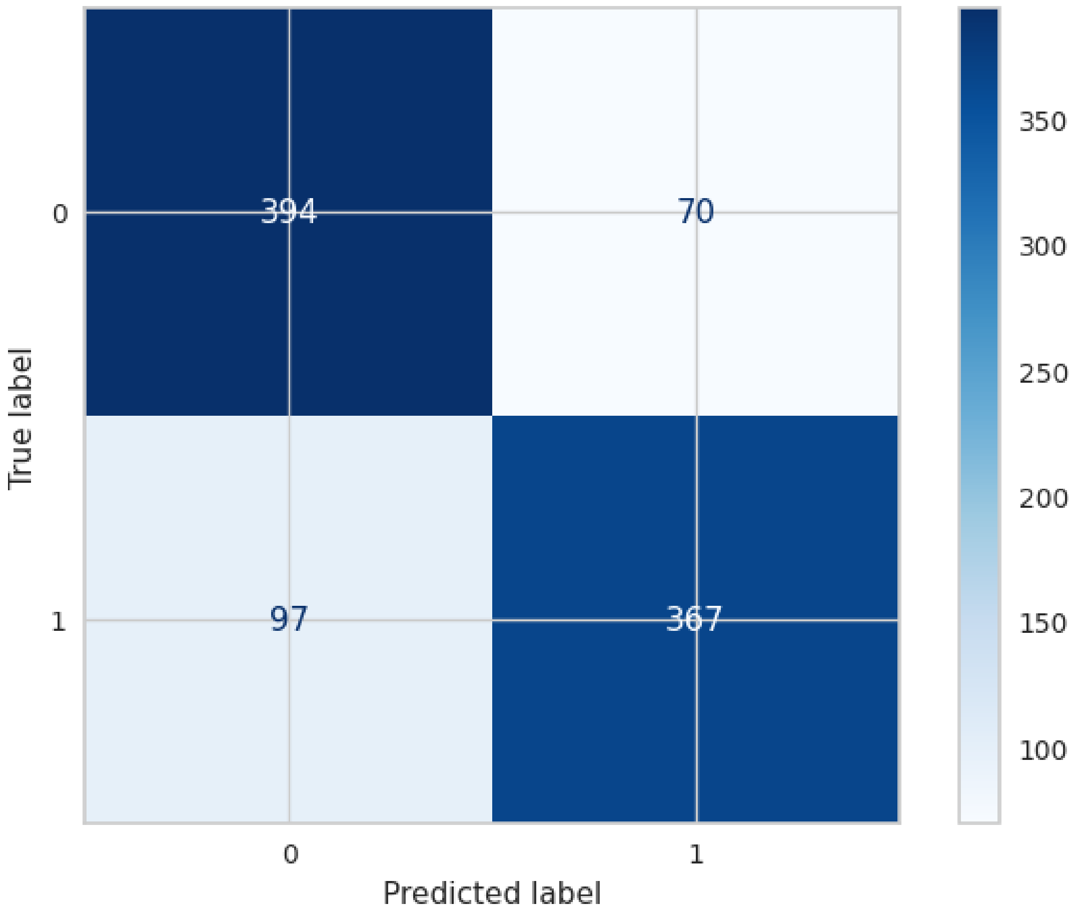
```
   macro avg       0.84      0.84      0.84       928
weighted avg       0.84      0.84      0.84       928

Confusion Matrix is :
 [[390  74]
 [ 78 386]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  1.0
Model Test Score is :  0.8168103448275862
F1 Score is :  0.8127753303964758
Recall Score is :  0.7952586206896551
Precision Score is :  0.831081081081081
AUC Value  :  0.8168103448275863
Classification Report is :                precision    recall  f1-score
support

           0       0.80      0.84      0.82       464
           1       0.83      0.80      0.81       464

    accuracy                           0.82       928
   macro avg       0.82      0.82      0.82       928
weighted avg       0.82      0.82      0.82       928

Confusion Matrix is :
 [[389  75]
 [ 95 369]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  1.0
Model Test Score is :  0.8232758620689655
F1 Score is :  0.8217391304347826
Recall Score is :  0.8146551724137931
Precision Score is :  0.8289473684210527
AUC Value  :  0.8232758620689655
Classification Report is :                precision    recall  f1-score
support

           0       0.82      0.83      0.82       464
           1       0.83      0.81      0.82       464

    accuracy                           0.82       928
   macro avg       0.82      0.82      0.82       928
```

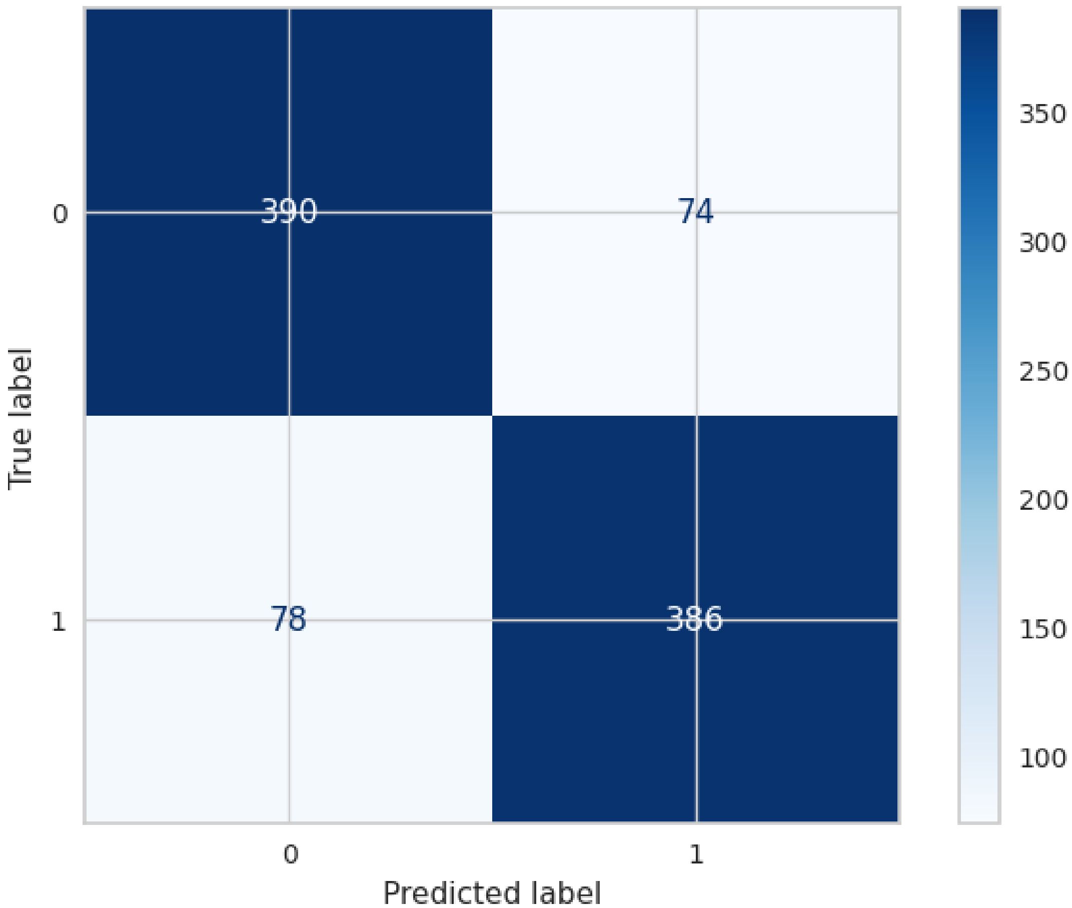
```
weighted avg       0.82      0.82      0.82       928

Confusion Matrix is :
 [[386  78]
 [ 86 378]]
```



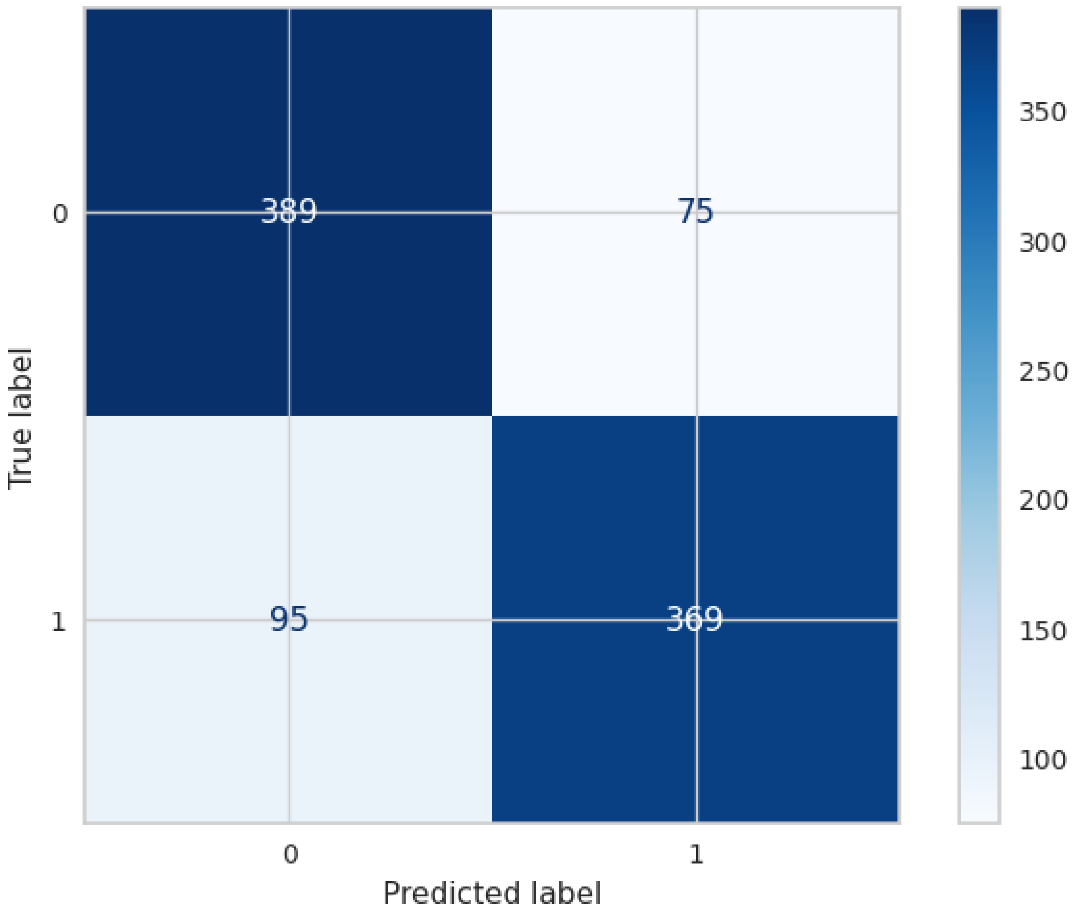

383
81
86
378
0
1
True label
0
1
Predicted label
350
300
250
200
150
100

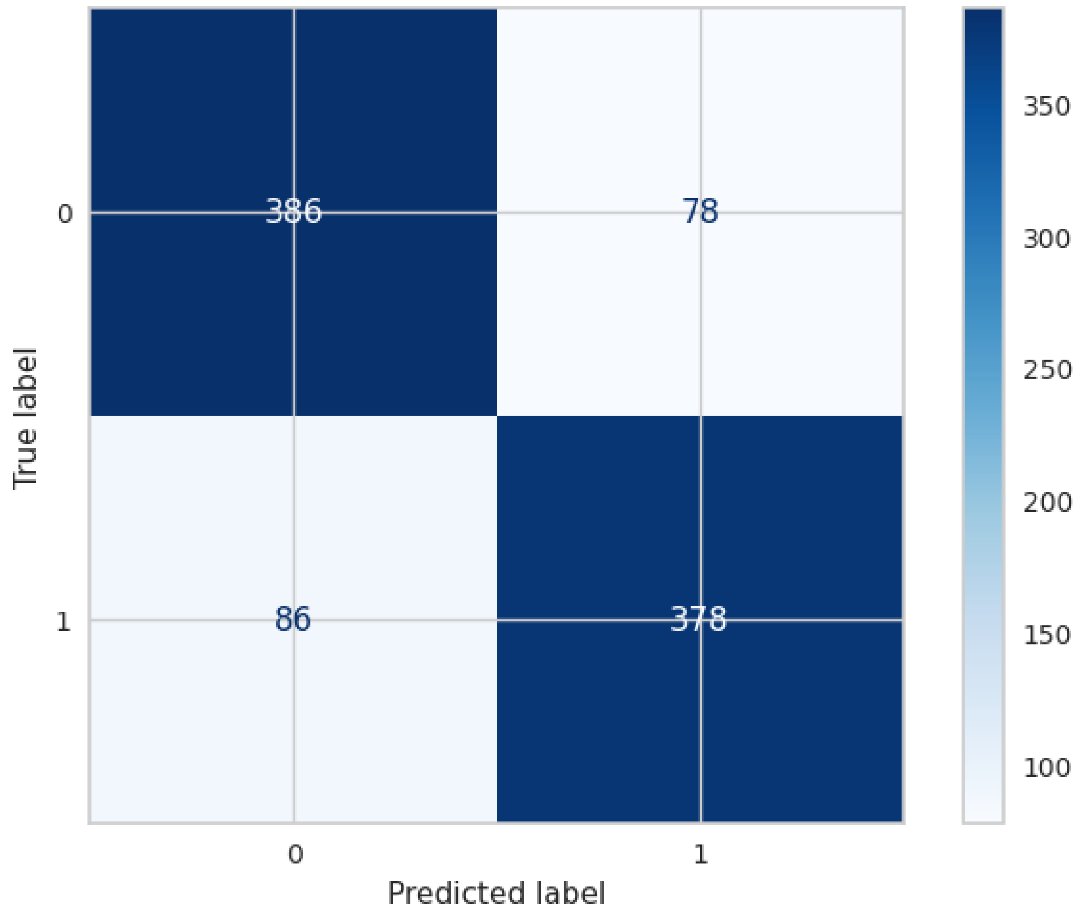

391
73
81
383
0
1
0
1
True label
Predicted label
350
300
250
200
150
100


394
70
97
367
0
1
True label
0
1
Predicted label
350
300
250
200
150
100



390
74
78
386
0
1
True label
0
1
Predicted label
350
300
250
200
150
100


389
75
95
369
0
1
True label
0
1
Predicted label
350
300
250
200
150
100



[241]:
```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
 ↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['Decision Under','Decision Under With Feature','Decision Under␣
 ↪Scaling','Decision Under With Normalize','Decision Under With PCA'
                ,'Decision Under With PCA and Scaling',
                'Decision Under With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[241]:
```
                                       Train Accuracy  Test Accuracy  \
Models
Decision Under                               1.000000       0.830819
Decision Under With Feature                  0.990539       0.820043
Decision Under Scaling                       1.000000       0.834052
Decision Under With Normalize                1.000000       0.820043
Decision Under With PCA                      1.000000       0.836207
Decision Under With PCA and Scaling          1.000000       0.816810
Decision Under With PCA and Normalize        1.000000       0.823276
```

```
                                      Test F1  Test Recall  Test Precision  \
Models
Decision Under                       0.829533     0.823276        0.835886
Decision Under With Feature          0.819068     0.814655        0.823529
Decision Under Scaling               0.832609     0.825431        0.839912
Decision Under With Normalize        0.814650     0.790948        0.839817
Decision Under With PCA              0.835498     0.831897        0.839130
Decision Under With PCA and Scaling  0.812775     0.795259        0.831081
Decision Under With PCA and Normalize 0.821739    0.814655        0.828947

                                          AUC
Models
Decision Under                       0.830819
Decision Under With Feature          0.820043
Decision Under Scaling               0.834052
Decision Under With Normalize        0.820043
Decision Under With PCA              0.836207
Decision Under With PCA and Scaling  0.816810
Decision Under With PCA and Normalize 0.823276
```

[242]:
```
models_draw(df)
```

KNeighborsClassifier

[243]:
```
X_train,y_train,X_test,y_test=Split(X_classification,y_classification)
```

```
X_train shape is  (37056, 20)
X_test shape is  (4118, 20)
y_train shape is  (37056,)
y_test shape is  (4118,)
```

[244]:
```
Search(KNeighborsClassifier(n_neighbors=3),{'n_neighbors':
 ↪[3,5,7,9,11]},X_train,y_train)
```

[244]:
```
KNeighborsClassifier(n_neighbors=11)
```

[245]:
```
cross_validation(KNeighborsClassifier(n_neighbors=11),X_train,y_train)
```

```
Train Score Value :  [0.92018621 0.91840108 0.91998651 0.9200877  0.91860347]
Mean 0.9194529950157511
Test Score Value :  [0.90326498 0.90622048 0.90352179 0.90500607 0.90999865]
Mean 0.905602394684234
```

[246]:
```
Values =␣
 ↪Models(KNeighborsClassifier(n_neighbors=11),X_train,y_train,X_test,y_test)
```

```
Apply Model With Normal Data :

Model Train Score is :  0.9197970639032815
```

```
Model Test Score is :  0.9108790675084992
F1 Score is :  0.5625744934445768
Recall Score is :  0.5086206896551724
Precision Score is :  0.6293333333333333
AUC Value  :  0.7352900930487138

Classification Report is :                precision    recall  f1-score
support

           0       0.94      0.96      0.95      3654
           1       0.63      0.51      0.56       464

    accuracy                           0.91      4118
   macro avg       0.78      0.74      0.76      4118
weighted avg       0.90      0.91      0.91      4118

Confusion Matrix is :
 [[3515  139]
 [ 228  236]]


 Apply Model With Feature Selection :

Model Train Score is :  0.9168015975820379
Model Test Score is :  0.9065080135988344
F1 Score is :  0.5299145299145299
Recall Score is :  0.4676724137931034
Precision Score is :  0.6112676056338028
AUC Value  :  0.7149527914614121

Classification Report is :                precision    recall  f1-score
support

           0       0.93      0.96      0.95      3654
           1       0.61      0.47      0.53       464

    accuracy                           0.91      4118
   macro avg       0.77      0.71      0.74      4118
weighted avg       0.90      0.91      0.90      4118

Confusion Matrix is :
 [[3516  138]
 [ 247  217]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.9145347582037997
Model Test Score is :  0.9023797960174842
```

```
F1 Score is :  0.41739130434782606
Recall Score is :  0.3103448275862069
Precision Score is :  0.6371681415929203
AUC Value  :  0.643951833607006

Classification Report is :                precision    recall  f1-score
support

           0       0.92      0.98      0.95      3654
           1       0.64      0.31      0.42       464

    accuracy                           0.90      4118
   macro avg       0.78      0.64      0.68      4118
weighted avg       0.89      0.90      0.89      4118

Confusion Matrix is :
 [[3572   82]
 [ 320  144]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.918825561312608
Model Test Score is :  0.9052938319572608
F1 Score is :  0.524390243902439
Recall Score is :  0.46336206896551724
Precision Score is :  0.6039325842696629
AUC Value  :  0.7123871100164204

Classification Report is :                precision    recall  f1-score
support

           0       0.93      0.96      0.95      3654
           1       0.60      0.46      0.52       464

    accuracy                           0.91      4118
   macro avg       0.77      0.71      0.74      4118
weighted avg       0.90      0.91      0.90      4118

Confusion Matrix is :
 [[3513  141]
 [ 249  215]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.9200399395509499
Model Test Score is :  0.9101505585235551
F1 Score is :  0.558472553699284
```

```
Recall Score is :  0.5043103448275862
Precision Score is :  0.6256684491978609
AUC Value  :  0.7329980842911878
```

```
Classification Report is :                precision    recall  f1-score
support

           0       0.94      0.96      0.95      3654
           1       0.63      0.50      0.56       464

    accuracy                           0.91      4118
   macro avg       0.78      0.73      0.75      4118
weighted avg       0.90      0.91      0.91      4118
```

```
Confusion Matrix is :
 [[3514  140]
 [ 230  234]]
```

```
 Apply Model With Normal Data With PCA and Scaling :
```

```
Model Train Score is :  0.9145347582037997
Model Test Score is :  0.9023797960174842
F1 Score is :  0.41739130434782606
Recall Score is :  0.3103448275862069
Precision Score is :  0.6371681415929203
AUC Value  :  0.643951833607006
```

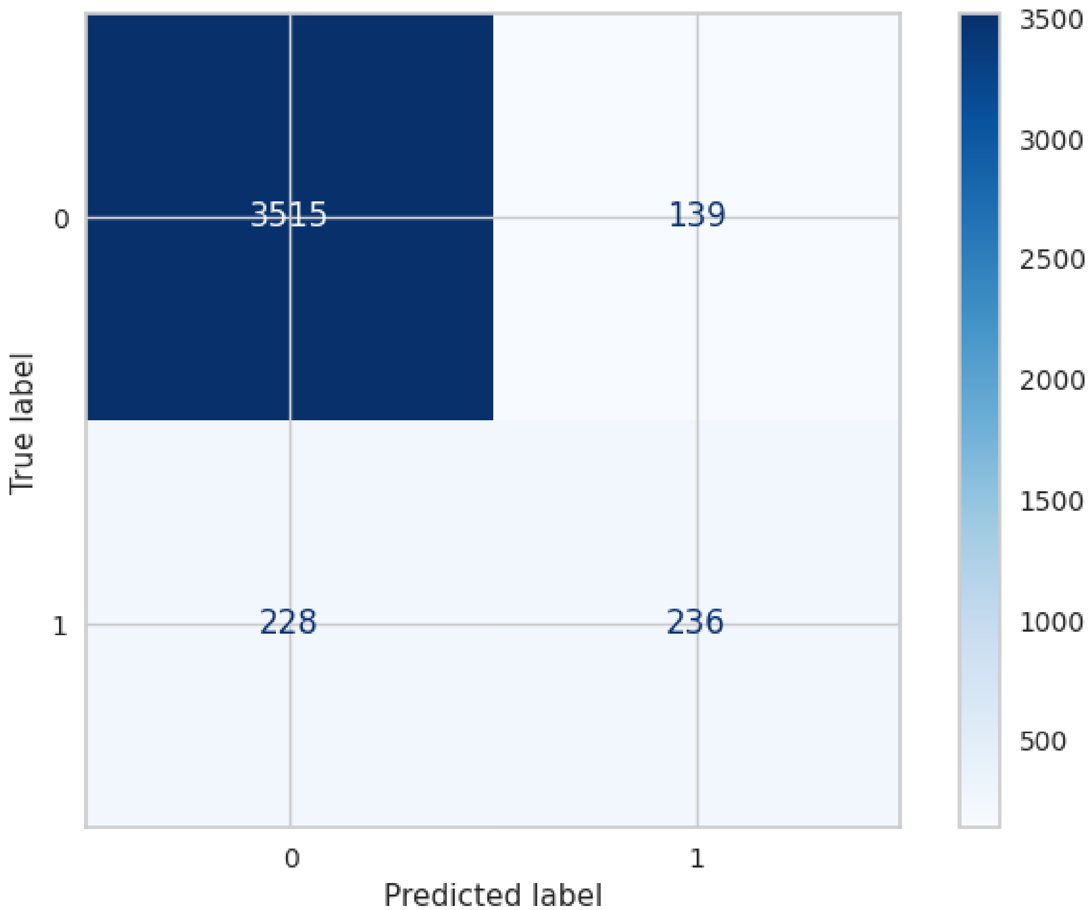
```
Classification Report is :                precision    recall  f1-score
support

           0       0.92      0.98      0.95      3654
           1       0.64      0.31      0.42       464

    accuracy                           0.90      4118
   macro avg       0.78      0.64      0.68      4118
weighted avg       0.89      0.90      0.89      4118
```

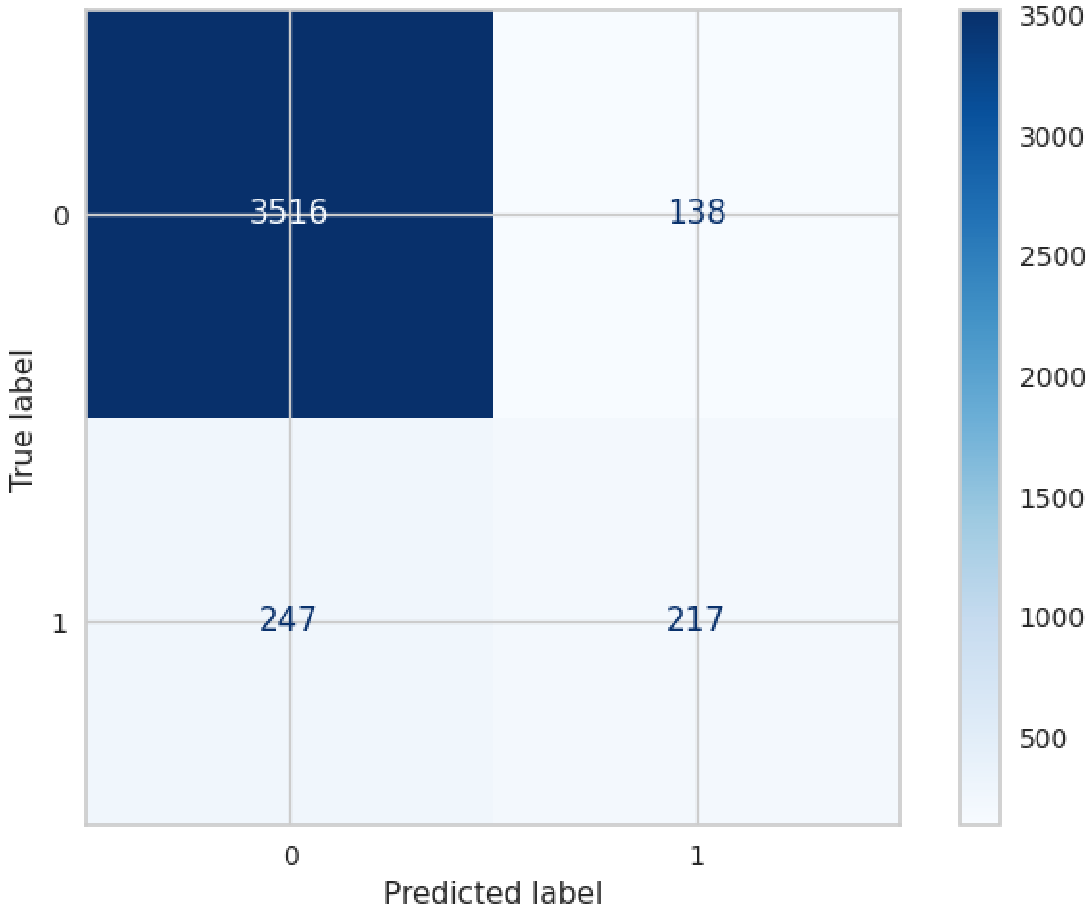
```
Confusion Matrix is :
 [[3572   82]
 [ 320  144]]
```

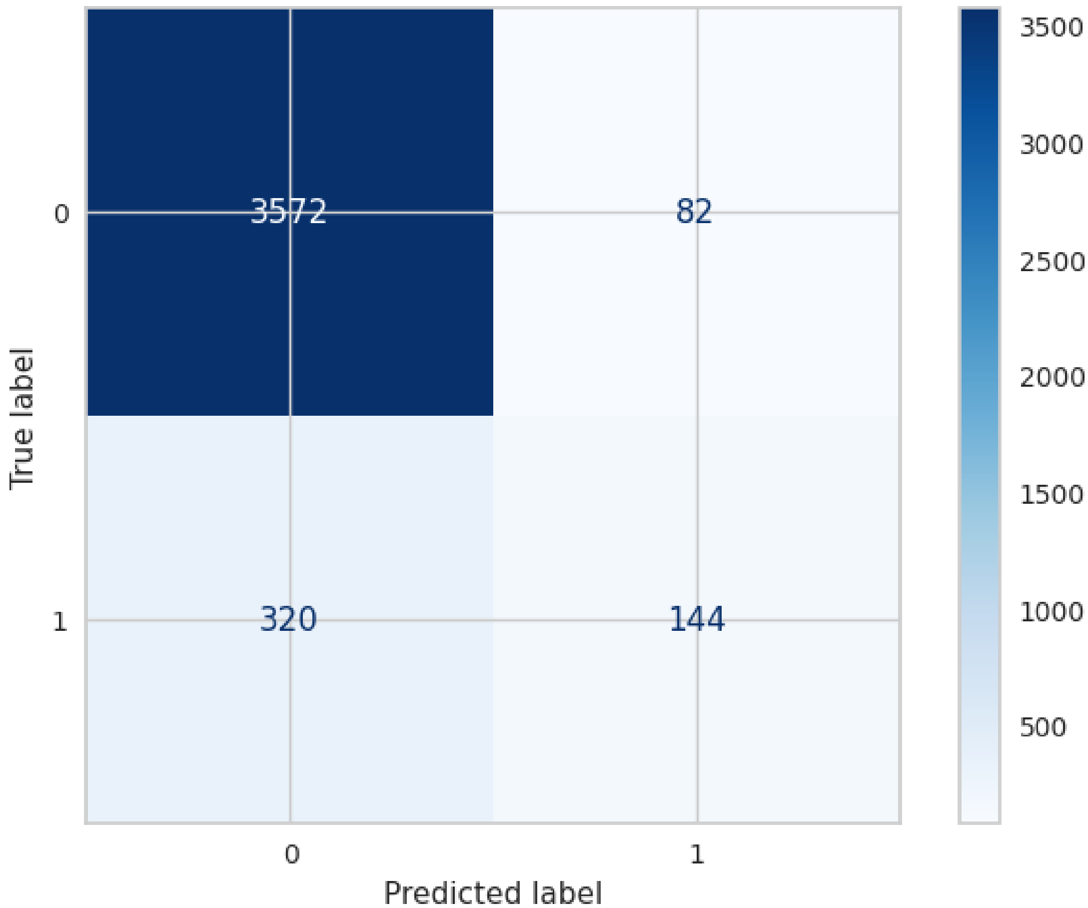
```
 Apply Model With Normal Data With PCA and Normalize :
```

```
Model Train Score is :  0.9188525474956822
Model Test Score is :  0.9052938319572608
F1 Score is :  0.524390243902439
Recall Score is :  0.46336206896551724
```

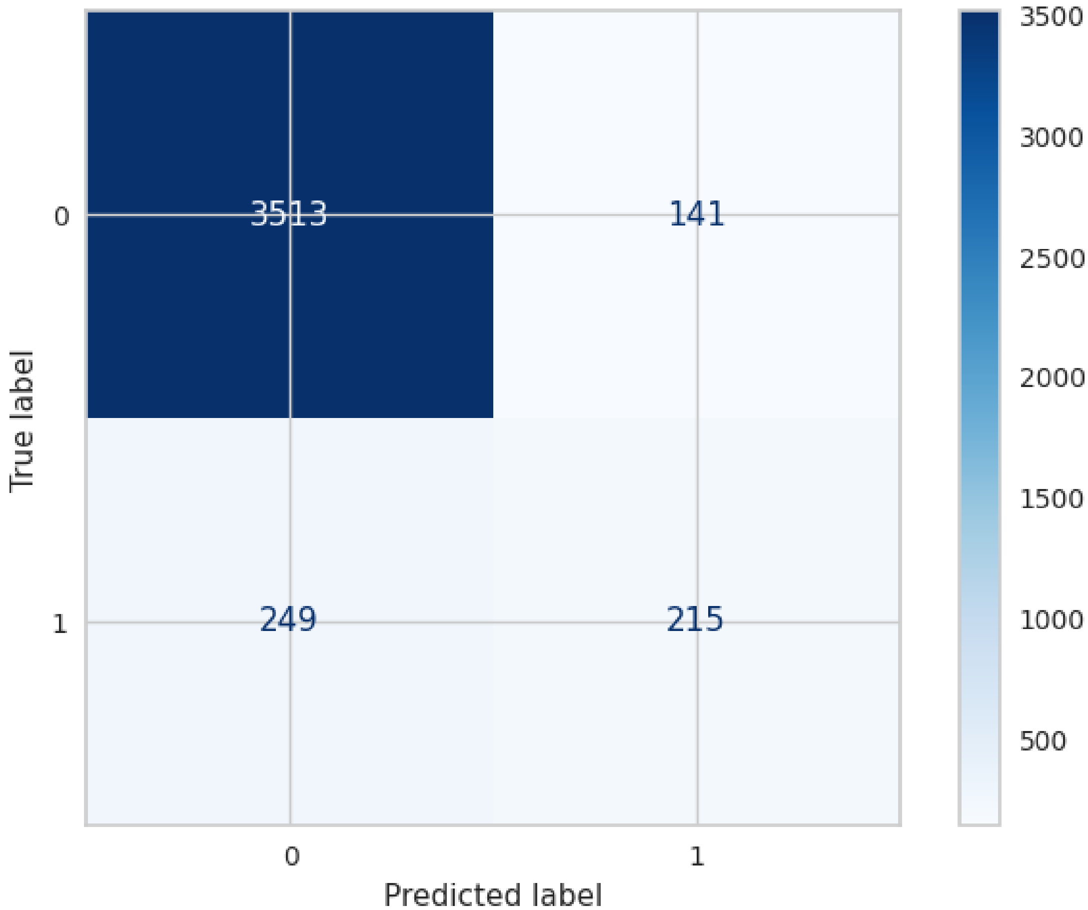
```
Precision Score is :  0.6039325842696629
AUC Value  :  0.7123871100164204
Classification Report is :                precision    recall  f1-score
support

           0       0.93      0.96      0.95      3654
           1       0.60      0.46      0.52       464

    accuracy                           0.91      4118
   macro avg       0.77      0.71      0.74      4118
weighted avg       0.90      0.91      0.90      4118

Confusion Matrix is :
 [[3513  141]
 [ 249  215]]
```

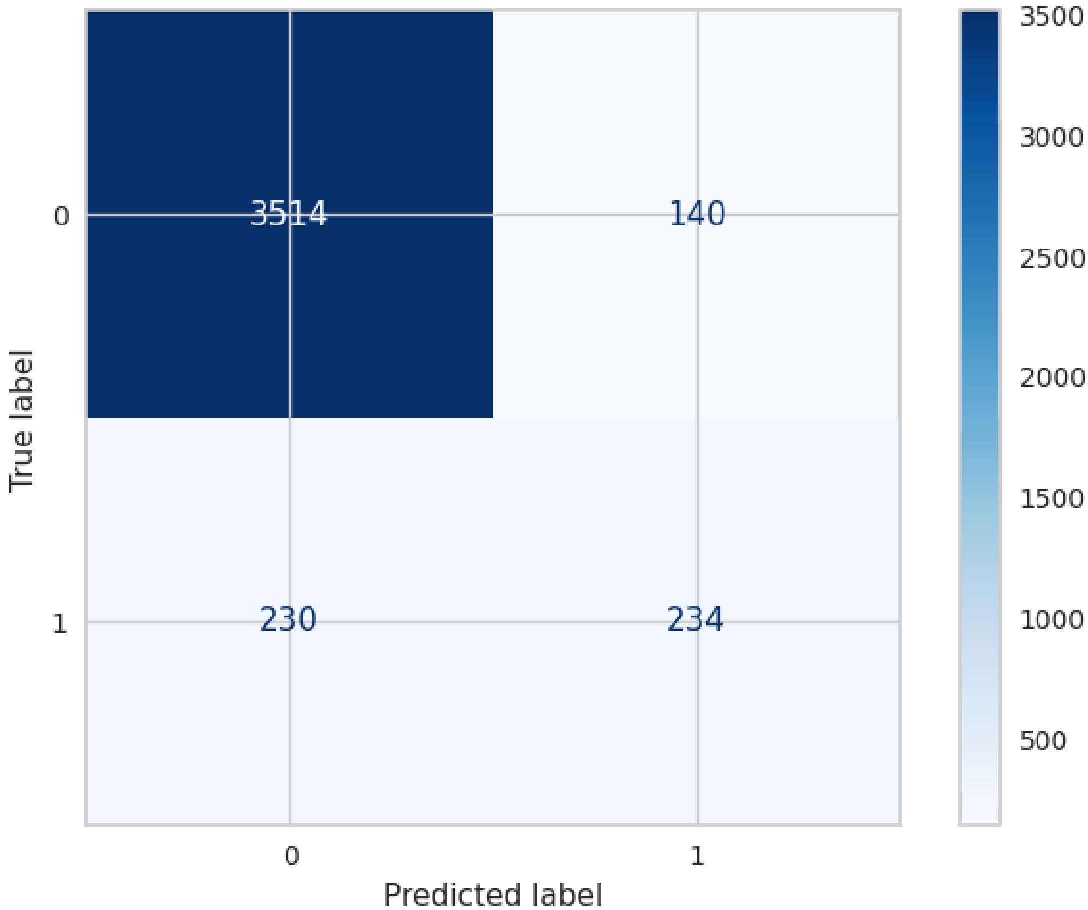


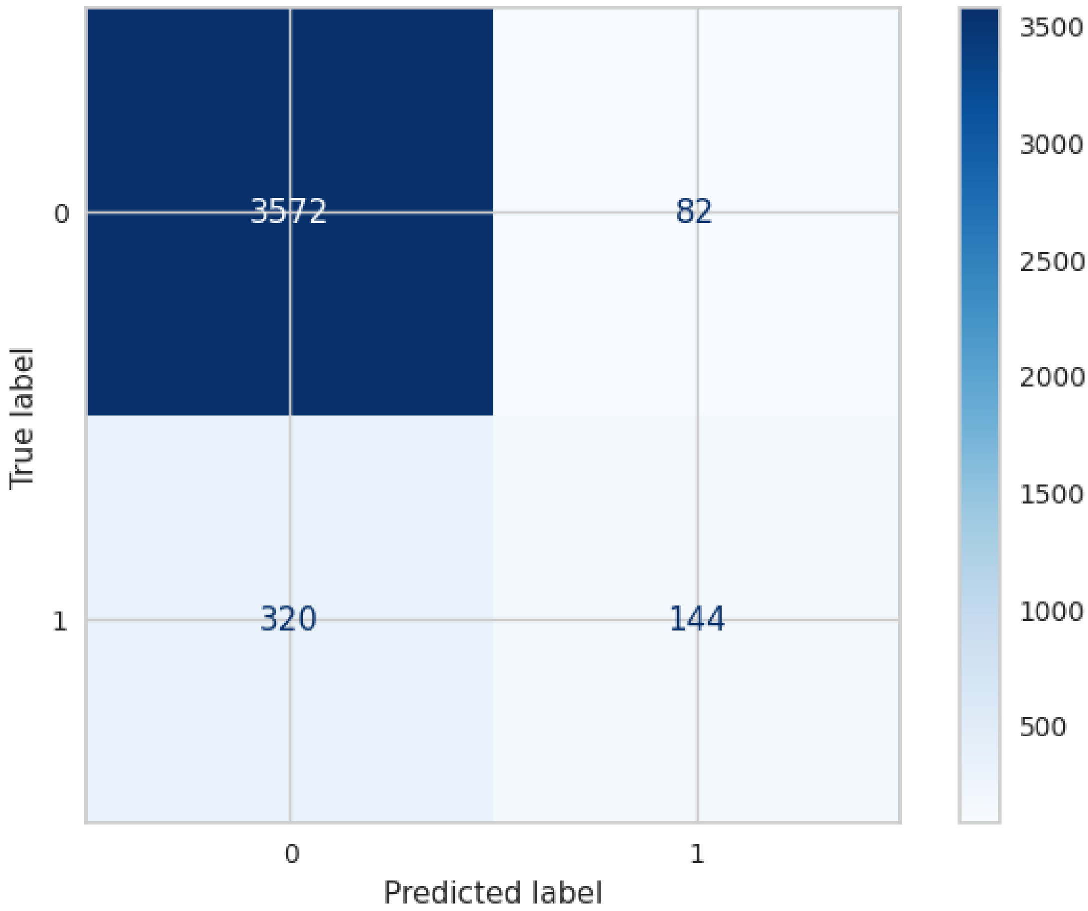

3516
138
247
217
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500

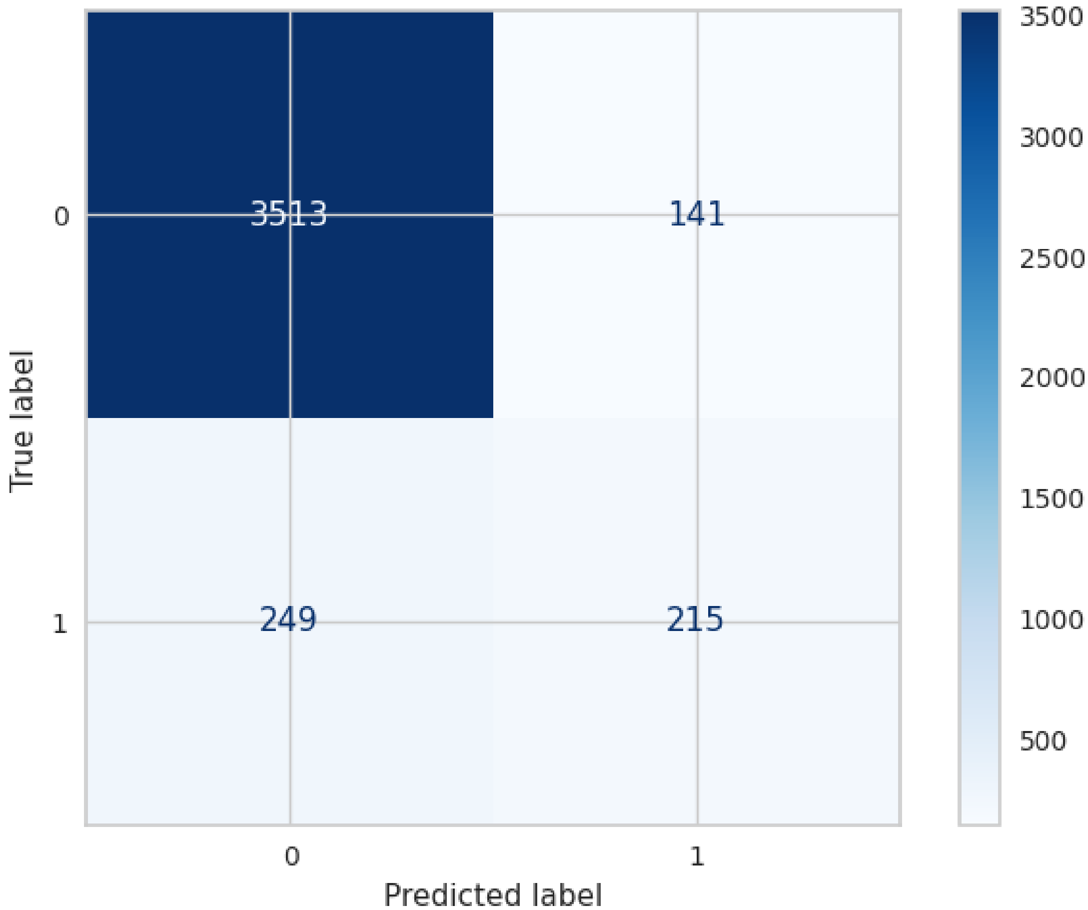


3572
82
320
144
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500


3513
141
249
215
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500


3514
140
230
234
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500


3572
82
320
144
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500



[247]:
```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
 ↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['KNN','KNN With Feature','KNN Scaling','KNN With␣
 ↪Normalize','KNN With PCA'
                ,'KNN With PCA and Scaling',
                'KNN With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[247]:
```
                            Train Accuracy  Test Accuracy   Test F1  \
Models
KNN                               0.919797       0.910879  0.562574
KNN With Feature                  0.916802       0.906508  0.529915
KNN Scaling                       0.914535       0.902380  0.417391
KNN With Normalize                0.918826       0.905294  0.524390
KNN With PCA                      0.920040       0.910151  0.558473
KNN With PCA and Scaling          0.914535       0.902380  0.417391
KNN With PCA and Normalize        0.918853       0.905294  0.524390
```

```
                                Test Recall  Test Precision       AUC
Models
KNN                                0.508621        0.629333  0.735290
KNN With Feature                   0.467672        0.611268  0.714953
KNN Scaling                        0.310345        0.637168  0.643952
KNN With Normalize                 0.463362        0.603933  0.712387
KNN With PCA                       0.504310        0.625668  0.732998
KNN With PCA and Scaling           0.310345        0.637168  0.643952
KNN With PCA and Normalize         0.463362        0.603933  0.712387
```

[248]:
```
models_draw(df)
```

RandomOverSampler

[249]:
```
X_train,y_train,X_test,y_test=Split(X_classification_over,y_classification_over)
```

```
X_train shape is  (65763, 20)
X_test shape is  (7307, 20)
y_train shape is  (65763,)
y_test shape is  (7307,)
```

[250]:
```
Search(KNeighborsClassifier(n_neighbors=3),{'n_neighbors':
↪[3,5,7,9,11]},X_train,y_train)
```

[250]:
```
KNeighborsClassifier(n_neighbors=3)
```

[251]:
```
cross_validation(KNeighborsClassifier(n_neighbors=3),X_train,y_train)
```

```
Train Score Value :  [0.96310587 0.96350504 0.96344801 0.96382886 0.96263139]
Mean 0.9633038340230147
Test Score Value :  [0.94062191 0.93818901 0.94161028 0.93917275 0.93970499]
Mean 0.9398597867822888
```

[252]:
```
Values =␣
↪Models(KNeighborsClassifier(n_neighbors=3),X_train,y_train,X_test,y_test)
```

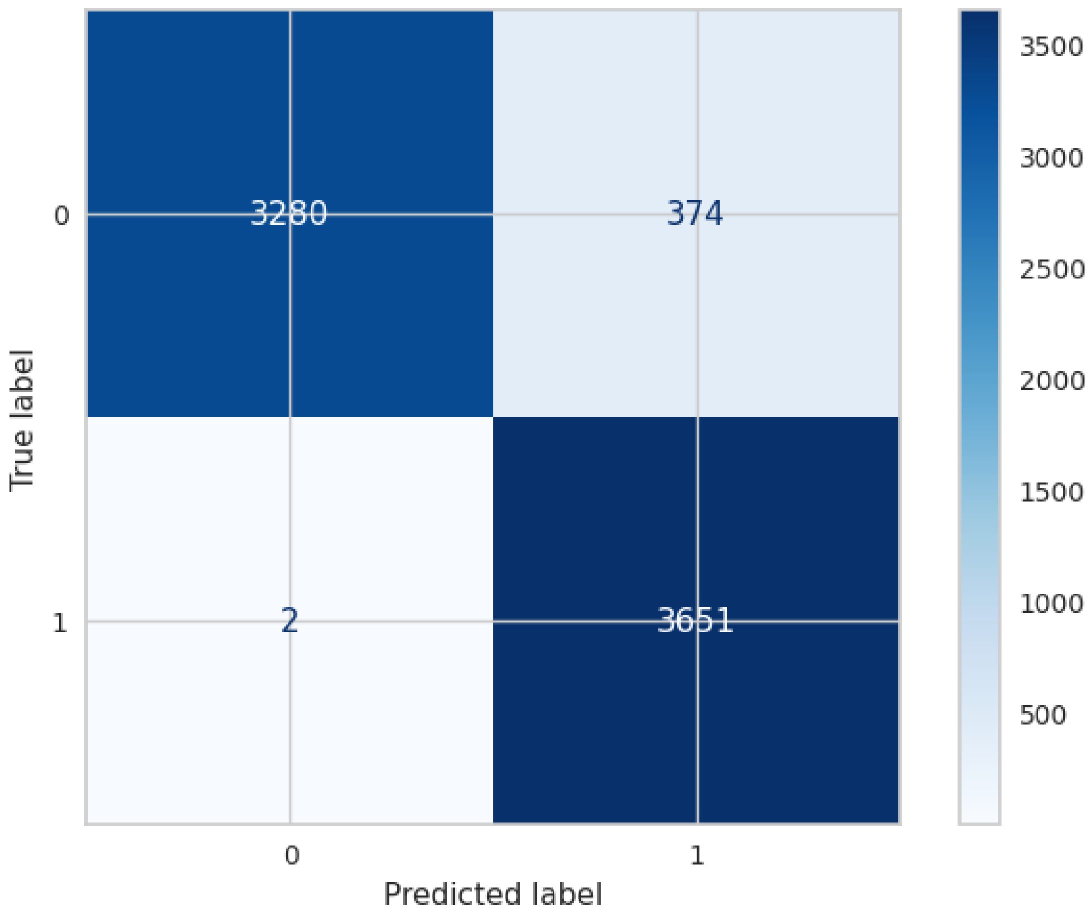
```
Apply Model With Normal Data :

Model Train Score is :  0.967595760534039
Model Test Score is :  0.9485424934993841
F1 Score is :  0.9510289137796301
Recall Score is :  0.9994525047905831
Precision Score is :  0.9070807453416149
AUC Value  :  0.9485494598391887

Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.90      0.95      3654
```

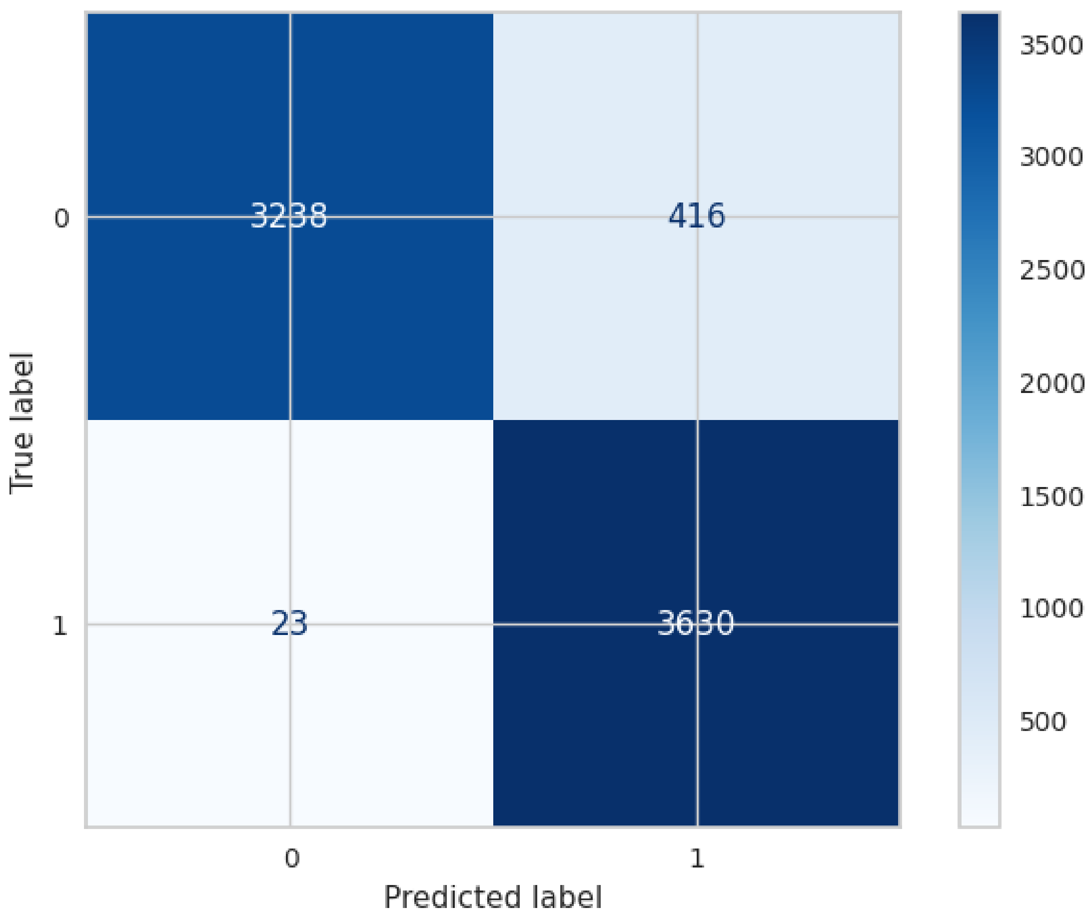
```
           1       0.91      1.00      0.95      3653

    accuracy                           0.95      7307
   macro avg       0.95      0.95      0.95      7307
weighted avg       0.95      0.95      0.95      7307

Confusion Matrix is :
 [[3280  374]
 [   2 3651]]


 Apply Model With Feature Selection :

Model Train Score is :  0.9595669297325243
Model Test Score is :  0.9399206240591214
F1 Score is :  0.9429796077412651
Recall Score is :  0.9937038050917054
Precision Score is :  0.8971824023727137
AUC Value  :  0.9399279835529681

Classification Report is :                precision    recall  f1-score
support

           0       0.99      0.89      0.94      3654
           1       0.90      0.99      0.94      3653

    accuracy                           0.94      7307
   macro avg       0.95      0.94      0.94      7307
weighted avg       0.95      0.94      0.94      7307

Confusion Matrix is :
 [[3238  416]
 [  23 3630]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.9663488587807734
Model Test Score is :  0.9482687833584235
F1 Score is :  0.9507940640458215
Recall Score is :  0.9997262523952916
Precision Score is :  0.9064283941424671
AUC Value  :  0.9482758246103442

Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.90      0.95      3654
           1       0.91      1.00      0.95      3653
```

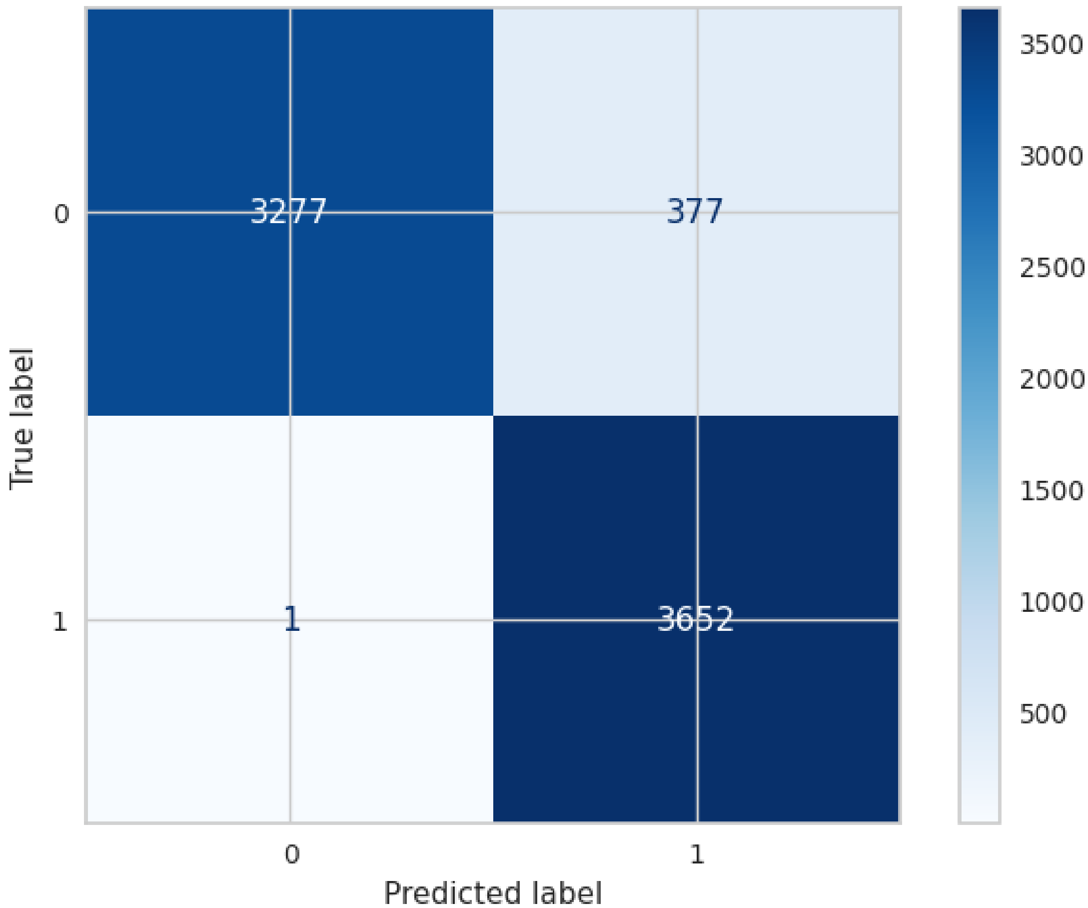
```
    accuracy                           0.95      7307
   macro avg       0.95      0.95      0.95      7307
weighted avg       0.95      0.95      0.95      7307

Confusion Matrix is :
 [[3277  377]
 [   1 3652]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.966379271018658
Model Test Score is :  0.948131928287943
F1 Score is :  0.9506703110764025
Recall Score is :  0.9997262523952916
Precision Score is :  0.9062034739454095
AUC Value  :  0.9481389882666114
Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.90      0.95      3654
           1       0.91      1.00      0.95      3653

    accuracy                           0.95      7307
   macro avg       0.95      0.95      0.95      7307
weighted avg       0.95      0.95      0.95      7307

Confusion Matrix is :
 [[3276  378]
 [   1 3652]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.9679911196265377
Model Test Score is :  0.9490899137813056
F1 Score is :  0.9515246286161062
Recall Score is :  0.9994525047905831
Precision Score is :  0.9079830887838846
AUC Value  :  0.9490968052141201
Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.90      0.95      3654
           1       0.91      1.00      0.95      3653
```

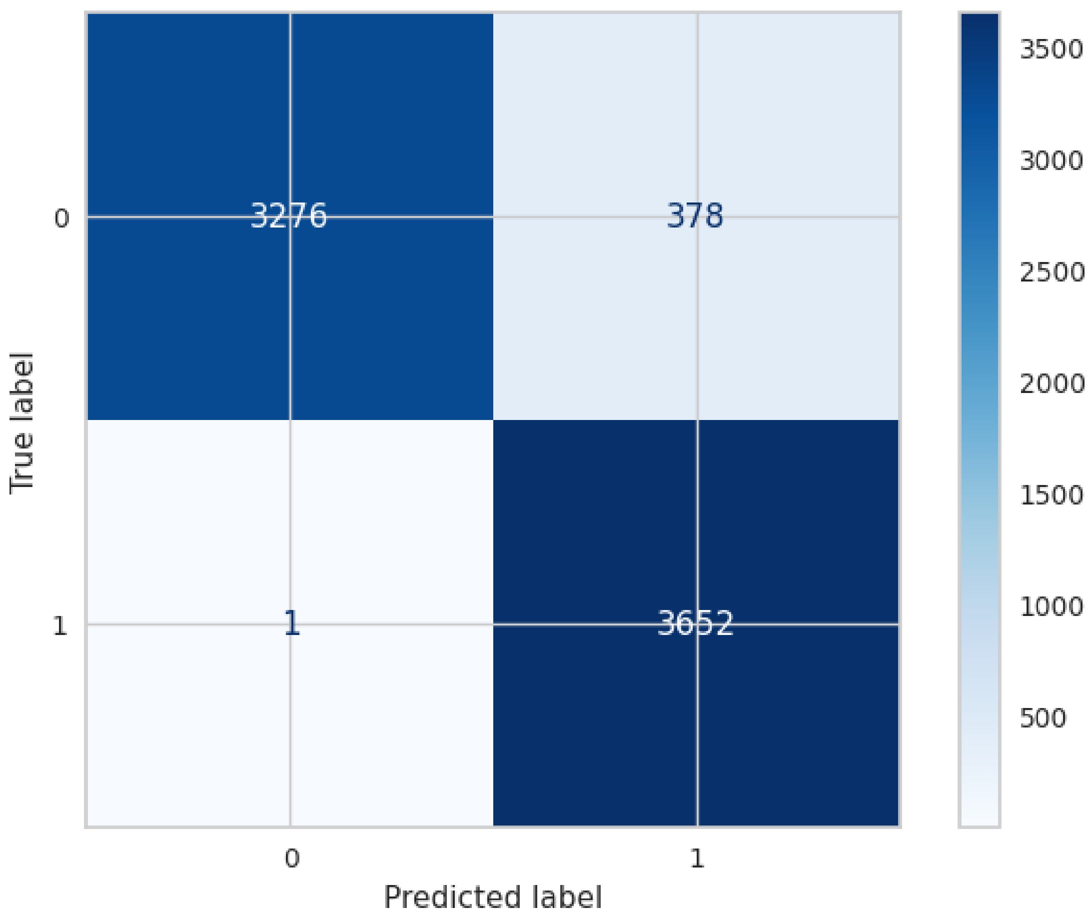
```
   accuracy                           0.95      7307
  macro avg       0.95      0.95      0.95      7307
weighted avg       0.95      0.95      0.95      7307

Confusion Matrix is :
 [[3284  370]
 [   2 3651]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  0.9663488587807734
Model Test Score is :  0.948131928287943
F1 Score is :  0.9506703110764025
Recall Score is :  0.9997262523952916
Precision Score is :  0.9062034739454095
AUC Value  :  0.9481389882666114
Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.90      0.95      3654
           1       0.91      1.00      0.95      3653

   accuracy                           0.95      7307
  macro avg       0.95      0.95      0.95      7307
weighted avg       0.95      0.95      0.95      7307

Confusion Matrix is :
 [[3276  378]
 [   1 3652]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  0.966379271018658
Model Test Score is :  0.948131928287943
F1 Score is :  0.9506703110764025
Recall Score is :  0.9997262523952916
Precision Score is :  0.9062034739454095
AUC Value  :  0.9481389882666114
Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.90      0.95      3654
           1       0.91      1.00      0.95      3653

   accuracy                           0.95      7307
```

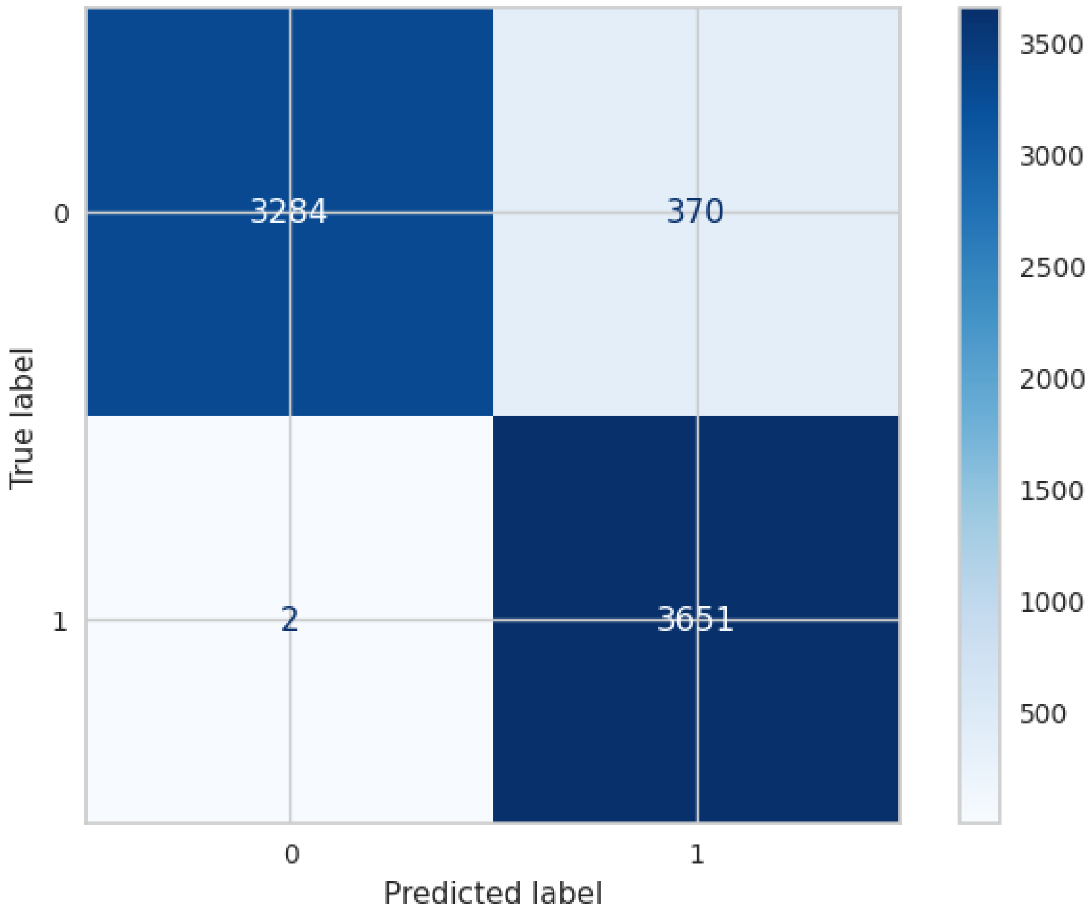
```
   macro avg       0.95      0.95      0.95      7307
weighted avg       0.95      0.95      0.95      7307

Confusion Matrix is :
 [[3276  378]
 [   1 3652]]
```



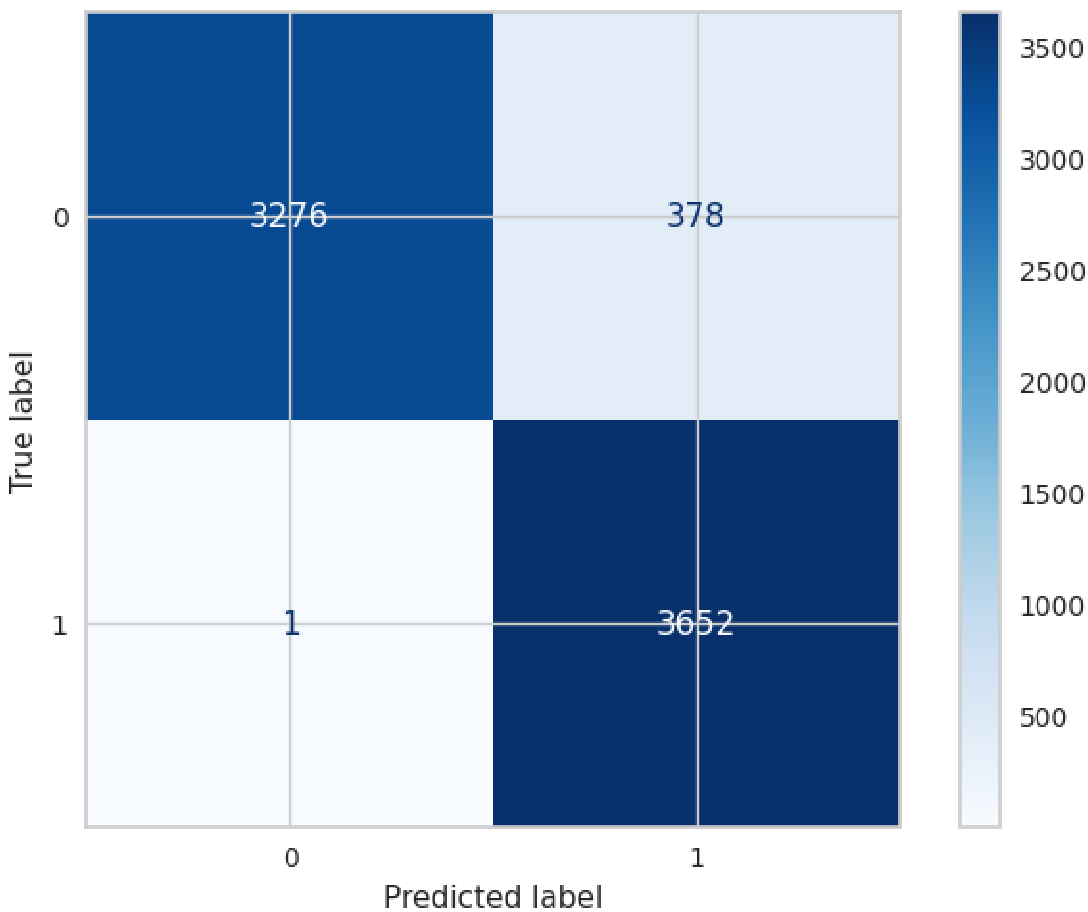

3238
416
23
3630
True label
0
1
Predicted label
0
1
3500
3000
2500
2000
1500
1000
500

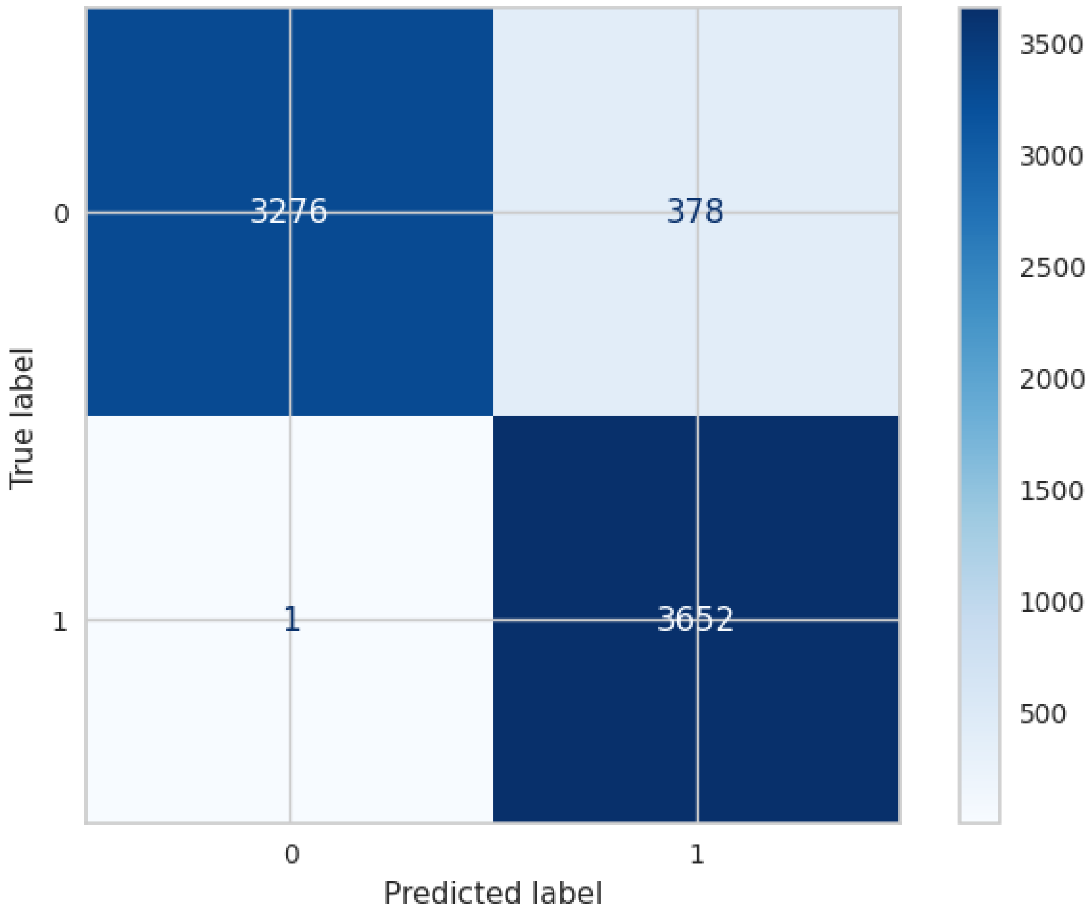

3277
377
1
3652
0
1
0
1
True label
Predicted label
3500
3000
2500
2000
1500
1000
500


3276
378
1
3652
True label
0
1
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500



3284
370
2
3651
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500


3276
378
1
3652
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500



[253]:
```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['KNN Over','KNN Over With Feature','KNN Over Scaling','KNN Over␣
↪With Normalize','KNN Over With PCA'
                ,'KNN Over With PCA and Scaling',
                'KNN Over With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[253]:
```
                                 Train Accuracy  Test Accuracy   Test F1  \
Models
KNN Over                               0.967596       0.948542  0.951029
KNN Over With Feature                  0.959567       0.939921  0.942980
KNN Over Scaling                       0.966349       0.948269  0.950794
KNN Over With Normalize                0.966379       0.948132  0.950670
KNN Over With PCA                      0.967991       0.949090  0.951525
KNN Over With PCA and Scaling          0.966349       0.948132  0.950670
KNN Over With PCA and Normalize        0.966379       0.948132  0.950670
```

```
                                   Test Recall  Test Precision       AUC
Models
KNN Over                              0.999453        0.907081  0.948549
KNN Over With Feature                 0.993704        0.897182  0.939928
KNN Over Scaling                      0.999726        0.906428  0.948276
KNN Over With Normalize               0.999726        0.906203  0.948139
KNN Over With PCA                     0.999453        0.907983  0.949097
KNN Over With PCA and Scaling         0.999726        0.906203  0.948139
KNN Over With PCA and Normalize       0.999726        0.906203  0.948139
```

[254]:
```
models_draw(df)
```

RandomUnderSampler

[255]:
```
X_train,y_train,X_test,y_test=Split(X_classification_under,y_classification_under)
```

```
X_train shape is  (8350, 20)
X_test shape is  (928, 20)
y_train shape is  (8350,)
y_test shape is  (928,)
```

[256]:
```
Search(KNeighborsClassifier(n_neighbors=3),{'n_neighbors':
 ↪[3,5,7,9,11]},X_train,y_train)
```

[256]:
```
KNeighborsClassifier(n_neighbors=7)
```

[257]:
```
cross_validation(KNeighborsClassifier(n_neighbors=11),X_train,y_train)
```

```
Train Score Value :  [0.87919162 0.87784431 0.88488024 0.88218563 0.88008982]
Mean 0.8808383233532935
Test Score Value :  [0.87305389 0.84491018 0.85508982 0.86047904 0.86946108]
Mean 0.8605988023952096
```

[258]:
```
Values =␣
 ↪Models(KNeighborsClassifier(n_neighbors=11),X_train,y_train,X_test,y_test)
```

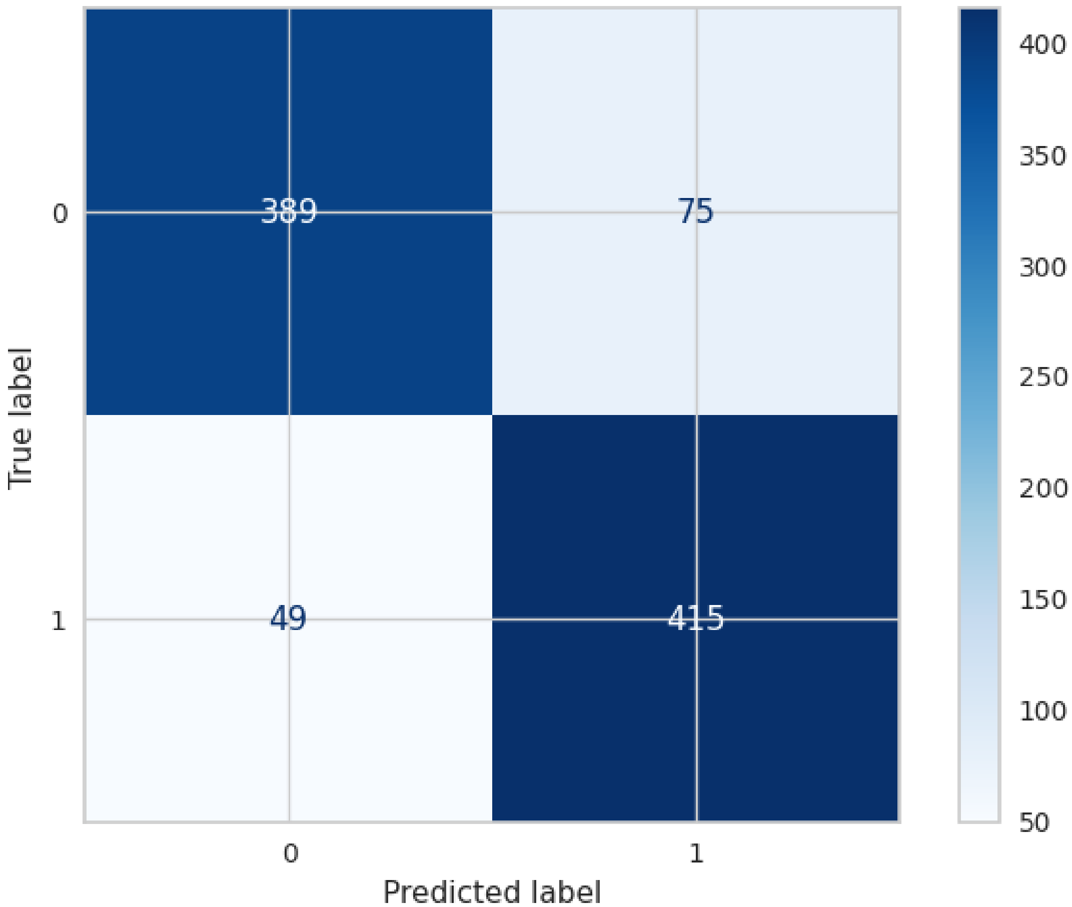
```
Apply Model With Normal Data :

Model Train Score is :  0.8832335329341318
Model Test Score is :  0.8663793103448276
F1 Score is :  0.870020964360587
Recall Score is :  0.8943965517241379
Precision Score is :  0.8469387755102041
AUC Value  :  0.8663793103448275

Classification Report is :               precision    recall  f1-score
support

           0       0.89      0.84      0.86       464
```

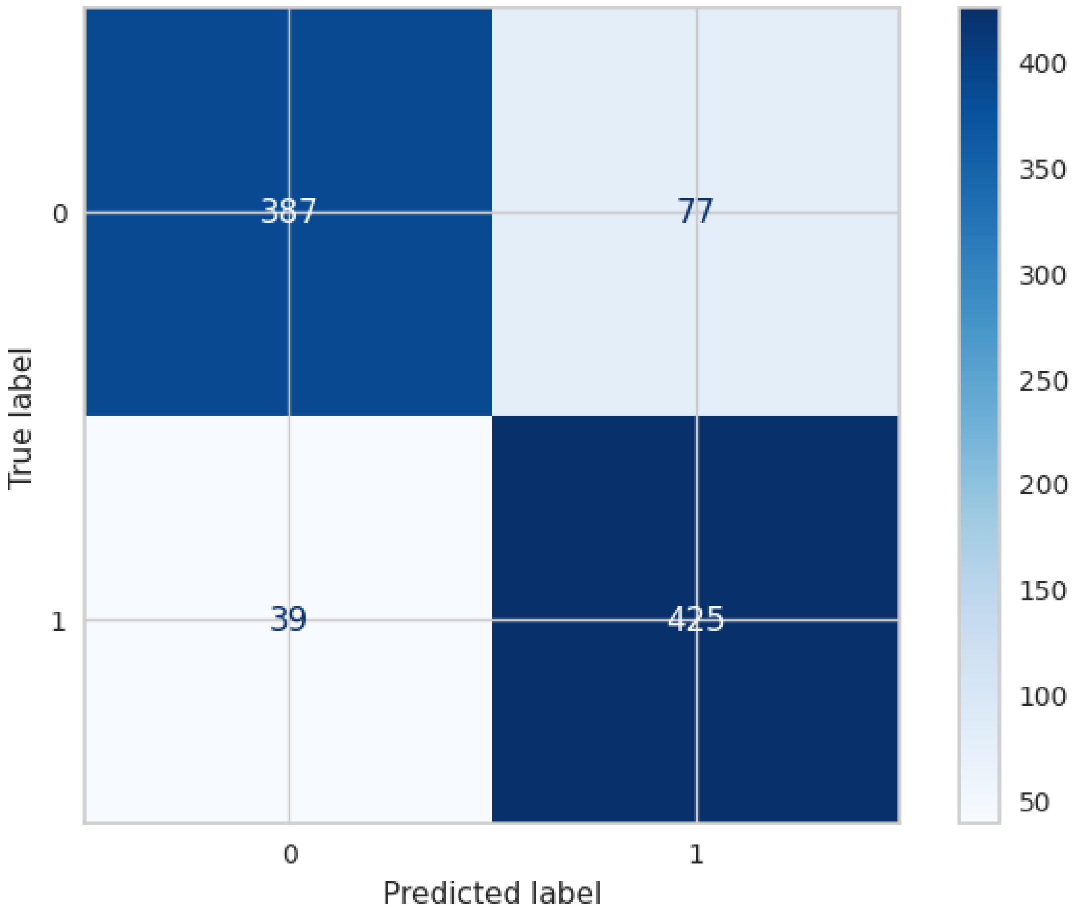
```
           1       0.85      0.89      0.87       464

    accuracy                           0.87       928
   macro avg       0.87      0.87      0.87       928
weighted avg       0.87      0.87      0.87       928

Confusion Matrix is :
 [[389  75]
 [ 49 415]]


 Apply Model With Feature Selection :

Model Train Score is :  0.8877844311377245
Model Test Score is :  0.875
F1 Score is :  0.8799171842650103
Recall Score is :  0.915948275862069
Precision Score is :  0.8466135458167331
AUC Value  :  0.875
Classification Report is :                precision    recall  f1-score
support

           0       0.91      0.83      0.87       464
           1       0.85      0.92      0.88       464

    accuracy                           0.88       928
   macro avg       0.88      0.88      0.87       928
weighted avg       0.88      0.88      0.87       928

Confusion Matrix is :
 [[387  77]
 [ 39 425]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.8708982035928143
Model Test Score is :  0.8588362068965517
F1 Score is :  0.8648090815273478
Recall Score is :  0.9030172413793104
Precision Score is :  0.8297029702970297
AUC Value  :  0.8588362068965518
Classification Report is :                precision    recall  f1-score
support

           0       0.89      0.81      0.85       464
           1       0.83      0.90      0.86       464
```

```
    accuracy                           0.86       928
   macro avg       0.86      0.86      0.86       928
weighted avg       0.86      0.86      0.86       928

Confusion Matrix is :
 [[378  86]
 [ 45 419]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.8779640718562874
Model Test Score is :  0.8609913793103449
F1 Score is :  0.8620320855614975
Recall Score is :  0.8685344827586207
Precision Score is :  0.8556263269639066
AUC Value  :  0.8609913793103448
Classification Report is :                precision    recall  f1-score
support

           0       0.87      0.85      0.86       464
           1       0.86      0.87      0.86       464

    accuracy                           0.86       928
   macro avg       0.86      0.86      0.86       928
weighted avg       0.86      0.86      0.86       928

Confusion Matrix is :
 [[396  68]
 [ 61 403]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.8833532934131737
Model Test Score is :  0.8674568965517241
F1 Score is :  0.8712041884816754
Recall Score is :  0.896551724137931
Precision Score is :  0.8472505091649695
AUC Value  :  0.8674568965517242
Classification Report is :                precision    recall  f1-score
support

           0       0.89      0.84      0.86       464
           1       0.85      0.90      0.87       464
```

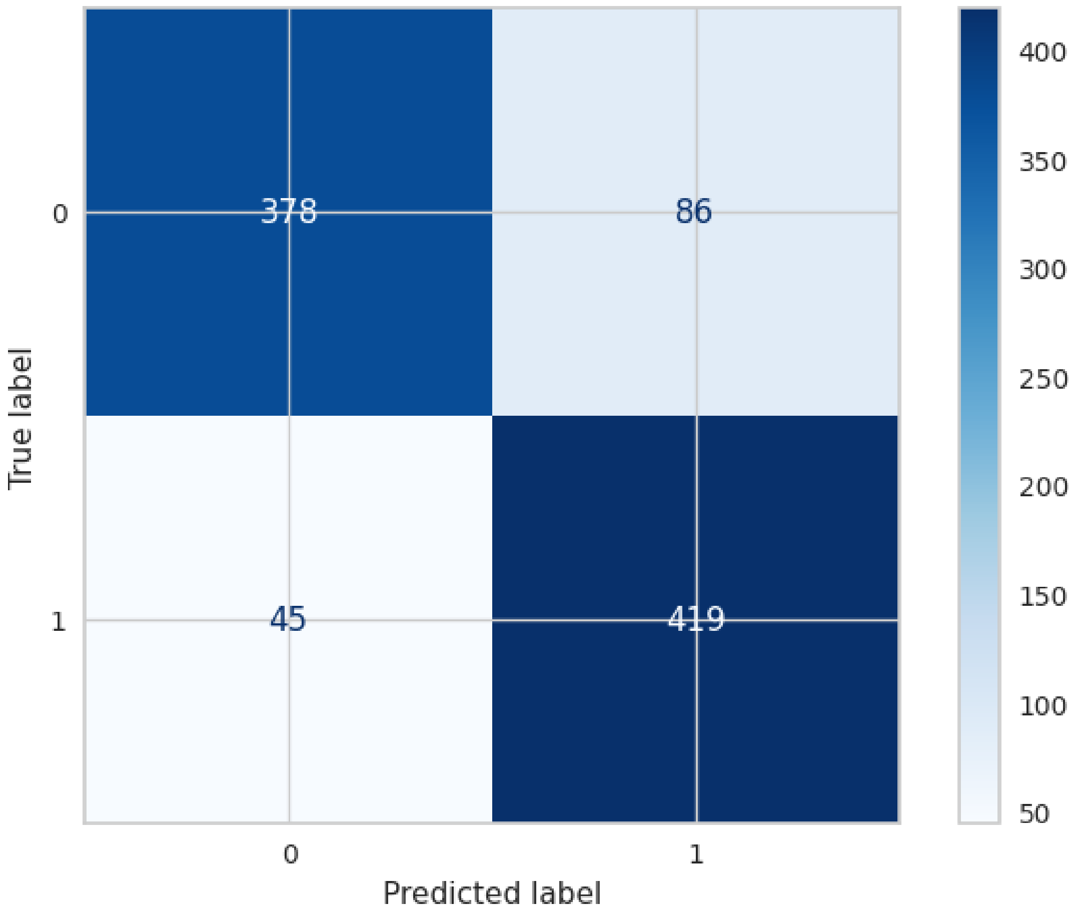
```
    accuracy                           0.87       928
   macro avg       0.87      0.87      0.87       928
weighted avg       0.87      0.87      0.87       928

Confusion Matrix is :
 [[389  75]
 [ 48 416]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  0.8708982035928143
Model Test Score is :  0.8588362068965517
F1 Score is :  0.8648090815273478
Recall Score is :  0.9030172413793104
Precision Score is :  0.8297029702970297
AUC Value  :  0.8588362068965518
Classification Report is :                precision    recall  f1-score
support

           0       0.89      0.81      0.85       464
           1       0.83      0.90      0.86       464

    accuracy                           0.86       928
   macro avg       0.86      0.86      0.86       928
weighted avg       0.86      0.86      0.86       928

Confusion Matrix is :
 [[378  86]
 [ 45 419]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  0.8779640718562874
Model Test Score is :  0.8609913793103449
F1 Score is :  0.8620320855614975
Recall Score is :  0.8685344827586207
Precision Score is :  0.8556263269639066
AUC Value  :  0.8609913793103448
Classification Report is :                precision    recall  f1-score
support

           0       0.87      0.85      0.86       464
           1       0.86      0.87      0.86       464

    accuracy                           0.86       928
```

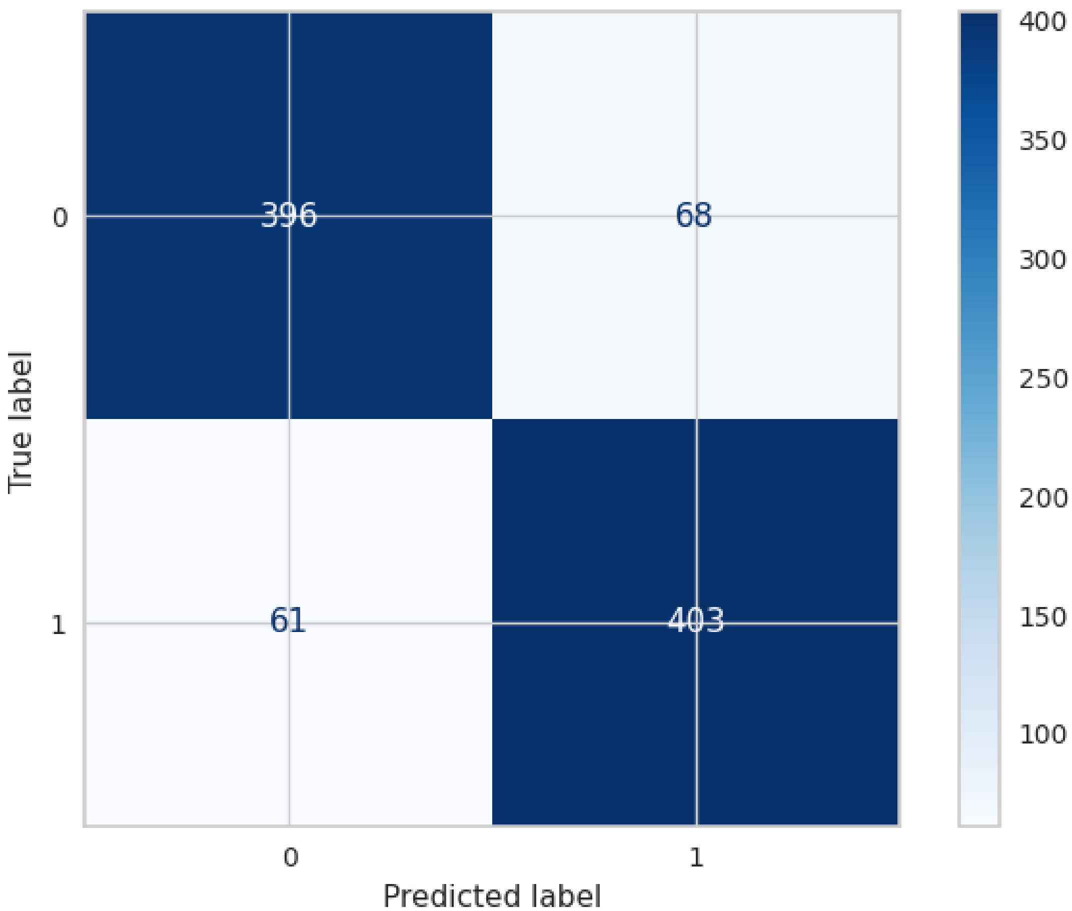
```
   macro avg       0.86      0.86      0.86       928
weighted avg       0.86      0.86      0.86       928

Confusion Matrix is :
 [[396  68]
 [ 61 403]]
```



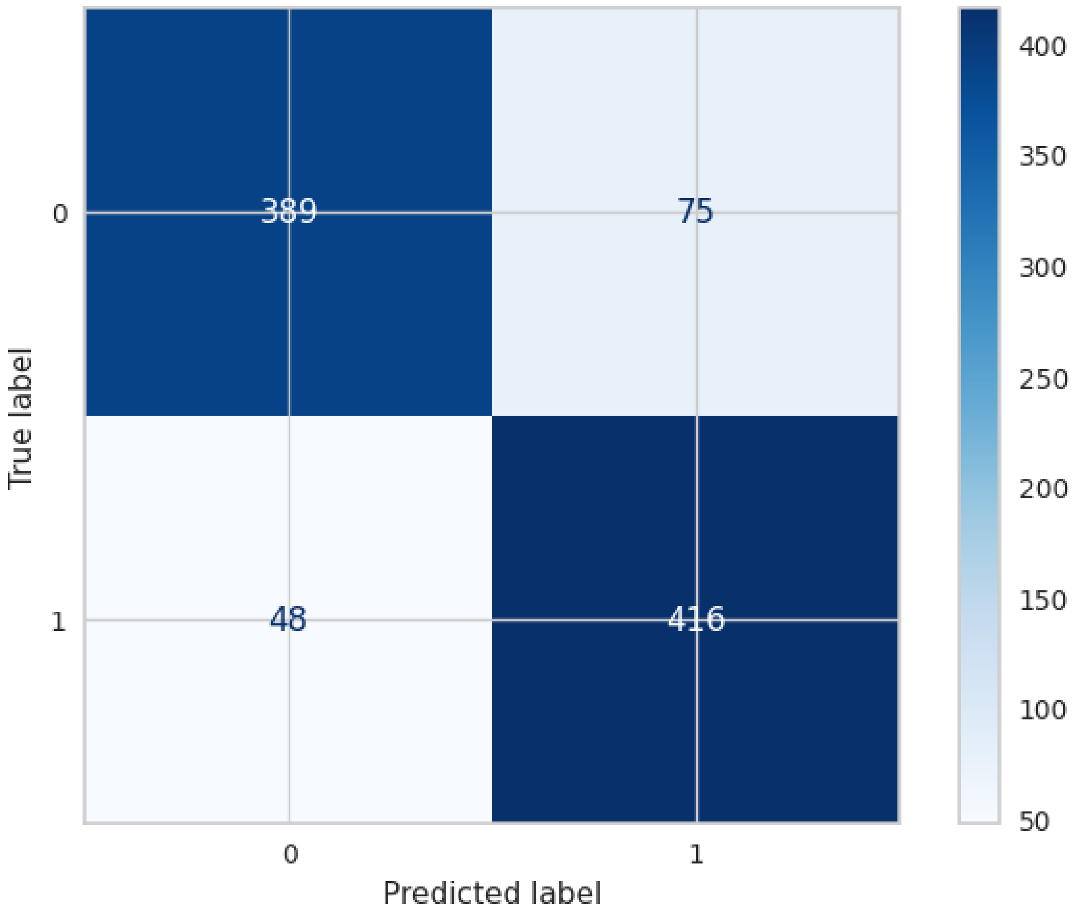


387
77
39
425
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50

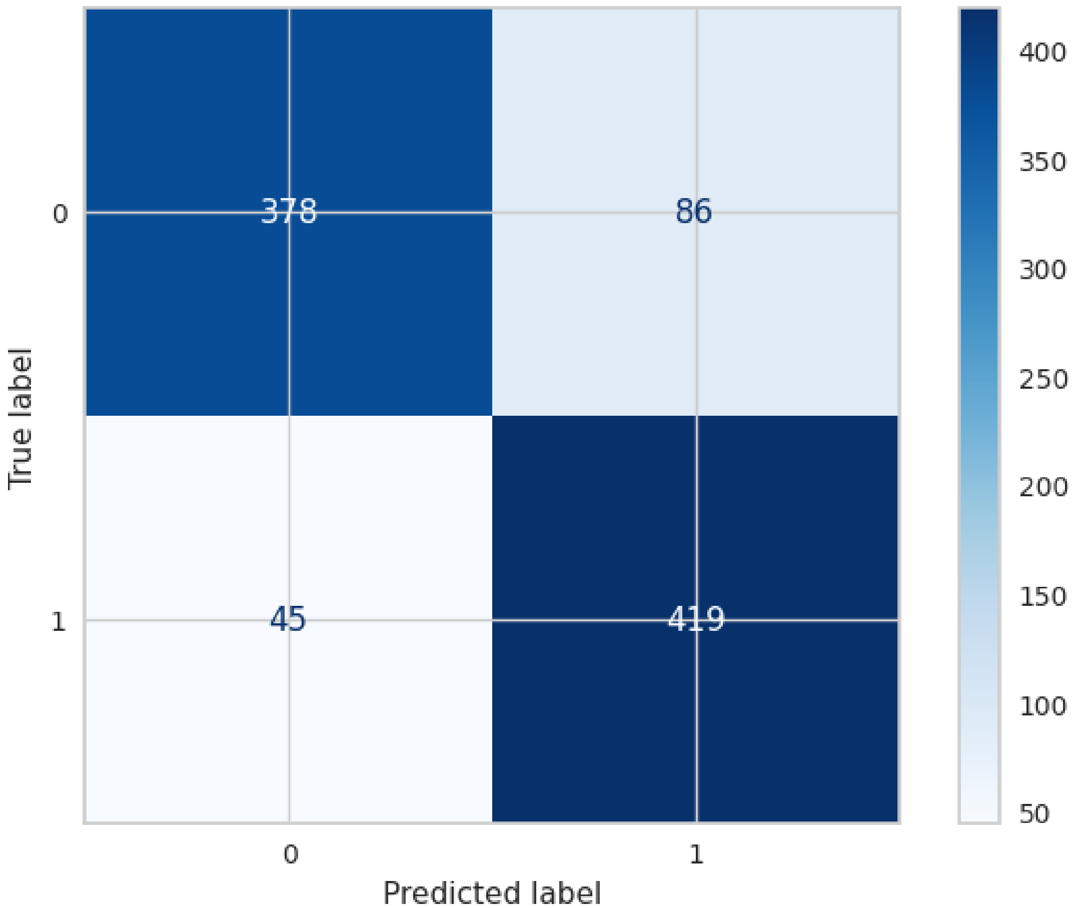

378
86
45
419
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50

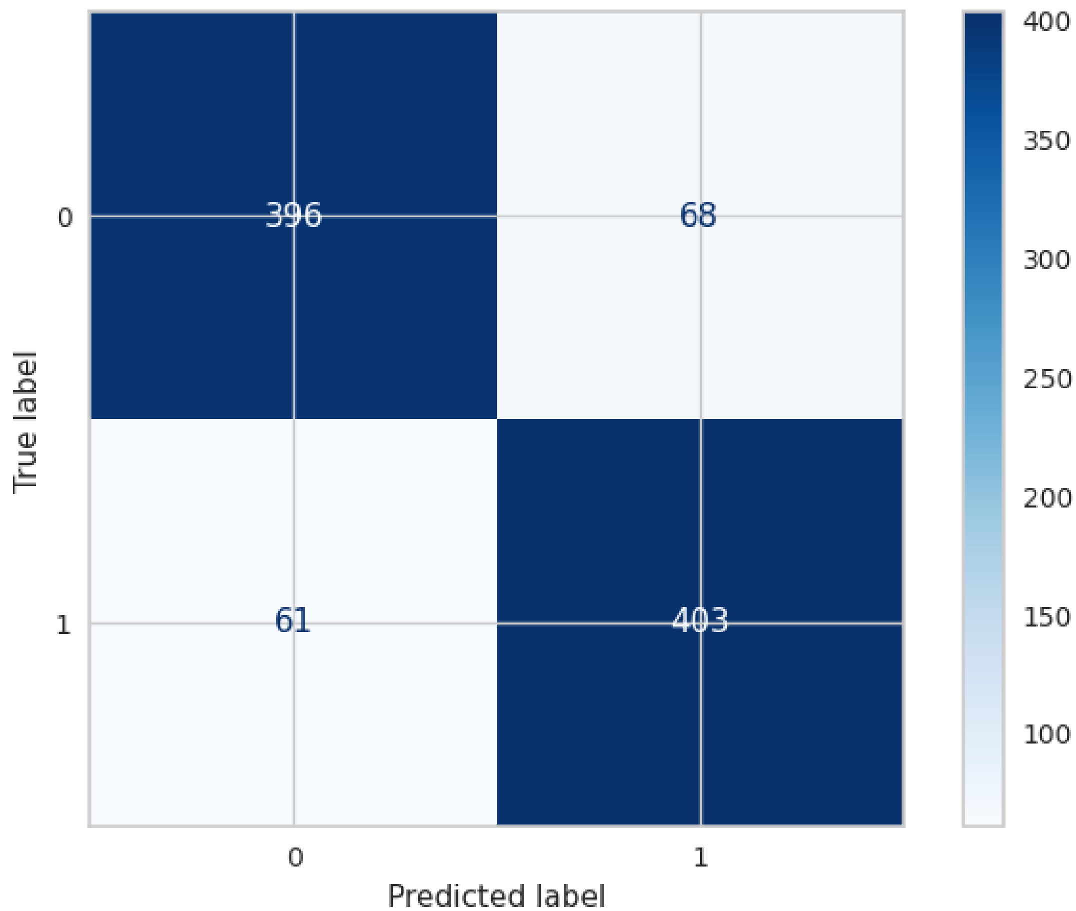


396
68
61
403
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100



389
75
48
416
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50


378
86
45
419
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50



[259]:
```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['KNN Under','KNN Under With Feature','KNN Under Scaling','KNN␣
↪Under With Normalize','KNN Under With PCA'
              ,'KNN Under With PCA and Scaling',
              'KNN Under With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[259]:
```
                                  Train Accuracy  Test Accuracy   Test F1  \
Models
KNN Under                               0.883234       0.866379  0.870021
KNN Under With Feature                  0.887784       0.875000  0.879917
KNN Under Scaling                       0.870898       0.858836  0.864809
KNN Under With Normalize                0.877964       0.860991  0.862032
KNN Under With PCA                      0.883353       0.867457  0.871204
KNN Under With PCA and Scaling          0.870898       0.858836  0.864809
KNN Under With PCA and Normalize        0.877964       0.860991  0.862032
```

```
                                    Test Recall  Test Precision       AUC
Models
KNN Under                              0.894397        0.846939  0.866379
KNN Under With Feature                 0.915948        0.846614  0.875000
KNN Under Scaling                      0.903017        0.829703  0.858836
KNN Under With Normalize               0.868534        0.855626  0.860991
KNN Under With PCA                     0.896552        0.847251  0.867457
KNN Under With PCA and Scaling         0.903017        0.829703  0.858836
KNN Under With PCA and Normalize       0.868534        0.855626  0.860991
```

[260]:
```
models_draw(df)
```

SVC

[261]:
```
X_train,y_train,X_test,y_test=Split(X_classification,y_classification)
```

```
X_train shape is  (37056, 20)
X_test shape is  (4118, 20)
y_train shape is  (37056,)
y_test shape is  (4118,)
```

[262]:
```
Search(SVC(kernel= 'rbf',max_iter=1000,C=1.0),{'C':[1,.
 ↪5,2,3,5,10]},X_train,y_train)
```

[262]:
```
SVC(C=0.5, max_iter=1000)
```

[263]:
```
cross_validation(SVC(kernel= 'rbf',max_iter=1000,C=.5),X_train,y_train)
```

```
Train Score Value :  [0.79624882 0.31151965 0.88257716 0.85093608 0.85710912]
Mean 0.7396781666305908
Test Score Value :  [0.79803022 0.30576171 0.87424099 0.85062745 0.86614492]
Mean 0.7389610570713463
```

[264]:
```
Values = Models(SVC(kernel= 'rbf',max_iter=1000,C=.
 ↪5),X_train,y_train,X_test,y_test)
```

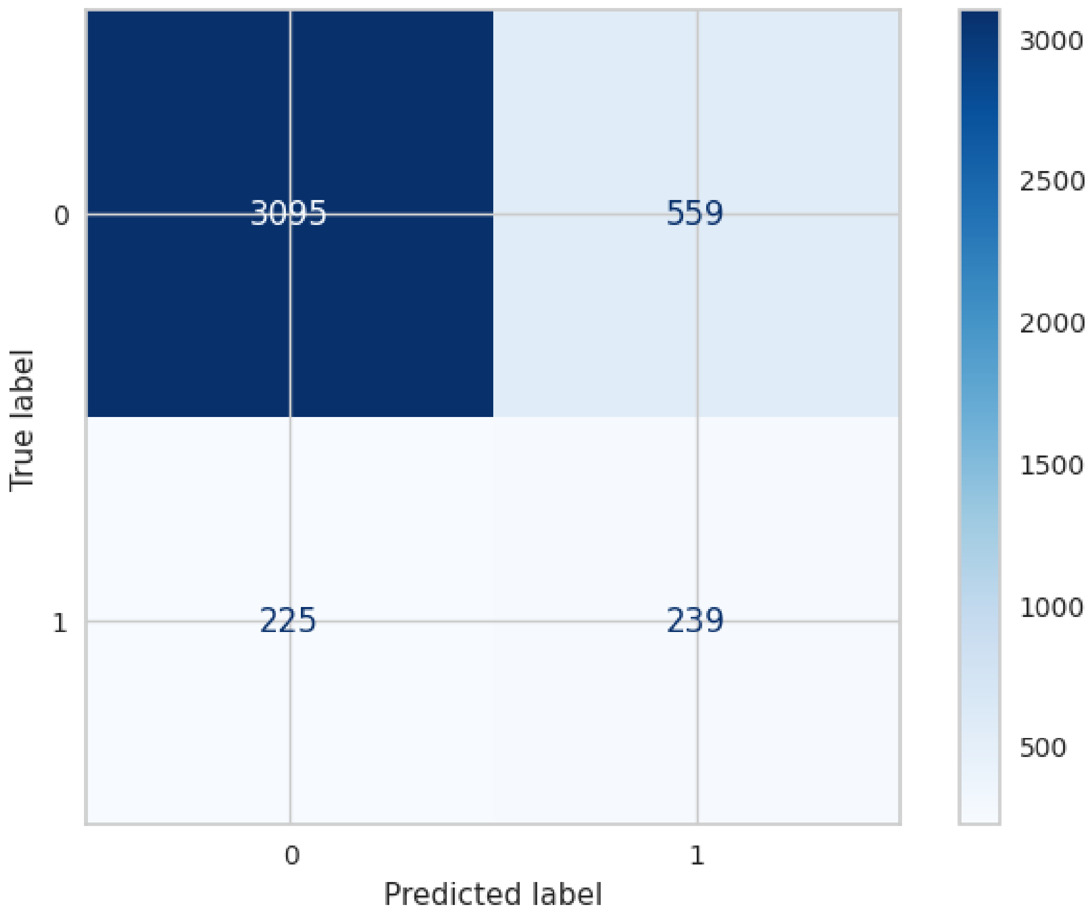
```
Apply Model With Normal Data :

Model Train Score is :  0.8246707685664939
Model Test Score is :  0.8096163186012627
F1 Score is :  0.3787638668779715
Recall Score is :  0.5150862068965517
Precision Score is :  0.29949874686716793
AUC Value  :  0.6810515873015872

Classification Report is :                precision    recall  f1-score
support

           0       0.93      0.85      0.89      3654
```

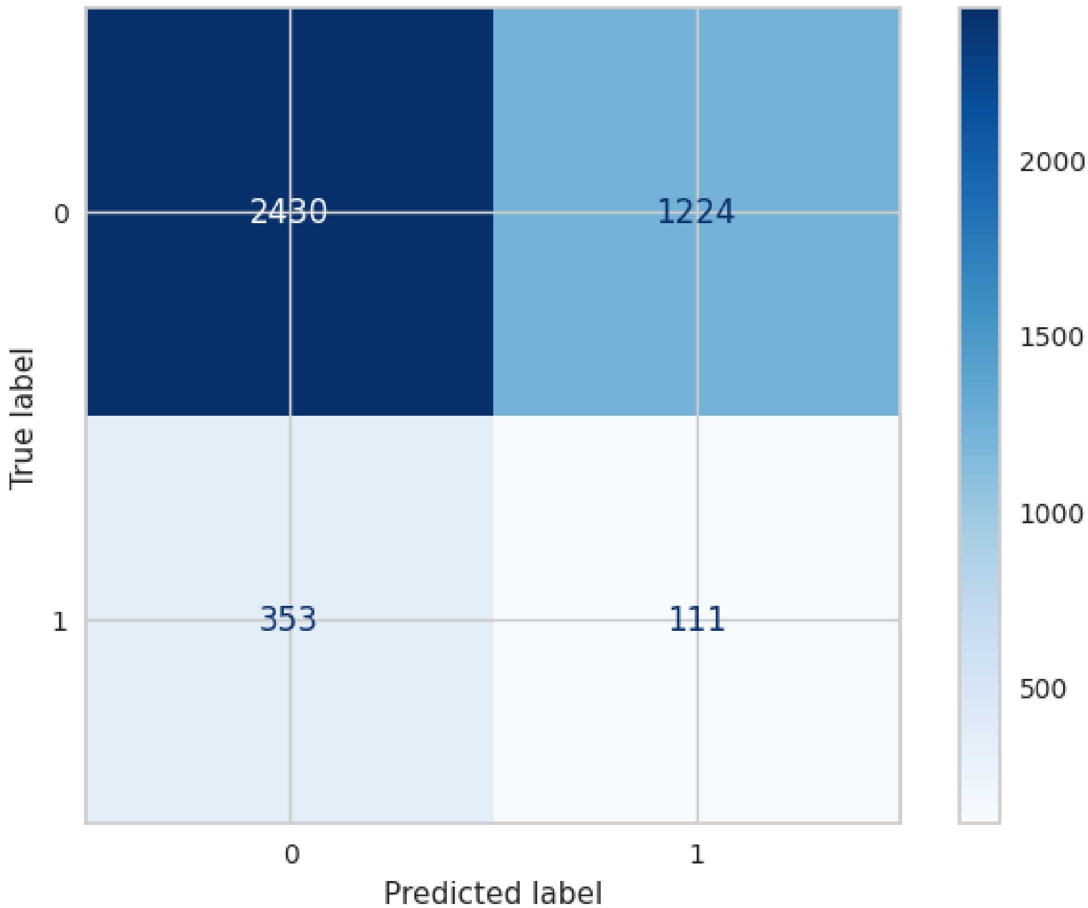
```
           1       0.30      0.52      0.38       464

    accuracy                           0.81      4118
   macro avg       0.62      0.68      0.63      4118
weighted avg       0.86      0.81      0.83      4118

Confusion Matrix is :
 [[3095  559]
 [ 225  239]]


 Apply Model With Feature Selection :

Model Train Score is :  0.6215997409326425
Model Test Score is :  0.6170471102476931
F1 Score is :  0.12340188993885493
Recall Score is :  0.23922413793103448
Precision Score is :  0.08314606741573034
AUC Value  :  0.4521243842364532

Classification Report is :                precision    recall  f1-score
support

           0       0.87      0.67      0.76      3654
           1       0.08      0.24      0.12       464

    accuracy                           0.62      4118
   macro avg       0.48      0.45      0.44      4118
weighted avg       0.78      0.62      0.68      4118

Confusion Matrix is :
 [[2430 1224]
 [ 353  111]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.8245628238341969
Model Test Score is :  0.8300145701796989
F1 Score is :  0.38488576449912126
Recall Score is :  0.47198275862068967
Precision Score is :  0.3249258160237389
AUC Value  :  0.6737308429118776

Classification Report is :                precision    recall  f1-score
support

           0       0.93      0.88      0.90      3654
           1       0.32      0.47      0.38       464
```

```
    accuracy                           0.83      4118
   macro avg       0.63      0.67      0.64      4118
weighted avg       0.86      0.83      0.84      4118

Confusion Matrix is :
 [[3199  455]
 [ 245  219]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.6014950345423143
Model Test Score is :  0.59834871296746
F1 Score is :  0.3508634222919937
Recall Score is :  0.9633620689655172
Precision Score is :  0.21449136276391556
AUC Value  :  0.7576799397920089
Classification Report is :                precision    recall  f1-score
support

           0       0.99      0.55      0.71      3654
           1       0.21      0.96      0.35       464

    accuracy                           0.60      4118
   macro avg       0.60      0.76      0.53      4118
weighted avg       0.90      0.60      0.67      4118

Confusion Matrix is :
 [[2017 1637]
 [  17  447]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.2692681347150259
Model Test Score is :  0.2799902865468674
F1 Score is :  0.15164520743919882
Recall Score is :  0.5711206896551724
Precision Score is :  0.08742989112504124
AUC Value  :  0.40707101806239737
Classification Report is :                precision    recall  f1-score
support

           0       0.82      0.24      0.37      3654
           1       0.09      0.57      0.15       464
```

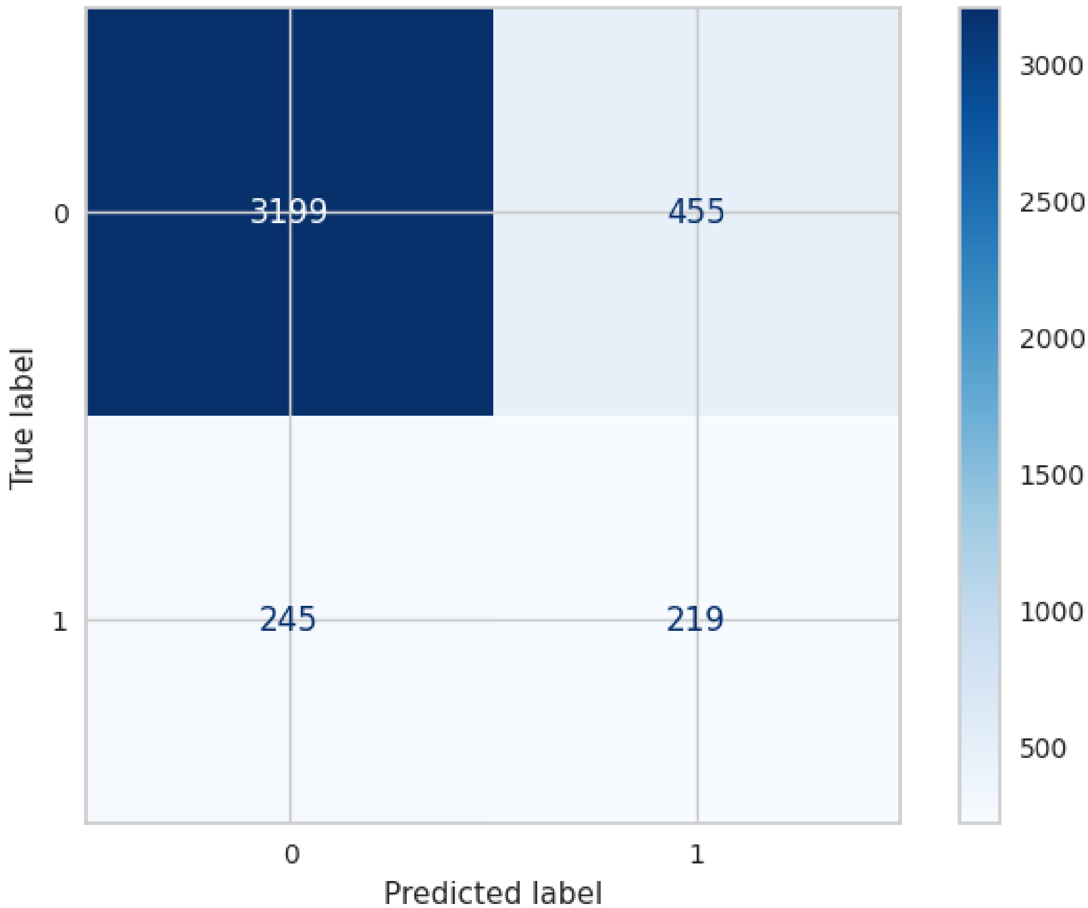
```
    accuracy                           0.28      4118
   macro avg       0.45      0.41      0.26      4118
weighted avg       0.73      0.28      0.35      4118

Confusion Matrix is :
 [[ 888 2766]
 [ 199  265]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  0.8845531088082902
Model Test Score is :  0.8841670713938805
F1 Score is :  0.49201277955271566
Recall Score is :  0.4978448275862069
Precision Score is :  0.4863157894736842
AUC Value  :  0.7155343459222769

Classification Report is :                precision    recall  f1-score
support

           0       0.94      0.93      0.93      3654
           1       0.49      0.50      0.49       464

    accuracy                           0.88      4118
   macro avg       0.71      0.72      0.71      4118
weighted avg       0.89      0.88      0.88      4118

Confusion Matrix is :
 [[3410  244]
 [ 233  231]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  0.7518890328151986
Model Test Score is :  0.7467217095677513
F1 Score is :  0.43834141087775985
Recall Score is :  0.8771551724137931
Precision Score is :  0.2921751615218952
AUC Value  :  0.8036569512862617

Classification Report is :                precision    recall  f1-score
support

           0       0.98      0.73      0.84      3654
           1       0.29      0.88      0.44       464

    accuracy                           0.75      4118
```

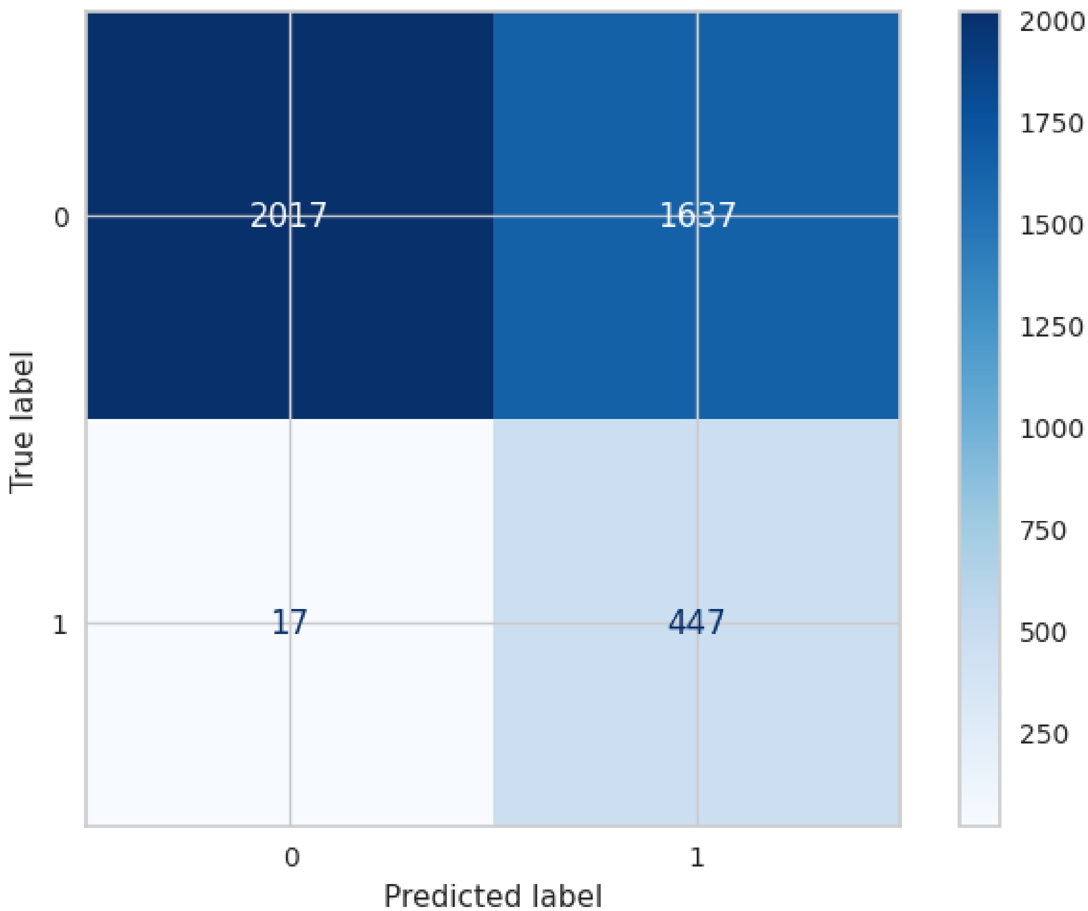
```
   macro avg       0.64      0.80      0.64      4118
weighted avg       0.90      0.75      0.79      4118

Confusion Matrix is :
 [[2668  986]
 [  57  407]]
```

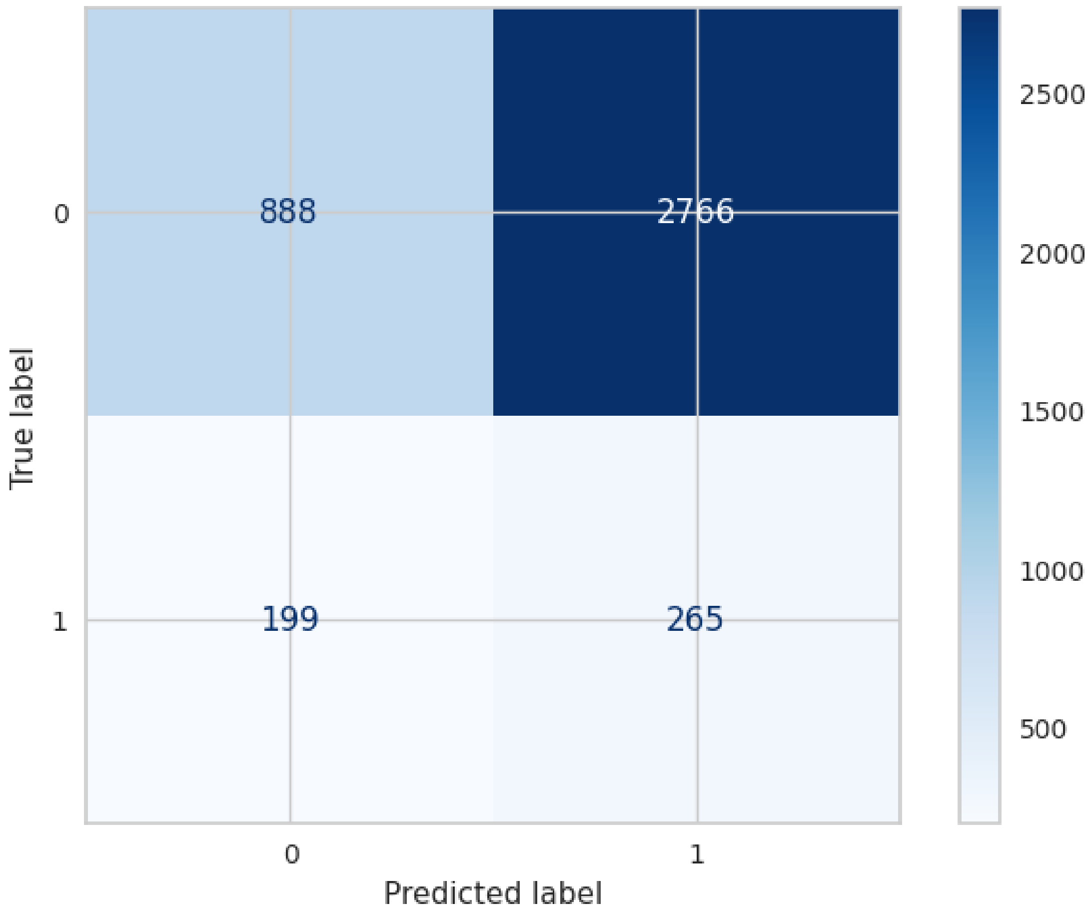




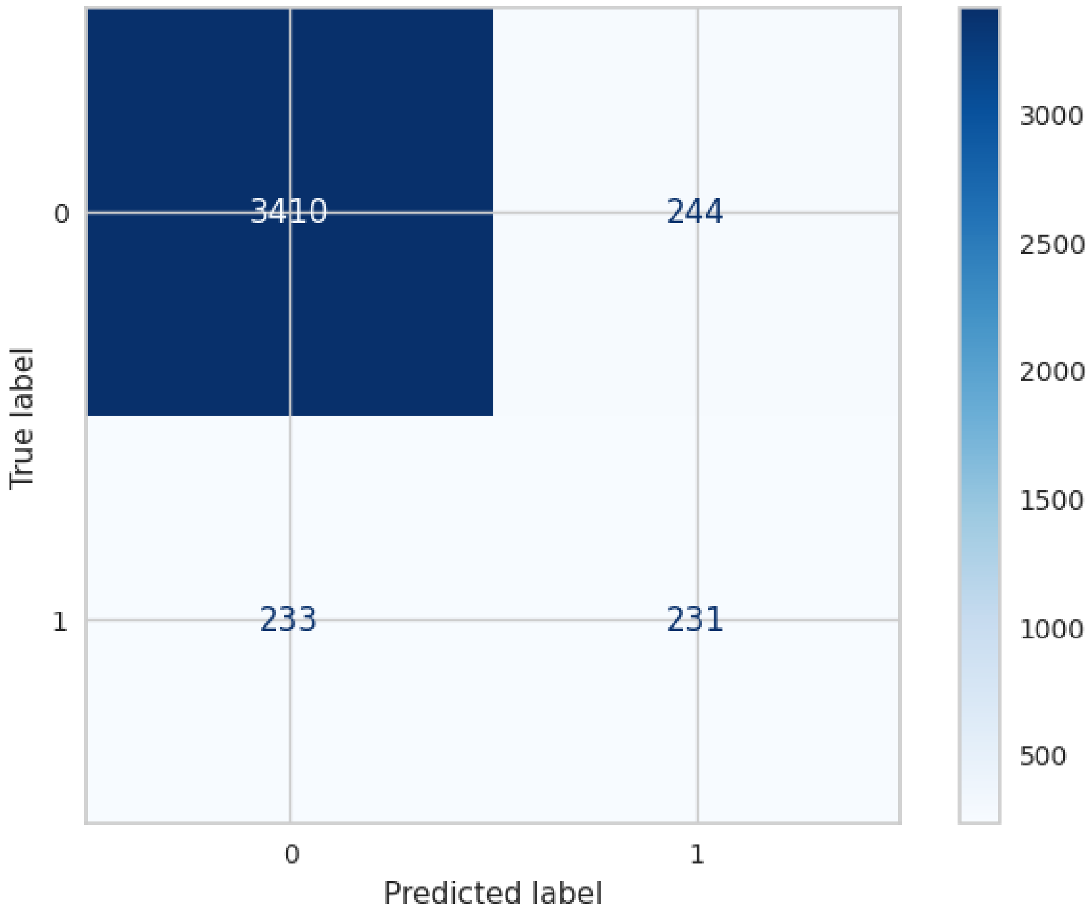
2430
1224
353
111
True label
0
1
Predicted label
2000
1500
1000
500

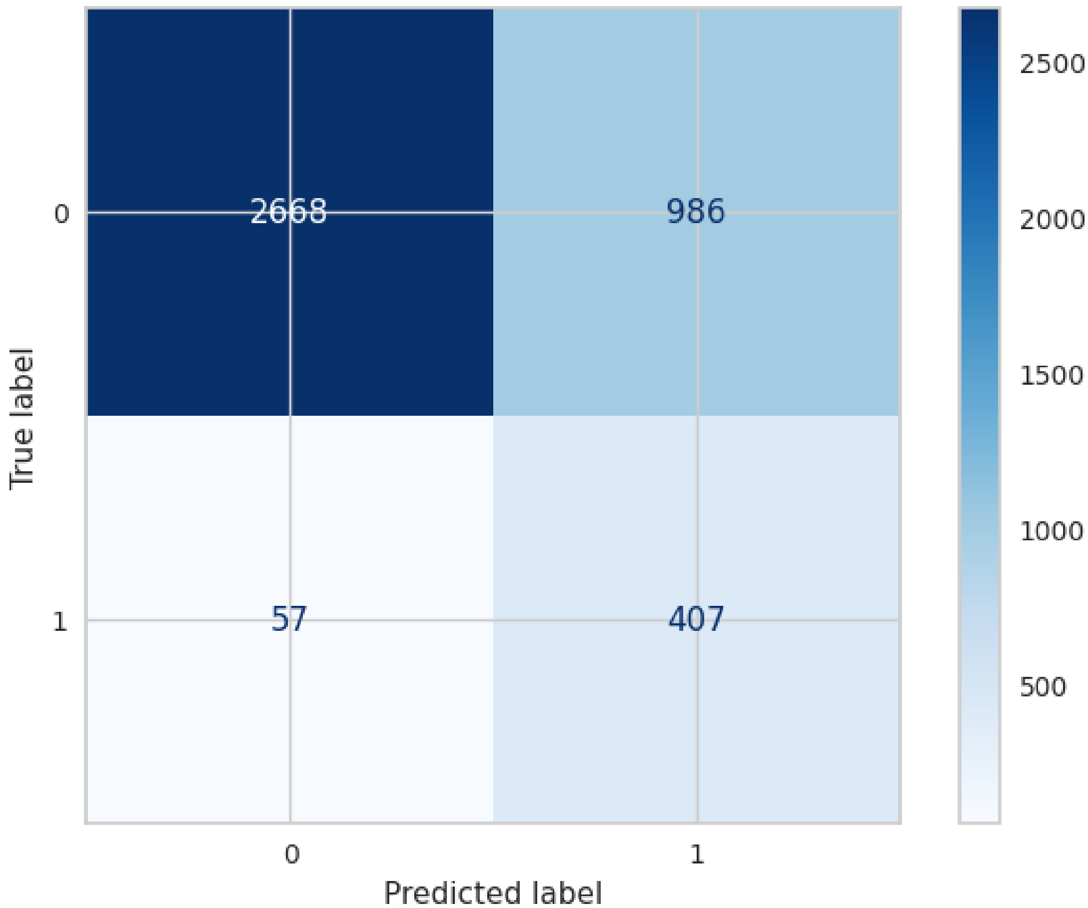

3199
455
245
219
0
1
True label
0
1
Predicted label
3000
2500
2000
1500
1000
500


2017
1637
17
447
0
1
True label
0
1
Predicted label
2000
1750
1500
1250
1000
750
500
250



888
2766
199
265
0
1
True label
0
1
Predicted label
2500
2000
1500
1000
500


3000
2500
2000
1500
1000
500
3410
244
233
231
0
1
True label
0
1
Predicted label



[265]:
```python
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['SVC','SVC With Feature','SVC Scaling','SVC With␣
↪Normalize','SVC With PCA'
                ,'SVC With PCA and Scaling',
                'SVC With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[265]:

Models	Train Accuracy	Test Accuracy	Test F1 \
SVC	0.824671	0.809616	0.378764
SVC With Feature	0.621600	0.617047	0.123402
SVC Scaling	0.824563	0.830015	0.384886
SVC With Normalize	0.601495	0.598349	0.350863
SVC With PCA	0.269268	0.279990	0.151645
SVC With PCA and Scaling	0.884553	0.884167	0.492013
SVC With PCA and Normalize	0.751889	0.746722	0.438341

```
                             Test Recall  Test Precision       AUC
Models
SVC                             0.515086        0.299499  0.681052
SVC With Feature                0.239224        0.083146  0.452124
SVC Scaling                     0.471983        0.324926  0.673731
SVC With Normalize              0.963362        0.214491  0.757680
SVC With PCA                    0.571121        0.087430  0.407071
SVC With PCA and Scaling        0.497845        0.486316  0.715534
SVC With PCA and Normalize      0.877155        0.292175  0.803657
```

[266]:
```
models_draw(df)
```

RandomOverSampler

[267]:
```
X_train,y_train,X_test,y_test=Split(X_classification_over,y_classification_over)
```

```
X_train shape is  (65763, 20)
X_test shape is  (7307, 20)
y_train shape is  (65763,)
y_test shape is  (7307,)
```

[268]:
```
Search(SVC(kernel= 'rbf',max_iter=1000,C=1.0),{'C':[1,.
↪5,2,3,5,10]},X_train,y_train)
```

[268]:
```
SVC(C=1, max_iter=1000)
```

[269]:
```
cross_validation(SVC(kernel= 'rbf',max_iter=1000,C=.5),X_train,y_train)
```

```
Train Score Value :  [0.81094849 0.61473104 0.59884052 0.57491779 0.53642774]
Mean 0.6271731178652876
Test Score Value :  [0.81228617 0.6121037  0.59971109 0.57694647 0.52851277]
Mean 0.6259120422602561
```

[270]:
```
Values = Models(SVC(kernel= 'rbf',max_iter=1000,C=.
↪5),X_train,y_train,X_test,y_test)
```

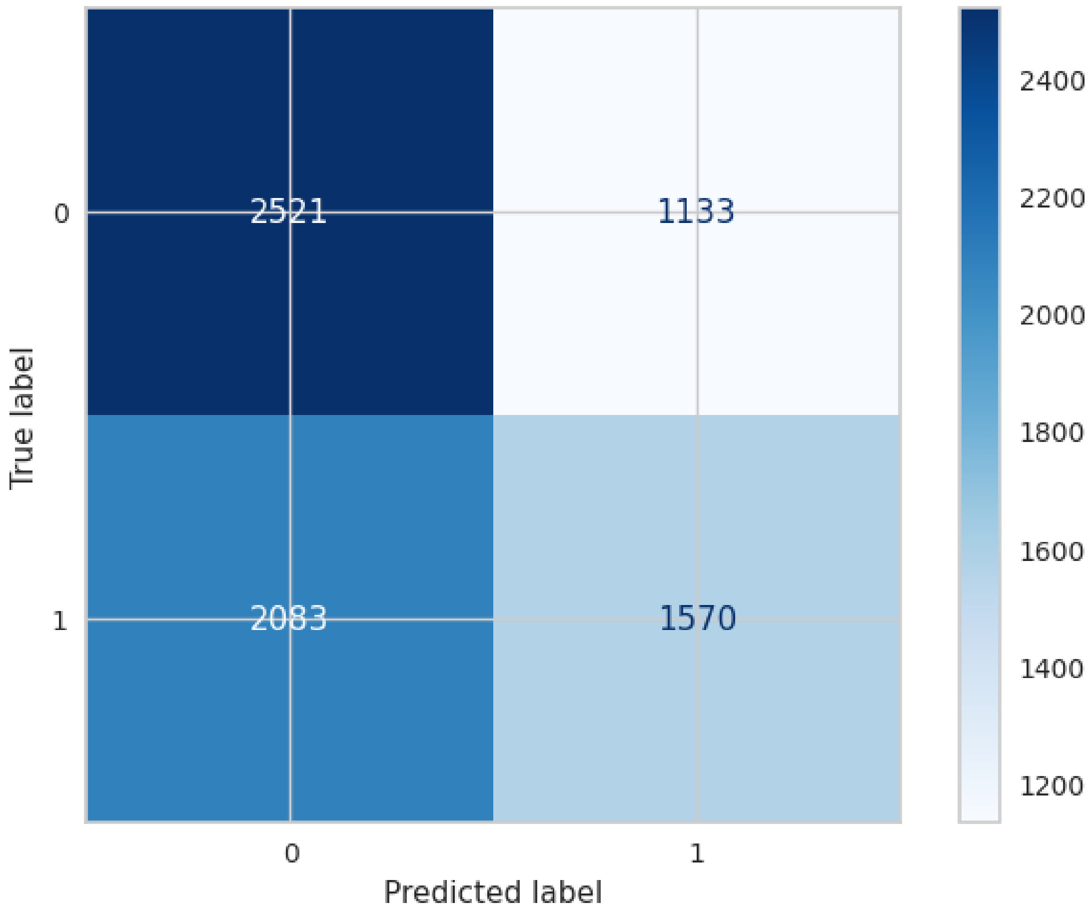
```
Apply Model With Normal Data :

Model Train Score is :  0.5604975442117909
Model Test Score is :  0.559874093335158
F1 Score is :  0.4940213971050976
Recall Score is :  0.4297837393922803
Precision Score is :  0.5808361080281169
AUC Value  :  0.5598562922467696

Classification Report is :               precision    recall  f1-score
support

           0       0.55      0.69      0.61      3654
```

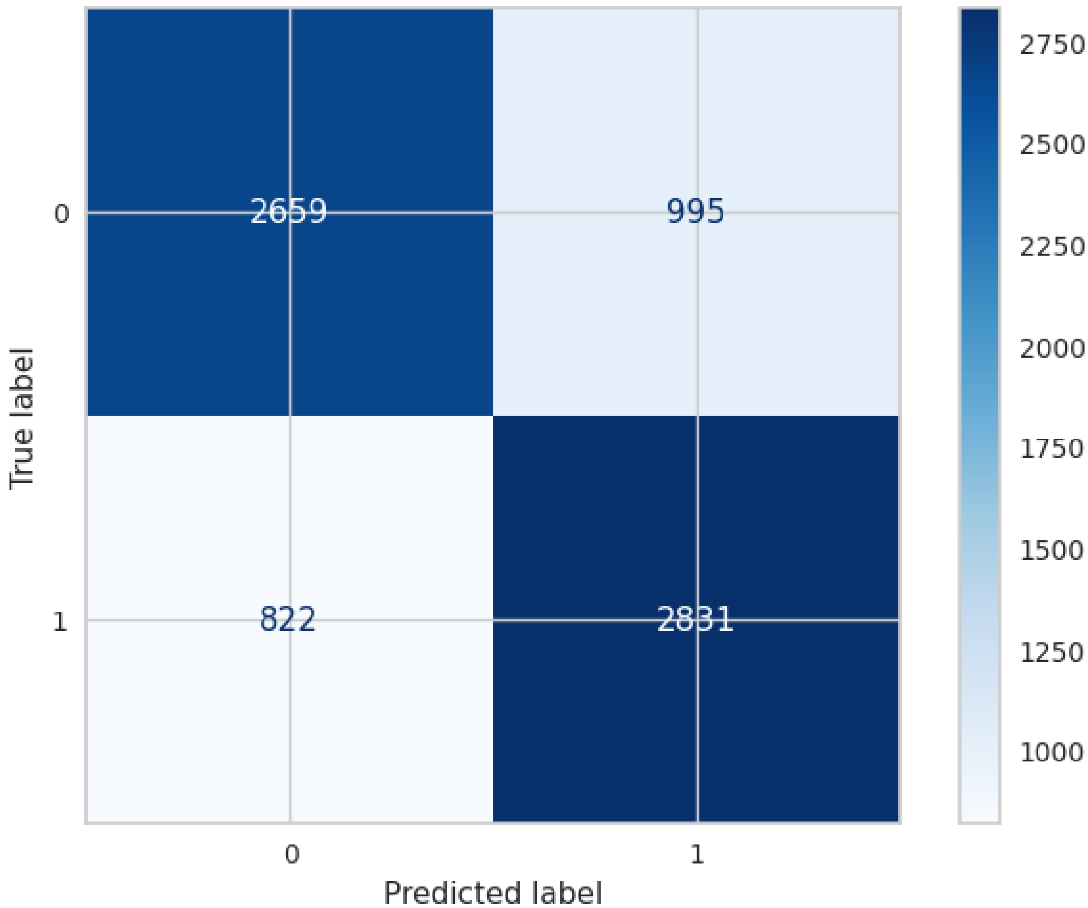
```
           1       0.58      0.43      0.49      3653

    accuracy                           0.56      7307
   macro avg       0.56      0.56      0.55      7307
weighted avg       0.56      0.56      0.55      7307

Confusion Matrix is :
 [[2521 1133]
 [2083 1570]]


 Apply Model With Feature Selection :

Model Train Score is :  0.7404467557745237
Model Test Score is :  0.7513343369371835
F1 Score is :  0.7570530819628293
Recall Score is :  0.7749794689296469
Precision Score is :  0.7399372713016205
AUC Value  :  0.7513375724505924
Classification Report is :                precision    recall  f1-score
support

           0       0.76      0.73      0.75      3654
           1       0.74      0.77      0.76      3653

    accuracy                           0.75      7307
   macro avg       0.75      0.75      0.75      7307
weighted avg       0.75      0.75      0.75      7307

Confusion Matrix is :
 [[2659  995]
 [ 822 2831]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.7213630765019844
Model Test Score is :  0.719310250444779
F1 Score is :  0.703312599450311
Recall Score is :  0.6654804270462633
Precision Score is :  0.7457055214723927
AUC Value  :  0.7193028845685614
Classification Report is :                precision    recall  f1-score
support

           0       0.70      0.77      0.73      3654
           1       0.75      0.67      0.70      3653
```

```
    accuracy                           0.72      7307
   macro avg       0.72      0.72      0.72      7307
weighted avg       0.72      0.72      0.72      7307

Confusion Matrix is :
 [[2825  829]
 [1222 2431]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.8147438529264176
Model Test Score is :  0.8160667852743945
F1 Score is :  0.825318429945412
Recall Score is :  0.8691486449493567
Precision Score is :  0.7856966097500618
AUC Value  :  0.8160740488019909
Classification Report is :                precision    recall  f1-score
support

           0       0.85      0.76      0.81      3654
           1       0.79      0.87      0.83      3653

    accuracy                           0.82      7307
   macro avg       0.82      0.82      0.82      7307
weighted avg       0.82      0.82      0.82      7307

Confusion Matrix is :
 [[2788  866]
 [ 478 3175]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.4318537779602512
Model Test Score is :  0.4362939646913918
F1 Score is :  0.4205936137290759
Recall Score is :  0.4092526690391459
Precision Score is :  0.43258101851851855
AUC Value  :  0.4362902644593649
Classification Report is :                precision    recall  f1-score
support

           0       0.44      0.46      0.45      3654
           1       0.43      0.41      0.42      3653
```

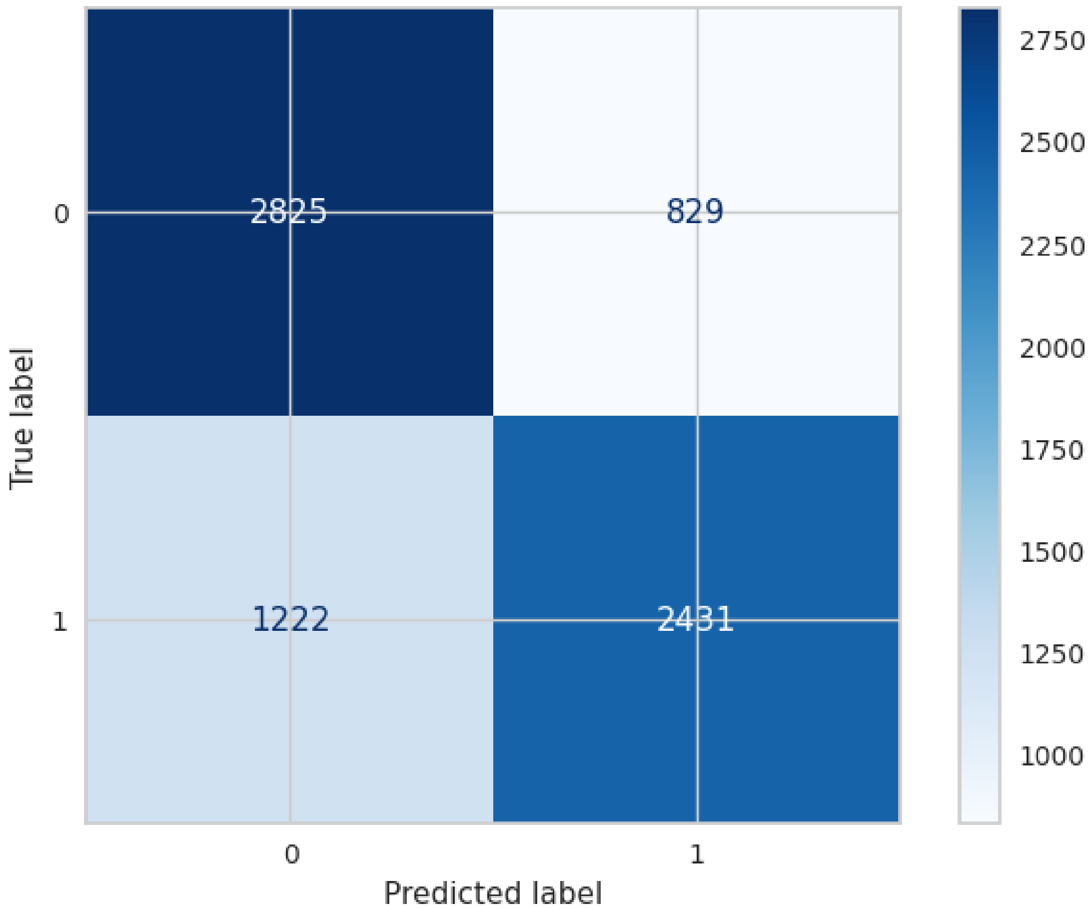
```
    accuracy                           0.44      7307
   macro avg       0.44      0.44      0.44      7307
weighted avg       0.44      0.44      0.44      7307

Confusion Matrix is :
 [[1693 1961]
 [2158 1495]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  0.6812797469701808
Model Test Score is :  0.6796222800054742
F1 Score is :  0.6573979218498464
Recall Score is :  0.6148371201751984
Precision Score is :  0.7062893081761006
AUC Value  :  0.6796134150410748

Classification Report is :                precision    recall  f1-score
support

           0       0.66      0.74      0.70      3654
           1       0.71      0.61      0.66      3653

    accuracy                           0.68      7307
   macro avg       0.68      0.68      0.68      7307
weighted avg       0.68      0.68      0.68      7307

Confusion Matrix is :
 [[2720  934]
 [1407 2246]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  0.8089199093715311
Model Test Score is :  0.8134665389352675
F1 Score is :  0.800585223116313
Recall Score is :  0.7489734464823433
Precision Score is :  0.8598365807668134
AUC Value  :  0.8134577139363004

Classification Report is :                precision    recall  f1-score
support

           0       0.78      0.88      0.82      3654
           1       0.86      0.75      0.80      3653

    accuracy                           0.81      7307
```

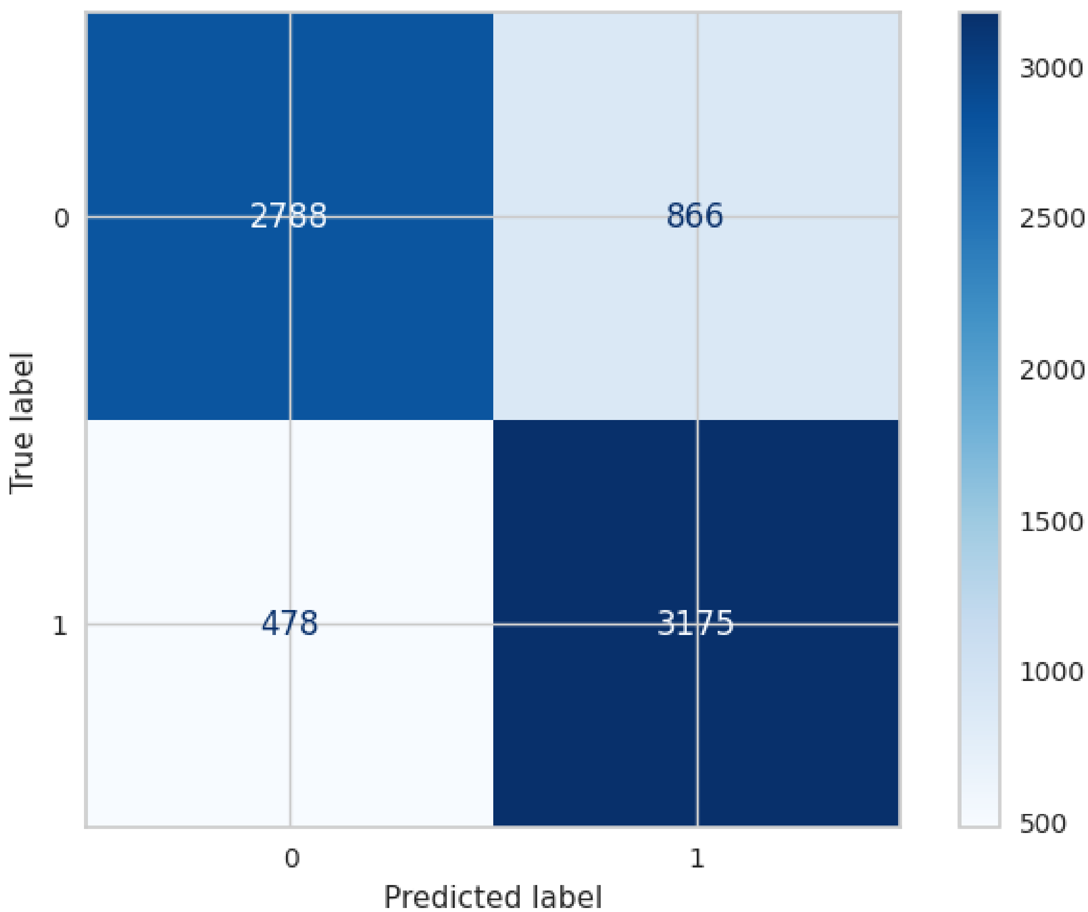
```
   macro avg       0.82      0.81      0.81      7307
weighted avg       0.82      0.81      0.81      7307

Confusion Matrix is :
 [[3208  446]
 [ 917 2736]]
```



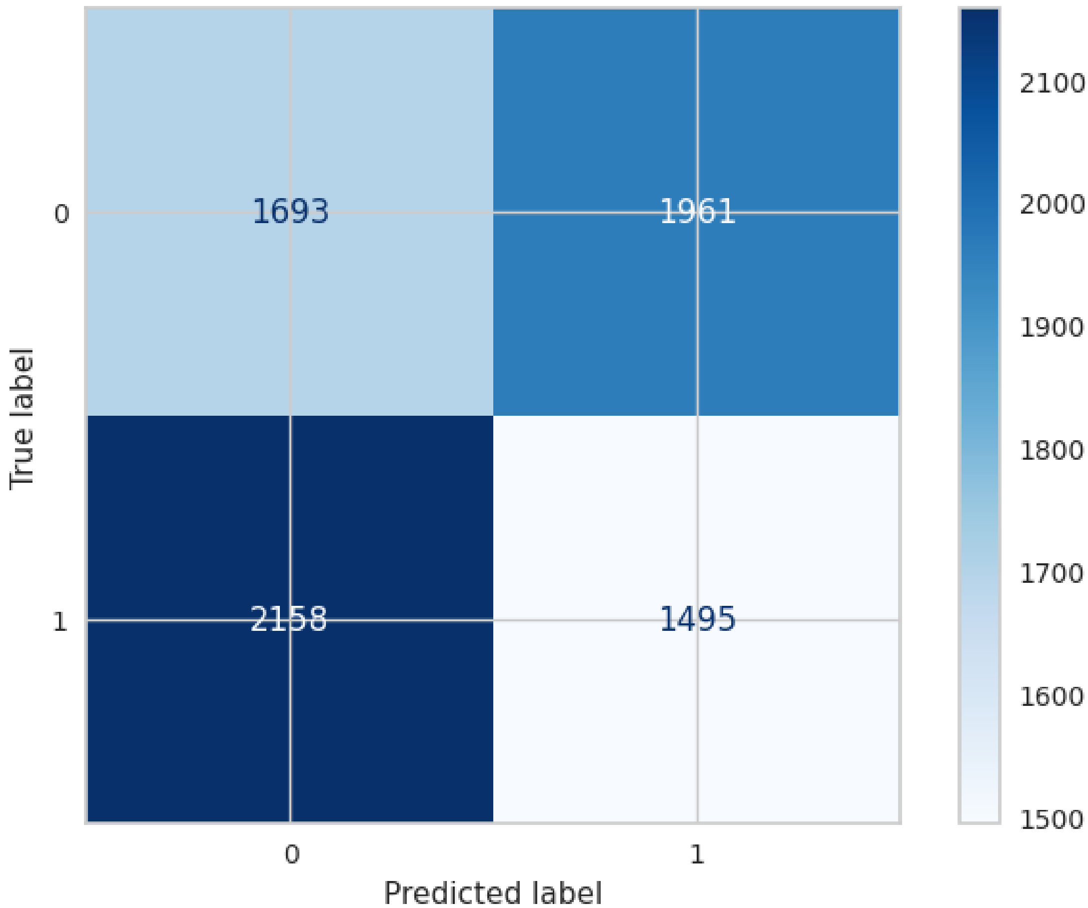


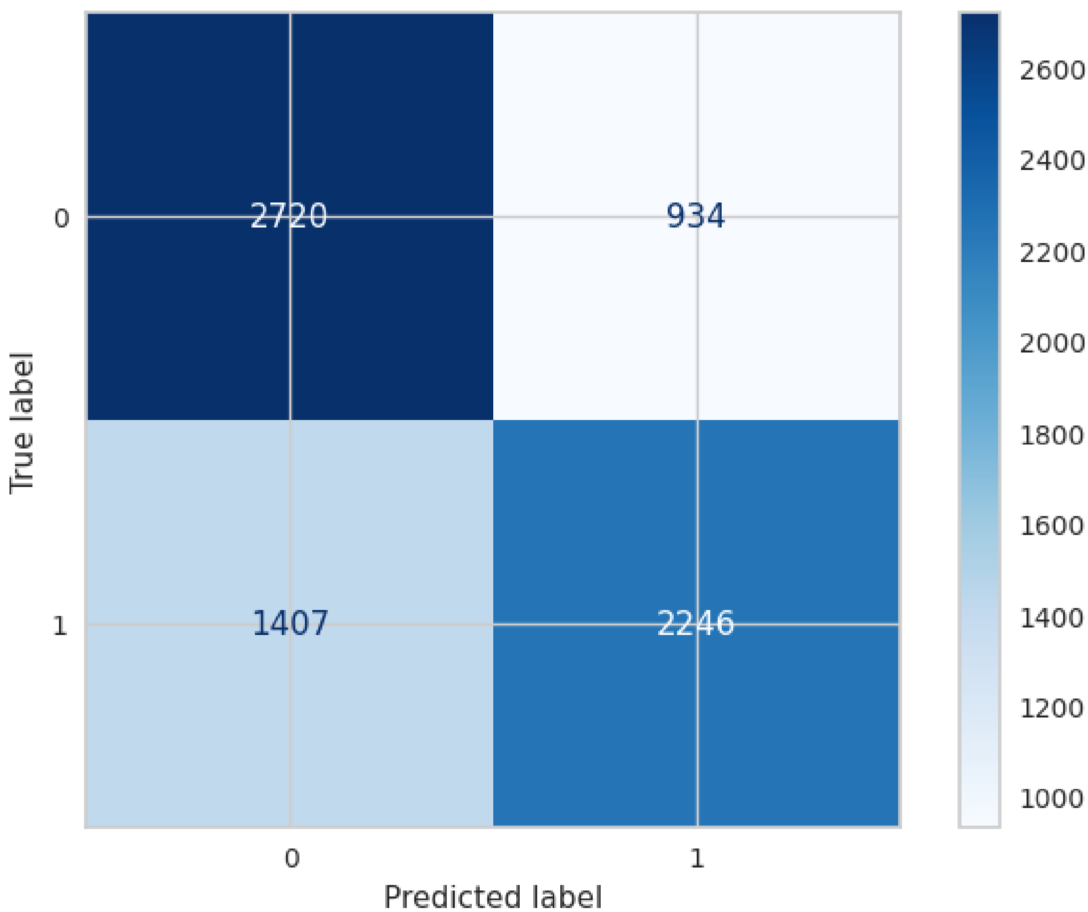
2659
995
822
2831
0
1
True label
0
1
Predicted label
2750
2500
2250
2000
1750
1500
1250
1000


2825
829
1222
2431
0
1
True label
0
1
Predicted label
2750
2500
2250
2000
1750
1500
1250
1000

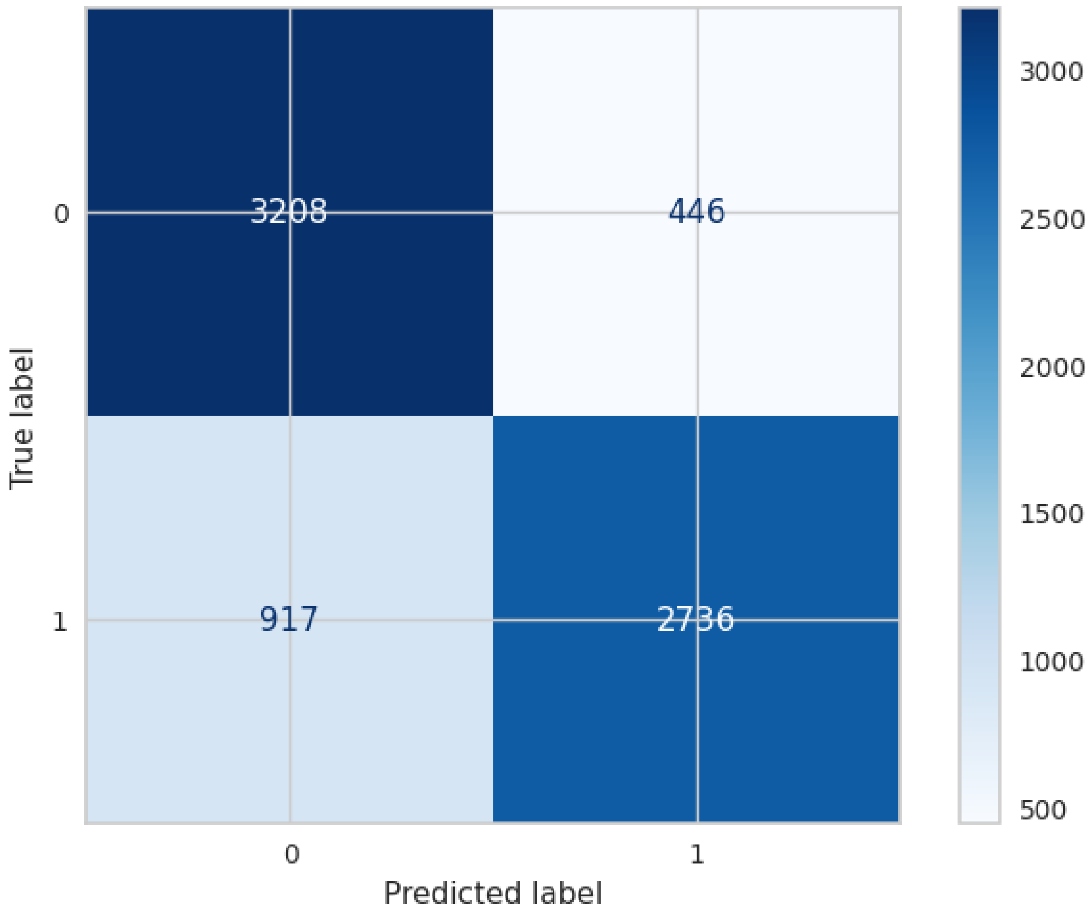


2788
866
478
3175
0
1
True label
0
1
Predicted label
3000
2500
2000
1500
1000
500


1693
1961
2158
1495
0
1
True label
0
1
Predicted label
2100
2000
1900
1800
1700
1600
1500



True label
0
1
2720
934
1407
2246
0
1
Predicted label
2600
2400
2200
2000
1800
1600
1400
1200
1000



[271]:
```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['SVC Under','SVC Under With Feature','SVC Under Scaling','SVC␣
↪Under With Normalize','SVC Under With PCA'
                ,'SVC Under With PCA and Scaling',
                'SVC Under With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[271]:
```
                                  Train Accuracy  Test Accuracy   Test F1  \
Models
SVC Under                               0.560498       0.559874  0.494021
SVC Under With Feature                  0.740447       0.751334  0.757053
SVC Under Scaling                       0.721363       0.719310  0.703313
SVC Under With Normalize                0.814744       0.816067  0.825318
SVC Under With PCA                      0.431854       0.436294  0.420594
SVC Under With PCA and Scaling          0.681280       0.679622  0.657398
SVC Under With PCA and Normalize        0.808920       0.813467  0.800585
```

```
                                     Test Recall  Test Precision       AUC
Models
SVC Under                               0.429784        0.580836  0.559856
SVC Under With Feature                  0.774979        0.739937  0.751338
SVC Under Scaling                       0.665480        0.745706  0.719303
SVC Under With Normalize                0.869149        0.785697  0.816074
SVC Under With PCA                      0.409253        0.432581  0.436290
SVC Under With PCA and Scaling          0.614837        0.706289  0.679613
SVC Under With PCA and Normalize        0.748973        0.859837  0.813458
```

[272]:
```
models_draw(df)
```

RandomUnderSampler

[273]:
```
X_train,y_train,X_test,y_test=Split(X_classification_under,y_classification_under)
```

```
X_train shape is  (8350, 20)
X_test shape is  (928, 20)
y_train shape is  (8350,)
y_test shape is  (928,)
```

[274]:
```
Search(SVC(kernel= 'rbf',max_iter=1000,C=1.0),{'C':[1,.
↪5,2,3,5,10]},X_train,y_train)
```

[274]:
```
SVC(C=1, max_iter=1000)
```

[275]:
```
cross_validation(SVC(kernel= 'rbf',max_iter=1000,C=1),X_train,y_train)
```

```
Train Score Value :  [0.67829341 0.80254491 0.68832335 0.6747006  0.80538922]
Mean 0.7298502994011976
Test Score Value :  [0.66227545 0.79640719 0.6994012  0.67065868 0.82335329]
Mean 0.7304191616766467
```

[276]:
```
Values = Models(SVC(kernel=␣
↪'rbf',max_iter=1000,C=1),X_train,y_train,X_test,y_test)
```

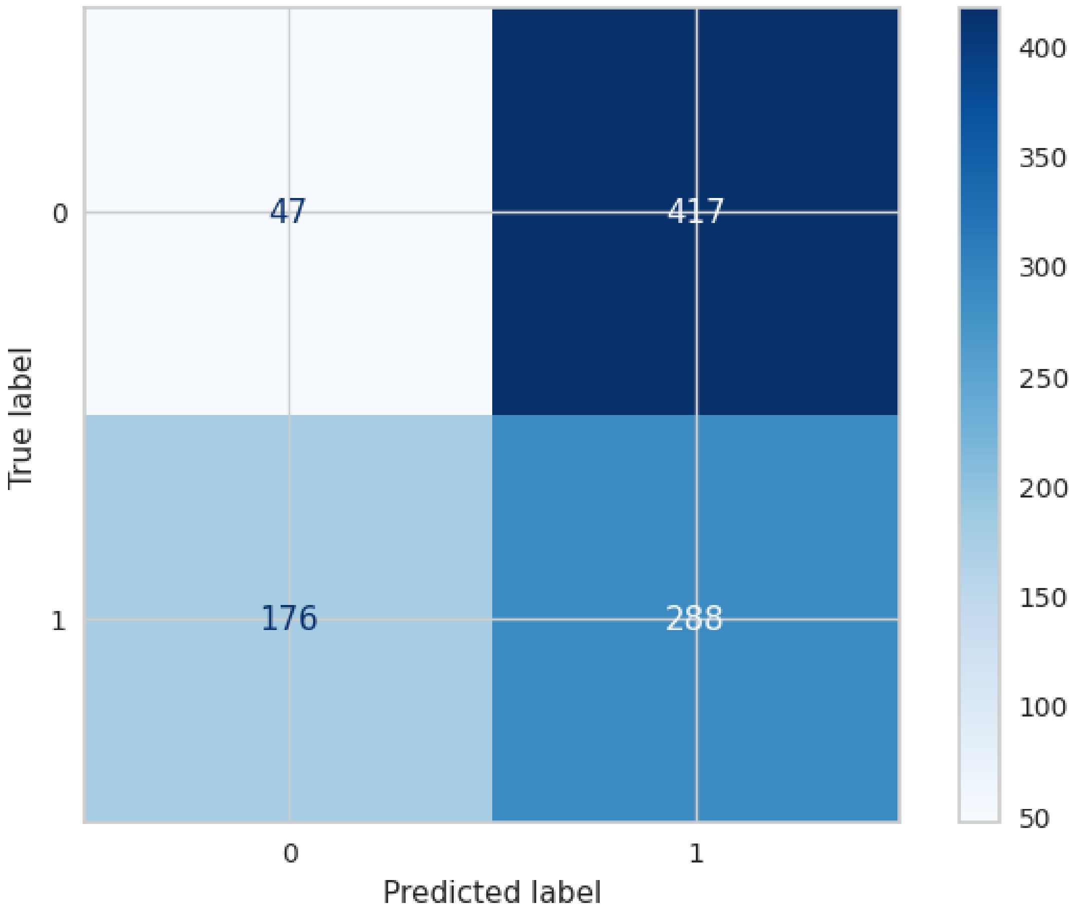
```
Apply Model With Normal Data :

Model Train Score is :  0.3635928143712575
Model Test Score is :  0.3609913793103448
F1 Score is :  0.49272882805816937
Recall Score is :  0.6206896551724138
Precision Score is :  0.4085106382978723
AUC Value  :  0.36099137931034486

Classification Report is :                precision    recall  f1-score
support

           0       0.21      0.10      0.14       464
```

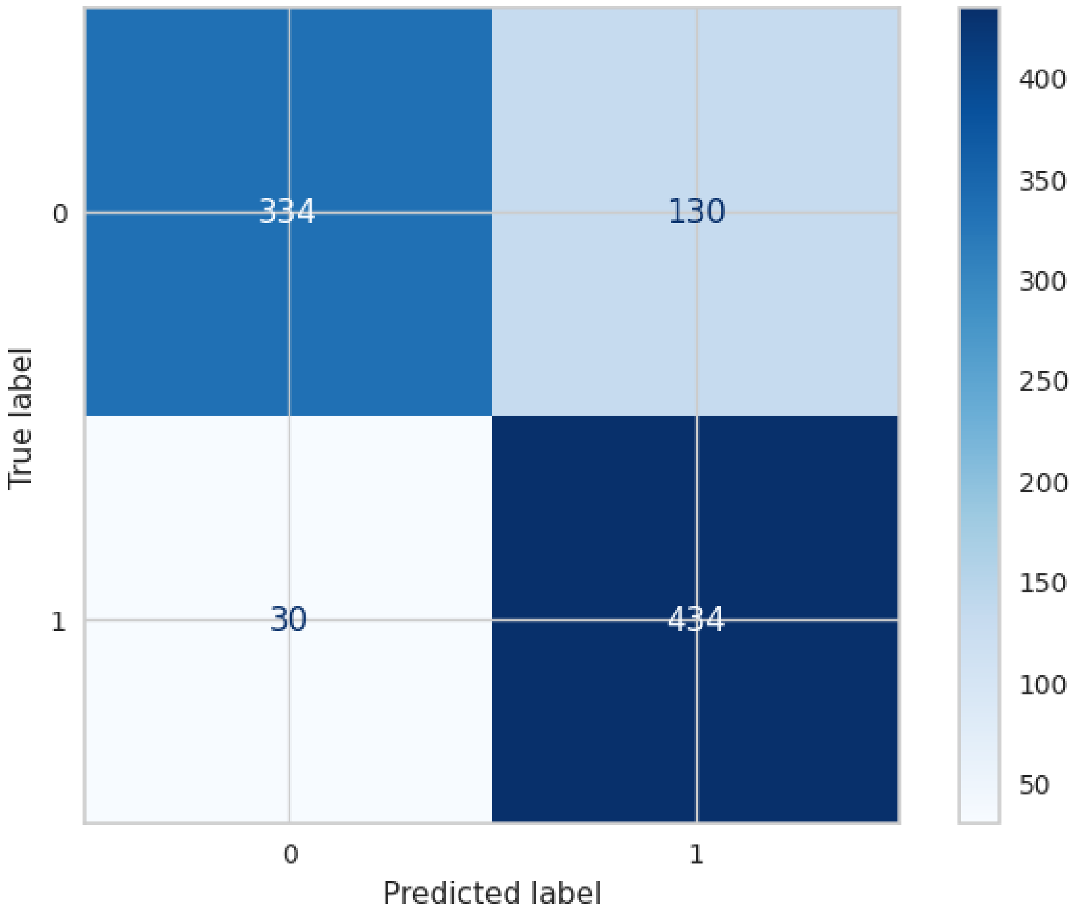
```
           1       0.41      0.62      0.49       464

    accuracy                           0.36       928
   macro avg       0.31      0.36      0.31       928
weighted avg       0.31      0.36      0.31       928

Confusion Matrix is :
 [[ 47 417]
 [176 288]]


 Apply Model With Feature Selection :

Model Train Score is :  0.8246706586826348
Model Test Score is :  0.8275862068965517
F1 Score is :  0.8443579766536964
Recall Score is :  0.9353448275862069
Precision Score is :  0.7695035460992907
AUC Value  :  0.8275862068965518
Classification Report is :                precision    recall  f1-score
support

           0       0.92      0.72      0.81       464
           1       0.77      0.94      0.84       464

    accuracy                           0.83       928
   macro avg       0.84      0.83      0.83       928
weighted avg       0.84      0.83      0.83       928

Confusion Matrix is :
 [[334 130]
 [ 30 434]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.8667065868263473
Model Test Score is :  0.8545258620689655
F1 Score is :  0.868804664723032
Recall Score is :  0.9633620689655172
Precision Score is :  0.7911504424778761
AUC Value  :  0.8545258620689655
Classification Report is :                precision    recall  f1-score
support

           0       0.95      0.75      0.84       464
           1       0.79      0.96      0.87       464
```

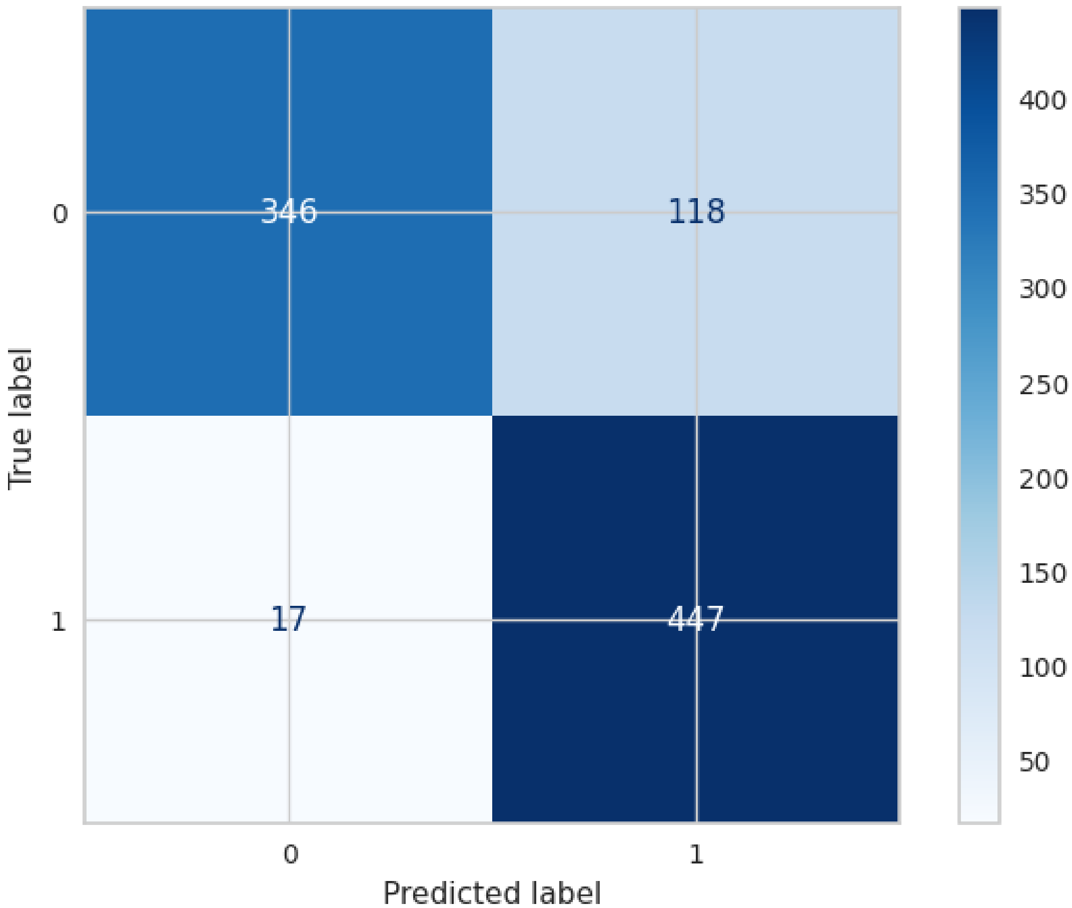
```
   accuracy                           0.85       928
  macro avg       0.87      0.85      0.85       928
weighted avg       0.87      0.85      0.85       928

Confusion Matrix is :
 [[346 118]
 [ 17 447]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.5210778443113773
Model Test Score is :  0.5344827586206896
F1 Score is :  0.6609105180533752
Recall Score is :  0.9073275862068966
Precision Score is :  0.519753086419753
AUC Value  :  0.5344827586206897
Classification Report is :                precision    recall  f1-score
support

           0       0.64      0.16      0.26       464
           1       0.52      0.91      0.66       464

   accuracy                           0.53       928
  macro avg       0.58      0.53      0.46       928
weighted avg       0.58      0.53      0.46       928

Confusion Matrix is :
 [[ 75 389]
 [ 43 421]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.3159281437125748
Model Test Score is :  0.32435344827586204
F1 Score is :  0.39537126325940214
Recall Score is :  0.4418103448275862
Precision Score is :  0.35776614310645727
AUC Value  :  0.3243534482758621
Classification Report is :                precision    recall  f1-score
support

           0       0.27      0.21      0.23       464
           1       0.36      0.44      0.40       464
```

```
    accuracy                           0.32       928
   macro avg       0.31      0.32      0.31       928
weighted avg       0.31      0.32      0.31       928

Confusion Matrix is :
 [[ 96 368]
 [259 205]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  0.8774850299401198
Model Test Score is :  0.8620689655172413
F1 Score is :  0.8740157480314961
Recall Score is :  0.9568965517241379
Precision Score is :  0.8043478260869565
AUC Value  :  0.8620689655172414

Classification Report is :                precision    recall  f1-score
support

           0       0.95      0.77      0.85       464
           1       0.80      0.96      0.87       464

    accuracy                           0.86       928
   macro avg       0.88      0.86      0.86       928
weighted avg       0.88      0.86      0.86       928

Confusion Matrix is :
 [[356 108]
 [ 20 444]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  0.5944910179640719
Model Test Score is :  0.6142241379310345
F1 Score is :  0.7084690553745928
Recall Score is :  0.9375
Precision Score is :  0.569371727748691
AUC Value  :  0.6142241379310345

Classification Report is :                precision    recall  f1-score
support

           0       0.82      0.29      0.43       464
           1       0.57      0.94      0.71       464

    accuracy                           0.61       928
```

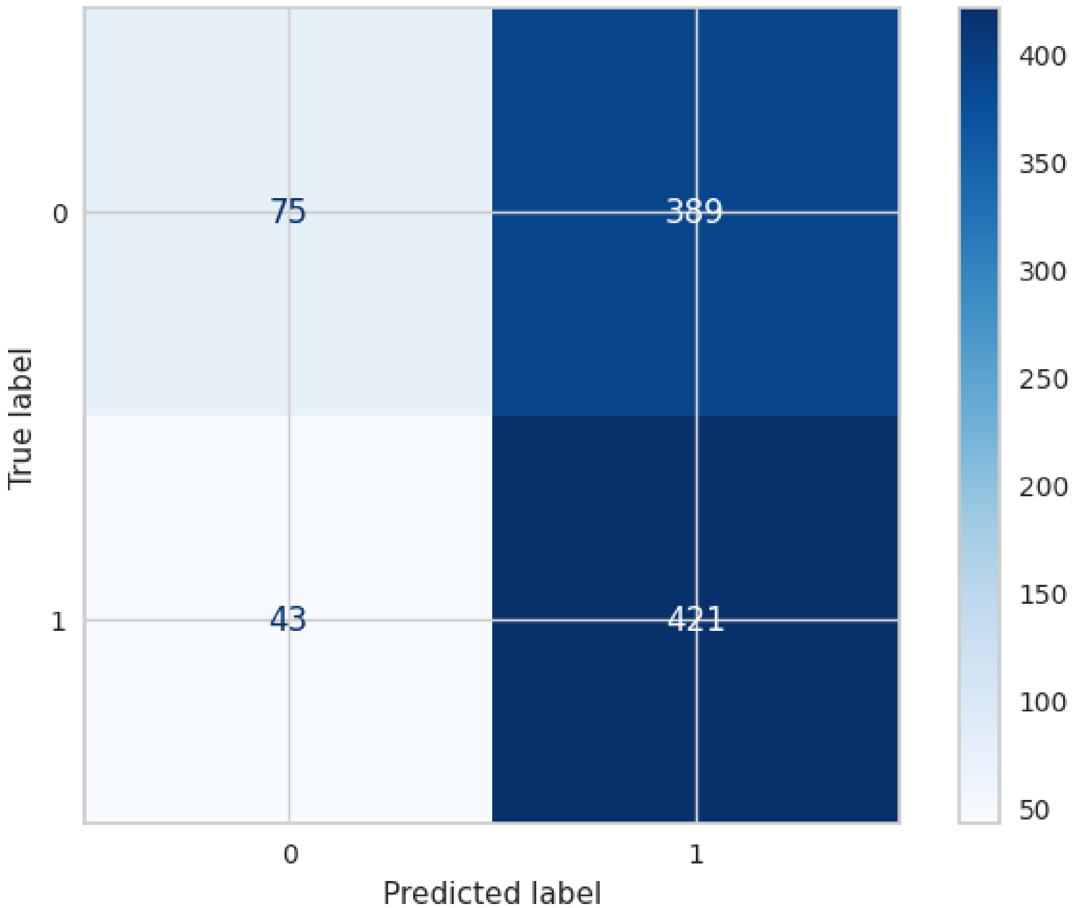
```
   macro avg       0.70      0.61      0.57       928
weighted avg       0.70      0.61      0.57       928

Confusion Matrix is :
 [[135 329]
 [ 29 435]]
```



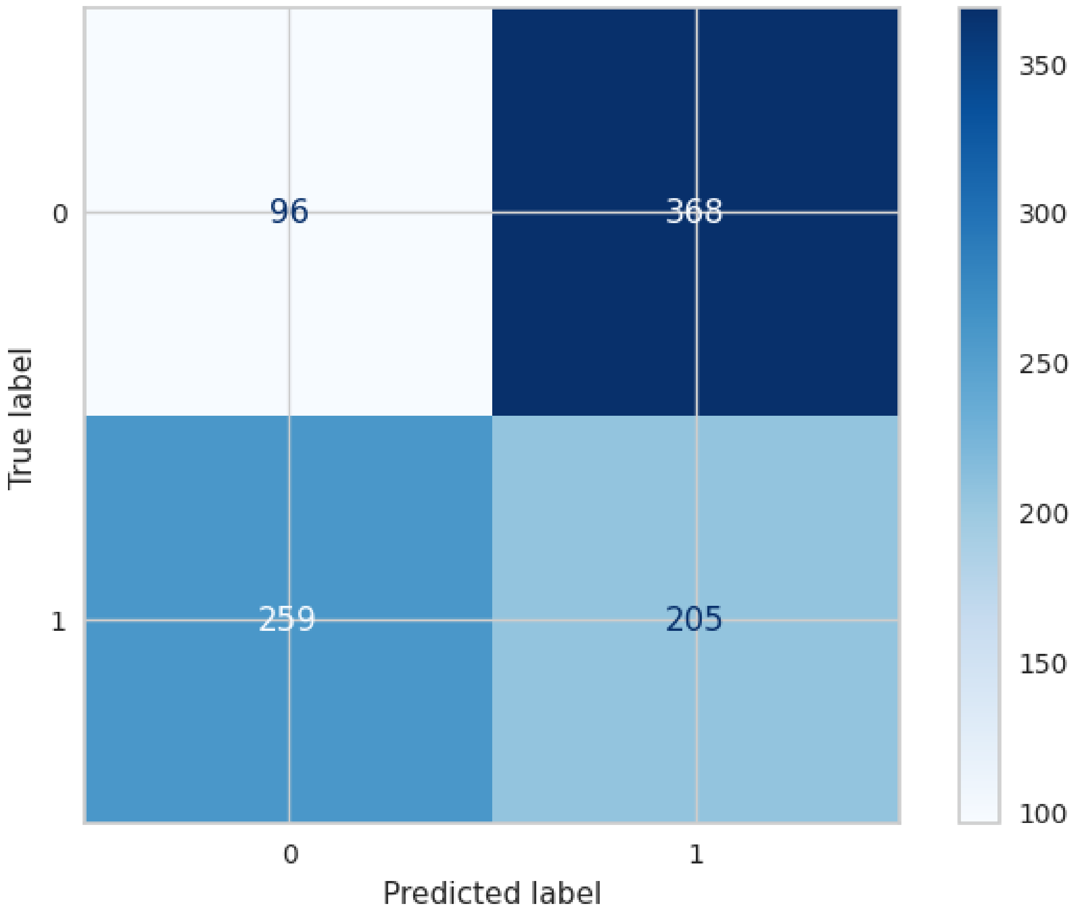

334
130
30
434
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50



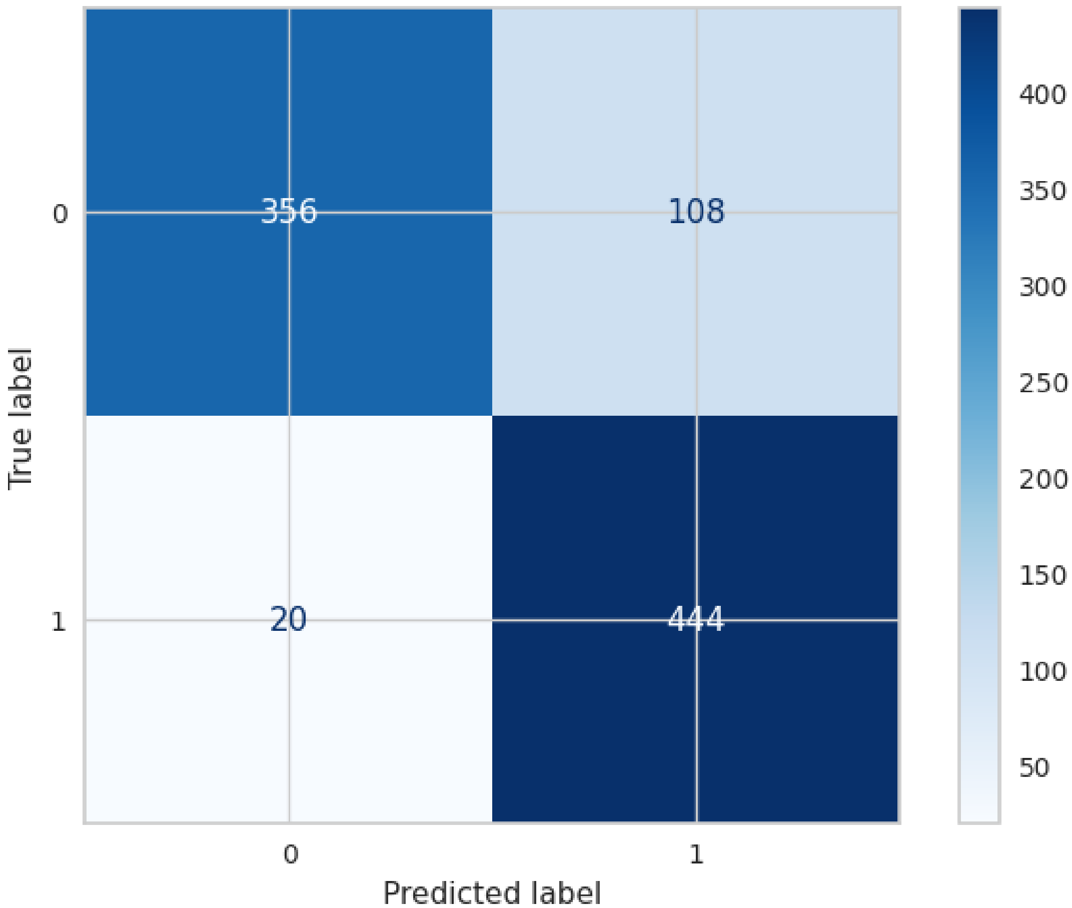
346
118
17
447
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50

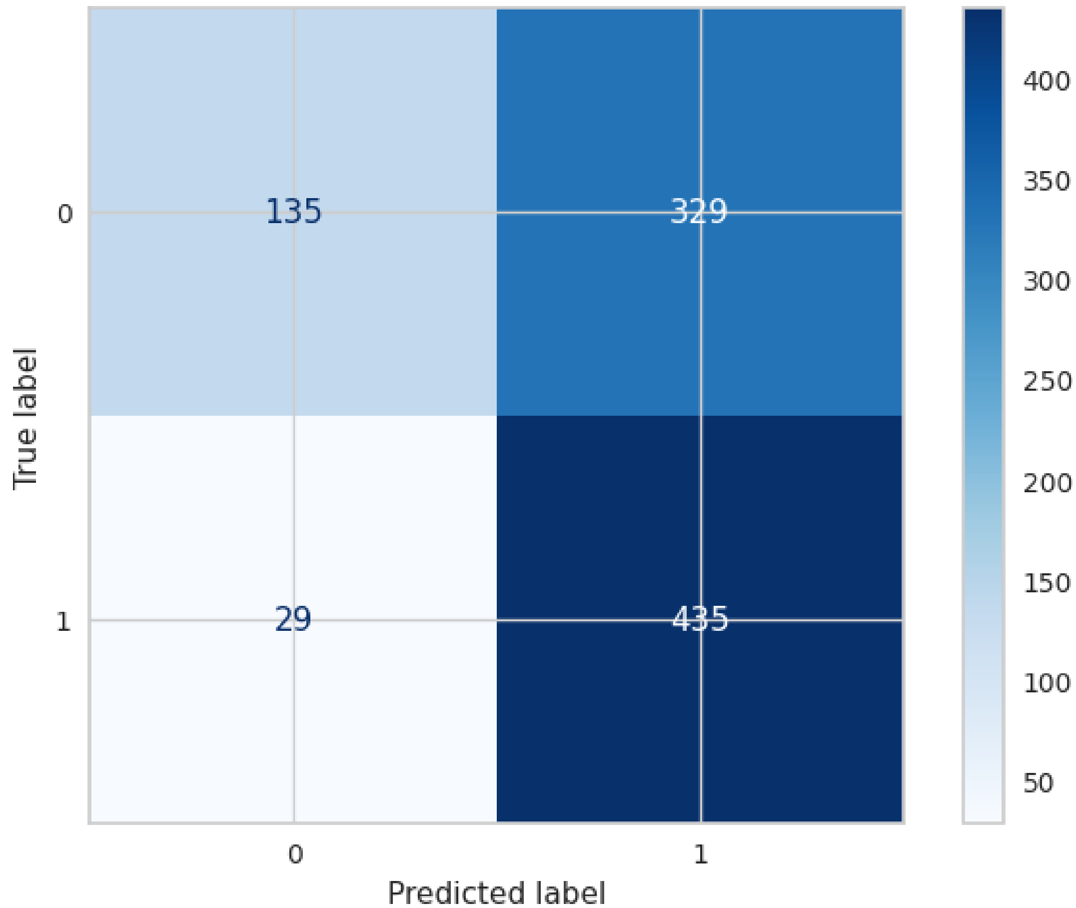


75
389
43
421
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50



96
368
259
205
0
1
True label
0
1
Predicted label
350
300
250
200
150
100


356
108
20
444
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50



[277]:
```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['SVC Under','SVC Under With Feature','SVC Under Scaling','SVC␣
↪Under With Normalize','SVC Under With PCA'
                ,'SVC Under With PCA and Scaling',
                'SVC Under With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[277]:
```
                                  Train Accuracy  Test Accuracy   Test F1  \
Models
SVC Under                               0.363593       0.360991  0.492729
SVC Under With Feature                  0.824671       0.827586  0.844358
SVC Under Scaling                       0.866707       0.854526  0.868805
SVC Under With Normalize                0.521078       0.534483  0.660911
SVC Under With PCA                      0.315928       0.324353  0.395371
SVC Under With PCA and Scaling          0.877485       0.862069  0.874016
SVC Under With PCA and Normalize        0.594491       0.614224  0.708469
```

```
                                   Test Recall  Test Precision       AUC
Models
SVC Under                             0.620690        0.408511  0.360991
SVC Under With Feature                0.935345        0.769504  0.827586
SVC Under Scaling                     0.963362        0.791150  0.854526
SVC Under With Normalize              0.907328        0.519753  0.534483
SVC Under With PCA                    0.441810        0.357766  0.324353
SVC Under With PCA and Scaling        0.956897        0.804348  0.862069
SVC Under With PCA and Normalize      0.937500        0.569372  0.614224
```

[278]:
```
models_draw(df)
```

LogisticRegression

[279]:
```
X_train,y_train,X_test,y_test=Split(X_classification,y_classification)
```

```
X_train shape is  (37056, 20)
X_test shape is  (4118, 20)
y_train shape is  (37056,)
y_test shape is  (4118,)
```

[280]:
```
Search(LogisticRegression(penalty='l2',solver='sag',C=1.0),{'C':[1,.
↪5,2,3,5,10]},X_train,y_train)
```

[280]: LogisticRegression(C=1, solver='sag')

[281]:
```
cross_validation(LogisticRegression(penalty='l2',solver='sag',C=3),X_train,y_train)
```

```
Train Score Value :  [0.90750236 0.90915837 0.90838253 0.90888851 0.90740428]
Mean 0.9082672120247013
Test Score Value :  [0.90974096 0.90500607 0.90689516 0.90703009 0.91377682]
Mean 0.9084898194316393
```

[282]:
```
Values =␣
↪Models(LogisticRegression(penalty='l2',solver='sag',C=3),X_train,y_train,X_test,y_test)
```

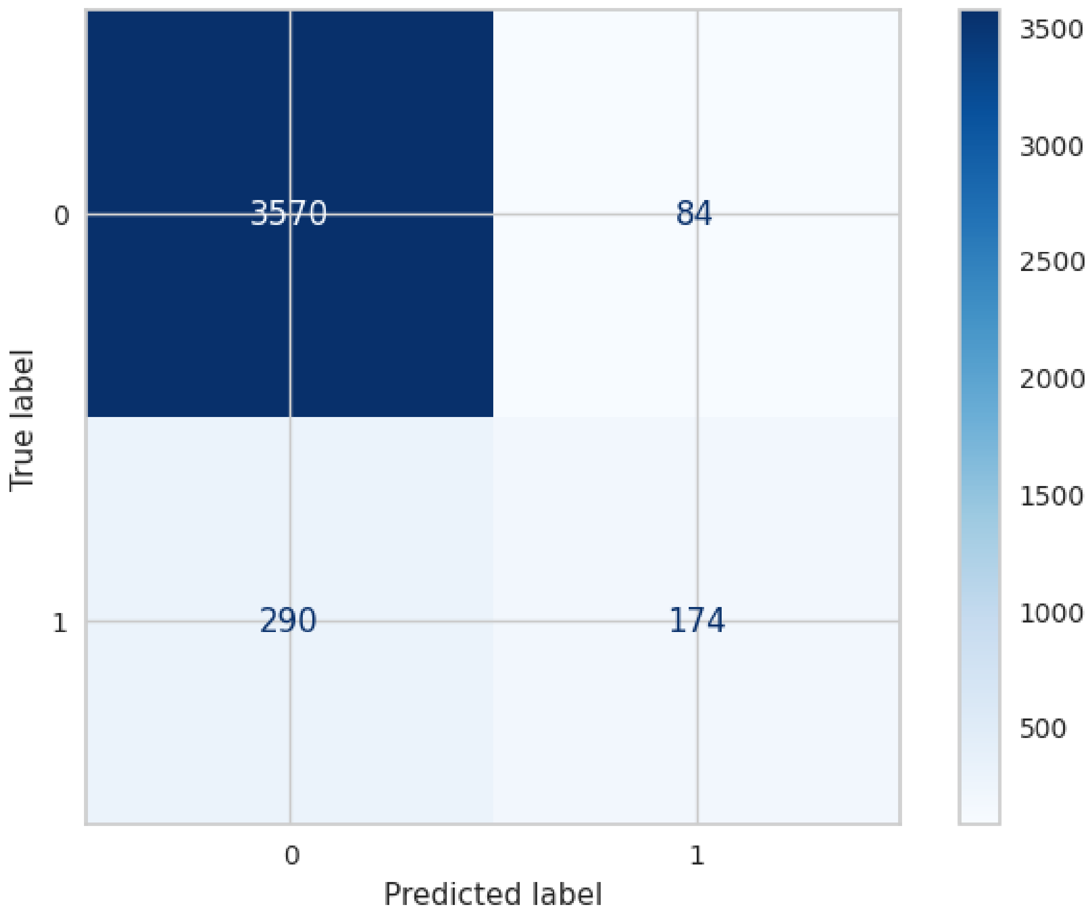
```
Apply Model With Normal Data :

Model Train Score is :  0.9085708117443869
Model Test Score is :  0.9091792132102963
F1 Score is :  0.4819944598337951
Recall Score is :  0.375
Precision Score is :  0.6744186046511628
AUC Value  :  0.6760057471264368

Classification Report is :                precision    recall  f1-score
support

           0       0.92      0.98      0.95      3654
```

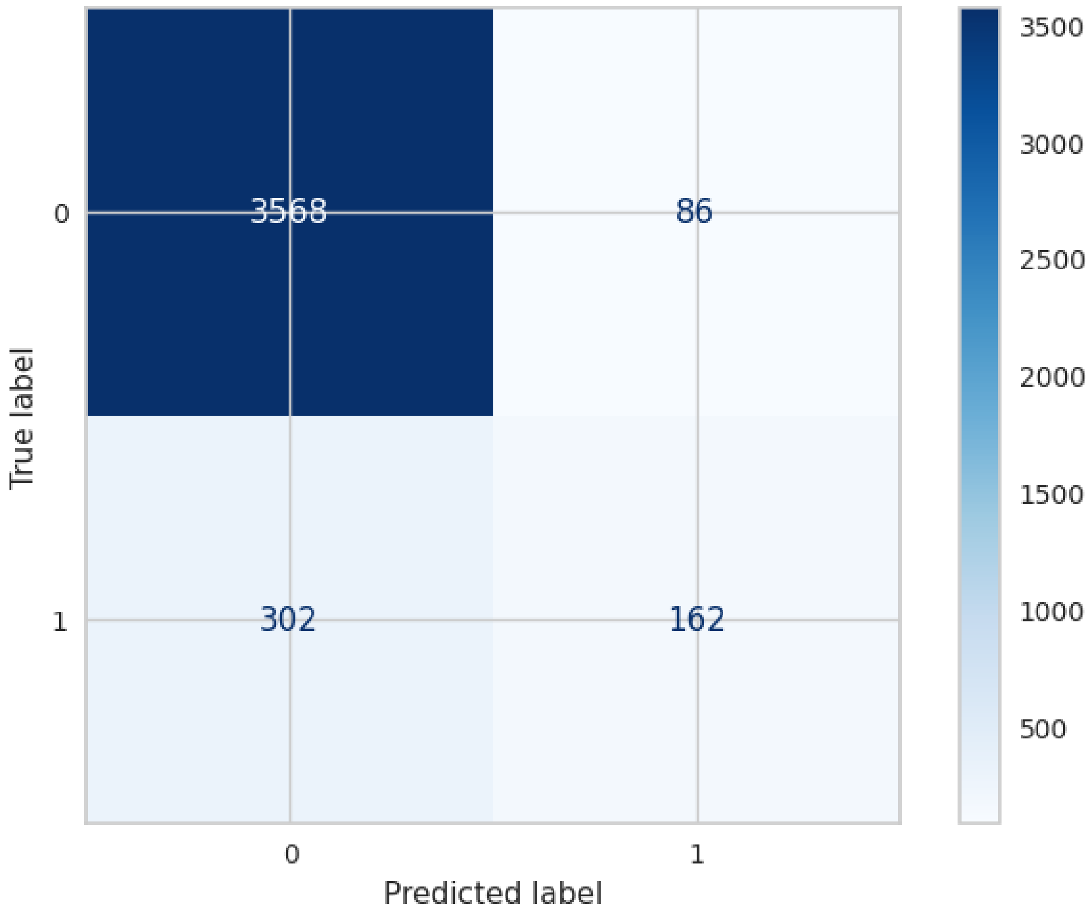
```
           1       0.67      0.38      0.48       464

    accuracy                           0.91      4118
   macro avg       0.80      0.68      0.72      4118
weighted avg       0.90      0.91      0.90      4118

Confusion Matrix is :
 [[3570   84]
 [ 290  174]]


 Apply Model With Feature Selection :

Model Train Score is :  0.9031735751295337
Model Test Score is :  0.9057795046138902
F1 Score is :  0.4550561797752809
Recall Score is :  0.34913793103448276
Precision Score is :  0.6532258064516129
AUC Value  :  0.6628010399562123

Classification Report is :                precision    recall  f1-score
support

           0       0.92      0.98      0.95      3654
           1       0.65      0.35      0.46       464

    accuracy                           0.91      4118
   macro avg       0.79      0.66      0.70      4118
weighted avg       0.89      0.91      0.89      4118

Confusion Matrix is :
 [[3568   86]
 [ 302  162]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.9086247841105354
Model Test Score is :  0.9091792132102963
F1 Score is :  0.48907103825136605
Recall Score is :  0.3857758620689655
Precision Score is :  0.667910447761194
AUC Value  :  0.6807094964422551

Classification Report is :                precision    recall  f1-score
support

           0       0.93      0.98      0.95      3654
           1       0.67      0.39      0.49       464
```

```
    accuracy                           0.91      4118
   macro avg       0.80      0.68      0.72      4118
weighted avg       0.90      0.91      0.90      4118

Confusion Matrix is :
 [[3565   89]
 [ 285  179]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.9045768566493955
Model Test Score is :  0.9084507042253521
F1 Score is :  0.45283018867924524
Recall Score is :  0.33620689655172414
Precision Score is :  0.6933333333333334
AUC Value  :  0.6586617405582923

Classification Report is :                precision    recall  f1-score
support

           0       0.92      0.98      0.95      3654
           1       0.69      0.34      0.45       464

    accuracy                           0.91      4118
   macro avg       0.81      0.66      0.70      4118
weighted avg       0.90      0.91      0.89      4118

Confusion Matrix is :
 [[3585   69]
 [ 308  156]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.9085438255613126
Model Test Score is :  0.9084507042253521
F1 Score is :  0.4814305364511692
Recall Score is :  0.3771551724137931
Precision Score is :  0.6653992395437263
AUC Value  :  0.6765359879584018

Classification Report is :                precision    recall  f1-score
support

           0       0.93      0.98      0.95      3654
           1       0.67      0.38      0.48       464
```

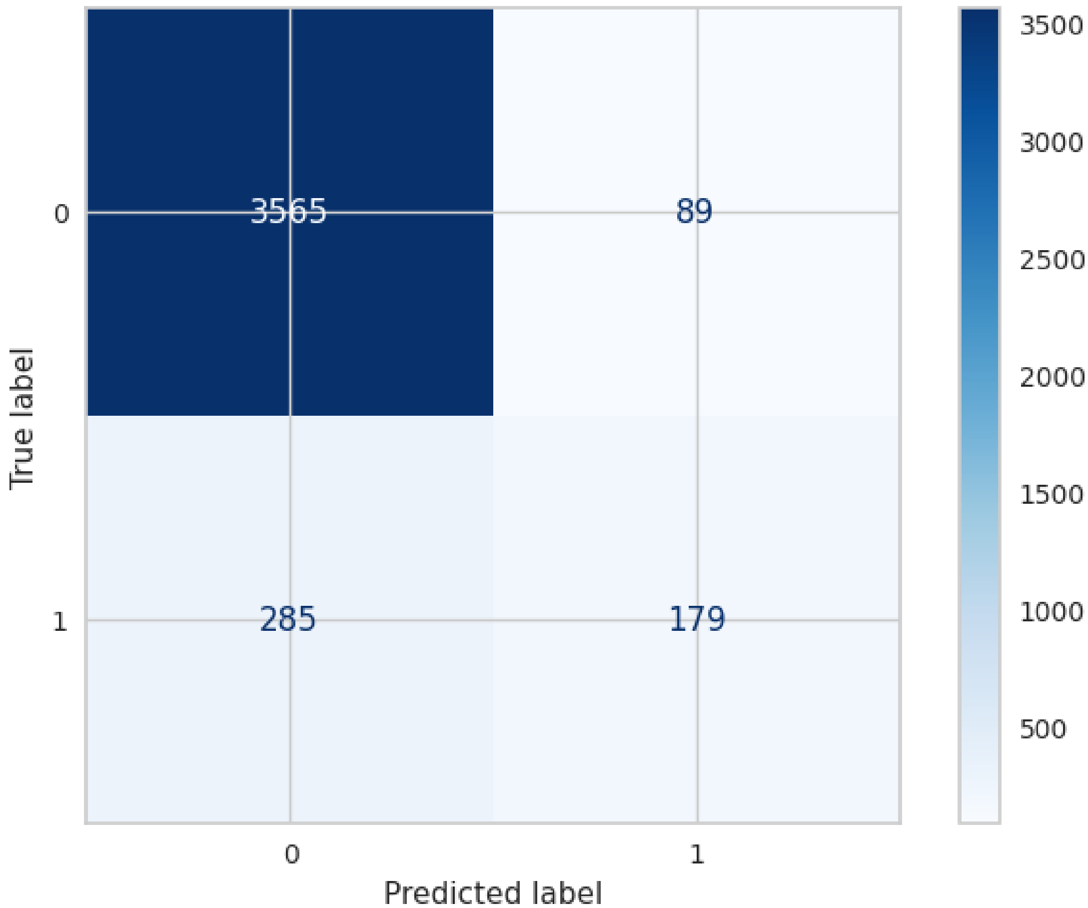
```
    accuracy                           0.91      4118
   macro avg       0.80      0.68      0.72      4118
weighted avg       0.90      0.91      0.90      4118

Confusion Matrix is :
 [[3566   88]
 [ 289  175]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  0.9086517702936097
Model Test Score is :  0.9091792132102963
F1 Score is :  0.48907103825136605
Recall Score is :  0.3857758620689655
Precision Score is :  0.667910447761194
AUC Value  :  0.6807094964422551

Classification Report is :                precision    recall  f1-score
support

           0       0.93      0.98      0.95      3654
           1       0.67      0.39      0.49       464

    accuracy                           0.91      4118
   macro avg       0.80      0.68      0.72      4118
weighted avg       0.90      0.91      0.90      4118

Confusion Matrix is :
 [[3565   89]
 [ 285  179]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  0.9044689119170984
Model Test Score is :  0.9084507042253521
F1 Score is :  0.45283018867924524
Recall Score is :  0.33620689655172414
Precision Score is :  0.6933333333333334
AUC Value  :  0.6586617405582923

Classification Report is :                precision    recall  f1-score
support

           0       0.92      0.98      0.95      3654
           1       0.69      0.34      0.45       464

    accuracy                           0.91      4118
```

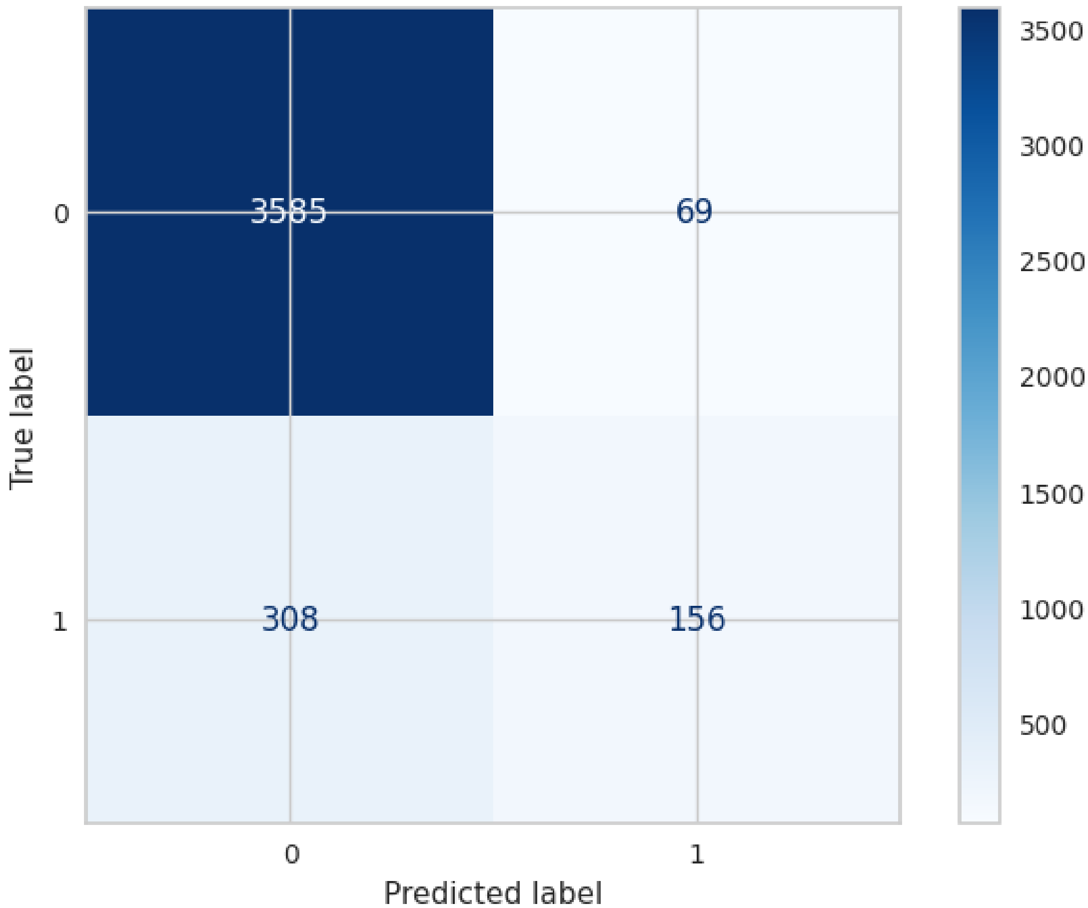
```
   macro avg       0.81      0.66      0.70      4118
weighted avg       0.90      0.91      0.89      4118

Confusion Matrix is :
 [[3585   69]
 [ 308  156]]
```



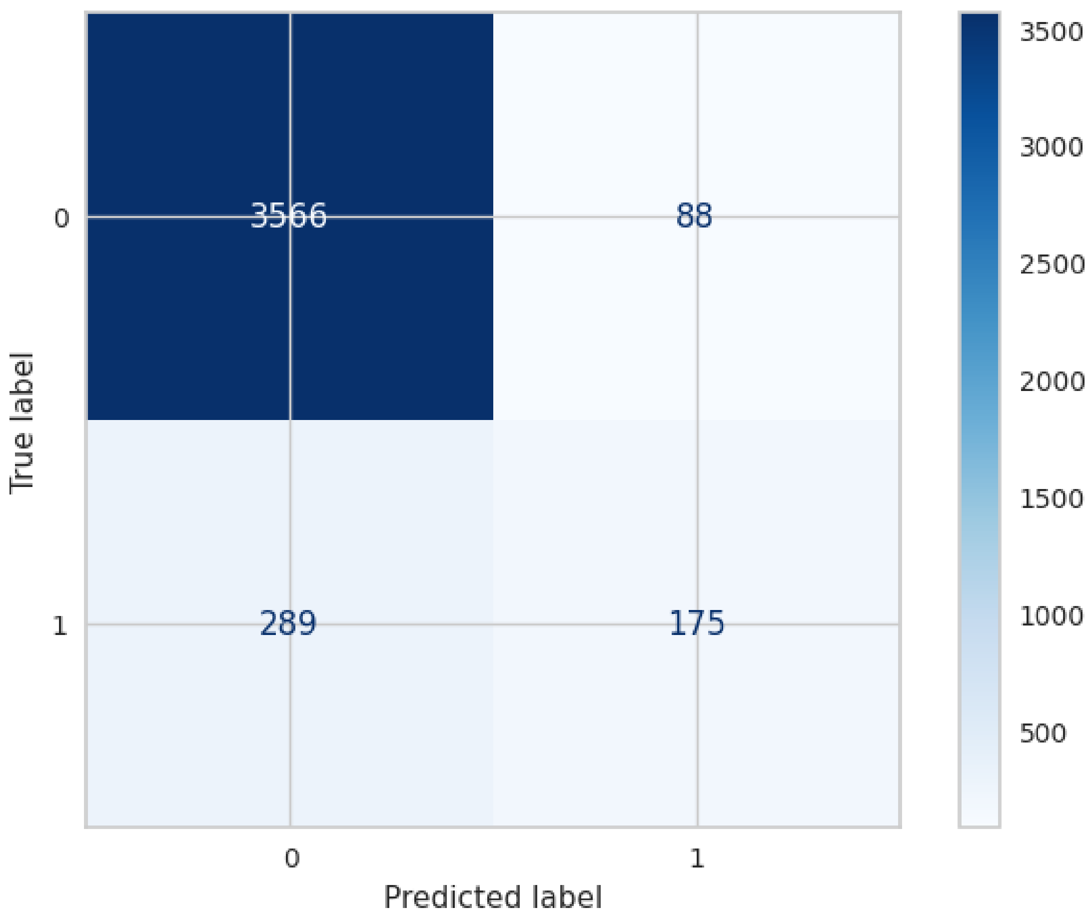


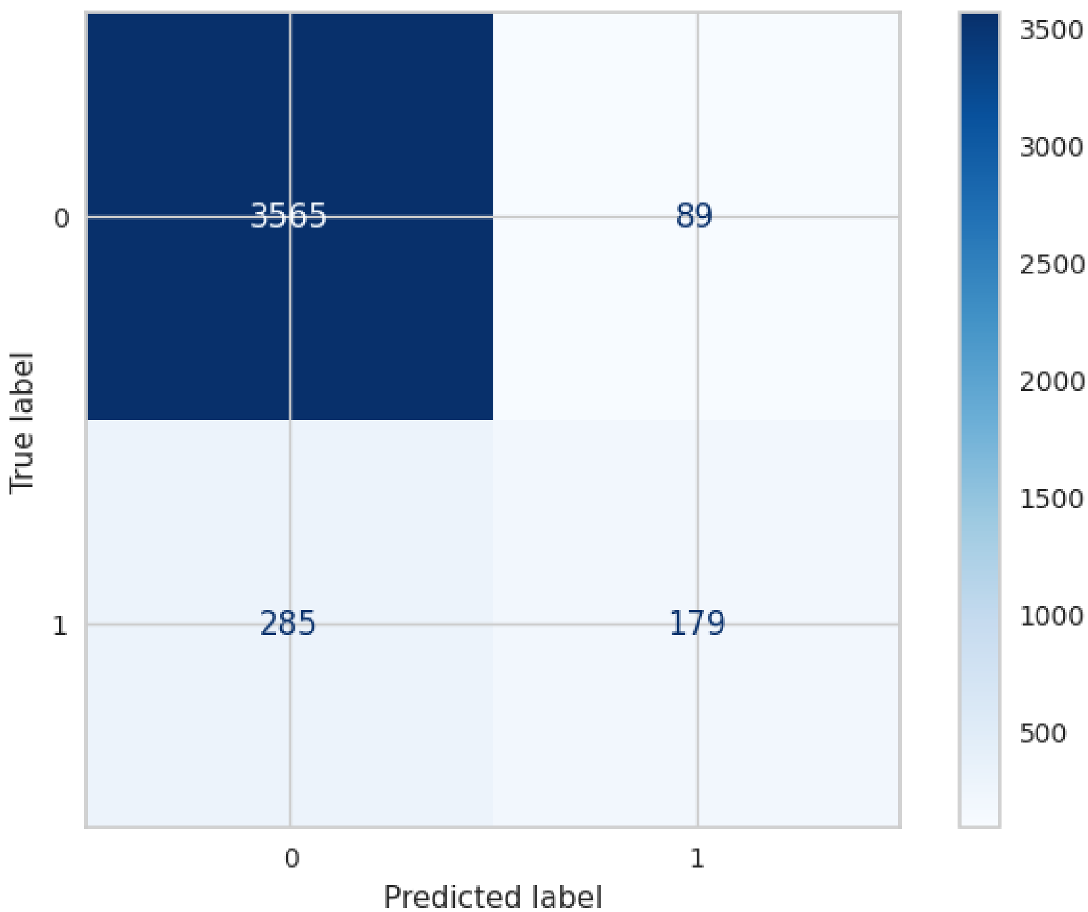
3568
86
302
162
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500

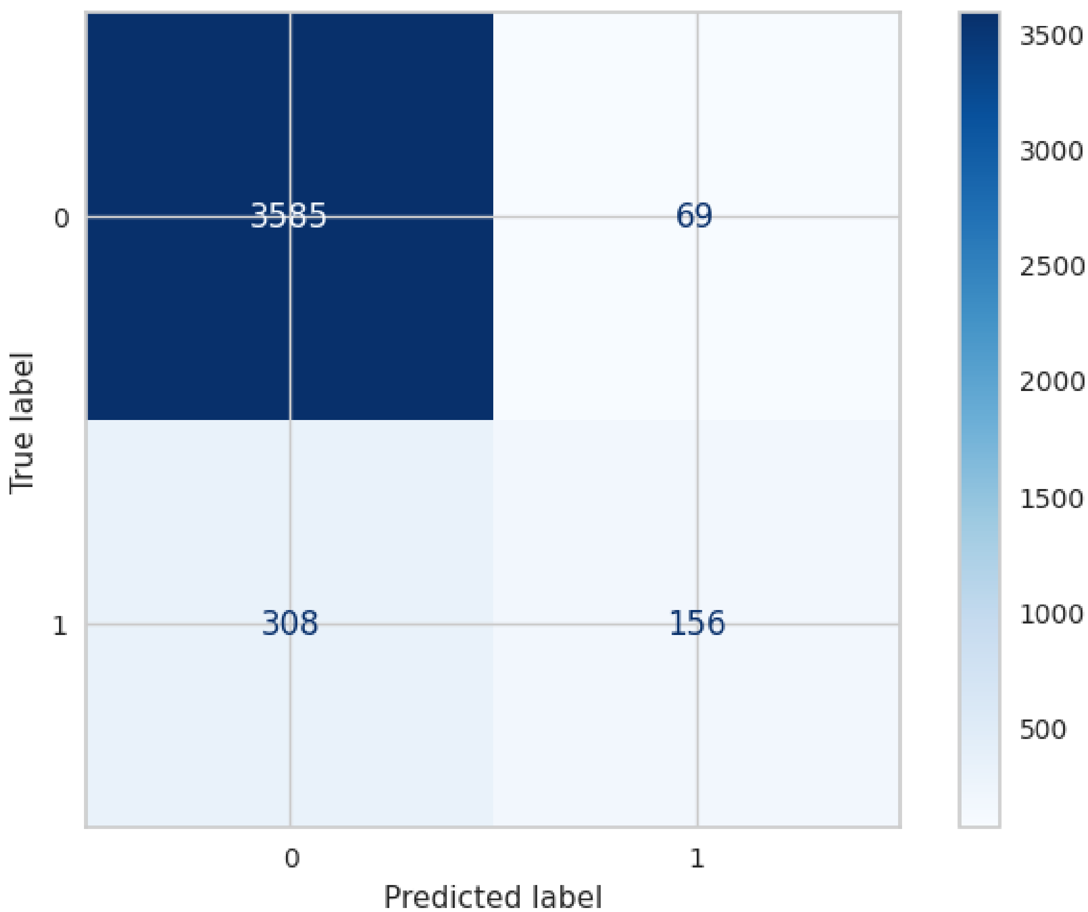

3565
89
285
179
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500



3585
69
308
156
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500


3566
88
289
175
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500


3565
89
285
179
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500



[283]:
```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['Logistic','Logistic With Feature','Logistic Scaling','Logistic␣
↪With Normalize','Logistic With PCA'
                ,'Logistic With PCA and Scaling',
                'Logistic With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[283]:
```
                               Train Accuracy  Test Accuracy   Test F1  \
Models
Logistic                             0.908571       0.909179  0.481994
Logistic With Feature                0.903174       0.905780  0.455056
Logistic Scaling                     0.908625       0.909179  0.489071
Logistic With Normalize              0.904577       0.908451  0.452830
Logistic With PCA                    0.908544       0.908451  0.481431
Logistic With PCA and Scaling        0.908652       0.909179  0.489071
Logistic With PCA and Normalize      0.904469       0.908451  0.452830
```

```
                                   Test Recall  Test Precision       AUC
 Models
 Logistic                             0.375000        0.674419  0.676006
 Logistic With Feature                0.349138        0.653226  0.662801
 Logistic Scaling                     0.385776        0.667910  0.680709
 Logistic With Normalize              0.336207        0.693333  0.658662
 Logistic With PCA                    0.377155        0.665399  0.676536
 Logistic With PCA and Scaling        0.385776        0.667910  0.680709
 Logistic With PCA and Normalize      0.336207        0.693333  0.658662
```

[284]:
```
models_draw(df)
```

RandomOverSampler

[285]:
```
X_train,y_train,X_test,y_test=Split(X_classification_over,y_classification_over)
```

```
X_train shape is  (65763, 20)
X_test shape is  (7307, 20)
y_train shape is  (65763,)
y_test shape is  (7307,)
```

[286]:
```
Search(LogisticRegression(penalty='l2',solver='sag',C=1.0),{'C':[1,.
↪5,2,3,5,10]},X_train,y_train)
```

[286]:
```
LogisticRegression(C=1, solver='sag')
```

[287]:
```
cross_validation(LogisticRegression(penalty='l2',solver='sag',C=1),X_train,y_train)
```

```
Train Score Value :  [0.86225052 0.86379015 0.86327694 0.86310848 0.864439  ]
Mean 0.8633730182570953
Test Score Value :  [0.86573405 0.86109633 0.86573405 0.8647354  0.86009732]
Mean 0.8634794318060528
```

[288]:
```
Values =␣
↪Models(LogisticRegression(penalty='l2',solver='sag',C=1),X_train,y_train,X_test,y_test)
```

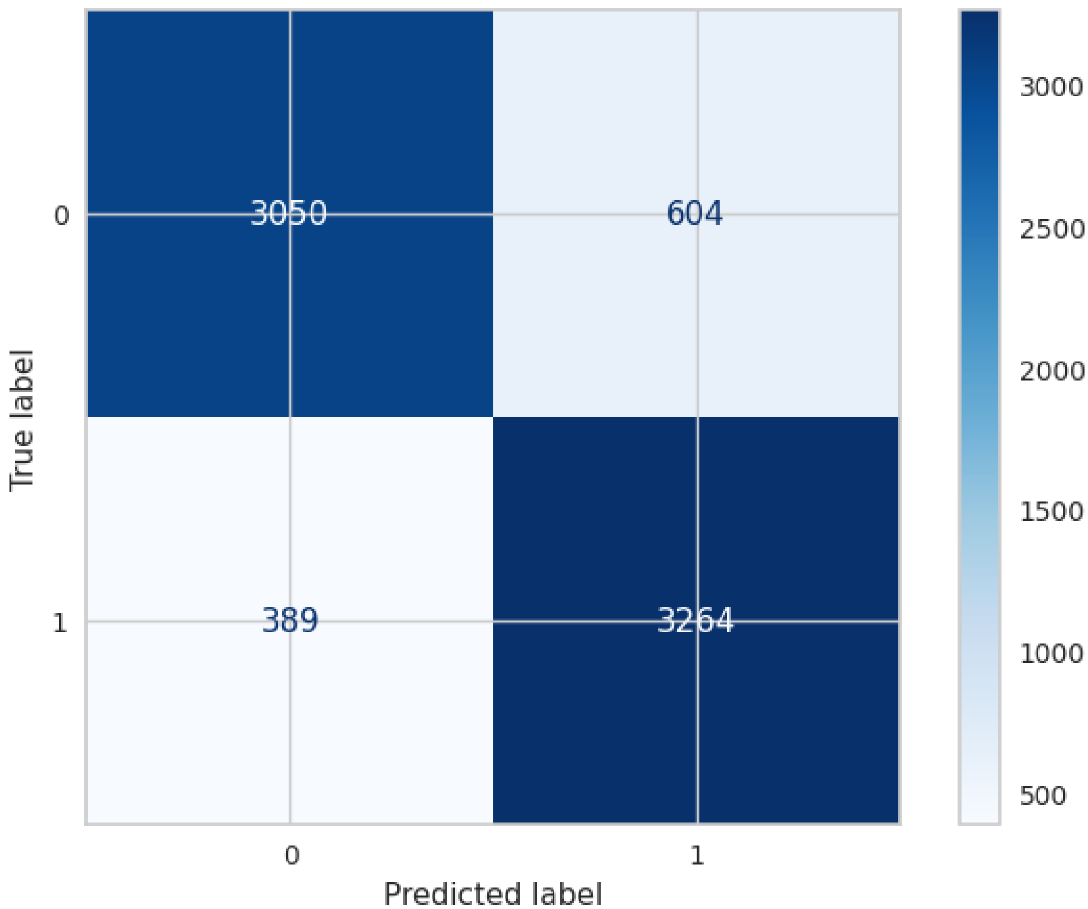
```
Apply Model With Normal Data :

Model Train Score is :  0.863160135638581
Model Test Score is :  0.8641029150130012
F1 Score is :  0.8679696848823295
Recall Score is :  0.8935121817684095
Precision Score is :  0.843846949327818
AUC Value  :  0.864106939269536

Classification Report is :                precision    recall  f1-score
support

           0       0.89      0.83      0.86      3654
```

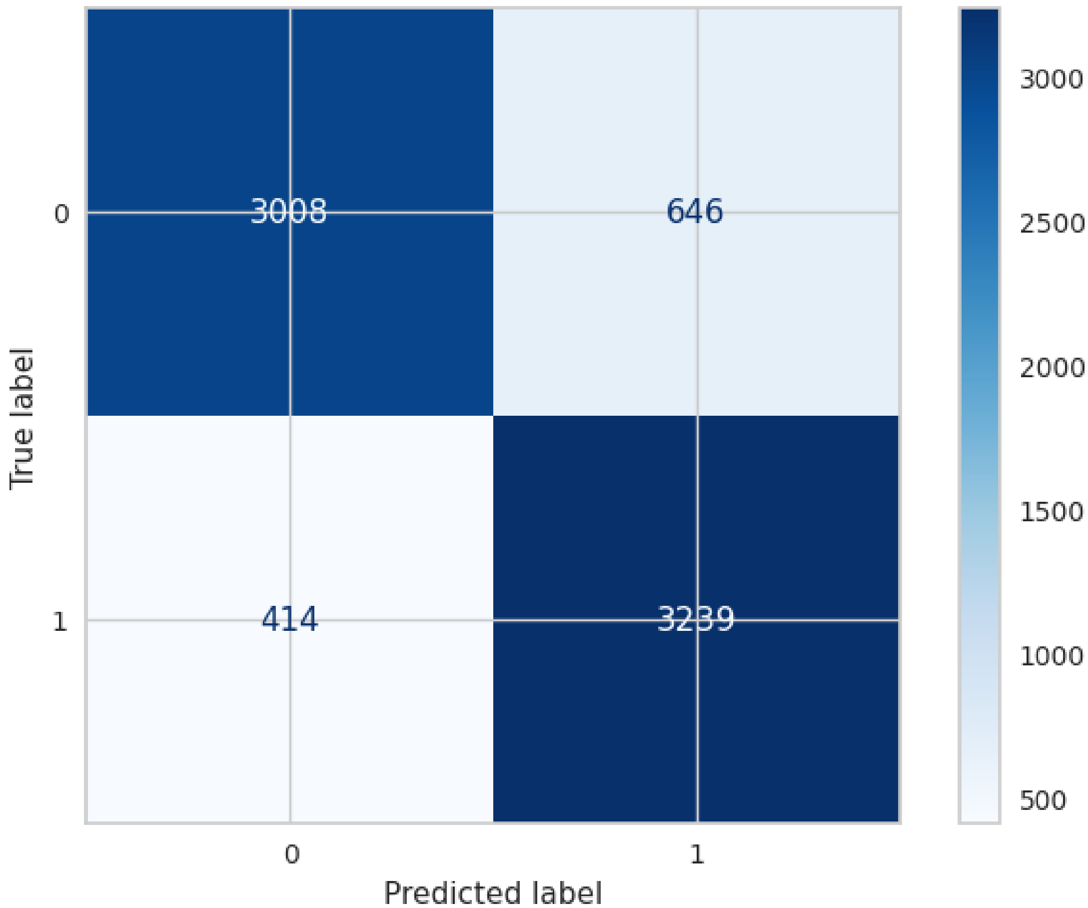
```
           1       0.84      0.89      0.87      3653

    accuracy                           0.86      7307
   macro avg       0.87      0.86      0.86      7307
weighted avg       0.87      0.86      0.86      7307

Confusion Matrix is :
 [[3050  604]
 [ 389 3264]]


 Apply Model With Feature Selection :

Model Train Score is :  0.8494290102337181
Model Test Score is :  0.8549336252908171
F1 Score is :  0.8593791456619793
Recall Score is :  0.886668491650698
Precision Score is :  0.8337194337194337
AUC Value  :  0.8549379677738986
Classification Report is :                precision    recall  f1-score
support

           0       0.88      0.82      0.85      3654
           1       0.83      0.89      0.86      3653

    accuracy                           0.85      7307
   macro avg       0.86      0.85      0.85      7307
weighted avg       0.86      0.85      0.85      7307

Confusion Matrix is :
 [[3008  646]
 [ 414 3239]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.8627799826650244
Model Test Score is :  0.8639660599425208
F1 Score is :  0.8678542940707258
Recall Score is :  0.8935121817684095
Precision Score is :  0.8436288446627035
AUC Value  :  0.8639701029258029
Classification Report is :                precision    recall  f1-score
support

           0       0.89      0.83      0.86      3654
           1       0.84      0.89      0.87      3653
```

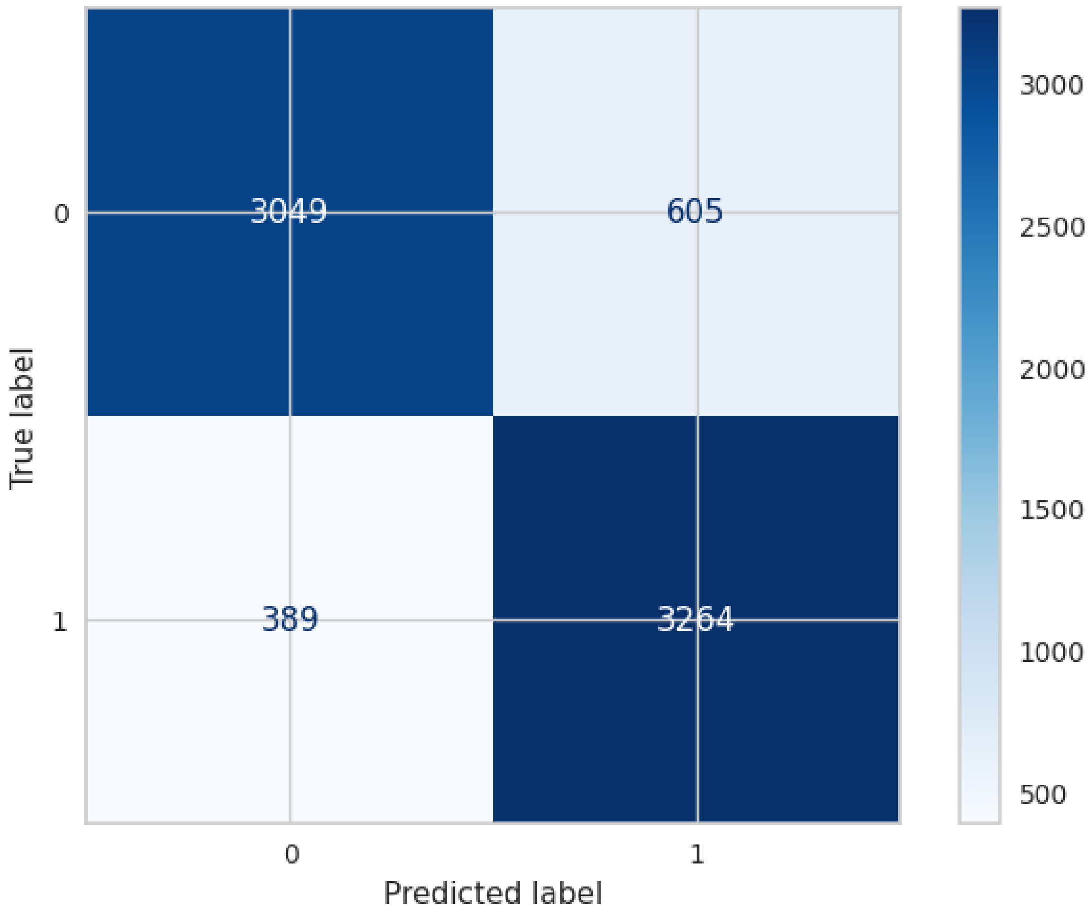
```
    accuracy                           0.86      7307
   macro avg       0.87      0.86      0.86      7307
weighted avg       0.87      0.86      0.86      7307

Confusion Matrix is :
 [[3049  605]
 [ 389 3264]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.8420388364277785
Model Test Score is :  0.8437115095114274
F1 Score is :  0.8508749020632019
Recall Score is :  0.8918696961401588
Precision Score is :  0.8134831460674158
AUC Value  :  0.8437180993016064

Classification Report is :                precision    recall  f1-score
support

           0       0.88      0.80      0.84      3654
           1       0.81      0.89      0.85      3653

    accuracy                           0.84      7307
   macro avg       0.85      0.84      0.84      7307
weighted avg       0.85      0.84      0.84      7307

Confusion Matrix is :
 [[2907  747]
 [ 395 3258]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.863449051898484
Model Test Score is :  0.8642397700834816
F1 Score is :  0.86805001330141
Recall Score is :  0.8932384341637011
Precision Score is :  0.8442432082794308
AUC Value  :  0.8642437381546474

Classification Report is :                precision    recall  f1-score
support

           0       0.89      0.84      0.86      3654
           1       0.84      0.89      0.87      3653
```

```
   accuracy                           0.86      7307
  macro avg       0.87      0.86      0.86      7307
weighted avg       0.87      0.86      0.86      7307

Confusion Matrix is :
 [[3052  602]
 [ 390 3263]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  0.8628560132597357
Model Test Score is :  0.8639660599425208
F1 Score is :  0.8678542940707258
Recall Score is :  0.8935121817684095
Precision Score is :  0.8436288446627035
AUC Value  :  0.8639701029258029

Classification Report is :                precision    recall  f1-score
support

           0       0.89      0.83      0.86      3654
           1       0.84      0.89      0.87      3653

    accuracy                           0.86      7307
   macro avg       0.87      0.86      0.86      7307
weighted avg       0.87      0.86      0.86      7307

Confusion Matrix is :
 [[3049  605]
 [ 389 3264]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  0.8419475997141249
Model Test Score is :  0.843574654440947
F1 Score is :  0.8507638072855463
Recall Score is :  0.8918696961401588
Precision Score is :  0.8132800798801797
AUC Value  :  0.8435812629578735

Classification Report is :                precision    recall  f1-score
support

           0       0.88      0.80      0.84      3654
           1       0.81      0.89      0.85      3653

    accuracy                           0.84      7307
```

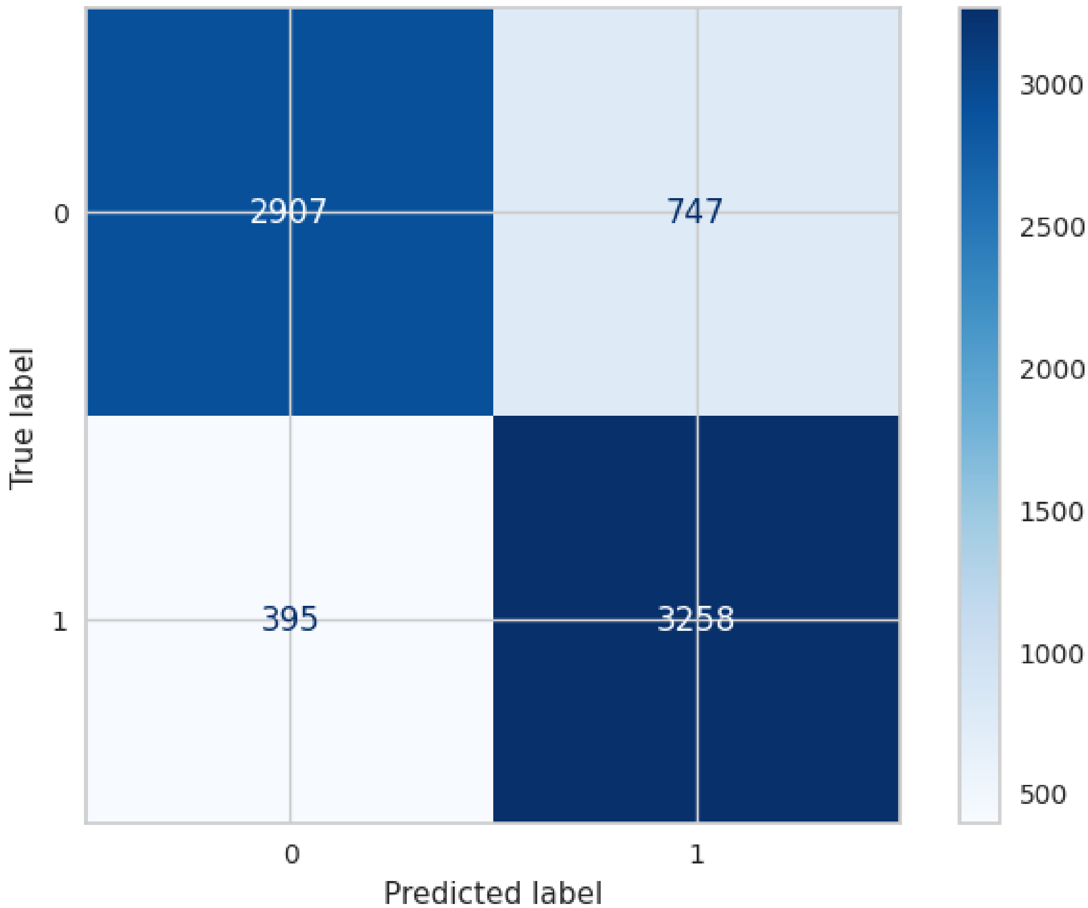
```
   macro avg       0.85      0.84      0.84      7307
weighted avg       0.85      0.84      0.84      7307

Confusion Matrix is :
 [[2906  748]
 [ 395 3258]]
```



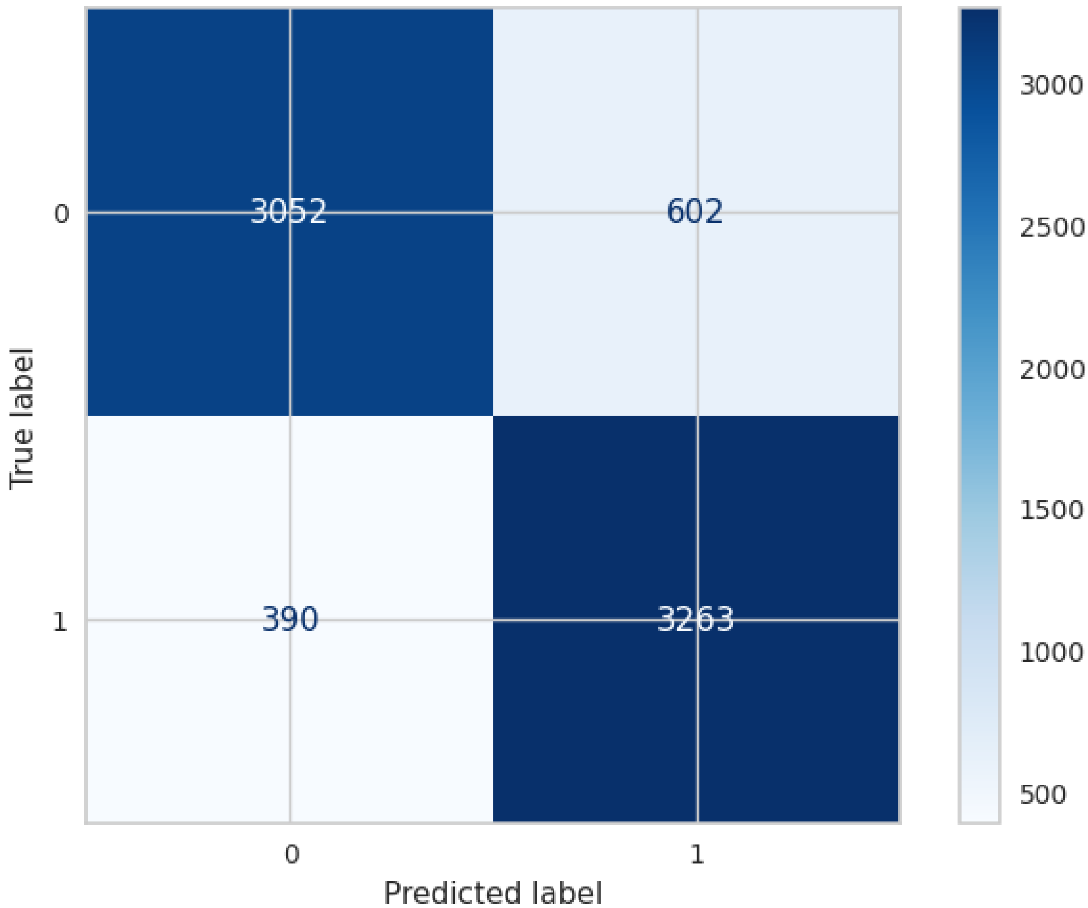

0
1
True label
3008
646
414
3239
0
1
Predicted label
3000
2500
2000
1500
1000
500

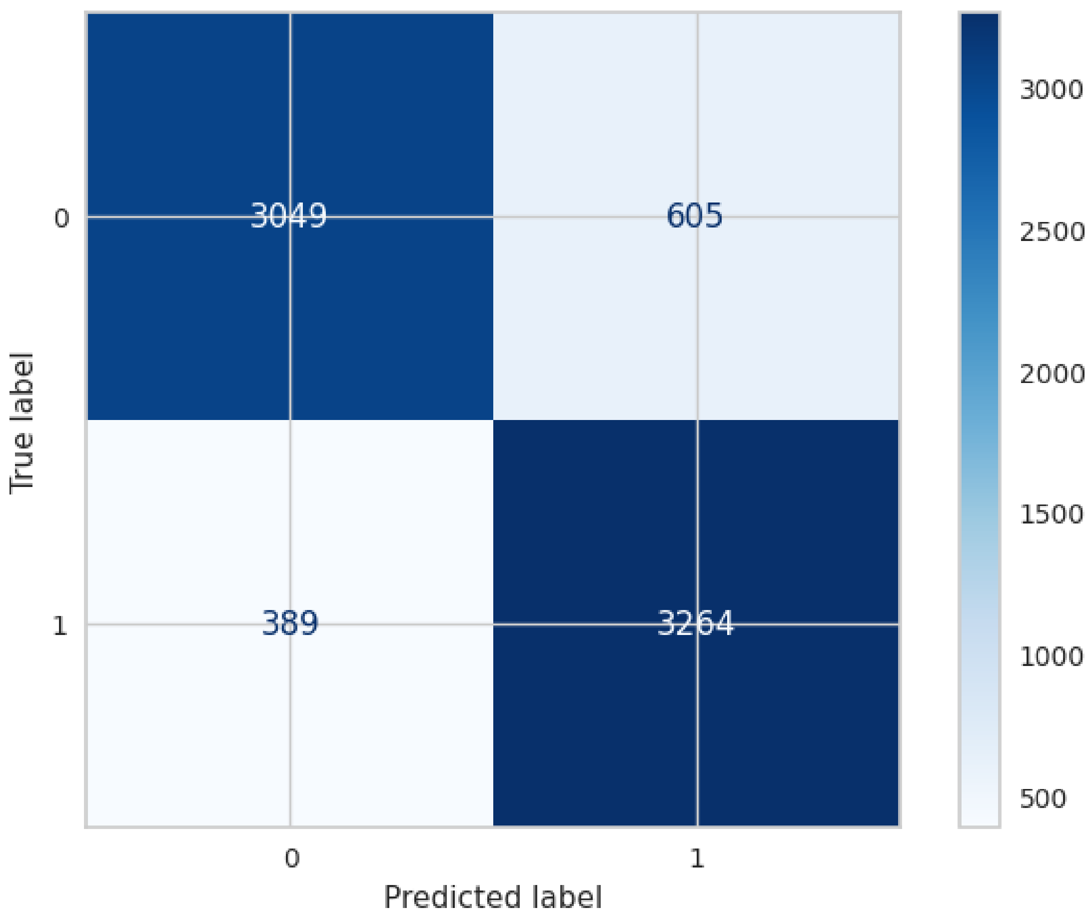


3049
605
389
3264
0
1
True label
0
1
Predicted label
3000
2500
2000
1500
1000
500

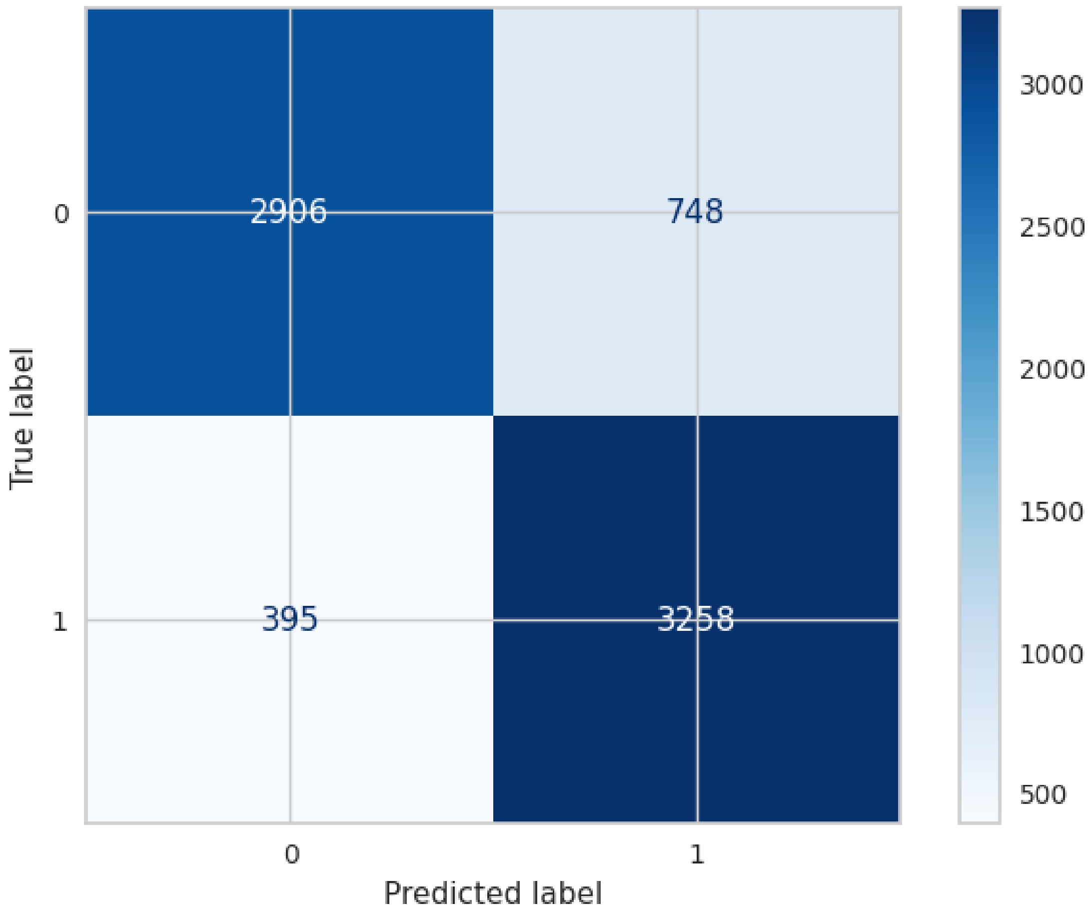


0
1
2907
747
395
3258
True label
0
1
Predicted label
3000
2500
2000
1500
1000
500


3052
602
390
3263
True label
Predicted label
0
1
0
1
3000
2500
2000
1500
1000
500



3049
605
389
3264
0
1
True label
0
1
Predicted label
3000
2500
2000
1500
1000
500



[289]:
```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['Logistic Over','Logistic Over With Feature','Logistic Over␣
↪Scaling','Logistic Over With Normalize','Logistic Over With PCA'
                ,'Logistic Over With PCA and Scaling',
                'Logistic Over With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[289]:

Models	Train Accuracy	Test Accuracy	Test F1	\
Logistic Over	0.863160	0.864103	0.867970	
Logistic Over With Feature	0.849429	0.854934	0.859379	
Logistic Over Scaling	0.862780	0.863966	0.867854	
Logistic Over With Normalize	0.842039	0.843712	0.850875	
Logistic Over With PCA	0.863449	0.864240	0.868050	
Logistic Over With PCA and Scaling	0.862856	0.863966	0.867854	
Logistic Over With PCA and Normalize	0.841948	0.843575	0.850764	

```
                                          Test Recall  Test Precision       AUC
 Models
 Logistic Over                               0.893512        0.843847  0.864107
 Logistic Over With Feature                  0.886668        0.833719  0.854938
 Logistic Over Scaling                       0.893512        0.843629  0.863970
 Logistic Over With Normalize                0.891870        0.813483  0.843718
 Logistic Over With PCA                      0.893238        0.844243  0.864244
 Logistic Over With PCA and Scaling          0.893512        0.843629  0.863970
 Logistic Over With PCA and Normalize        0.891870        0.813280  0.843581
```

[290]:
```
models_draw(df)
```

RandomUnderSampler

[291]:
```
X_train,y_train,X_test,y_test=Split(X_classification_under,y_classification_under)
```

```
X_train shape is  (8350, 20)
X_test shape is  (928, 20)
y_train shape is  (8350,)
y_test shape is  (928,)
```

[292]:
```
Search(LogisticRegression(penalty='l2',solver='sag',C=1.0),{'C':[1,.
 ↪5,2,3,5,10]},X_train,y_train)
```

[292]:
```
LogisticRegression(C=0.5, solver='sag')
```

[293]:
```
cross_validation(LogisticRegression(penalty='l2',solver='sag',C=10),X_train,y_train)
```

```
Train Score Value :  [0.86047904 0.86182635 0.85928144 0.86347305 0.85733533]
Mean 0.8604790419161675
Test Score Value :  [0.85928144 0.85149701 0.86227545 0.85508982 0.86826347]
Mean 0.8592814371257484
```

[294]:
```
Values =␣
 ↪Models(LogisticRegression(penalty='l2',solver='sag',C=10),X_train,y_train,X_test,y_test)
```

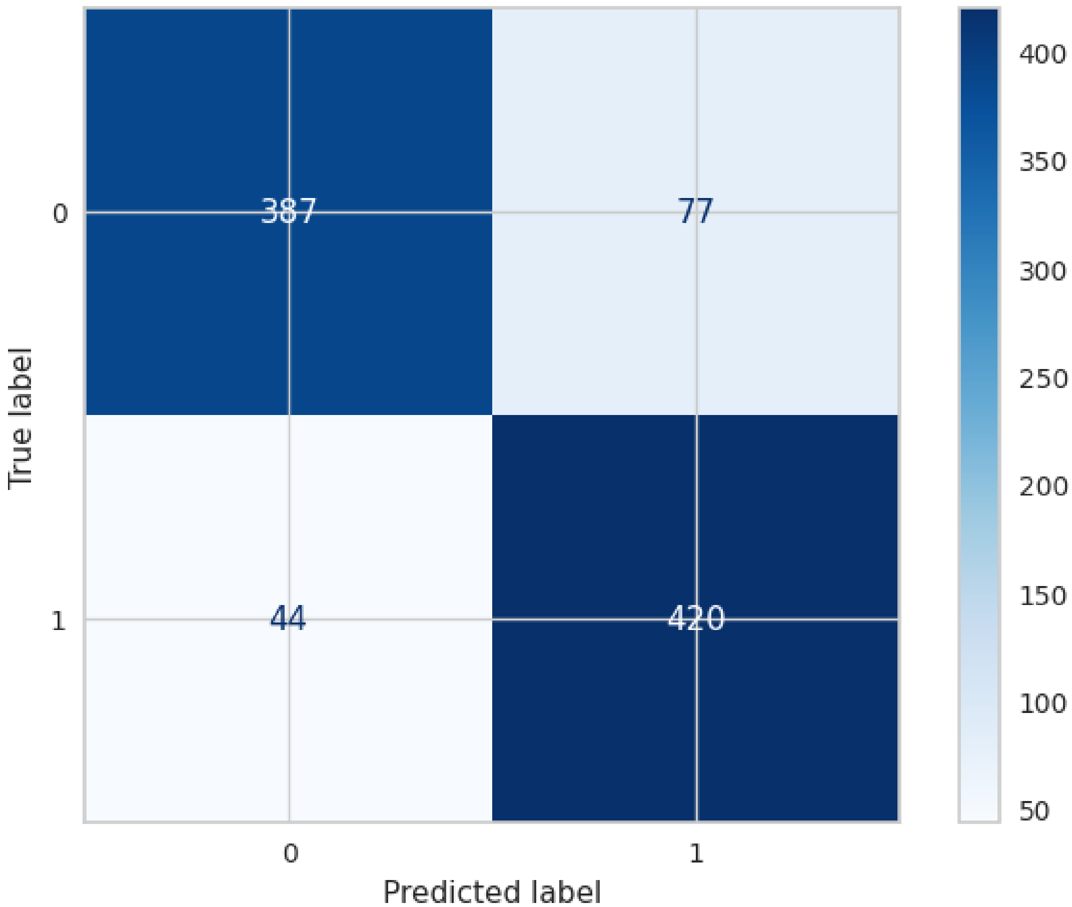
```
Apply Model With Normal Data :

Model Train Score is :  0.8608383233532935
Model Test Score is :  0.8696120689655172
F1 Score is :  0.874089490114464
Recall Score is :  0.9051724137931034
Precision Score is :  0.8450704225352113
AUC Value  :  0.8696120689655172

Classification Report is :                precision    recall  f1-score
support

           0       0.90      0.83      0.86       464
```

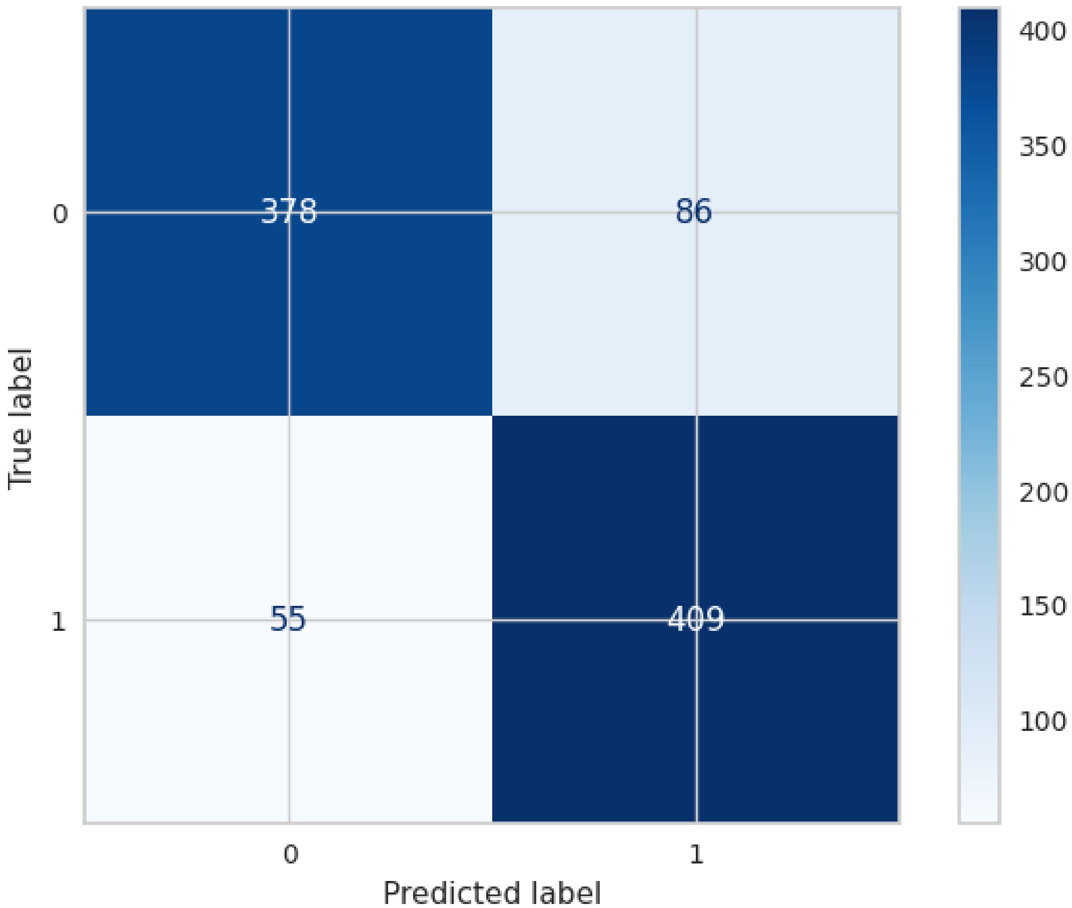
```
           1       0.85      0.91      0.87       464

    accuracy                           0.87       928
   macro avg       0.87      0.87      0.87       928
weighted avg       0.87      0.87      0.87       928

Confusion Matrix is :
 [[387  77]
 [ 44 420]]


 Apply Model With Feature Selection :

Model Train Score is :  0.8508982035928143
Model Test Score is :  0.8480603448275862
F1 Score is :  0.8529718456725757
Recall Score is :  0.8814655172413793
Precision Score is :  0.8262626262626263
AUC Value  :  0.8480603448275863
Classification Report is :                precision    recall  f1-score
support

           0       0.87      0.81      0.84       464
           1       0.83      0.88      0.85       464

    accuracy                           0.85       928
   macro avg       0.85      0.85      0.85       928
weighted avg       0.85      0.85      0.85       928

Confusion Matrix is :
 [[378  86]
 [ 55 409]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.864311377245509
Model Test Score is :  0.8739224137931034
F1 Score is :  0.8767123287671231
Recall Score is :  0.896551724137931
Precision Score is :  0.8577319587628865
AUC Value  :  0.8739224137931034
Classification Report is :                precision    recall  f1-score
support

           0       0.89      0.85      0.87       464
           1       0.86      0.90      0.88       464
```

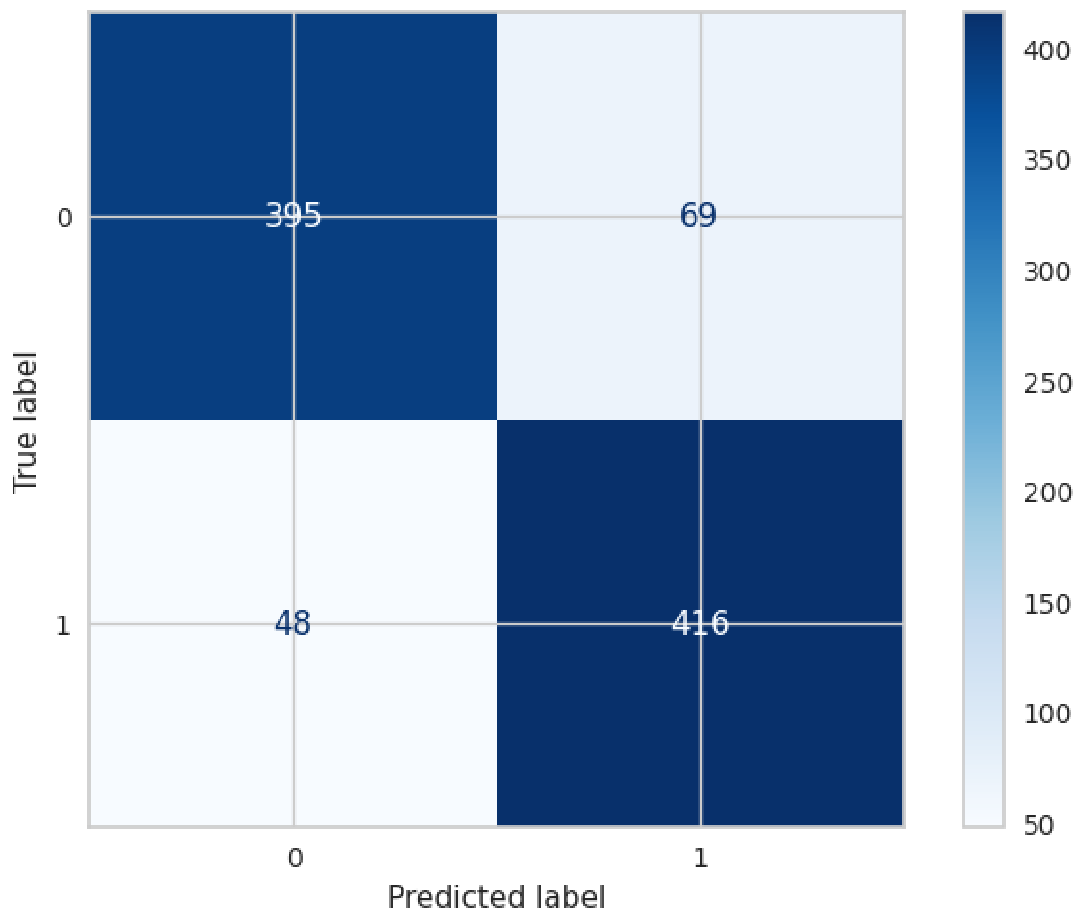
```
    accuracy                           0.87       928
   macro avg       0.87      0.87      0.87       928
weighted avg       0.87      0.87      0.87       928

Confusion Matrix is :
 [[395  69]
 [ 48 416]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.844311377245509
Model Test Score is :  0.8491379310344828
F1 Score is :  0.8553719008264462
Recall Score is :  0.8922413793103449
Precision Score is :  0.8214285714285714
AUC Value  :  0.8491379310344828
Classification Report is :                precision    recall  f1-score
support

           0       0.88      0.81      0.84       464
           1       0.82      0.89      0.86       464

    accuracy                           0.85       928
   macro avg       0.85      0.85      0.85       928
weighted avg       0.85      0.85      0.85       928

Confusion Matrix is :
 [[374  90]
 [ 50 414]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.8616766467065868
Model Test Score is :  0.8696120689655172
F1 Score is :  0.8730325288562434
Recall Score is :  0.896551724137931
Precision Score is :  0.8507157464212679
AUC Value  :  0.8696120689655171
Classification Report is :                precision    recall  f1-score
support

           0       0.89      0.84      0.87       464
           1       0.85      0.90      0.87       464
```

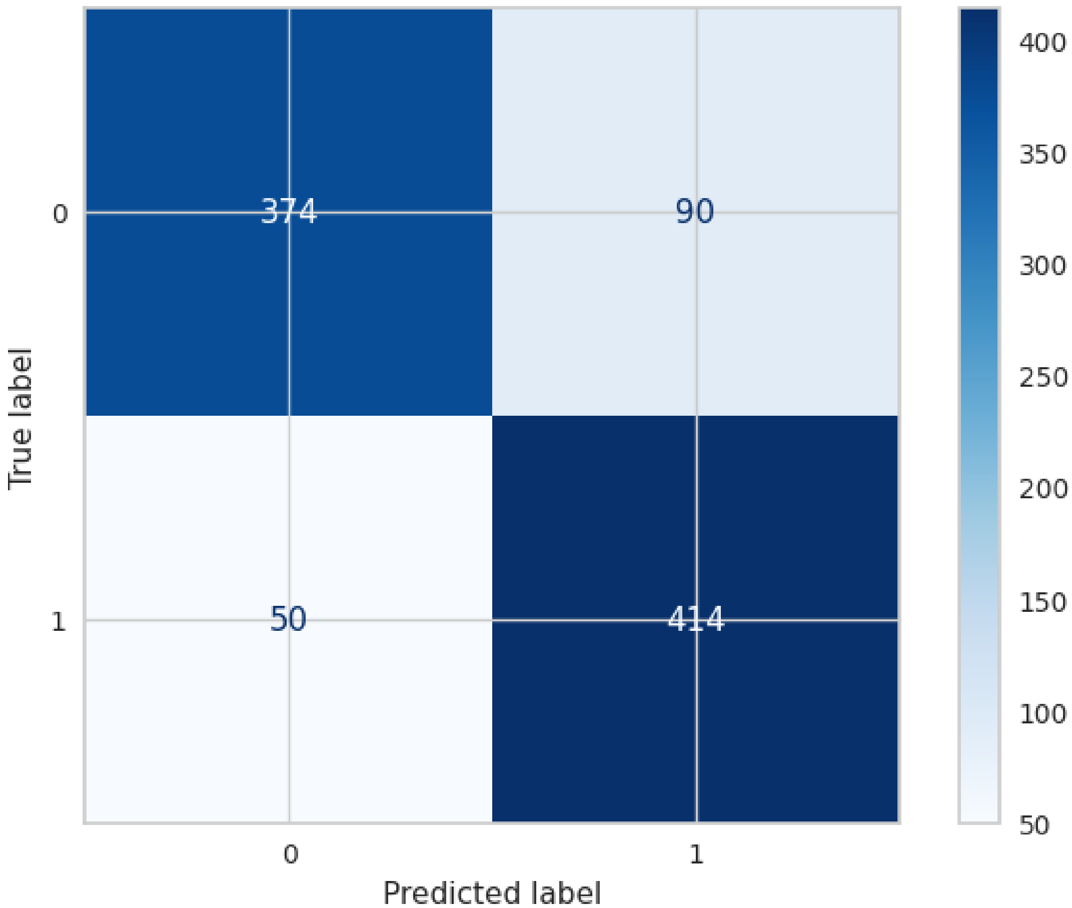
```
    accuracy                           0.87       928
   macro avg       0.87      0.87      0.87       928
weighted avg       0.87      0.87      0.87       928

Confusion Matrix is :
 [[391  73]
 [ 48 416]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  0.864311377245509
Model Test Score is :  0.8739224137931034
F1 Score is :  0.8767123287671231
Recall Score is :  0.896551724137931
Precision Score is :  0.8577319587628865
AUC Value  :  0.8739224137931034

Classification Report is :                precision    recall  f1-score
support

           0       0.89      0.85      0.87       464
           1       0.86      0.90      0.88       464

    accuracy                           0.87       928
   macro avg       0.87      0.87      0.87       928
weighted avg       0.87      0.87      0.87       928

Confusion Matrix is :
 [[395  69]
 [ 48 416]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  0.844311377245509
Model Test Score is :  0.8491379310344828
F1 Score is :  0.8553719008264462
Recall Score is :  0.8922413793103449
Precision Score is :  0.8214285714285714
AUC Value  :  0.8491379310344828

Classification Report is :                precision    recall  f1-score
support

           0       0.88      0.81      0.84       464
           1       0.82      0.89      0.86       464

    accuracy                           0.85       928
```

```
   macro avg       0.85      0.85      0.85       928
weighted avg       0.85      0.85      0.85       928

Confusion Matrix is :
 [[374  90]
 [ 50 414]]
```

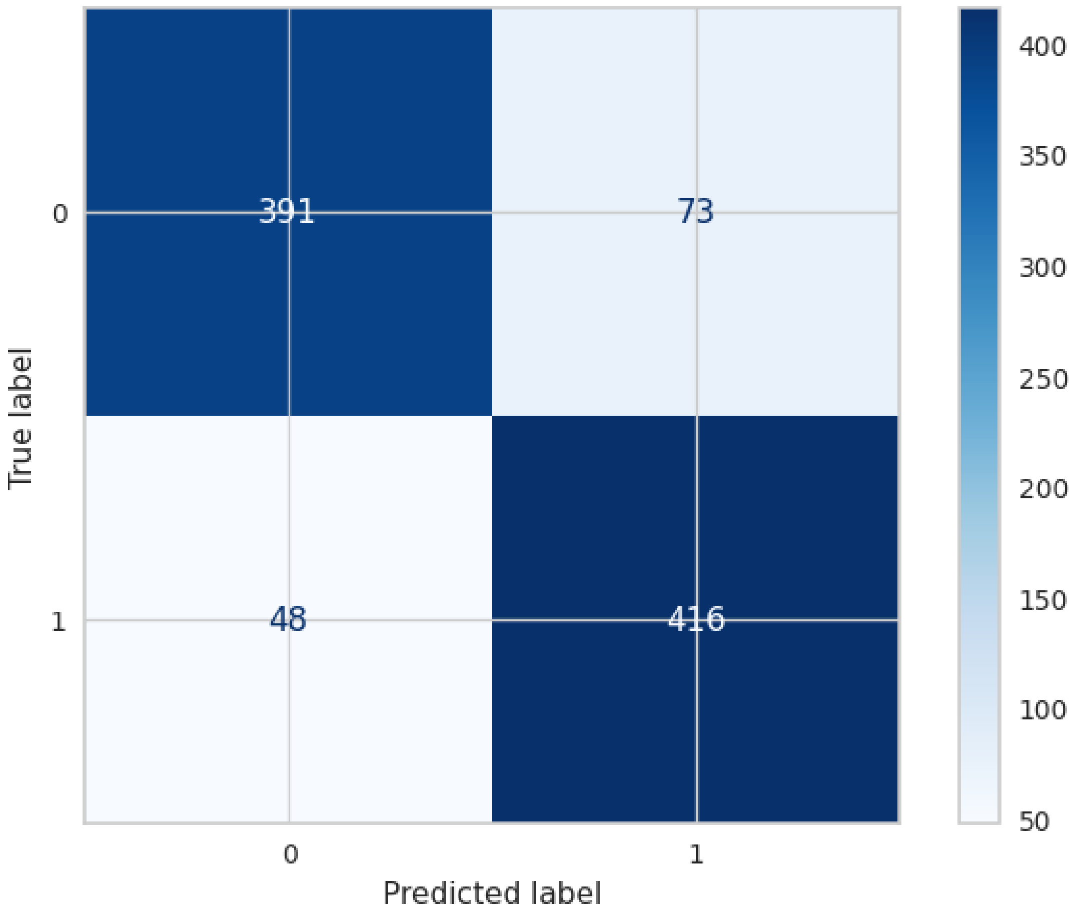



378
86
55
409
True label
Predicted label
0
1
0
1
400
350
300
250
200
150
100

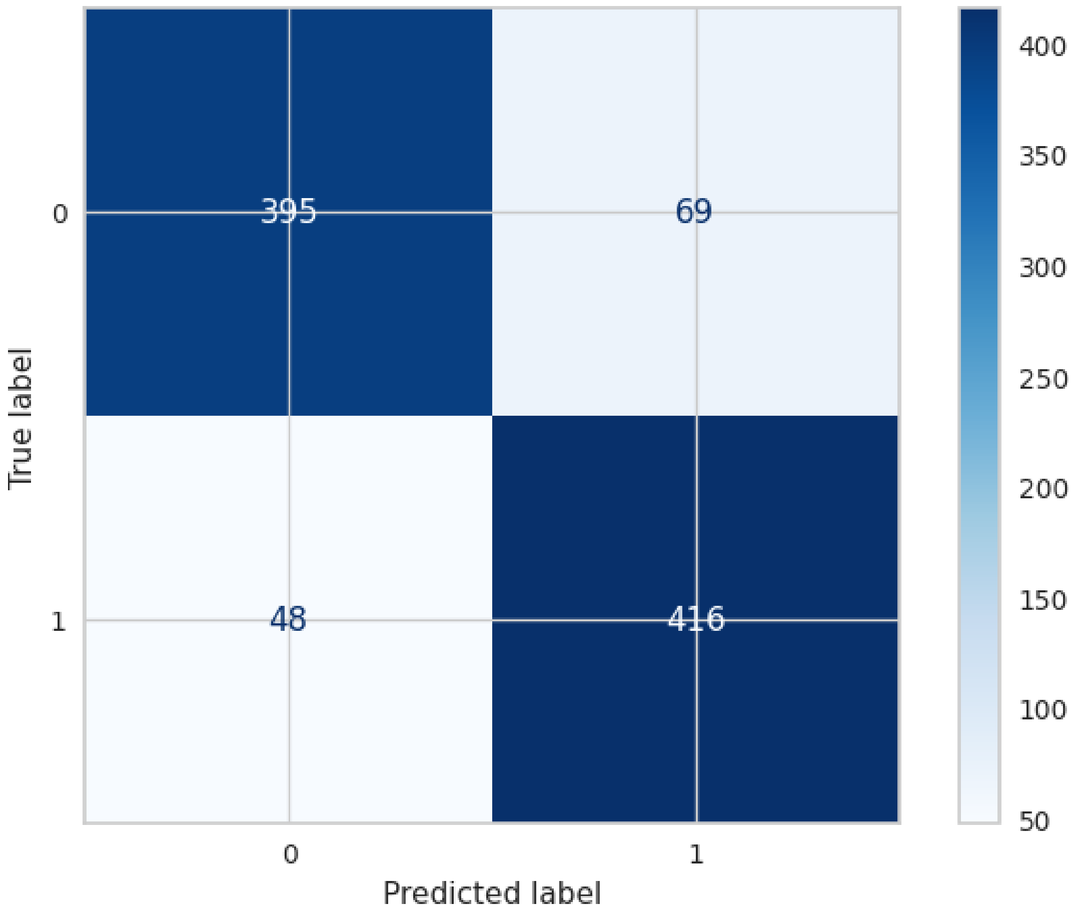

395
69
48
416
True label
Predicted label
0
1
0
1
400
350
300
250
200
150
100
50

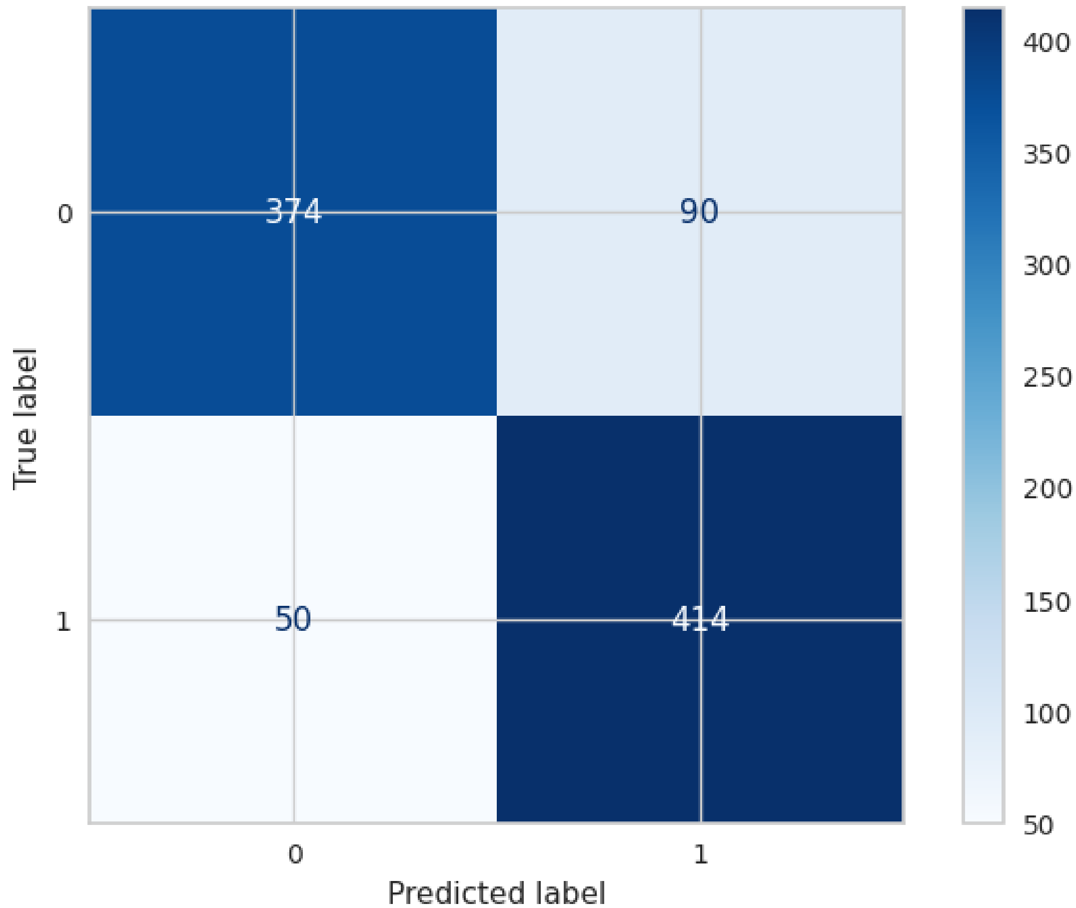


374
90
50
414
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50



391
73
48
416
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50


395
69
48
416
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50



[295]:
```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['Logistic Under','Logistic Under With Feature','Logistic Under␣
↪Scaling','Logistic Under With Normalize','Logistic Under With PCA'
                ,'Logistic Under With PCA and Scaling',
                'Logistic Under With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[295]:
```
                                       Train Accuracy  Test Accuracy  \
Models
Logistic Under                               0.860838       0.869612
Logistic Under With Feature                  0.850898       0.848060
Logistic Under Scaling                       0.864311       0.873922
Logistic Under With Normalize                0.844311       0.849138
Logistic Under With PCA                      0.861677       0.869612
Logistic Under With PCA and Scaling          0.864311       0.873922
Logistic Under With PCA and Normalize        0.844311       0.849138
```

```
                                    Test F1  Test Recall  Test Precision  \
Models
Logistic Under                      0.874089     0.905172        0.845070
Logistic Under With Feature         0.852972     0.881466        0.826263
Logistic Under Scaling              0.876712     0.896552        0.857732
Logistic Under With Normalize       0.855372     0.892241        0.821429
Logistic Under With PCA             0.873033     0.896552        0.850716
Logistic Under With PCA and Scaling 0.876712     0.896552        0.857732
Logistic Under With PCA and Normalize 0.855372   0.892241        0.821429

                                         AUC
Models
Logistic Under                      0.869612
Logistic Under With Feature         0.848060
Logistic Under Scaling              0.873922
Logistic Under With Normalize       0.849138
Logistic Under With PCA             0.869612
Logistic Under With PCA and Scaling 0.873922
Logistic Under With PCA and Normalize 0.849138
```

[296]:
```
models_draw(df)
```

GaussianNB

[297]:
```
X_train,y_train,X_test,y_test=Split(X_classification,y_classification)
```

```
X_train shape is  (37056, 20)
X_test shape is  (4118, 20)
y_train shape is  (37056,)
y_test shape is  (4118,)
```

[298]:
```
cross_validation(GaussianNB(),X_train,y_train)
```

```
Train Score Value :  [0.83946161 0.84091752 0.84358239 0.85208298 0.83973689]
Mean 0.8431562790461198
Test Score Value :  [0.83904479 0.83969775 0.84347591 0.84995277 0.84172176]
Mean 0.8427785981704972
```

[299]:
```
Values = Models(GaussianNB(),X_train,y_train,X_test,y_test)
```

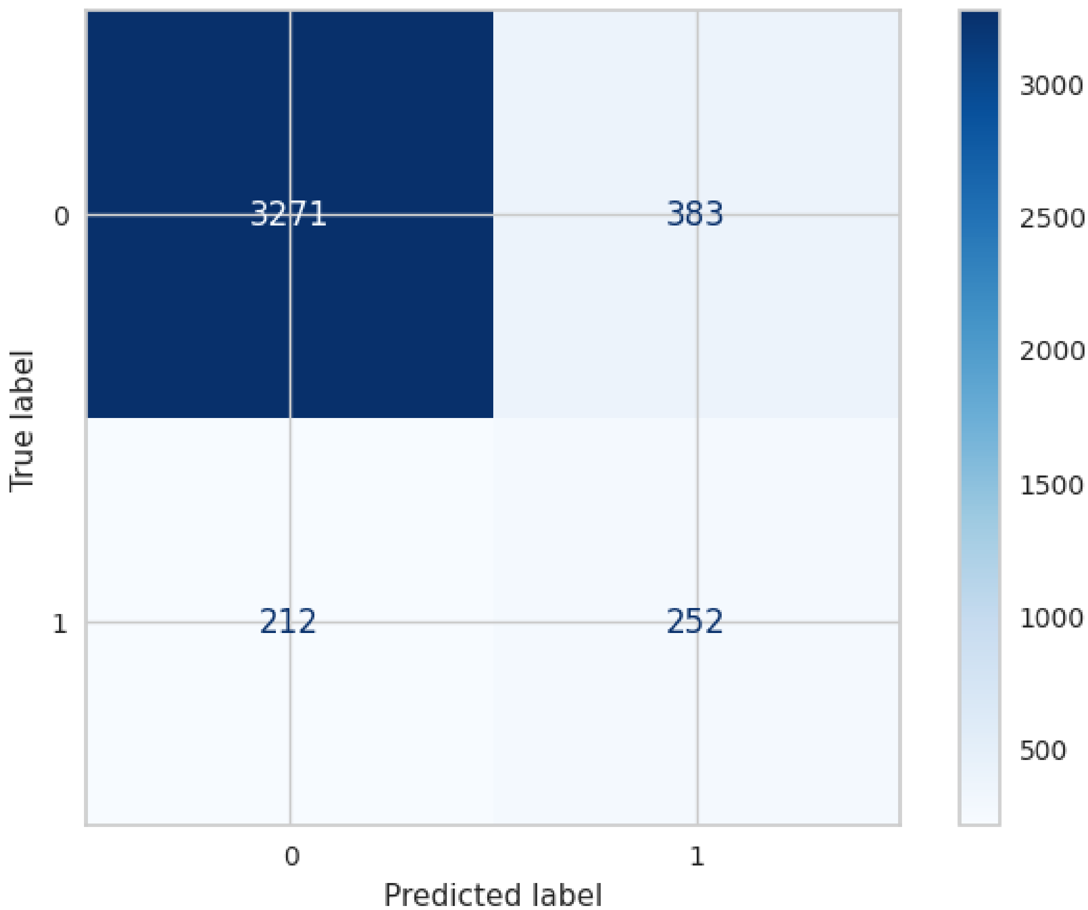
```
Apply Model With Normal Data :

Model Train Score is :  0.8424006908462867
Model Test Score is :  0.855512384652744
F1 Score is :  0.4585987261146497
Recall Score is :  0.5431034482758621
Precision Score is :  0.3968503937007874
AUC Value  :  0.7191434044882321
```

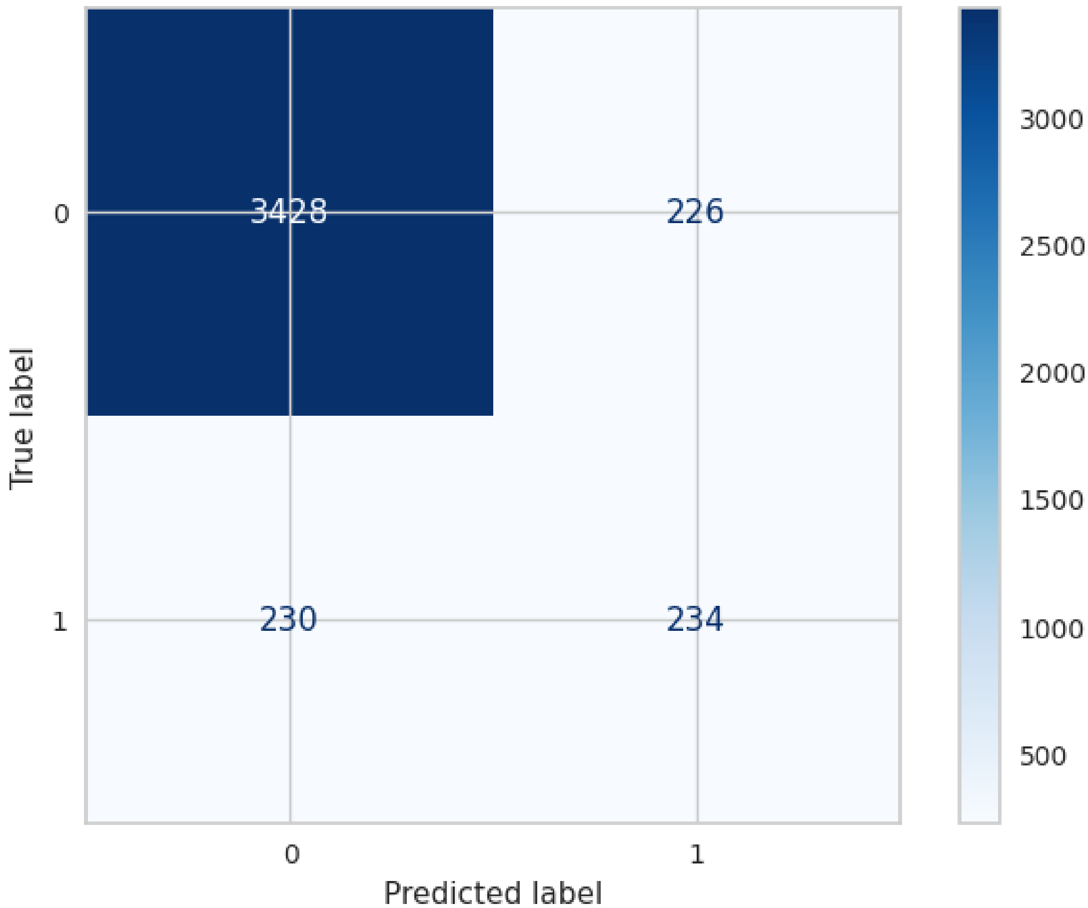
```
Classification Report is :               precision    recall  f1-score
support

           0       0.94      0.90      0.92      3654
           1       0.40      0.54      0.46       464

    accuracy                           0.86      4118
   macro avg       0.67      0.72      0.69      4118
weighted avg       0.88      0.86      0.87      4118

Confusion Matrix is :
 [[3271  383]
 [ 212  252]]


 Apply Model With Feature Selection :

Model Train Score is :  0.8853626943005182
Model Test Score is :  0.8892666342884895
F1 Score is :  0.5064935064935064
Recall Score is :  0.5043103448275862
Precision Score is :  0.508695652173913
AUC Value  :  0.7212301587301588
Classification Report is :               precision    recall  f1-score
support

           0       0.94      0.94      0.94      3654
           1       0.51      0.50      0.51       464

    accuracy                           0.89      4118
   macro avg       0.72      0.72      0.72      4118
weighted avg       0.89      0.89      0.89      4118

Confusion Matrix is :
 [[3428  226]
 [ 230  234]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.7029900690846287
Model Test Score is :  0.7102962603205439
F1 Score is :  0.406172224987556
Recall Score is :  0.8793103448275862
Precision Score is :  0.26407766990291265
AUC Value  :  0.784072249589491
Classification Report is :               precision    recall  f1-score
```

```
support

           0       0.98      0.69      0.81      3654
           1       0.26      0.88      0.41       464

    accuracy                           0.71      4118
   macro avg       0.62      0.78      0.61      4118
weighted avg       0.90      0.71      0.76      4118

Confusion Matrix is :
 [[2517 1137]
 [  56  408]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.8132286269430051
Model Test Score is :  0.8142302088392424
F1 Score is :  0.5215759849906191
Recall Score is :  0.8987068965517241
Precision Score is :  0.3674008810572687
AUC Value  :  0.8511049534756432

Classification Report is :                precision    recall  f1-score
support

           0       0.98      0.80      0.88      3654
           1       0.37      0.90      0.52       464

    accuracy                           0.81      4118
   macro avg       0.68      0.85      0.70      4118
weighted avg       0.91      0.81      0.84      4118

Confusion Matrix is :
 [[2936  718]
 [  47  417]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.8778875215889465
Model Test Score is :  0.8827100534239922
F1 Score is :  0.4627363737486096
Recall Score is :  0.4482758620689655
Precision Score is :  0.4781609195402299
AUC Value  :  0.6930760810071155

Classification Report is :                precision    recall  f1-score
support
```

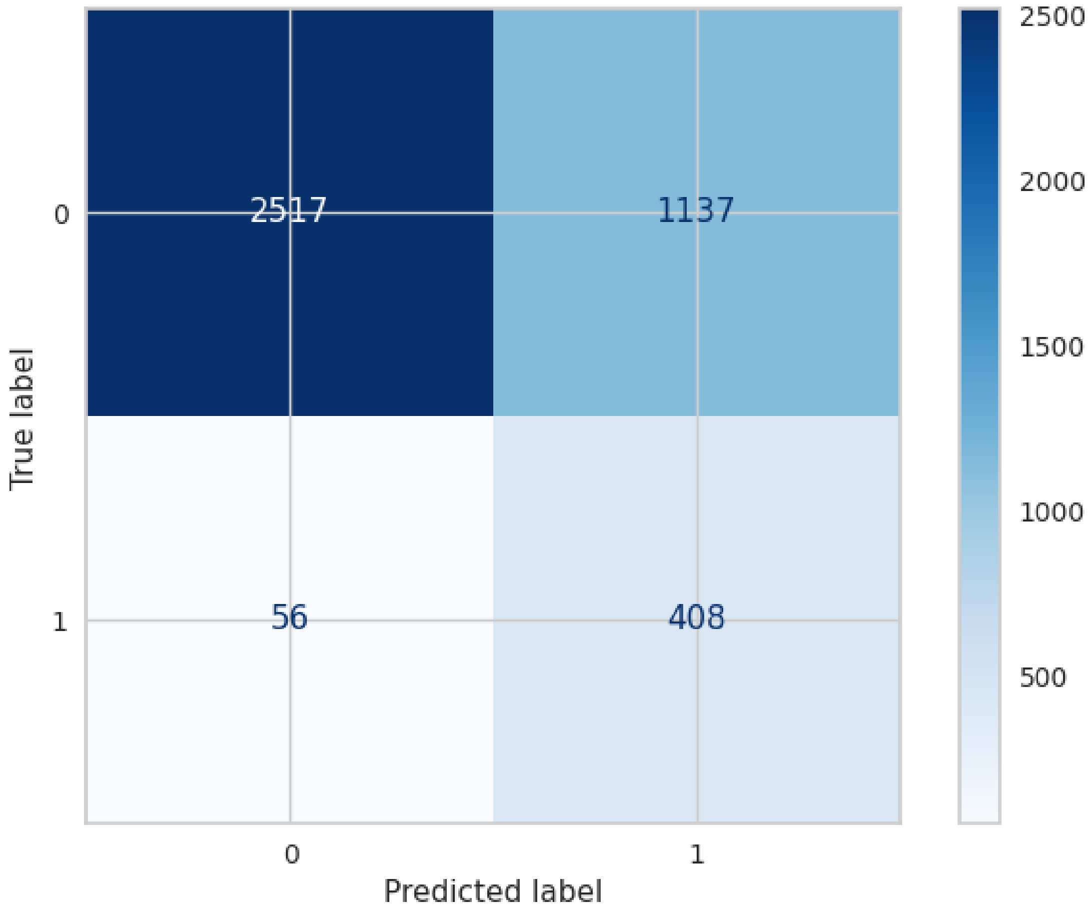
```
           0       0.93      0.94      0.93      3654
           1       0.48      0.45      0.46       464

    accuracy                           0.88      4118
   macro avg       0.70      0.69      0.70      4118
weighted avg       0.88      0.88      0.88      4118

Confusion Matrix is :
 [[3427  227]
 [ 256  208]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  0.8537618739205527
Model Test Score is :  0.861583292860612
F1 Score is :  0.5086206896551724
Recall Score is :  0.6357758620689655
Precision Score is :  0.4238505747126437
AUC Value  :  0.7630165571975918

Classification Report is :               precision    recall  f1-score
support

           0       0.95      0.89      0.92      3654
           1       0.42      0.64      0.51       464

    accuracy                           0.86      4118
   macro avg       0.69      0.76      0.71      4118
weighted avg       0.89      0.86      0.87      4118

Confusion Matrix is :
 [[3253  401]
 [ 169  295]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  0.8816655872193437
Model Test Score is :  0.8858669256920836
F1 Score is :  0.5164609053497943
Recall Score is :  0.540948275862069
Precision Score is :  0.4940944881889764
AUC Value  :  0.7353071975916803

Classification Report is :               precision    recall  f1-score
support
```

```
           0       0.94      0.93      0.94      3654
           1       0.49      0.54      0.52       464

    accuracy                           0.89      4118
   macro avg       0.72      0.74      0.73      4118
weighted avg       0.89      0.89      0.89      4118

Confusion Matrix is :
 [[3397  257]
 [ 213  251]]
```

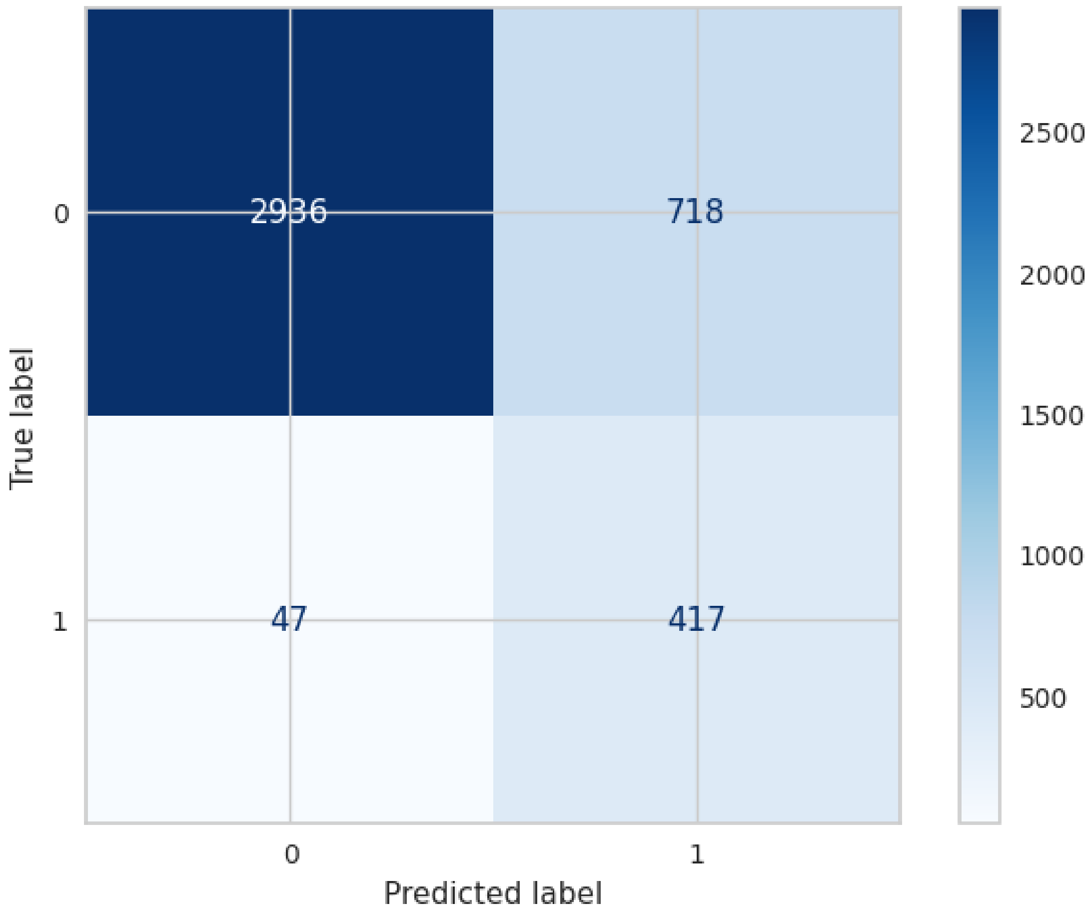


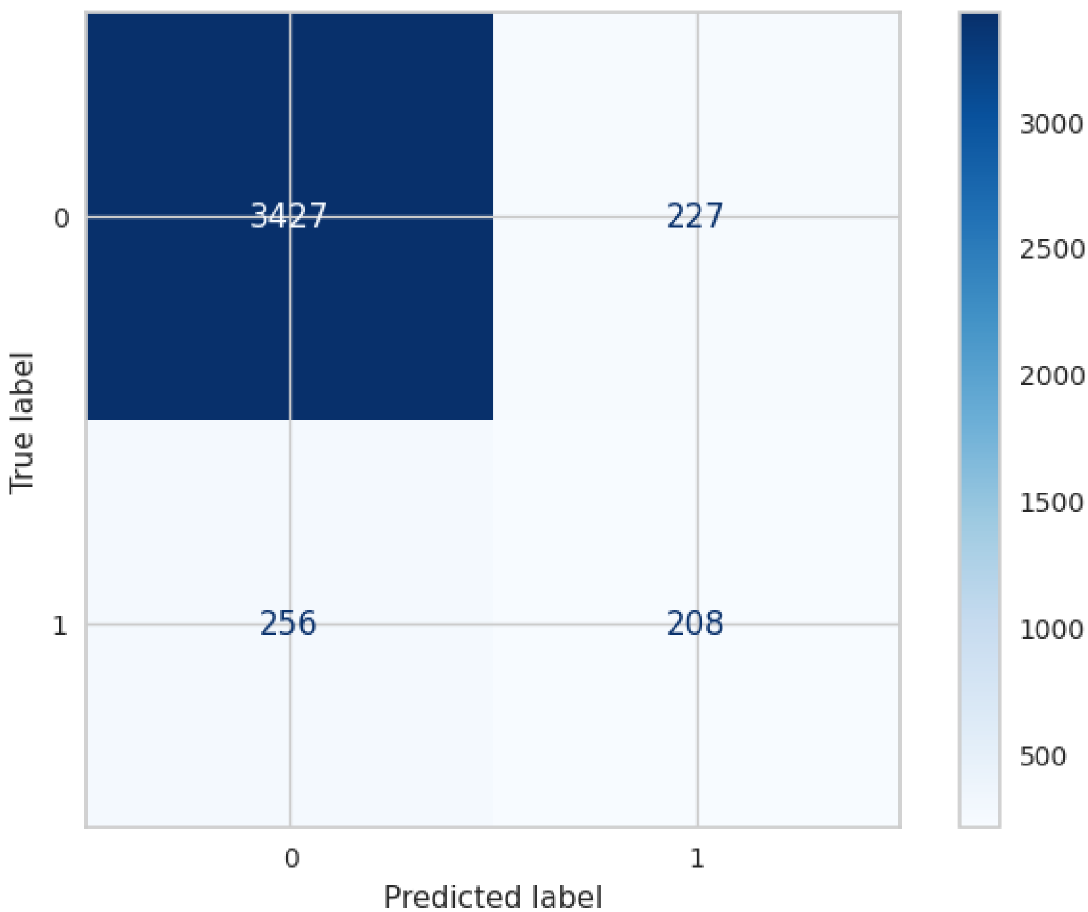


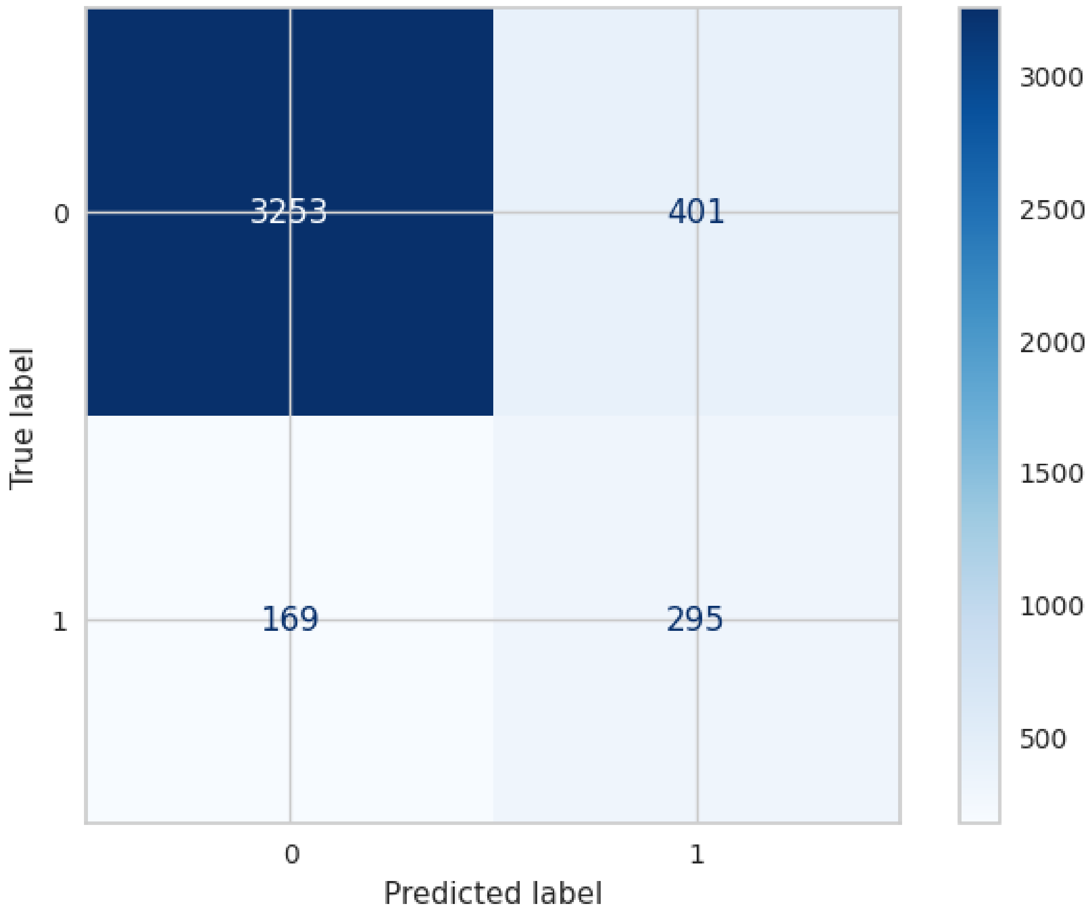
3428
226
230
234
0
1
True label
0
1
Predicted label
3000
2500
2000
1500
1000
500

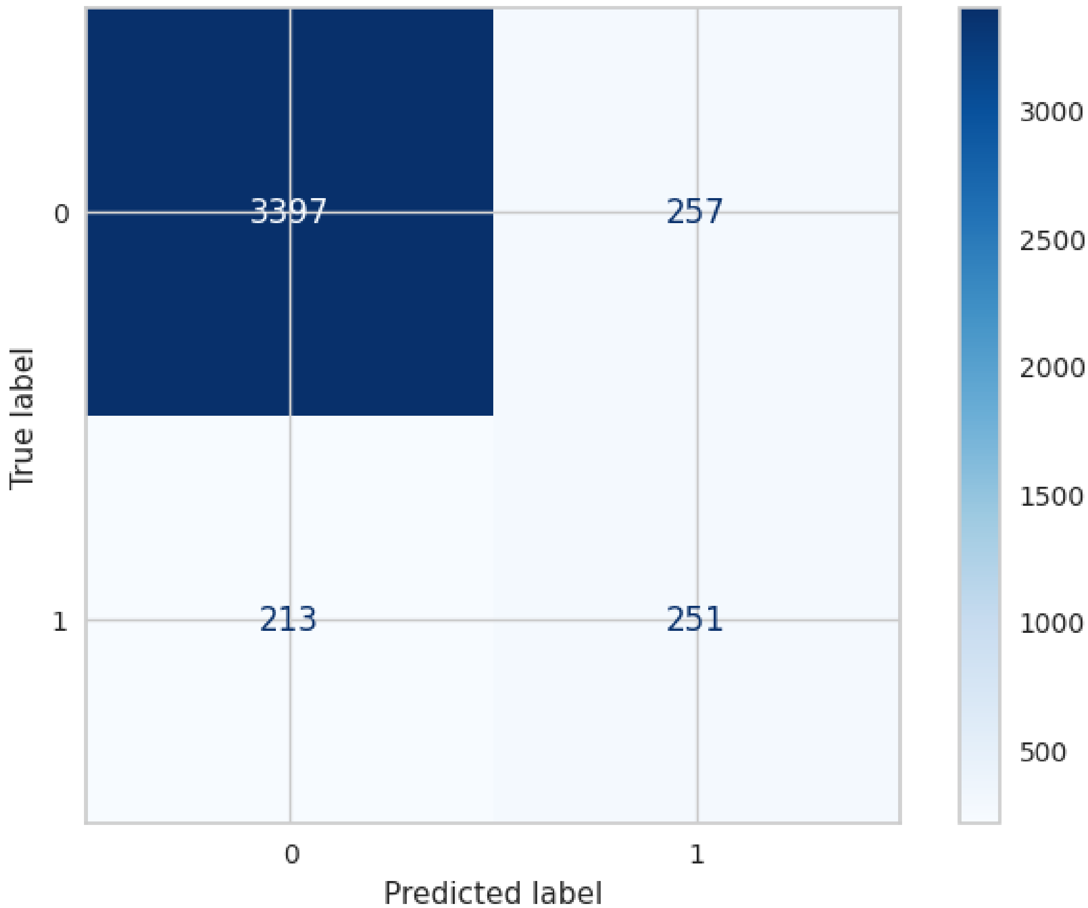


2517
1137
56
408
0
1
True label
0
1
Predicted label
2500
2000
1500
1000
500


2936
718
47
417
0
1
True label
0
1
Predicted label
2500
2000
1500
1000
500


3427
227
256
208
0
1
0
1
True label
Predicted label
3000
2500
2000
1500
1000
500


3253
401
169
295
0
1
True label
0
1
Predicted label
3000
2500
2000
1500
1000
500



[300]:

```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['NB','NB With Feature','NB Scaling','NB With Normalize','NB␣
↪With PCA'
                ,'NB With PCA and Scaling',
                'NB With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[300]:

```
                           Train Accuracy  Test Accuracy   Test F1  \
Models
NB                               0.842401       0.855512  0.458599
NB With Feature                  0.885363       0.889267  0.506494
NB Scaling                       0.702990       0.710296  0.406172
NB With Normalize                0.813229       0.814230  0.521576
NB With PCA                      0.877888       0.882710  0.462736
NB With PCA and Scaling          0.853762       0.861583  0.508621
NB With PCA and Normalize        0.881666       0.885867  0.516461
```

```
                            Test Recall  Test Precision       AUC
Models
NB                             0.543103        0.396850  0.719143
NB With Feature                0.504310        0.508696  0.721230
NB Scaling                     0.879310        0.264078  0.784072
NB With Normalize              0.898707        0.367401  0.851105
NB With PCA                    0.448276        0.478161  0.693076
NB With PCA and Scaling        0.635776        0.423851  0.763017
NB With PCA and Normalize      0.540948        0.494094  0.735307
```

[301]:
```
models_draw(df)
```

RandomOverSampler

[302]:
```
X_train,y_train,X_test,y_test=Split(X_classification_over,y_classification_over)
```

```
X_train shape is  (65763, 20)
X_test shape is  (7307, 20)
y_train shape is  (65763,)
y_test shape is  (7307,)
```

[303]:
```
cross_validation(GaussianNB(),X_train,y_train)
```

```
Train Score Value :  [0.73157194 0.73461319 0.7343851  0.7347893  0.73381992]
Mean 0.7338358920006607
Test Score Value :  [0.73671406 0.73618186 0.73777845 0.72825426 0.73000304]
Mean 0.7337863340272069
```

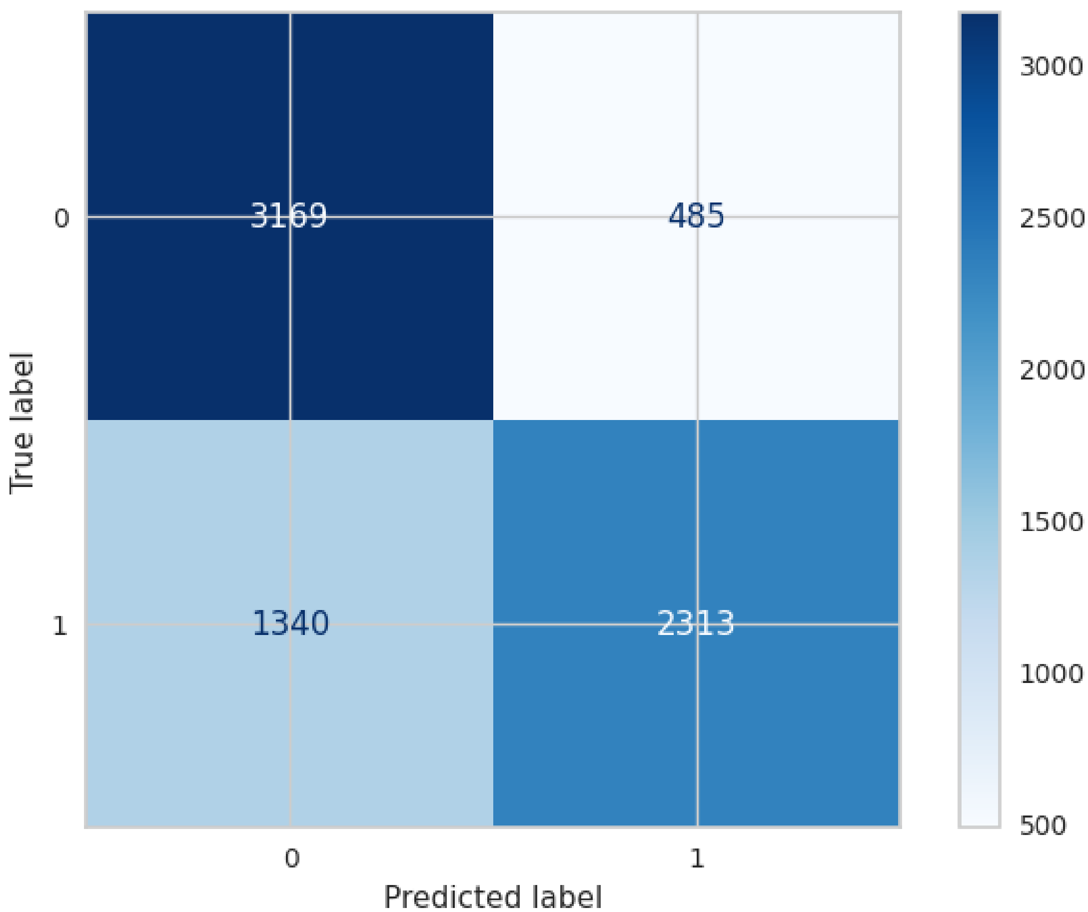
[304]:
```
Values = Models(GaussianNB(),X_train,y_train,X_test,y_test)
```

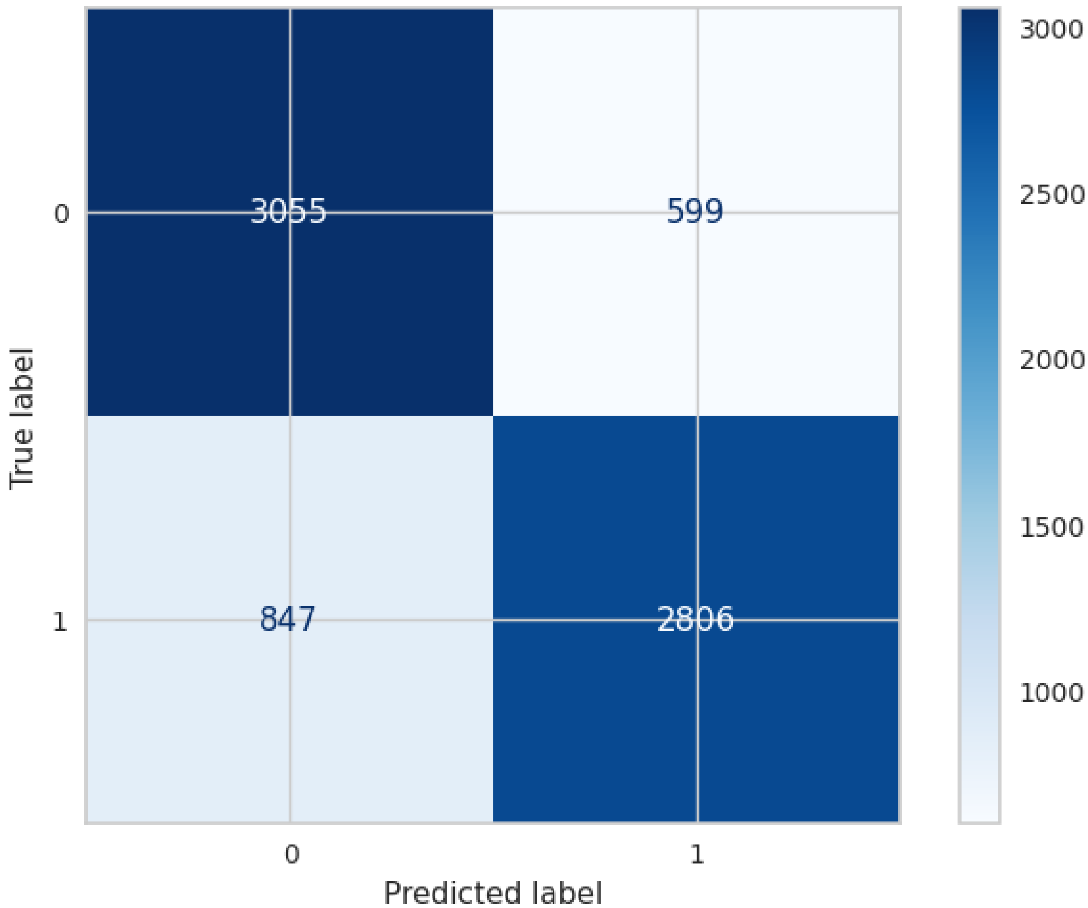
```
Apply Model With Normal Data :

Model Train Score is :  0.7342274531271383
Model Test Score is :  0.7502394963733406
F1 Score is :  0.7170981243218105
Recall Score is :  0.6331782096906652
Precision Score is :  0.8266619013581129
AUC Value  :  0.7502234781348783

Classification Report is :                precision    recall  f1-score
support

           0       0.70      0.87      0.78      3654
           1       0.83      0.63      0.72      3653

    accuracy                           0.75      7307
   macro avg       0.76      0.75      0.75      7307
weighted avg       0.76      0.75      0.75      7307
```

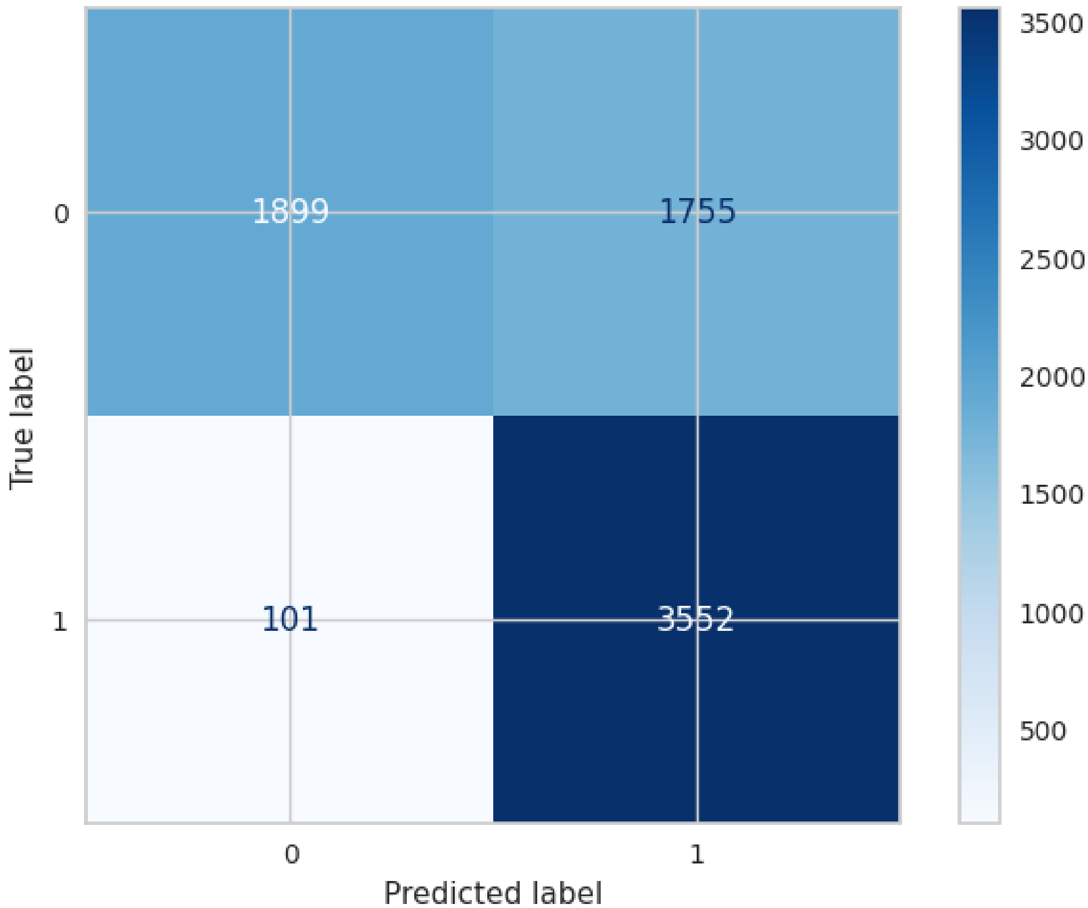
```
Confusion Matrix is :
 [[3169  485]
 [1340 2313]]


 Apply Model With Feature Selection :

Model Train Score is :  0.7964204796009915
Model Test Score is :  0.8021075680853975
F1 Score is :  0.7951260980447719
Recall Score is :  0.7681357788119354
Precision Score is :  0.8240822320117475
AUC Value  :  0.8021029195099634

Classification Report is :                precision    recall  f1-score
support

           0       0.78      0.84      0.81      3654
           1       0.82      0.77      0.80      3653

    accuracy                           0.80      7307
   macro avg       0.80      0.80      0.80      7307
weighted avg       0.80      0.80      0.80      7307

Confusion Matrix is :
 [[3055  599]
 [ 847 2806]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.7412526800784636
Model Test Score is :  0.7459969891884495
F1 Score is :  0.7928571428571429
Recall Score is :  0.9723514919244457
Precision Score is :  0.6693046919163369
AUC Value  :  0.7460279627109914

Classification Report is :                precision    recall  f1-score
support

           0       0.95      0.52      0.67      3654
           1       0.67      0.97      0.79      3653

    accuracy                           0.75      7307
   macro avg       0.81      0.75      0.73      7307
weighted avg       0.81      0.75      0.73      7307

Confusion Matrix is :
```

```
 [[1899 1755]
 [ 101 3552]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.8391344677098064
Model Test Score is :  0.8424798138771041
F1 Score is :  0.8543222376914315
Recall Score is :  0.9238981658910485
Precision Score is :  0.7944915254237288
AUC Value  :  0.8424909548667064
Classification Report is :               precision    recall  f1-score
support

           0       0.91      0.76      0.83      3654
           1       0.79      0.92      0.85      3653

    accuracy                           0.84      7307
   macro avg       0.85      0.84      0.84      7307
weighted avg       0.85      0.84      0.84      7307

Confusion Matrix is :
 [[2781  873]
 [ 278 3375]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.7973936712132962
Model Test Score is :  0.8086766114684549
F1 Score is :  0.7988489208633093
Recall Score is :  0.7599233506706816
Precision Score is :  0.8419775553533515
AUC Value  :  0.8086699402505022
Classification Report is :               precision    recall  f1-score
support

           0       0.78      0.86      0.82      3654
           1       0.84      0.76      0.80      3653

    accuracy                           0.81      7307
   macro avg       0.81      0.81      0.81      7307
weighted avg       0.81      0.81      0.81      7307

Confusion Matrix is :
 [[3133  521]
```

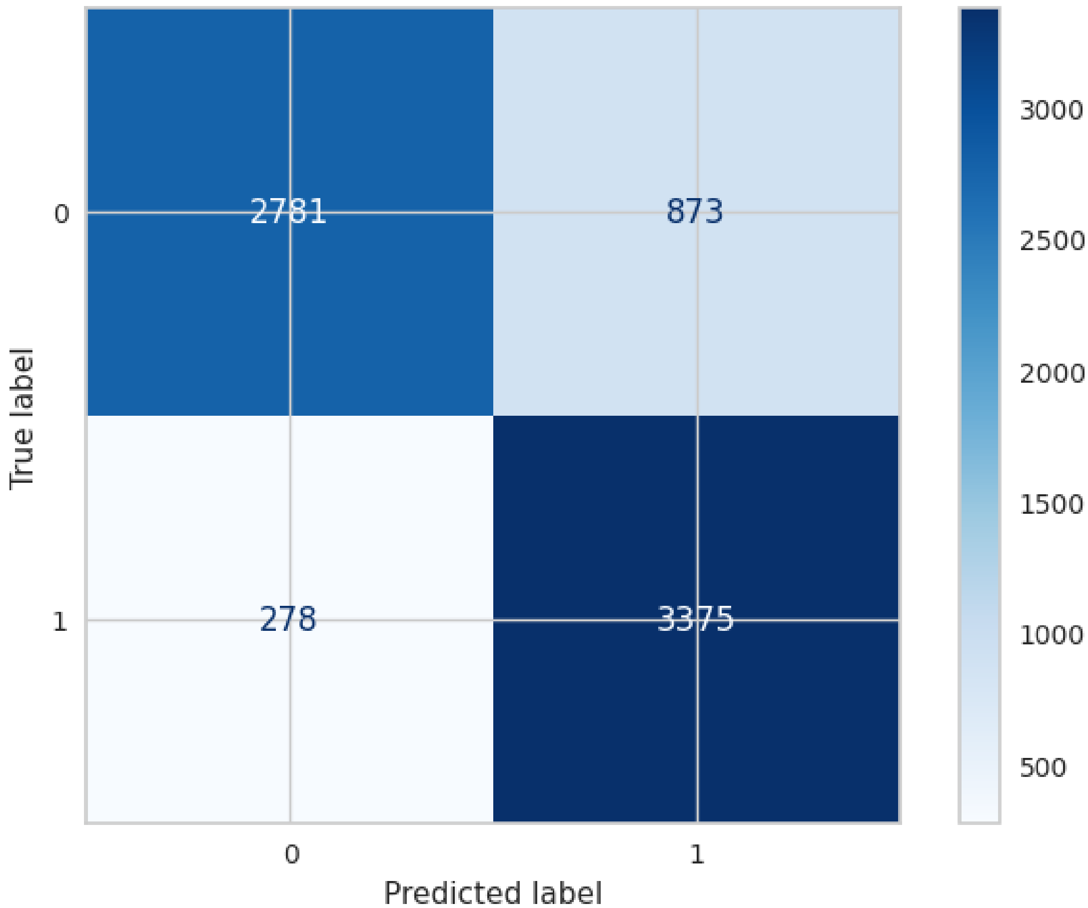
```
 [ 877 2776]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  0.694797986709852
Model Test Score is :  0.7020665115642535
F1 Score is :  0.7683796148526439
Recall Score is :  0.9885026006022447
Precision Score is :  0.6284371736860425
AUC Value  :  0.7021057064313906
Classification Report is :                precision    recall  f1-score
support

           0       0.97      0.42      0.58      3654
           1       0.63      0.99      0.77      3653

    accuracy                           0.70      7307
   macro avg       0.80      0.70      0.68      7307
weighted avg       0.80      0.70      0.68      7307

Confusion Matrix is :
 [[1519 2135]
 [  42 3611]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  0.826999984793881
Model Test Score is :  0.830299712604352
F1 Score is :  0.846268286635259
Recall Score is :  0.9343005748699699
Precision Score is :  0.7733967822343077
AUC Value  :  0.8303139437020894
Classification Report is :                precision    recall  f1-score
support

           0       0.92      0.73      0.81      3654
           1       0.77      0.93      0.85      3653

    accuracy                           0.83      7307
   macro avg       0.85      0.83      0.83      7307
weighted avg       0.85      0.83      0.83      7307

Confusion Matrix is :
 [[2654 1000]
 [ 240 3413]]
```


3169
485
1340
2313
0
1
0
1
True label
Predicted label
3000
2500
2000
1500
1000
500

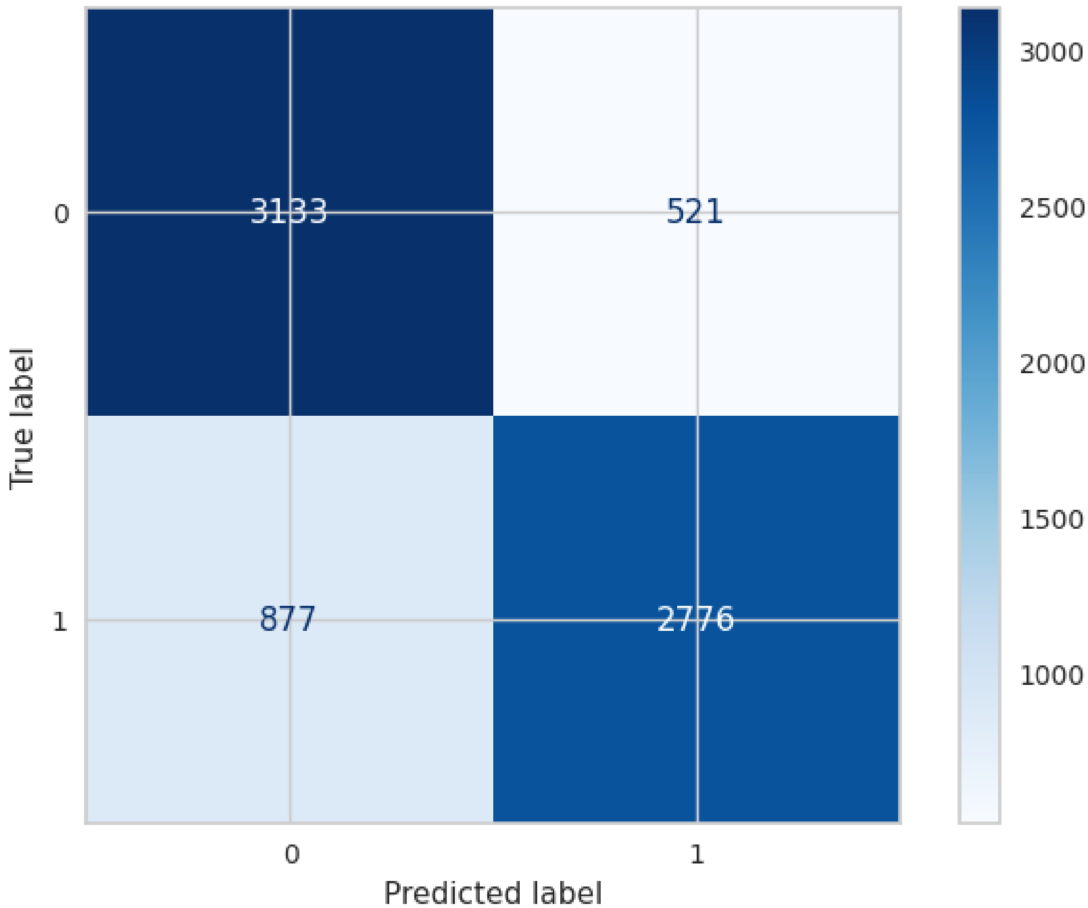

3055
599
847
2806
0
1
True label
0
1
Predicted label
3000
2500
2000
1500
1000

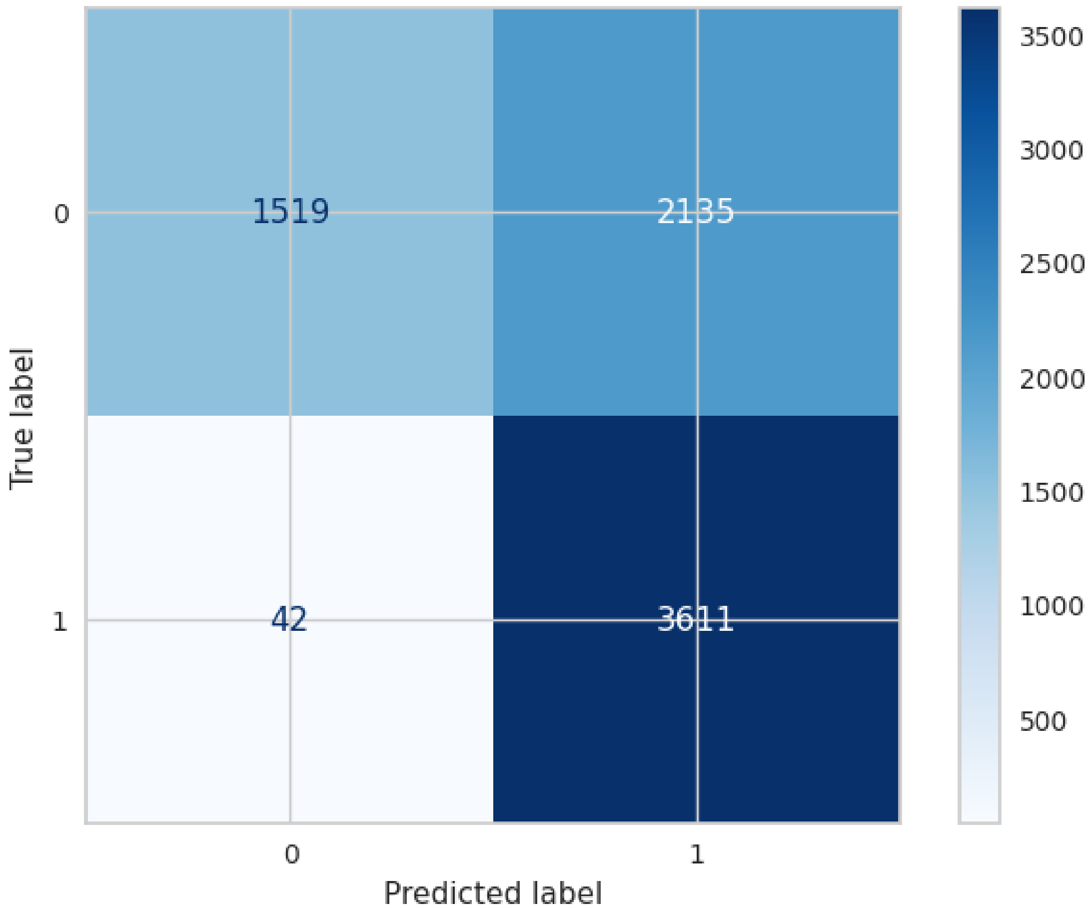


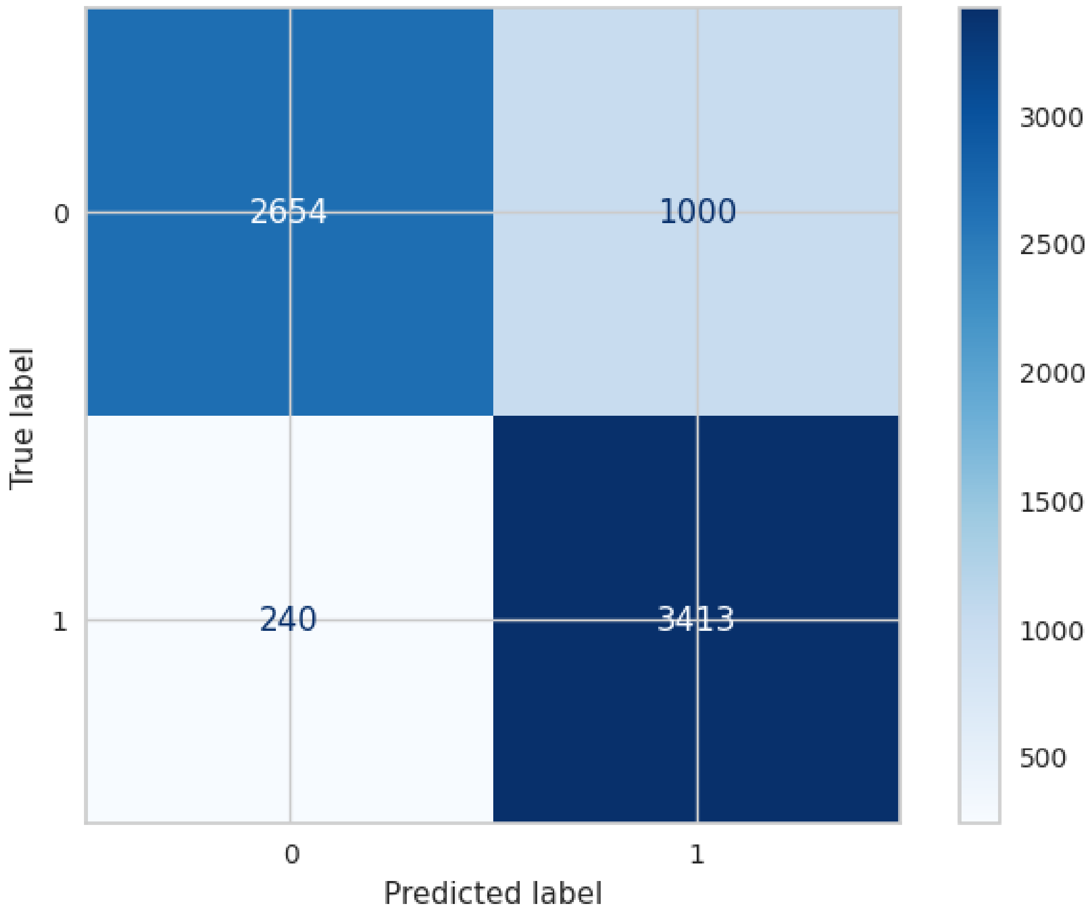
1899
1755
101
3552
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500



True label
0
1
2781
873
278
3375
0
1
Predicted label
3000
2500
2000
1500
1000
500


3133
521
877
2776
0
1
True label
0
1
Predicted label
3000
2500
2000
1500
1000


1519
2135
42
3611
0
1
True label
Predicted label
3500
3000
2500
2000
1500
1000
500



[305]:
```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['NB Over','NB Over With Feature','NB Over Scaling','NB Over␣
↪With Normalize','NB Over With PCA'
                ,'NB Over With PCA and Scaling',
                'NB Over With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[305]:
```
                                Train Accuracy  Test Accuracy   Test F1  \
Models
NB Over                               0.734227       0.750239  0.717098
NB Over With Feature                  0.796420       0.802108  0.795126
NB Over Scaling                       0.741253       0.745997  0.792857
NB Over With Normalize                0.839134       0.842480  0.854322
NB Over With PCA                      0.797394       0.808677  0.798849
NB Over With PCA and Scaling          0.694798       0.702067  0.768380
NB Over With PCA and Normalize        0.827000       0.830300  0.846268
```

```
                                   Test Recall  Test Precision       AUC
Models
NB Over                               0.633178        0.826662  0.750223
NB Over With Feature                  0.768136        0.824082  0.802103
NB Over Scaling                       0.972351        0.669305  0.746028
NB Over With Normalize                0.923898        0.794492  0.842491
NB Over With PCA                      0.759923        0.841978  0.808670
NB Over With PCA and Scaling          0.988503        0.628437  0.702106
NB Over With PCA and Normalize        0.934301        0.773397  0.830314
```

[306]:
```
models_draw(df)
```

RandomUnderSampler

[307]:
```
X_train,y_train,X_test,y_test=Split(X_classification_under,y_classification_under)
```

```
X_train shape is  (8350, 20)
X_test shape is  (928, 20)
y_train shape is  (8350,)
y_test shape is  (928,)
```

[308]:
```
cross_validation(GaussianNB(),X_train,y_train)
```

```
Train Score Value :  [0.7251497  0.73098802 0.73053892 0.72919162 0.72739521]
Mean 0.7286526946107784
Test Score Value :  [0.73473054 0.71976048 0.7257485  0.72754491 0.73173653]
Mean 0.7279041916167666
```

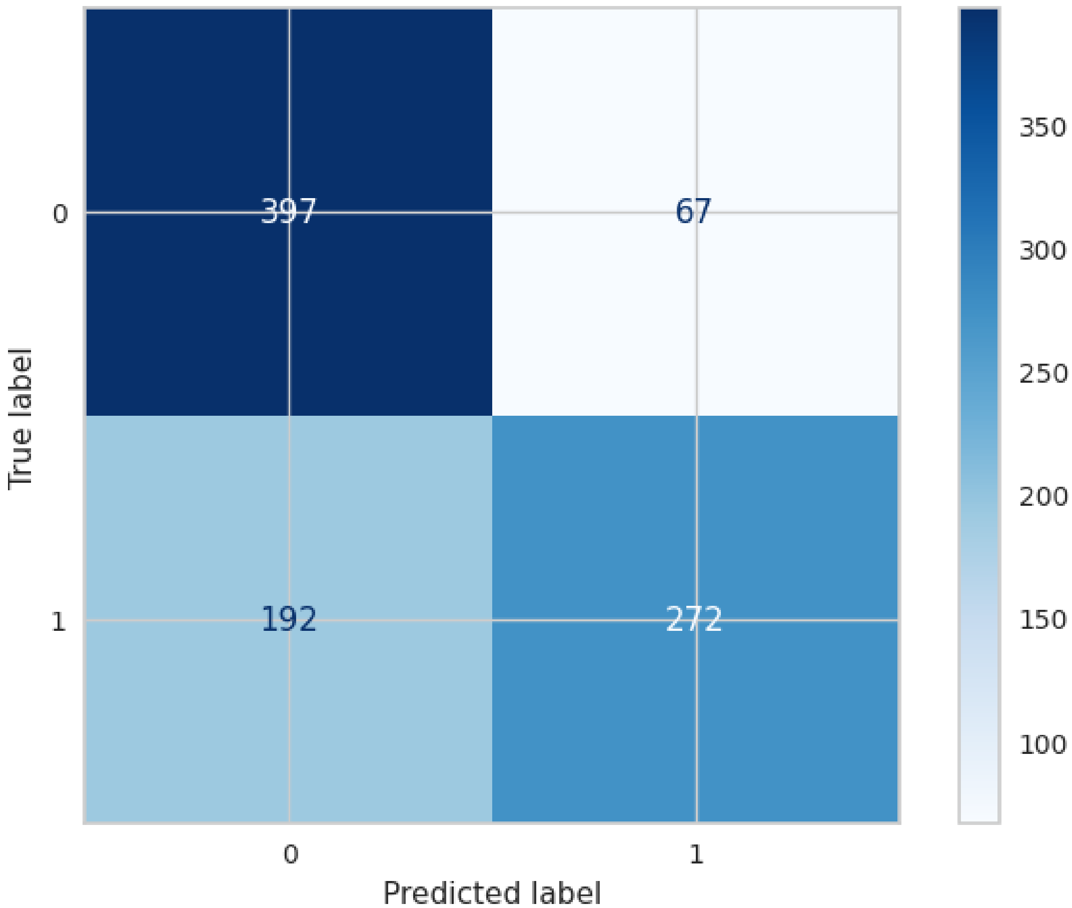
[309]:
```
Values = Models(GaussianNB(),X_train,y_train,X_test,y_test)
```

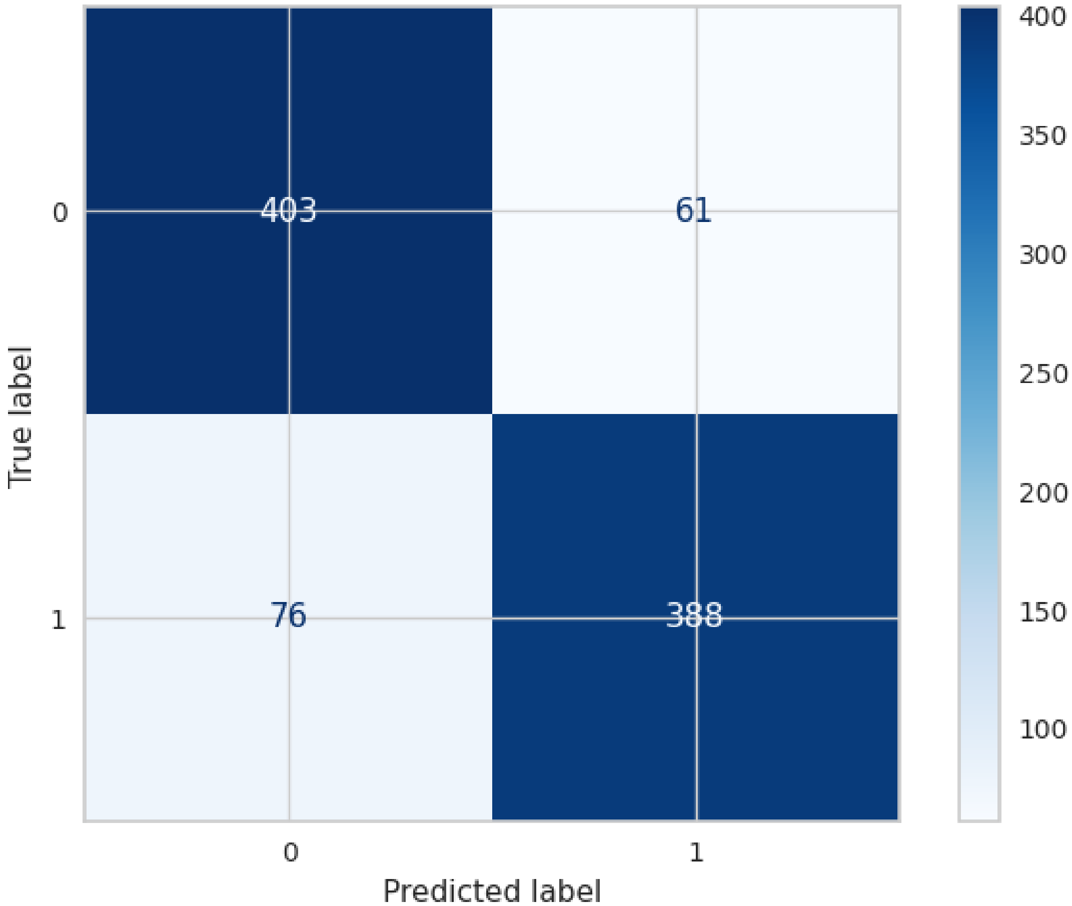
```
Apply Model With Normal Data :

Model Train Score is :  0.7288622754491018
Model Test Score is :  0.7209051724137931
F1 Score is :  0.6774595267745953
Recall Score is :  0.5862068965517241
Precision Score is :  0.8023598820058997
AUC Value  :  0.7209051724137931

Classification Report is :                precision    recall  f1-score
support

           0       0.67      0.86      0.75       464
           1       0.80      0.59      0.68       464

    accuracy                           0.72       928
   macro avg       0.74      0.72      0.72       928
weighted avg       0.74      0.72      0.72       928
```

```
Confusion Matrix is :
 [[397  67]
 [192 272]]


 Apply Model With Feature Selection :

Model Train Score is :  0.8483832335329341
Model Test Score is :  0.8523706896551724
F1 Score is :  0.8499452354874042
Recall Score is :  0.8362068965517241
Precision Score is :  0.8641425389755011
AUC Value  :  0.8523706896551725

Classification Report is :                precision    recall  f1-score
support

           0       0.84      0.87      0.85       464
           1       0.86      0.84      0.85       464

    accuracy                           0.85       928
   macro avg       0.85      0.85      0.85       928
weighted avg       0.85      0.85      0.85       928

Confusion Matrix is :
 [[403  61]
 [ 76 388]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.7288622754491018
Model Test Score is :  0.7209051724137931
F1 Score is :  0.6774595267745953
Recall Score is :  0.5862068965517241
Precision Score is :  0.8023598820058997
AUC Value  :  0.7209051724137931

Classification Report is :                precision    recall  f1-score
support

           0       0.67      0.86      0.75       464
           1       0.80      0.59      0.68       464

    accuracy                           0.72       928
   macro avg       0.74      0.72      0.72       928
weighted avg       0.74      0.72      0.72       928

Confusion Matrix is :
```

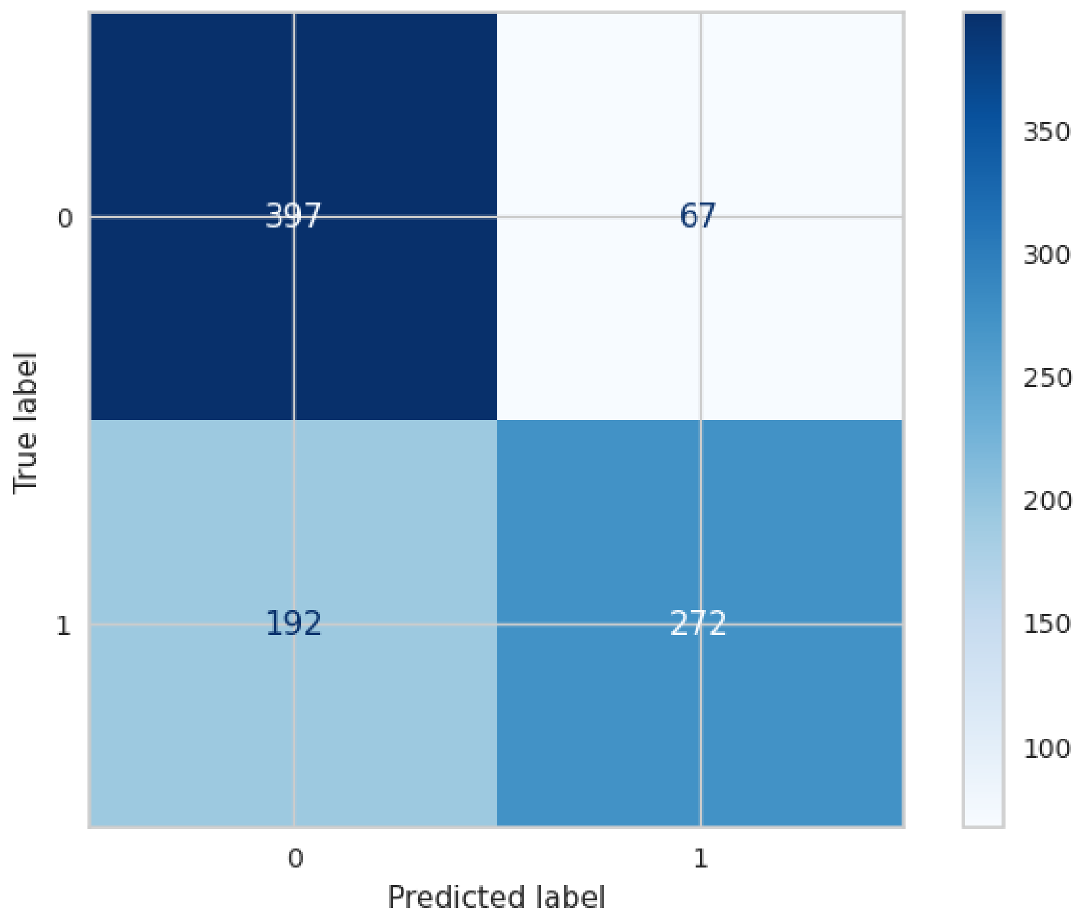
```
 [[397  67]
 [192 272]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.8411976047904192
Model Test Score is :  0.853448275862069
F1 Score is :  0.8609406952965236
Recall Score is :  0.9073275862068966
Precision Score is :  0.8190661478599222
AUC Value  :  0.853448275862069
Classification Report is :                precision    recall  f1-score
support

           0       0.90      0.80      0.85       464
           1       0.82      0.91      0.86       464

    accuracy                           0.85       928
   macro avg       0.86      0.85      0.85       928
weighted avg       0.86      0.85      0.85       928

Confusion Matrix is :
 [[371  93]
 [ 43 421]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.7547305389221557
Model Test Score is :  0.7543103448275862
F1 Score is :  0.7199017199017199
Recall Score is :  0.6314655172413793
Precision Score is :  0.8371428571428572
AUC Value  :  0.7543103448275863
Classification Report is :                precision    recall  f1-score
support

           0       0.70      0.88      0.78       464
           1       0.84      0.63      0.72       464

    accuracy                           0.75       928
   macro avg       0.77      0.75      0.75       928
weighted avg       0.77      0.75      0.75       928

Confusion Matrix is :
 [[407  57]
```

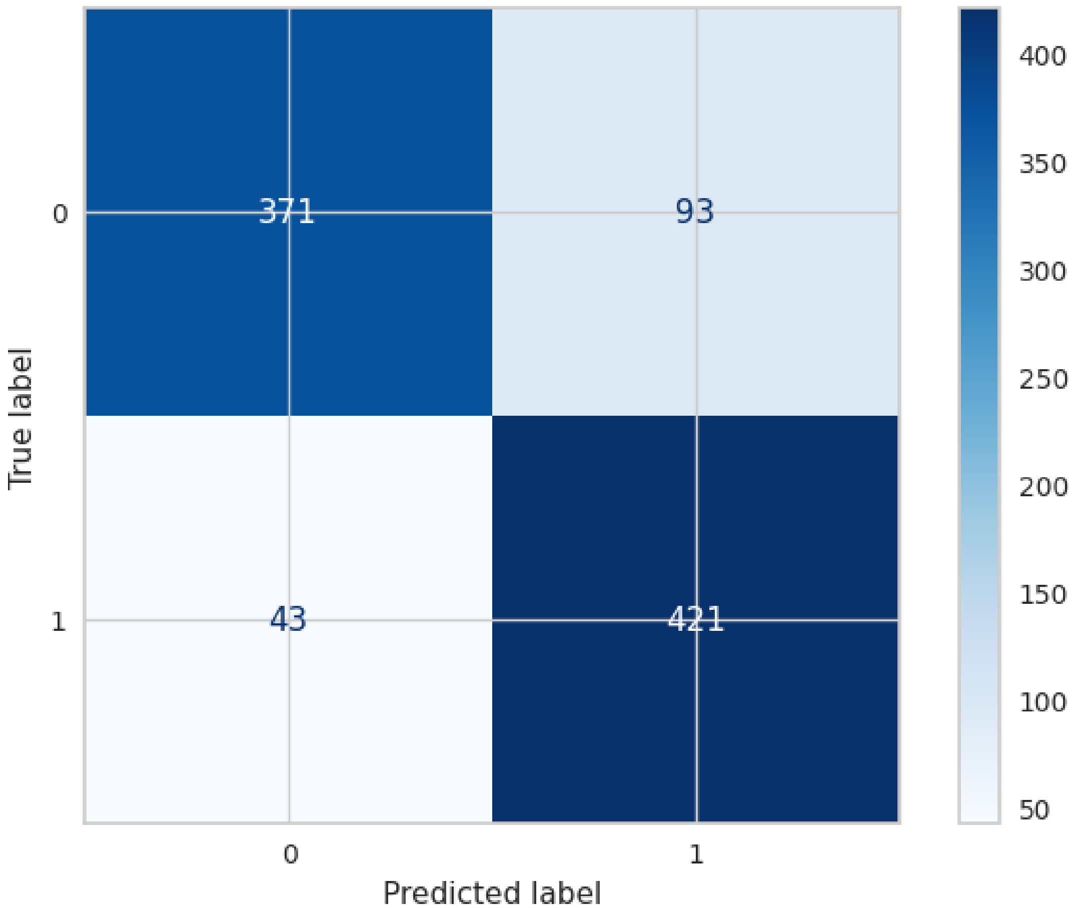
```
 [171 293]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  0.7595209580838324
Model Test Score is :  0.7672413793103449
F1 Score is :  0.7391304347826088
Recall Score is :  0.6594827586206896
Precision Score is :  0.8406593406593407
AUC Value  :  0.7672413793103448
Classification Report is :                precision    recall  f1-score
support

           0       0.72      0.88      0.79       464
           1       0.84      0.66      0.74       464

    accuracy                           0.77       928
   macro avg       0.78      0.77      0.76       928
weighted avg       0.78      0.77      0.76       928

Confusion Matrix is :
 [[406  58]
 [158 306]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  0.8512574850299401
Model Test Score is :  0.8631465517241379
F1 Score is :  0.8700102354145343
Recall Score is :  0.915948275862069
Precision Score is :  0.8284600389863548
AUC Value  :  0.8631465517241379
Classification Report is :                precision    recall  f1-score
support

           0       0.91      0.81      0.86       464
           1       0.83      0.92      0.87       464

    accuracy                           0.86       928
   macro avg       0.87      0.86      0.86       928
weighted avg       0.87      0.86      0.86       928

Confusion Matrix is :
 [[376  88]
 [ 39 425]]
```


397
67
192
272
0
1
True label
Predicted label
350
300
250
200
150
100

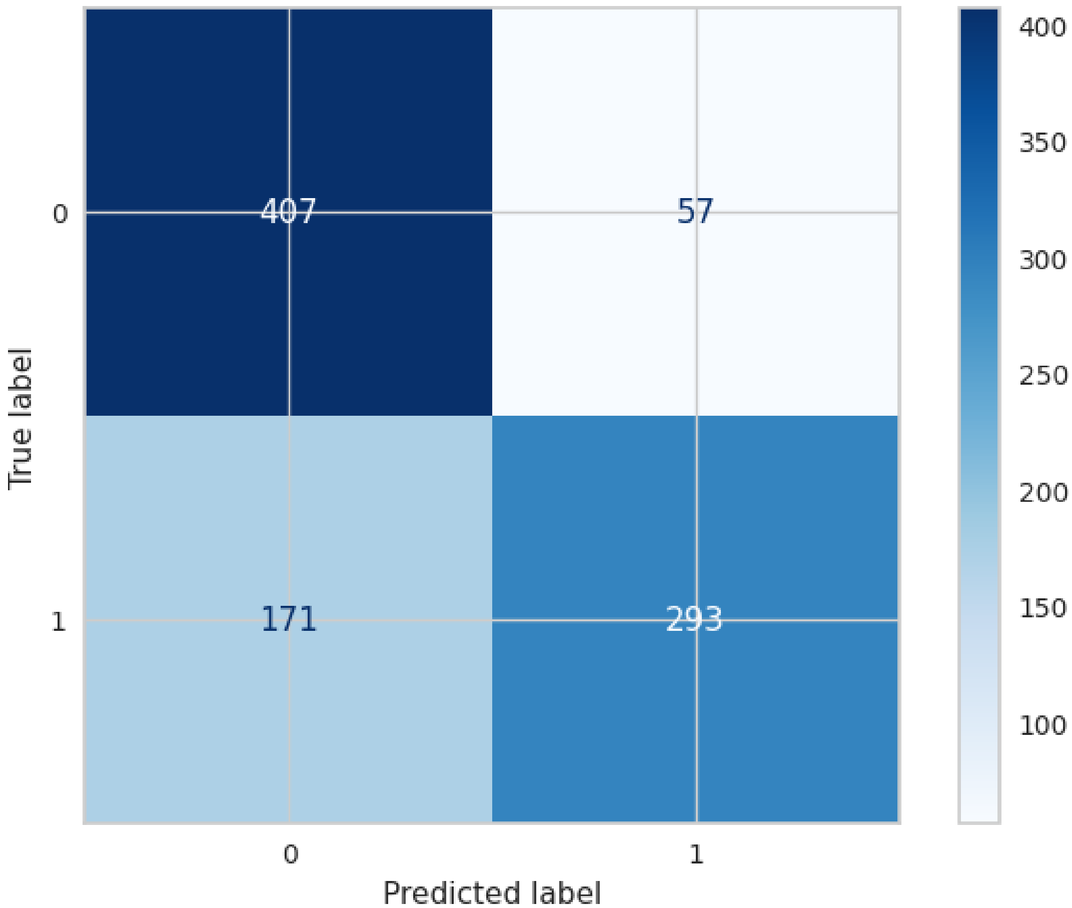


403
61
76
388
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100

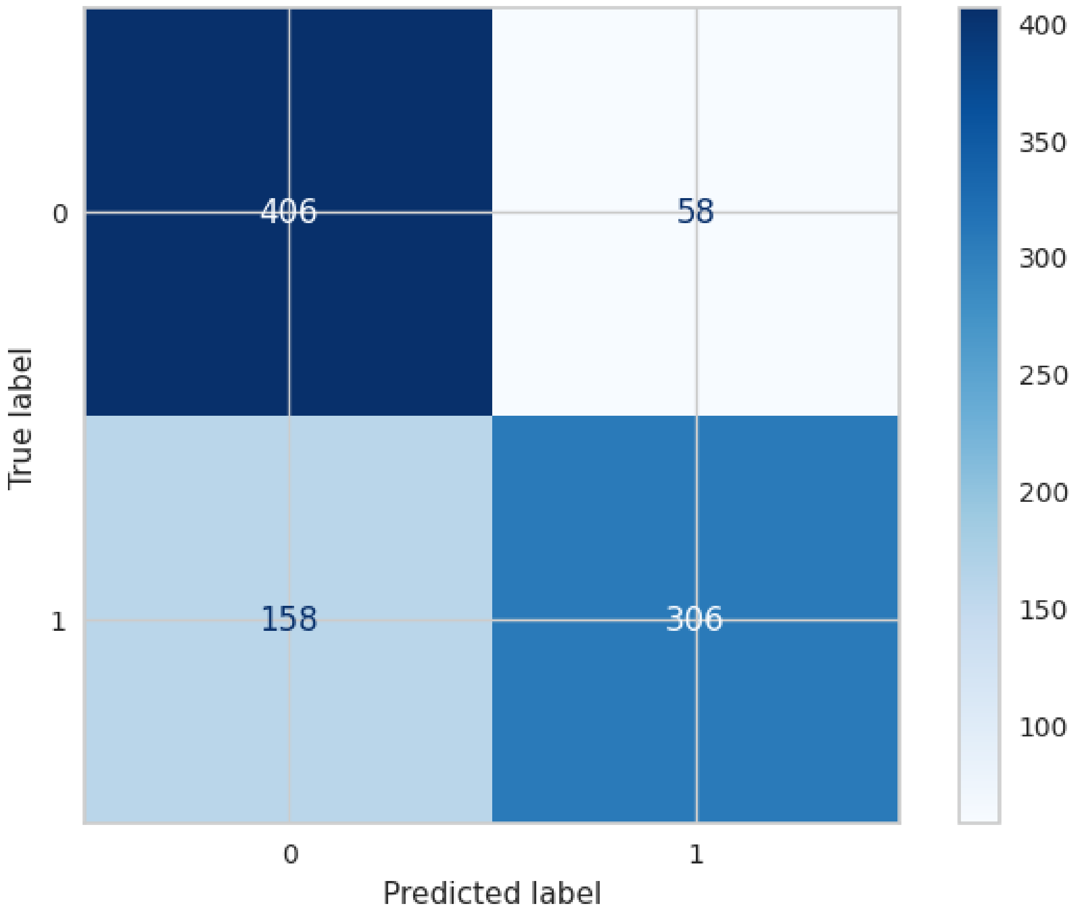

397
67
192
272
0
1
True label
0
1
Predicted label
350
300
250
200
150
100



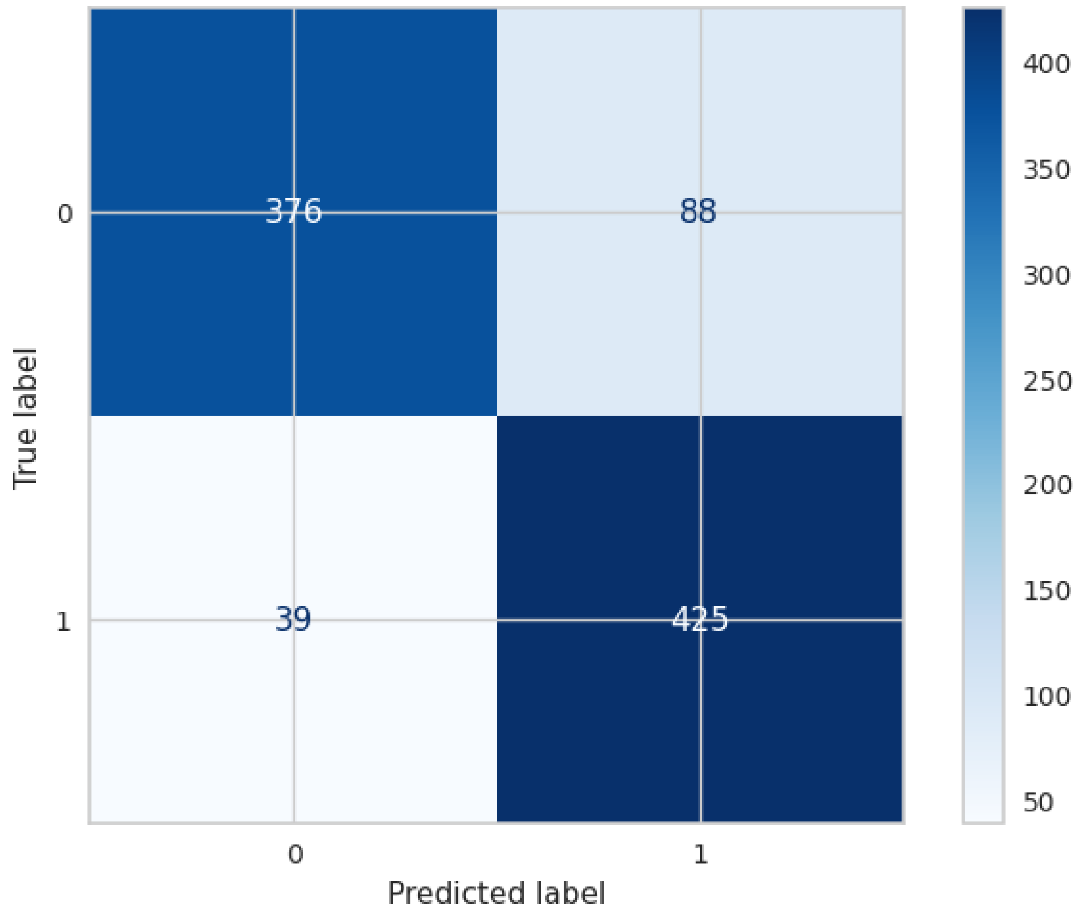
0
1
True label
371
93
43
421
0
1
Predicted label
400
350
300
250
200
150
100
50



407
57
171
293
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100



406
58
158
306
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100



[310]:

```python
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['NB Under','NB Under With Feature','NB Under Scaling','NB Under␣
↪With Normalize','NB Under With PCA'
                ,'NB Under With PCA and Scaling',
                'NB Under With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[310]:

```
                                 Train Accuracy  Test Accuracy   Test F1  \
Models
NB Under                               0.728862       0.720905  0.677460
NB Under With Feature                  0.848383       0.852371  0.849945
NB Under Scaling                       0.728862       0.720905  0.677460
NB Under With Normalize                0.841198       0.853448  0.860941
NB Under With PCA                      0.754731       0.754310  0.719902
NB Under With PCA and Scaling          0.759521       0.767241  0.739130
NB Under With PCA and Normalize        0.851257       0.863147  0.870010
```

Models	Test Recall	Test Precision	AUC
NB Under	0.586207	0.802360	0.720905
NB Under With Feature	0.836207	0.864143	0.852371
NB Under Scaling	0.586207	0.802360	0.720905
NB Under With Normalize	0.907328	0.819066	0.853448
NB Under With PCA	0.631466	0.837143	0.754310
NB Under With PCA and Scaling	0.659483	0.840659	0.767241
NB Under With PCA and Normalize	0.915948	0.828460	0.863147

[311]:
```
models_draw(df)
```

GradientBoostingClassifier

[312]:
```
X_train,y_train,X_test,y_test=Split(X_classification,y_classification)
```

```
X_train shape is  (37056, 20)
X_test shape is  (4118, 20)
y_train shape is  (37056,)
y_test shape is  (4118,)
```

[313]:
```
Search(GradientBoostingClassifier(n_estimators=100,max_depth=3) ,{'max_depth':
 ↪[5,10,20,25,30,40]},X_train,y_train)
```

[313]:
```
GradientBoostingClassifier(max_depth=5)
```

[314]:
```
cross_validation(GradientBoostingClassifier(n_estimators=100,max_depth=5),X_train,y_train)
```

```
Train Score Value :  [0.93529888 0.93452522 0.93631304 0.93688649 0.93280486]
Mean 0.9351656960626078
Test Score Value :  [0.91203454 0.91634057 0.91229254 0.90851437 0.91930914]
Mean 0.9136982314252169
```

[315]:
```
Values =
 ↪Models(GradientBoostingClassifier(n_estimators=100,max_depth=5),X_train,y_train,X_test,y_te
```

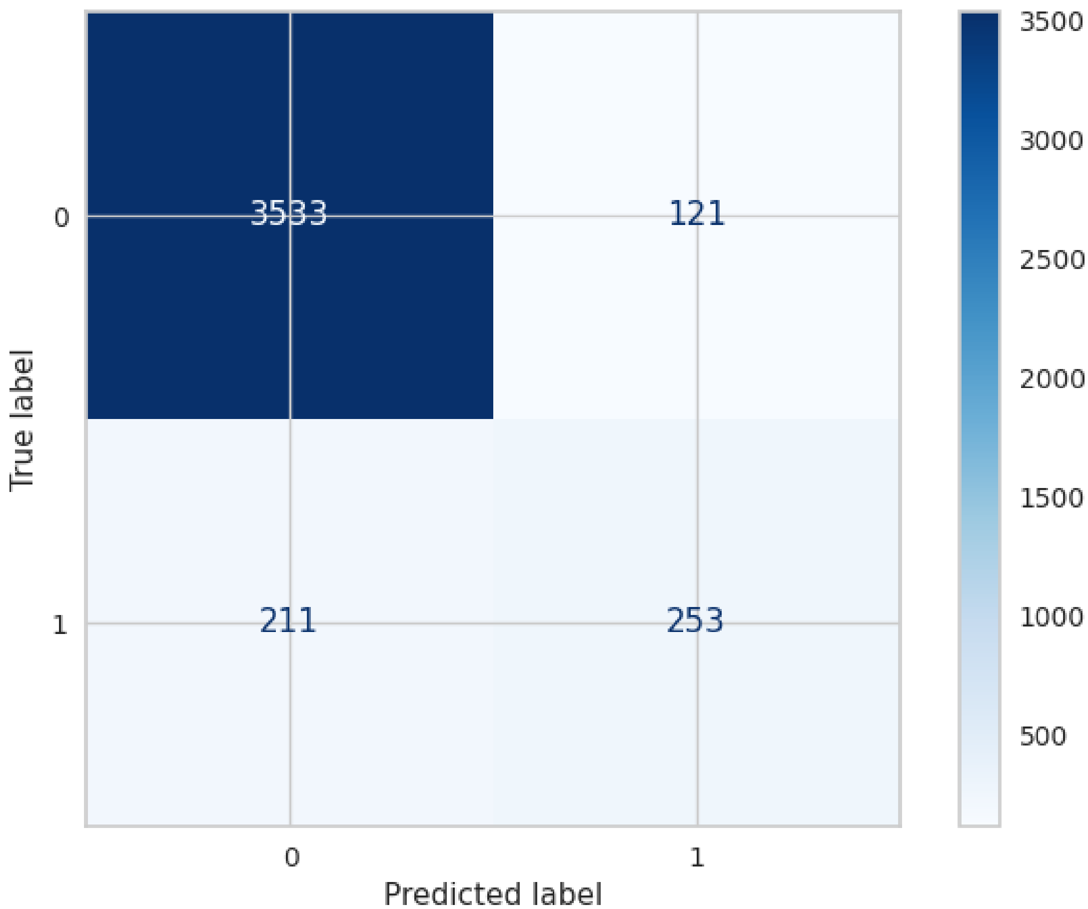
```
Apply Model With Normal Data :

Model Train Score is :  0.9332361830742659
Model Test Score is :  0.9193783389995144
F1 Score is :  0.6038186157517901
Recall Score is :  0.5452586206896551
Precision Score is :  0.6764705882352942
AUC Value  :  0.7560721127531472

Classification Report is :               precision    recall  f1-score
support

           0       0.94      0.97      0.96      3654
```

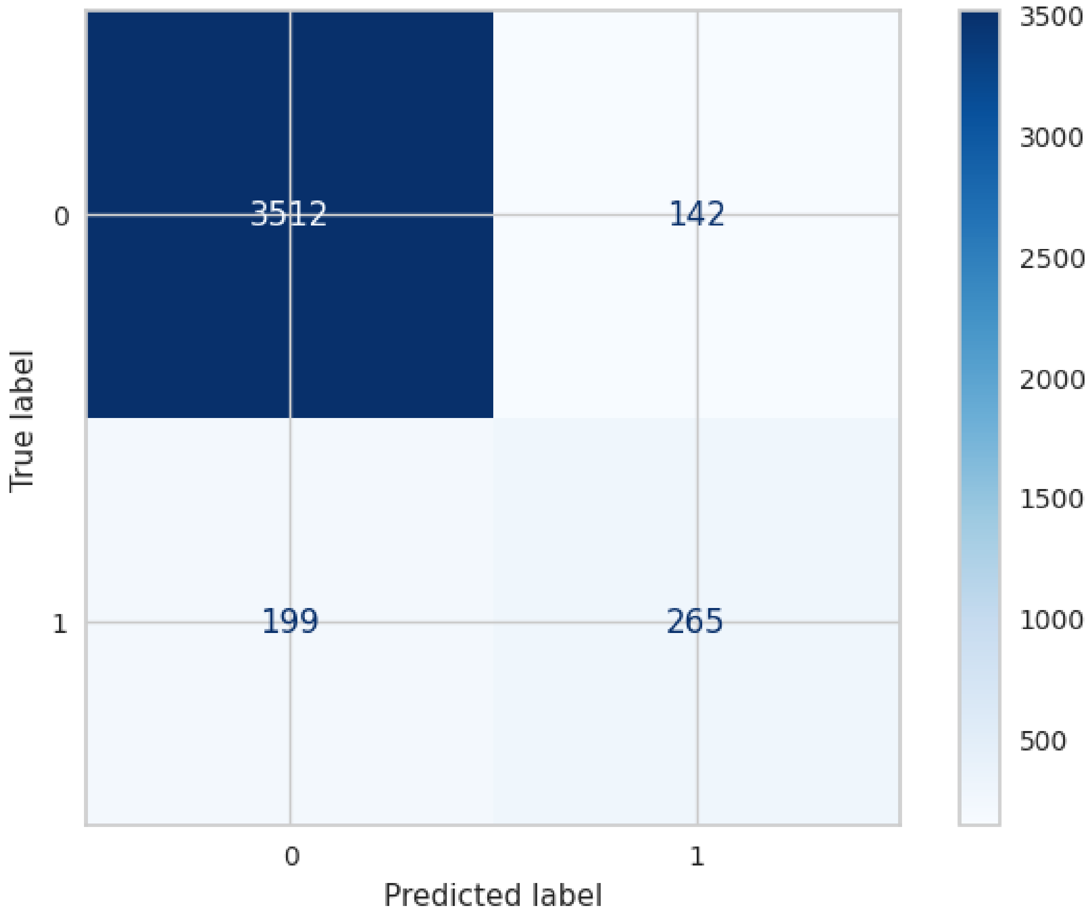
```
           1       0.68      0.55      0.60       464

    accuracy                           0.92      4118
   macro avg       0.81      0.76      0.78      4118
weighted avg       0.91      0.92      0.92      4118

Confusion Matrix is :
 [[3533  121]
 [ 211  253]]


 Apply Model With Feature Selection :

Model Train Score is :  0.9240878670120898
Model Test Score is :  0.9171928120446818
F1 Score is :  0.6084959816303099
Recall Score is :  0.5711206896551724
Precision Score is :  0.6511056511056511
AUC Value  :  0.766129584017515
Classification Report is :                precision    recall  f1-score
support

           0       0.95      0.96      0.95      3654
           1       0.65      0.57      0.61       464

    accuracy                           0.92      4118
   macro avg       0.80      0.77      0.78      4118
weighted avg       0.91      0.92      0.91      4118

Confusion Matrix is :
 [[3512  142]
 [ 199  265]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.9332361830742659
Model Test Score is :  0.9193783389995144
F1 Score is :  0.6038186157517901
Recall Score is :  0.5452586206896551
Precision Score is :  0.6764705882352942
AUC Value  :  0.7560721127531472
Classification Report is :                precision    recall  f1-score
support

           0       0.94      0.97      0.96      3654
           1       0.68      0.55      0.60       464
```

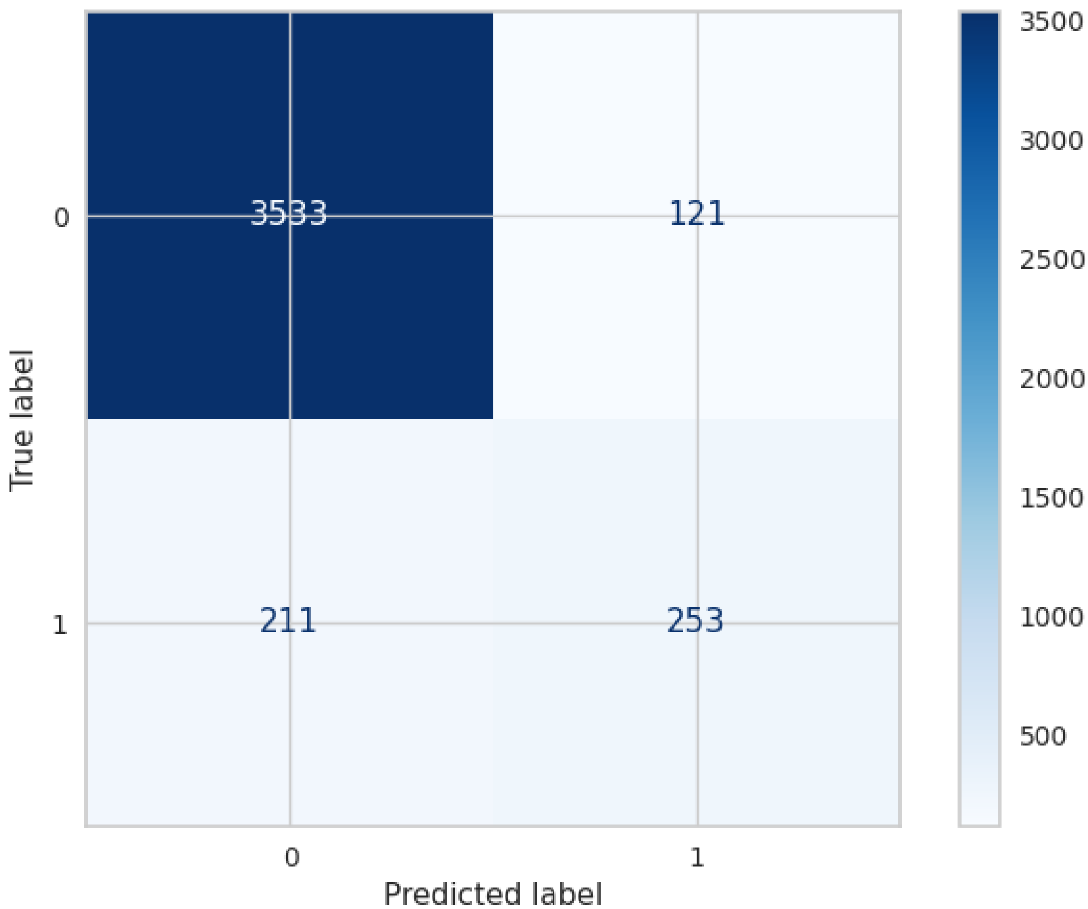
```
    accuracy                           0.92      4118
   macro avg       0.81      0.76      0.78      4118
weighted avg       0.91      0.92      0.92      4118

Confusion Matrix is :
 [[3533  121]
 [ 211  253]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.9356649395509499
Model Test Score is :  0.9135502671199611
F1 Score is :  0.5658536585365854
Recall Score is :  0.5
Precision Score is :  0.651685393258427
AUC Value  :  0.733032293377121
Classification Report is :                precision    recall  f1-score
support

           0       0.94      0.97      0.95      3654
           1       0.65      0.50      0.57       464

    accuracy                           0.91      4118
   macro avg       0.80      0.73      0.76      4118
weighted avg       0.91      0.91      0.91      4118

Confusion Matrix is :
 [[3530  124]
 [ 232  232]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.9418987478411054
Model Test Score is :  0.9176784847013113
F1 Score is :  0.592057761732852
Recall Score is :  0.5301724137931034
Precision Score is :  0.670299727520436
AUC Value  :  0.7485290093048713
Classification Report is :                precision    recall  f1-score
support

           0       0.94      0.97      0.95      3654
           1       0.67      0.53      0.59       464
```

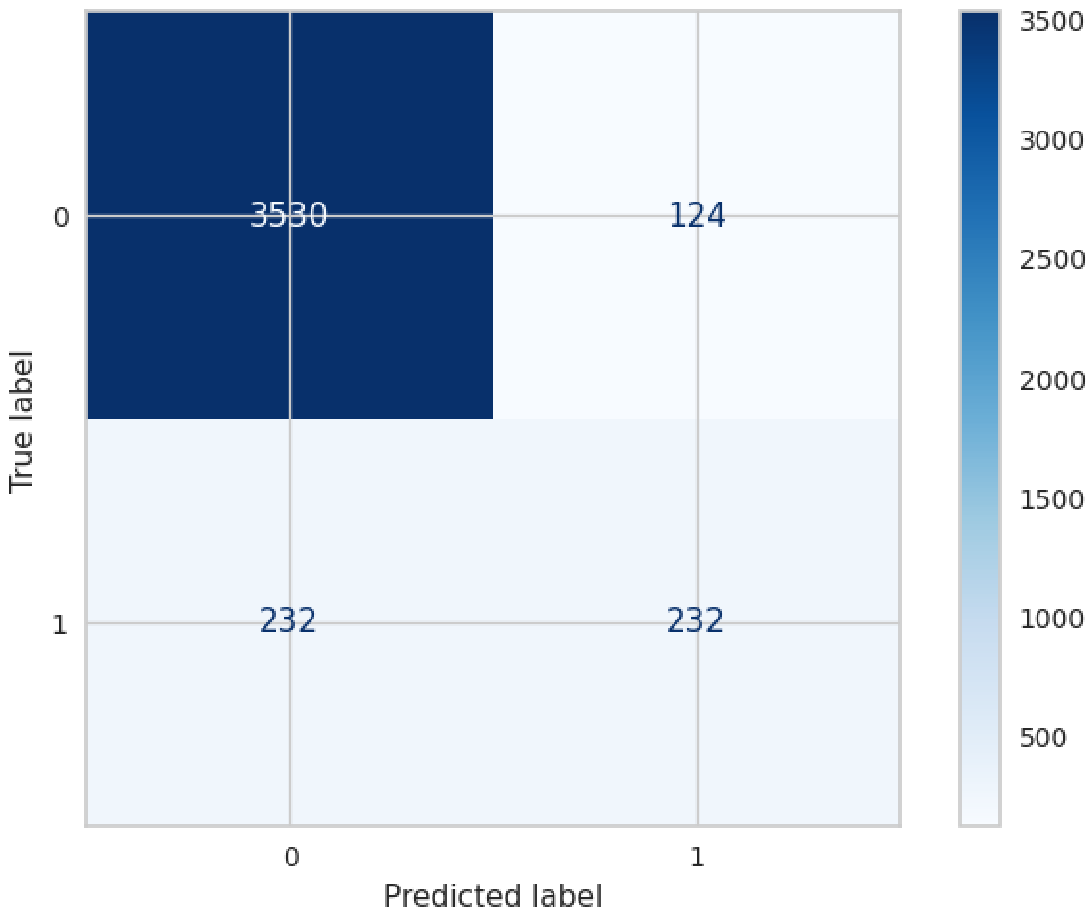
```
    accuracy                           0.92      4118
   macro avg       0.81      0.75      0.77      4118
weighted avg       0.91      0.92      0.91      4118

Confusion Matrix is :
 [[3533  121]
 [ 218  246]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  0.9397938255613126
Model Test Score is :  0.9142787761049053
F1 Score is :  0.5593008739076155
Recall Score is :  0.4827586206896552
Precision Score is :  0.6646884272997032
AUC Value  :  0.7259168035030105

Classification Report is :                precision    recall  f1-score
support

           0       0.94      0.97      0.95      3654
           1       0.66      0.48      0.56       464

    accuracy                           0.91      4118
   macro avg       0.80      0.73      0.76      4118
weighted avg       0.91      0.91      0.91      4118

Confusion Matrix is :
 [[3541  113]
 [ 240  224]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  0.9359078151986183
Model Test Score is :  0.9157357940747936
F1 Score is :  0.5678704856787049
Recall Score is :  0.49137931034482757
Precision Score is :  0.672566371681416
AUC Value  :  0.7305008210180625

Classification Report is :                precision    recall  f1-score
support

           0       0.94      0.97      0.95      3654
           1       0.67      0.49      0.57       464

    accuracy                           0.92      4118
```

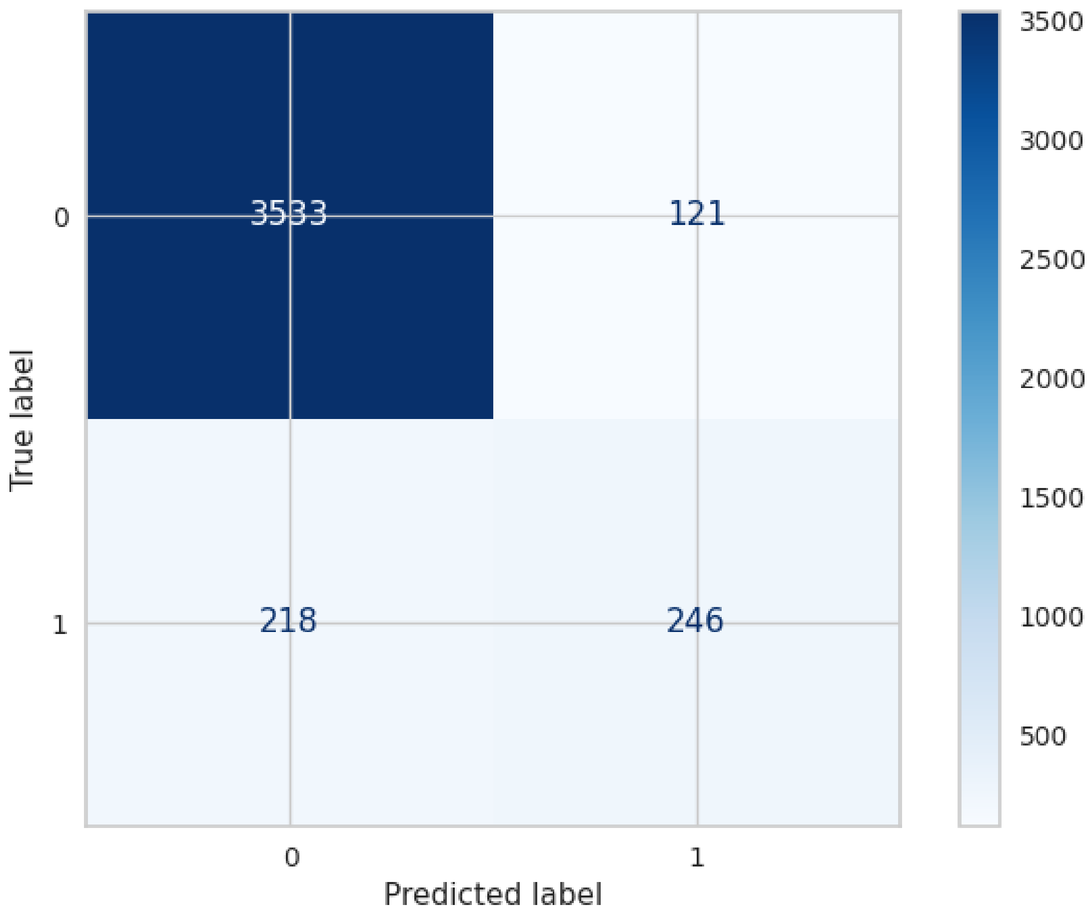
```
   macro avg       0.81      0.73      0.76      4118
weighted avg       0.91      0.92      0.91      4118

Confusion Matrix is :
 [[3543  111]
 [ 236  228]]
```

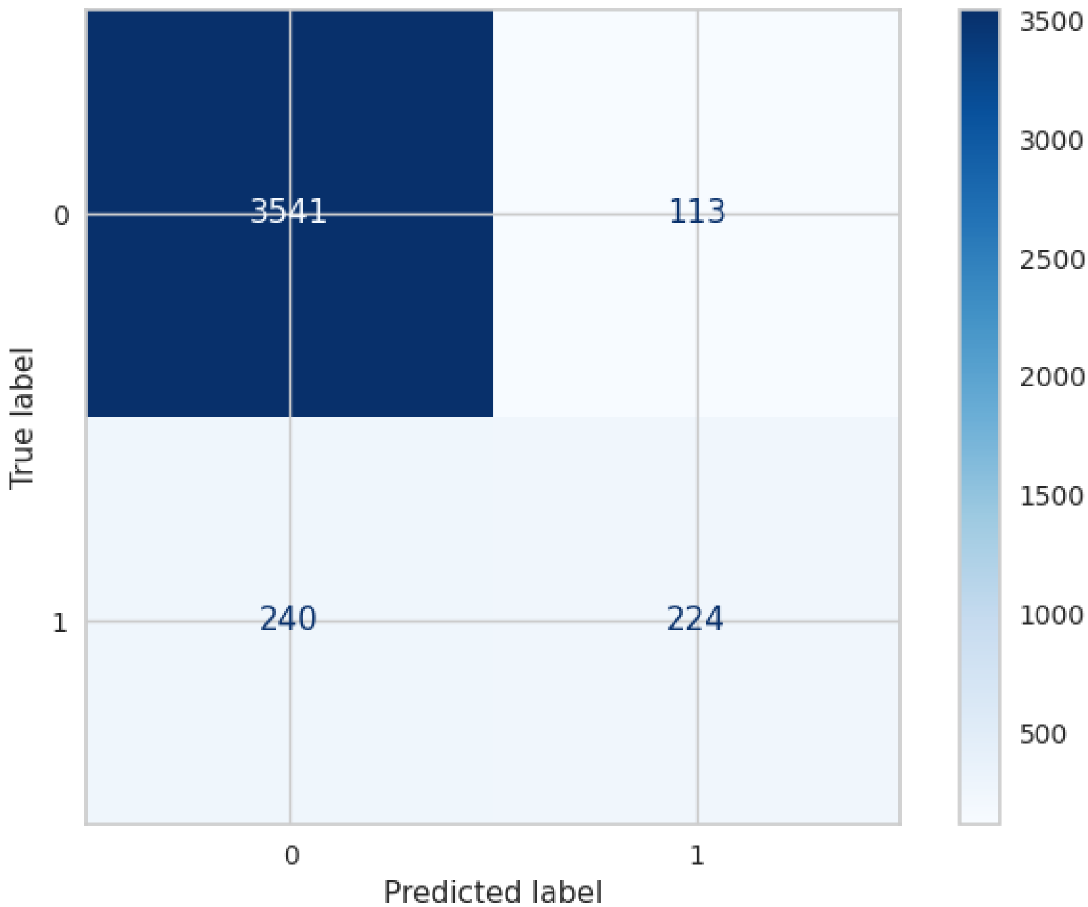


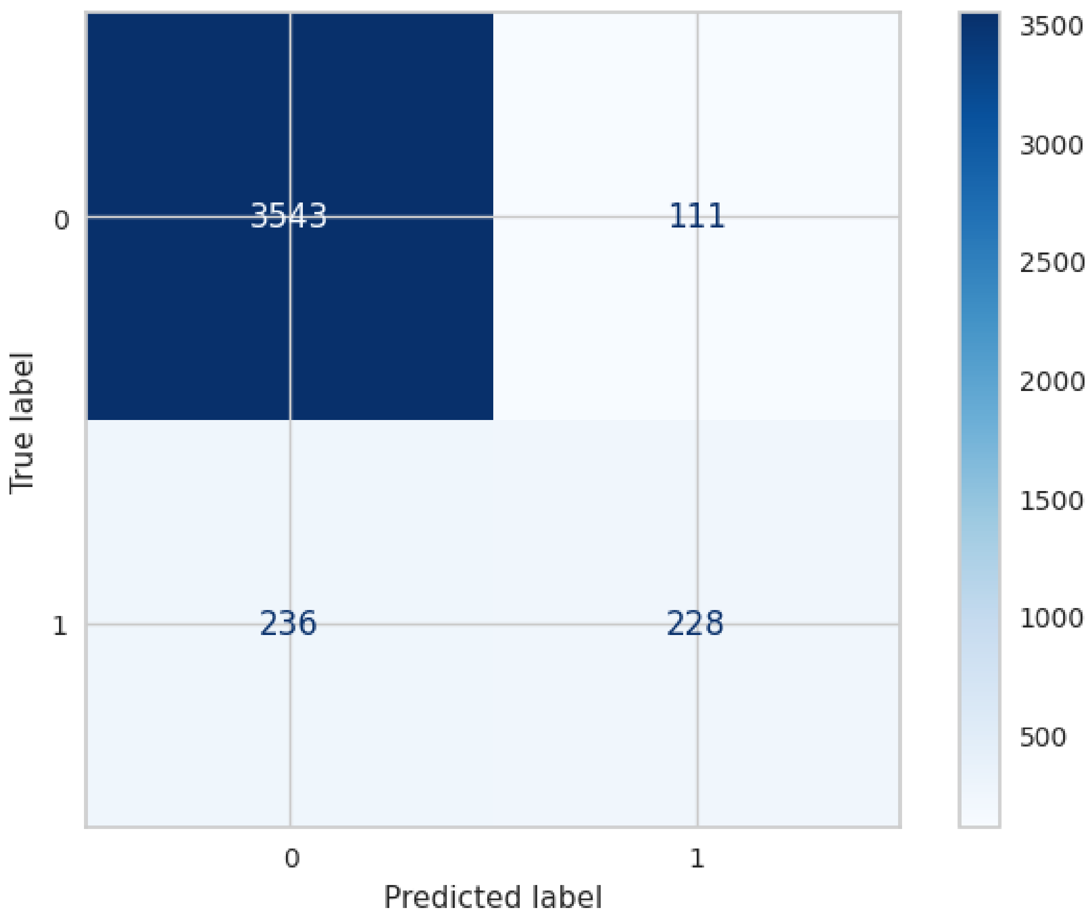

3512
142
199
265
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500


3533
121
211
253
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500


3530
124
232
232
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500


3533
121
218
246
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500


3541
113
240
224
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500



[316]:
```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['Gradient','Gradient With Feature','Gradient Scaling','Gradient␣
↪With Normalize','Gradient With PCA'
                ,'Gradient With PCA and Scaling',
                'Gradient With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[316]:
```
                                 Train Accuracy  Test Accuracy   Test F1  \
Models
Gradient                               0.933236       0.919378  0.603819
Gradient With Feature                  0.924088       0.917193  0.608496
Gradient Scaling                       0.933236       0.919378  0.603819
Gradient With Normalize                0.935665       0.913550  0.565854
Gradient With PCA                      0.941899       0.917678  0.592058
Gradient With PCA and Scaling          0.939794       0.914279  0.559301
Gradient With PCA and Normalize        0.935908       0.915736  0.567870
```

```
                                  Test Recall  Test Precision       AUC
Models
Gradient                             0.545259        0.676471  0.756072
Gradient With Feature                0.571121        0.651106  0.766130
Gradient Scaling                     0.545259        0.676471  0.756072
Gradient With Normalize              0.500000        0.651685  0.733032
Gradient With PCA                    0.530172        0.670300  0.748529
Gradient With PCA and Scaling        0.482759        0.664688  0.725917
Gradient With PCA and Normalize      0.491379        0.672566  0.730501
```

[317]:
```
models_draw(df)
```

RandomOverSampler

[318]:
```
X_train,y_train,X_test,y_test=Split(X_classification_over,y_classification_over)
```

```
X_train shape is  (65763, 20)
X_test shape is  (7307, 20)
y_train shape is  (65763,)
y_test shape is  (7307,)
```

[319]:
```
Search(GradientBoostingClassifier(n_estimators=100,max_depth=3) ,{'max_depth':
 ↪[5,10,20,25,30,40]},X_train,y_train)
```

[319]:
```
GradientBoostingClassifier(max_depth=20)
```

[320]:
```
cross_validation(GradientBoostingClassifier(n_estimators=100,max_depth=20),X_train,y_train)
```

```
Train Score Value :  [0.99992397 0.99990496 0.99992397 0.99990496 0.99996199]
Mean 0.9999239693330242
Test Score Value :  [0.9618338  0.96457082 0.96761195 0.96677311 0.9656326 ]
Mean 0.9652844583854293
```

[321]:
```
Values =␣
 ↪Models(GradientBoostingClassifier(n_estimators=100,max_depth=20),X_train,y_train,X_test,y_t
```

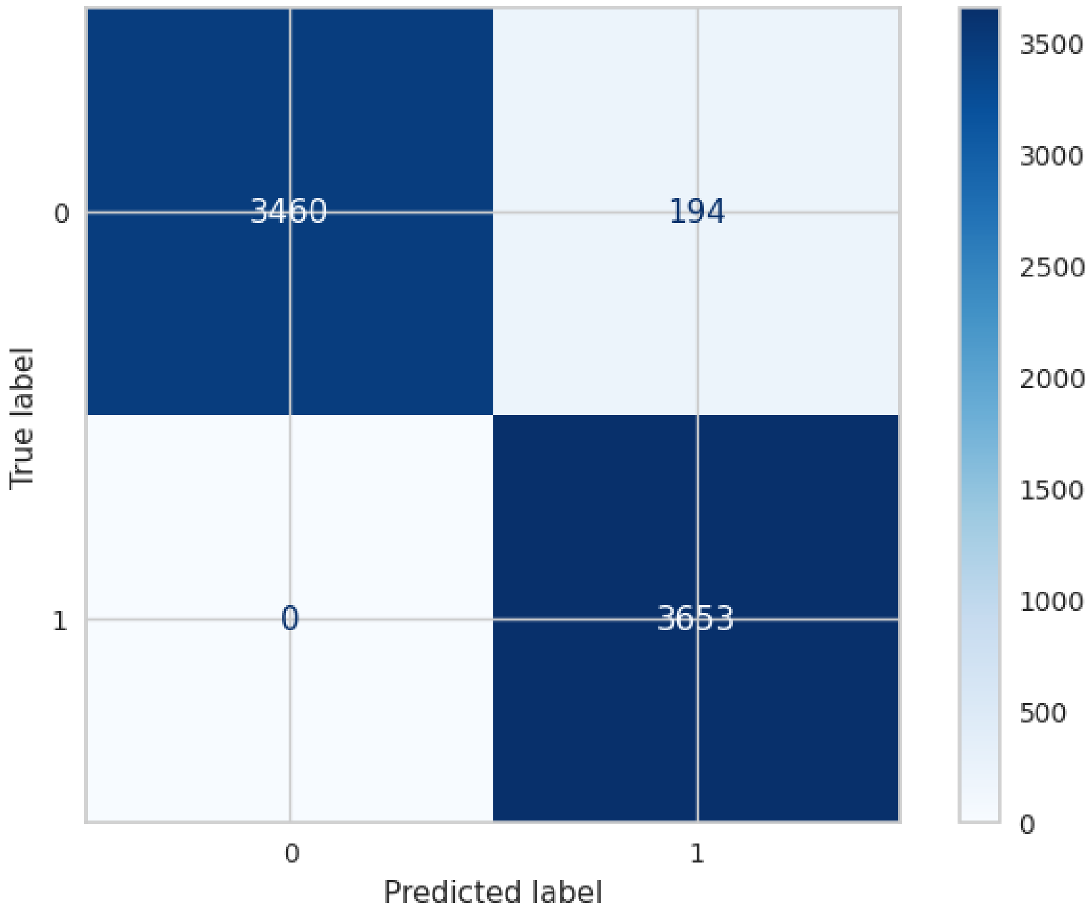
```
Apply Model With Normal Data :

Model Train Score is :  0.9999239694052887
Model Test Score is :  0.9734501163268099
F1 Score is :  0.9741333333333333
Recall Score is :  1.0
Precision Score is :  0.9495710943592409
AUC Value  :  0.9734537493158183

Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.95      0.97      3654
```

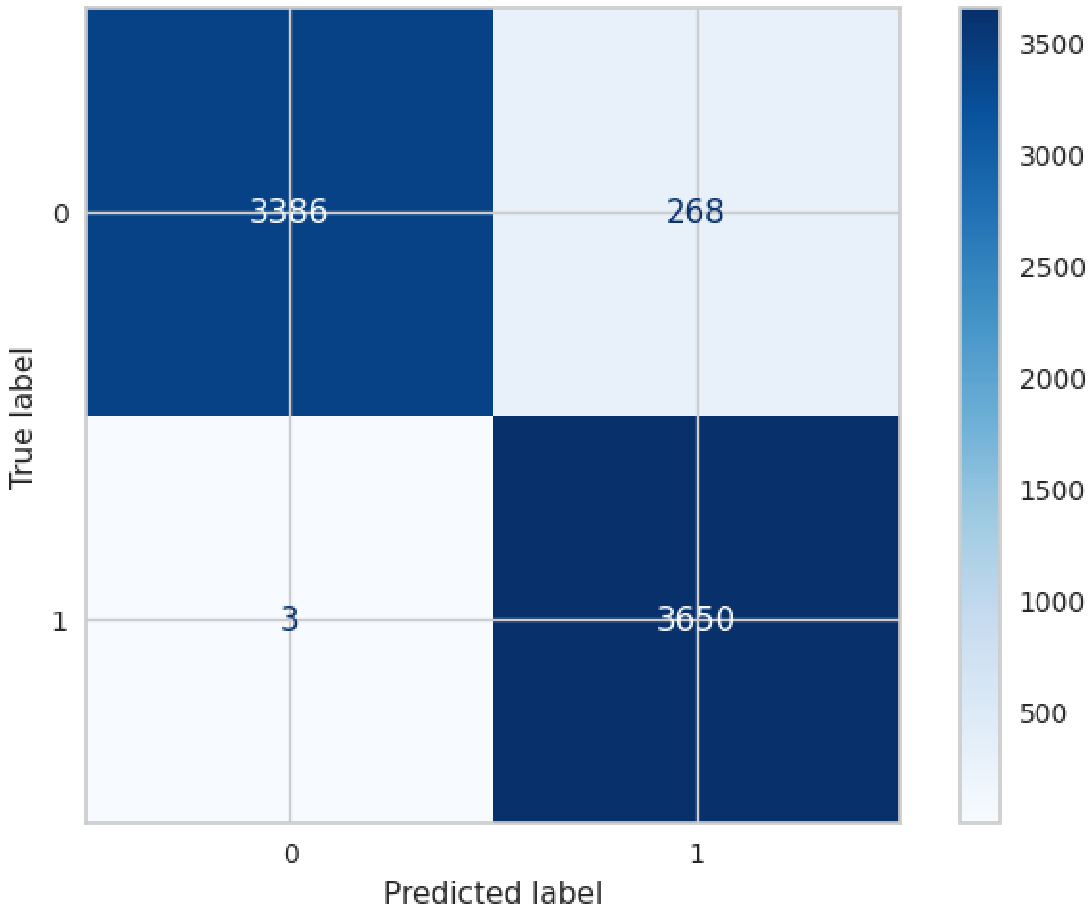
```
           1       0.95      1.00      0.97      3653

    accuracy                           0.97      7307
   macro avg       0.97      0.97      0.97      7307
weighted avg       0.97      0.97      0.97      7307

Confusion Matrix is :
 [[3460  194]
 [   0 3653]]


 Apply Model With Feature Selection :

Model Train Score is :  0.9883825251281115
Model Test Score is :  0.9629122758998221
F1 Score is :  0.9642055210672301
Recall Score is :  0.9991787571858747
Precision Score is :  0.9315977539561
AUC Value  :  0.9629172384725213
Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.93      0.96      3654
           1       0.93      1.00      0.96      3653

    accuracy                           0.96      7307
   macro avg       0.97      0.96      0.96      7307
weighted avg       0.97      0.96      0.96      7307

Confusion Matrix is :
 [[3386  268]
 [   3 3650]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.9999239694052887
Model Test Score is :  0.972355275762967
F1 Score is :  0.9730953649440597
Recall Score is :  1.0
Precision Score is :  0.9476005188067445
AUC Value  :  0.9723590585659551
Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.94      0.97      3654
           1       0.95      1.00      0.97      3653
```

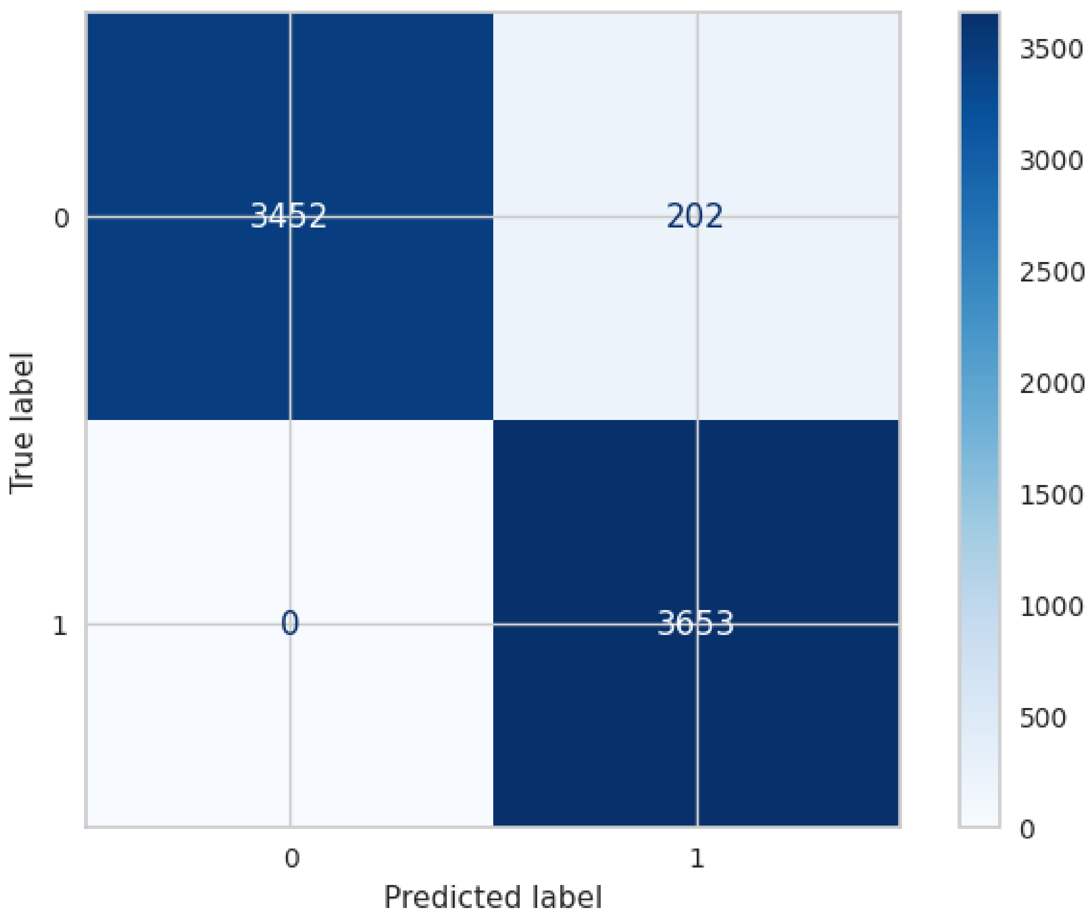
```
   accuracy                           0.97      7307
  macro avg       0.97      0.97      0.97      7307
weighted avg       0.97      0.97      0.97      7307

Confusion Matrix is :
 [[3452  202]
 [   0 3653]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.9999239694052887
Model Test Score is :  0.9670179280142329
F1 Score is :  0.9680667815025837
Recall Score is :  1.0
Precision Score is :  0.9381099126861838
AUC Value  :  0.9670224411603722
Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.93      0.97      3654
           1       0.94      1.00      0.97      3653

   accuracy                           0.97      7307
  macro avg       0.97      0.97      0.97      7307
weighted avg       0.97      0.97      0.97      7307

Confusion Matrix is :
 [[3413  241]
 [   0 3653]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.9999239694052887
Model Test Score is :  0.9709867250581634
F1 Score is :  0.9718010109071562
Recall Score is :  1.0
Precision Score is :  0.9451487710219922
AUC Value  :  0.9709906951286262
Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.94      0.97      3654
           1       0.95      1.00      0.97      3653
```

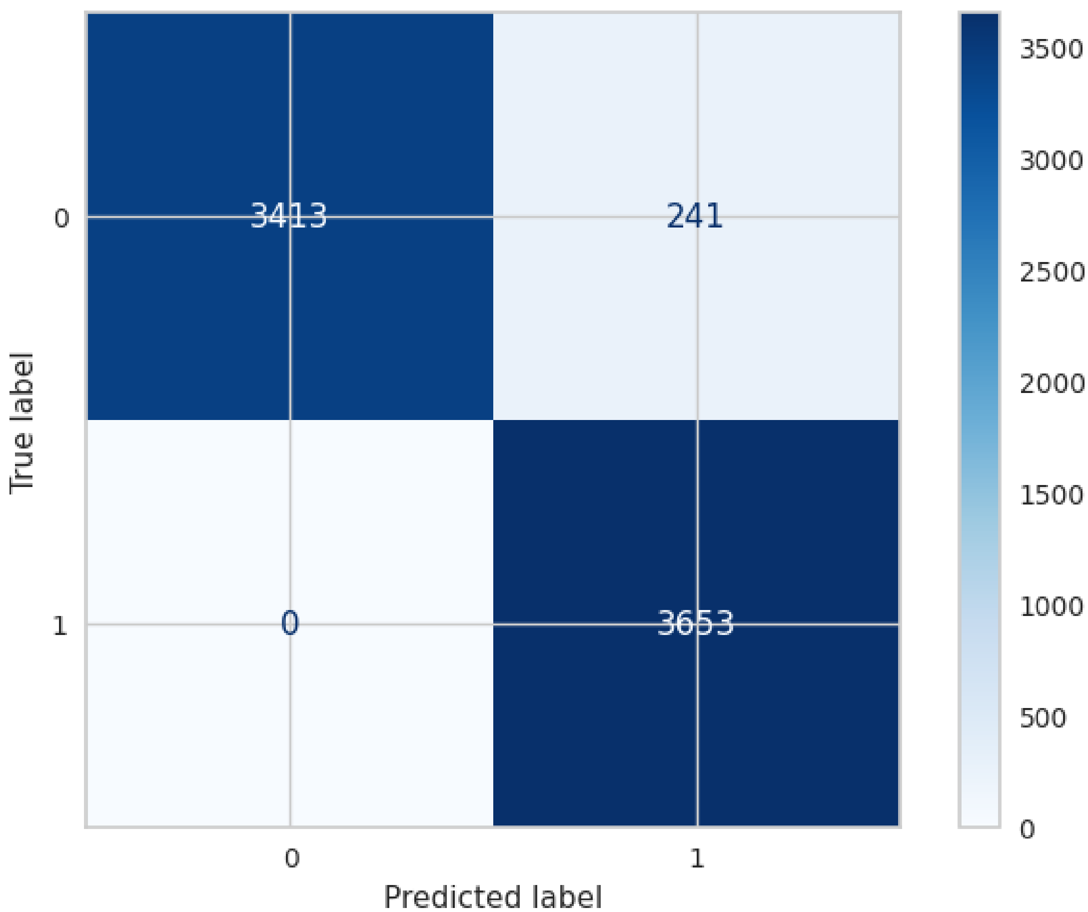
```
    accuracy                           0.97      7307
   macro avg       0.97      0.97      0.97      7307
weighted avg       0.97      0.97      0.97      7307

Confusion Matrix is :
 [[3442  212]
 [   0 3653]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  0.9999239694052887
Model Test Score is :  0.9742712467496921
F1 Score is :  0.974913263944489
Recall Score is :  1.0
Precision Score is :  0.9510544129133038
AUC Value  :  0.9742747673782157

Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.95      0.97      3654
           1       0.95      1.00      0.97      3653

    accuracy                           0.97      7307
   macro avg       0.98      0.97      0.97      7307
weighted avg       0.98      0.97      0.97      7307

Confusion Matrix is :
 [[3466  188]
 [   0 3653]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  0.9999239694052887
Model Test Score is :  0.9693444642123991
F1 Score is :  0.9702523240371846
Recall Score is :  1.0
Precision Score is :  0.9422233685839567
AUC Value  :  0.9693486590038315

Classification Report is :                precision    recall  f1-score
support

           0       1.00      0.94      0.97      3654
           1       0.94      1.00      0.97      3653

    accuracy                           0.97      7307
```

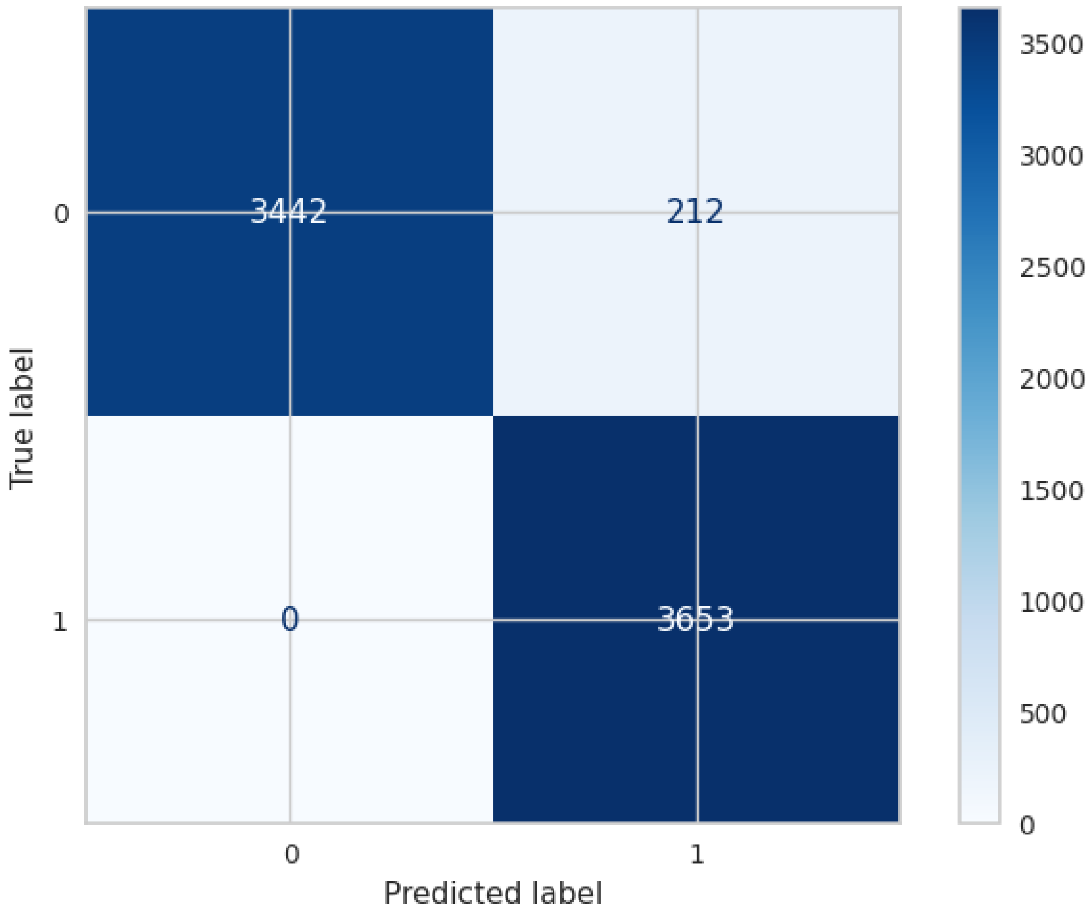
```
   macro avg       0.97      0.97      0.97      7307
weighted avg       0.97      0.97      0.97      7307

Confusion Matrix is :
 [[3430  224]
 [   0 3653]]
```



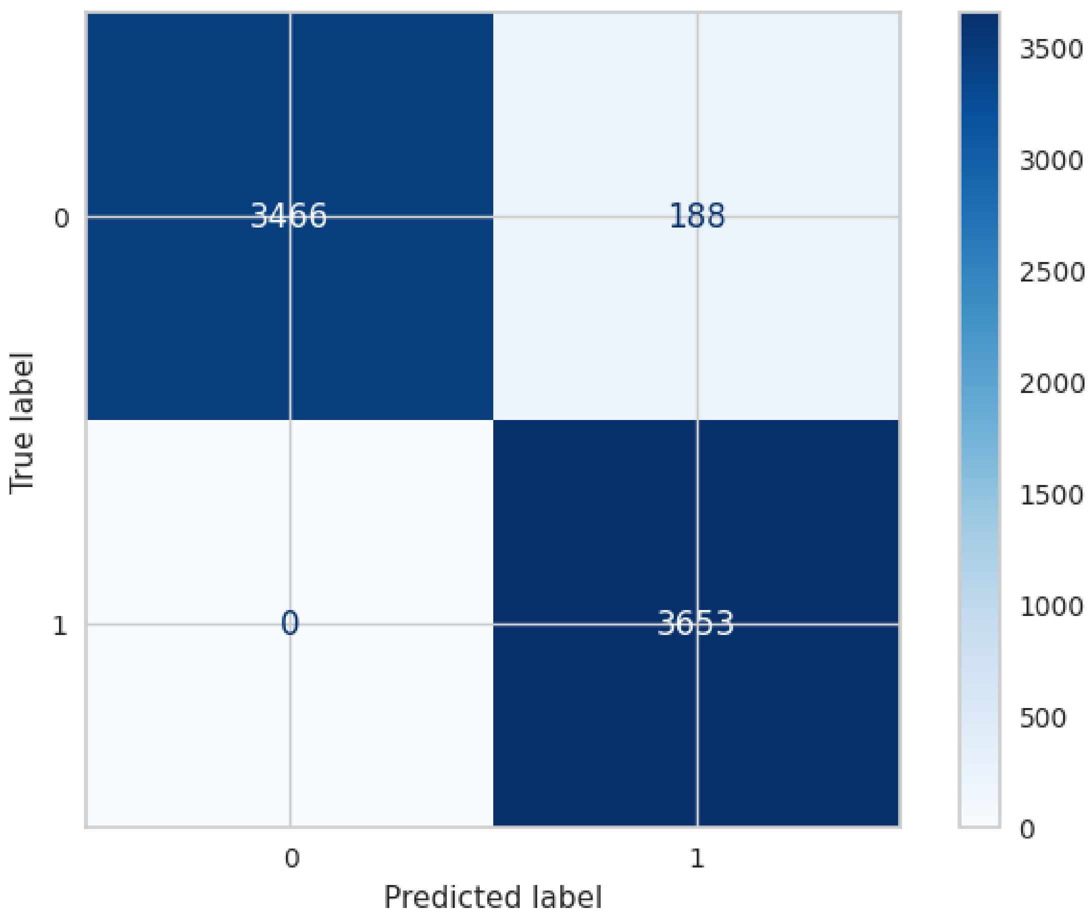

3386
268
3
3650
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500

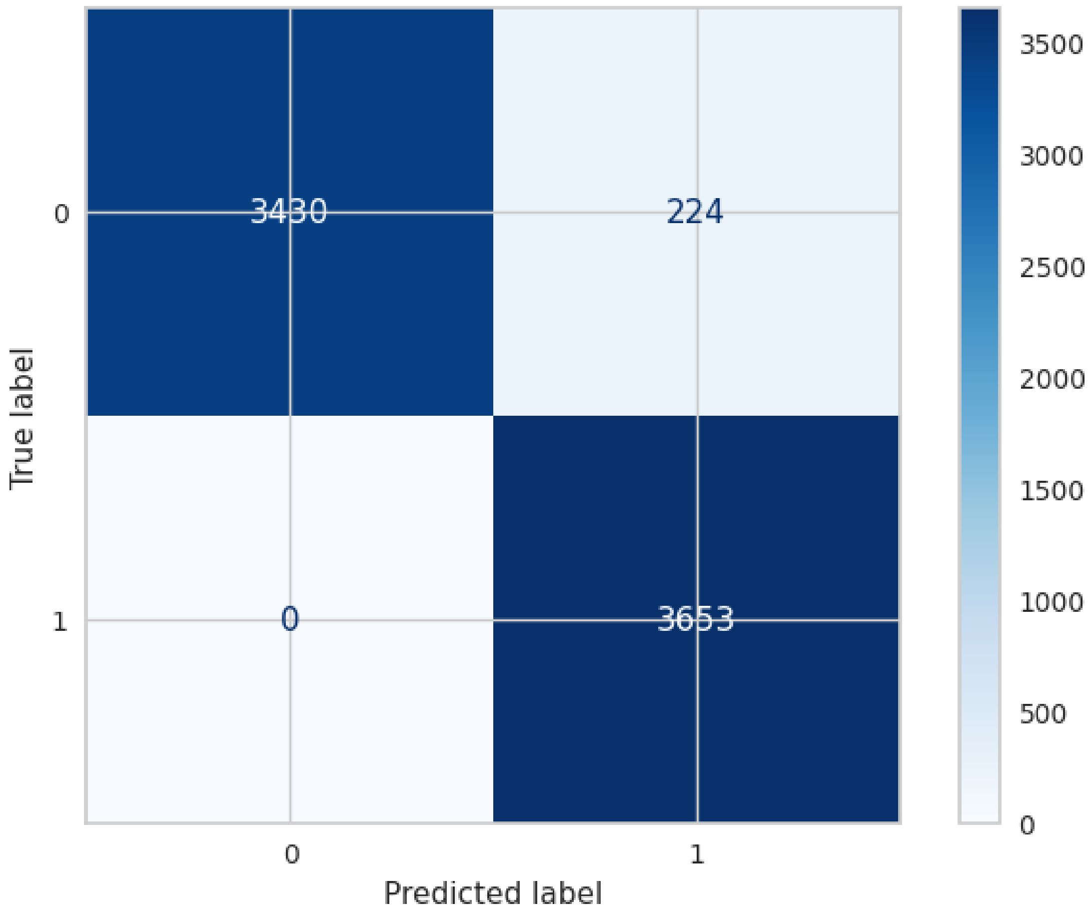

3452
202
0
3653
True label
Predicted label
0
1
0
1
3500
3000
2500
2000
1500
1000
500
0


3413
241
0
3653
True label
Predicted label
0
1
0
1
3500
3000
2500
2000
1500
1000
500
0



3442
212
0
3653
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500
0


3466
188
0
3653
True label
Predicted label
0
1
0
1
3500
3000
2500
2000
1500
1000
500
0



[322]:
```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
 ↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['Gradient Over','Gradient Over With Feature','Gradient Over␣
 ↪Scaling','Gradient Over With Normalize','Gradient Over With PCA'
                ,'Gradient Over With PCA and Scaling',
                'Gradient Over With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[322]:
```
                                      Train Accuracy  Test Accuracy   Test F1  \
Models
Gradient Over                               0.999924       0.973450  0.974133
Gradient Over With Feature                  0.988383       0.962912  0.964206
Gradient Over Scaling                       0.999924       0.972355  0.973095
Gradient Over With Normalize                0.999924       0.967018  0.968067
Gradient Over With PCA                      0.999924       0.970987  0.971801
Gradient Over With PCA and Scaling          0.999924       0.974271  0.974913
Gradient Over With PCA and Normalize        0.999924       0.969344  0.970252
```

```
                                        Test Recall  Test Precision       AUC
Models
Gradient Over                              1.000000        0.949571  0.973454
Gradient Over With Feature                 0.999179        0.931598  0.962917
Gradient Over Scaling                      1.000000        0.947601  0.972359
Gradient Over With Normalize               1.000000        0.938110  0.967022
Gradient Over With PCA                     1.000000        0.945149  0.970991
Gradient Over With PCA and Scaling         1.000000        0.951054  0.974275
Gradient Over With PCA and Normalize       1.000000        0.942223  0.969349
```

[323]:
```
models_draw(df)
```

RandomUnderSampler

[324]:
```
X_train,y_train,X_test,y_test=Split(X_classification_under,y_classification_under)
```

```
X_train shape is  (8350, 20)
X_test shape is  (928, 20)
y_train shape is  (8350,)
y_test shape is  (928,)
```

[325]:
```
Search(GradientBoostingClassifier(n_estimators=100,max_depth=3) ,{'max_depth':
↪[5,10,20,25,30,40]},X_train,y_train)
```

[325]:
```
GradientBoostingClassifier(max_depth=5)
```

[326]:
```
cross_validation(GradientBoostingClassifier(n_estimators=100,max_depth=5),X_train,y_train)
```

```
Train Score Value :  [0.93233533 0.93547904 0.93547904 0.93637725 0.93607784]
Mean 0.9351497005988024
Test Score Value :  [0.90179641 0.88622754 0.89221557 0.88682635 0.89161677]
Mean 0.8917365269461077
```

[327]:
```
Values =␣
↪Models(GradientBoostingClassifier(n_estimators=100,max_depth=5),X_train,y_train,X_test,y_te
```

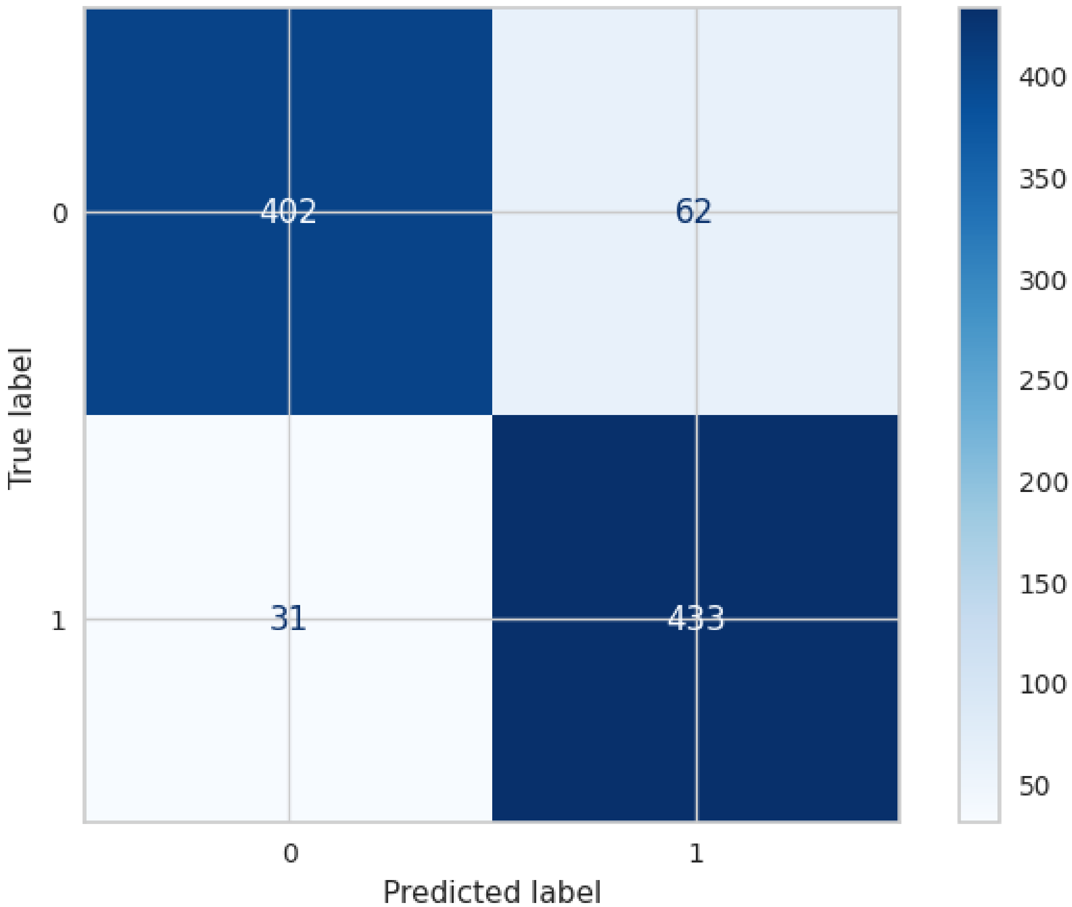
```
Apply Model With Normal Data :

Model Train Score is :  0.9286227544910179
Model Test Score is :  0.8997844827586207
F1 Score is :  0.9030239833159541
Recall Score is :  0.9331896551724138
Precision Score is :  0.8747474747474747
AUC Value  :  0.8997844827586207

Classification Report is :               precision    recall  f1-score
support

           0       0.93      0.87      0.90       464
```

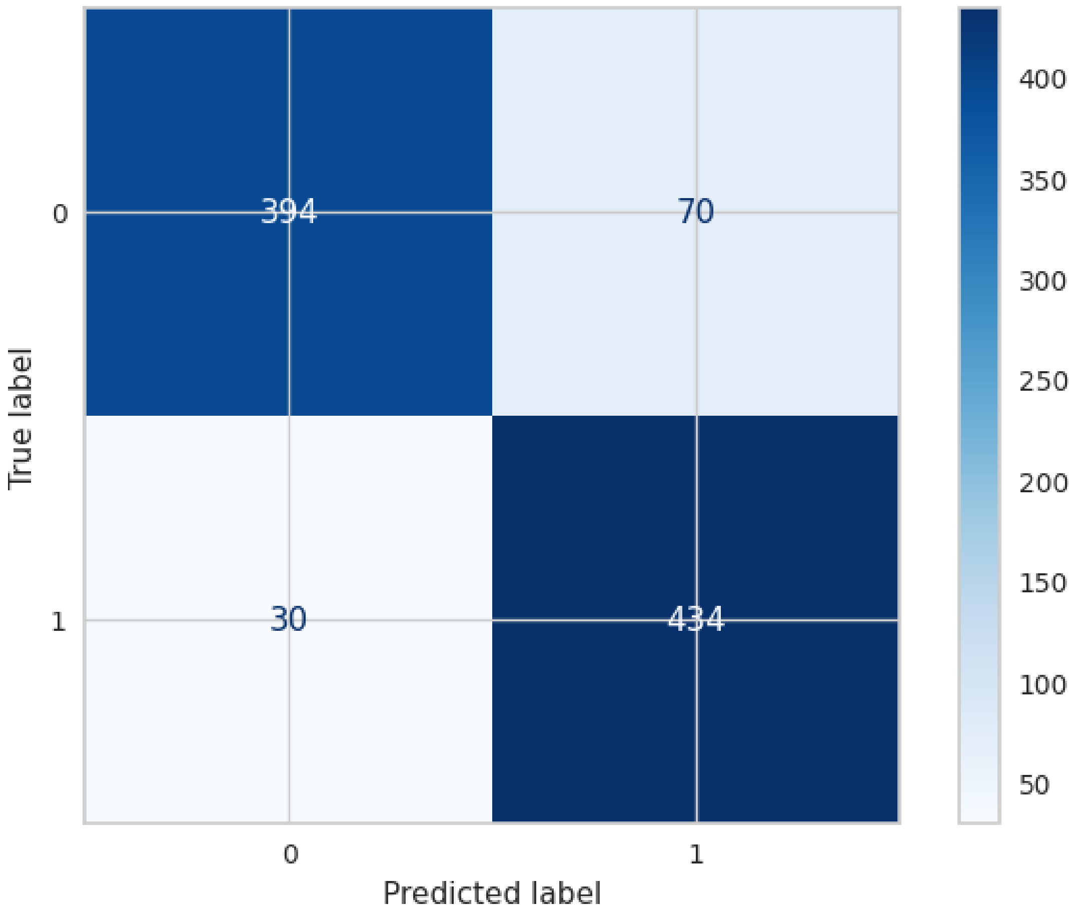
```
           1       0.87      0.93      0.90       464

    accuracy                           0.90       928
   macro avg       0.90      0.90      0.90       928
weighted avg       0.90      0.90      0.90       928

Confusion Matrix is :
 [[402  62]
 [ 31 433]]


 Apply Model With Feature Selection :

Model Train Score is :  0.9196407185628742
Model Test Score is :  0.8922413793103449
F1 Score is :  0.8966942148760331
Recall Score is :  0.9353448275862069
Precision Score is :  0.8611111111111112
AUC Value  :  0.8922413793103449
Classification Report is :                precision    recall  f1-score
support

           0       0.93      0.85      0.89       464
           1       0.86      0.94      0.90       464

    accuracy                           0.89       928
   macro avg       0.90      0.89      0.89       928
weighted avg       0.90      0.89      0.89       928

Confusion Matrix is :
 [[394  70]
 [ 30 434]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.9286227544910179
Model Test Score is :  0.8997844827586207
F1 Score is :  0.9030239833159541
Recall Score is :  0.9331896551724138
Precision Score is :  0.8747474747474747
AUC Value  :  0.8997844827586207
Classification Report is :                precision    recall  f1-score
support

           0       0.93      0.87      0.90       464
           1       0.87      0.93      0.90       464
```

```
    accuracy                           0.90       928
   macro avg       0.90      0.90      0.90       928
weighted avg       0.90      0.90      0.90       928

Confusion Matrix is :
 [[402  62]
 [ 31 433]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.9281437125748503
Model Test Score is :  0.8976293103448276
F1 Score is :  0.9007314524555905
Recall Score is :  0.9288793103448276
Precision Score is :  0.8742393509127789
AUC Value  :  0.8976293103448276

Classification Report is :                precision    recall  f1-score
support

           0       0.92      0.87      0.89       464
           1       0.87      0.93      0.90       464

    accuracy                           0.90       928
   macro avg       0.90      0.90      0.90       928
weighted avg       0.90      0.90      0.90       928

Confusion Matrix is :
 [[402  62]
 [ 33 431]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.9335329341317365
Model Test Score is :  0.8987068965517241
F1 Score is :  0.9024896265560166
Recall Score is :  0.9375
Precision Score is :  0.87
AUC Value  :  0.8987068965517242

Classification Report is :                precision    recall  f1-score
support

           0       0.93      0.86      0.89       464
           1       0.87      0.94      0.90       464
```

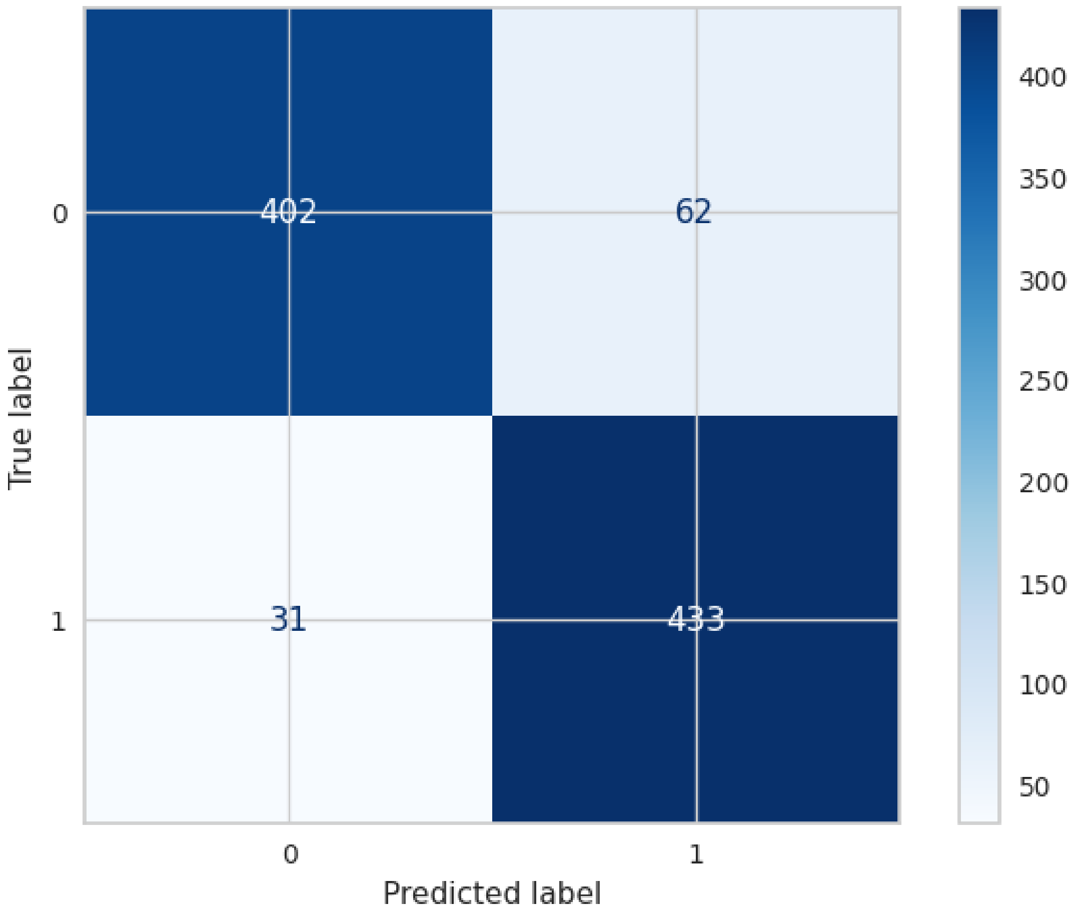
```
   accuracy                           0.90       928
  macro avg       0.90      0.90      0.90       928
weighted avg      0.90      0.90      0.90       928

Confusion Matrix is :
 [[399  65]
 [ 29 435]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  0.9341317365269461
Model Test Score is :  0.8782327586206896
F1 Score is :  0.8819226750261233
Recall Score is :  0.9094827586206896
Precision Score is :  0.8559837728194726
AUC Value  :  0.8782327586206896
Classification Report is :                precision    recall  f1-score
support

           0      0.90      0.85      0.87       464
           1      0.86      0.91      0.88       464

    accuracy                          0.88       928
   macro avg      0.88      0.88      0.88       928
weighted avg      0.88      0.88      0.88       928

Confusion Matrix is :
 [[393  71]
 [ 42 422]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  0.934251497005988
Model Test Score is :  0.8890086206896551
F1 Score is :  0.8923719958202716
Recall Score is :  0.9202586206896551
Precision Score is :  0.8661257606490872
AUC Value  :  0.8890086206896551
Classification Report is :                precision    recall  f1-score
support

           0      0.91      0.86      0.89       464
           1      0.87      0.92      0.89       464

    accuracy                          0.89       928
```

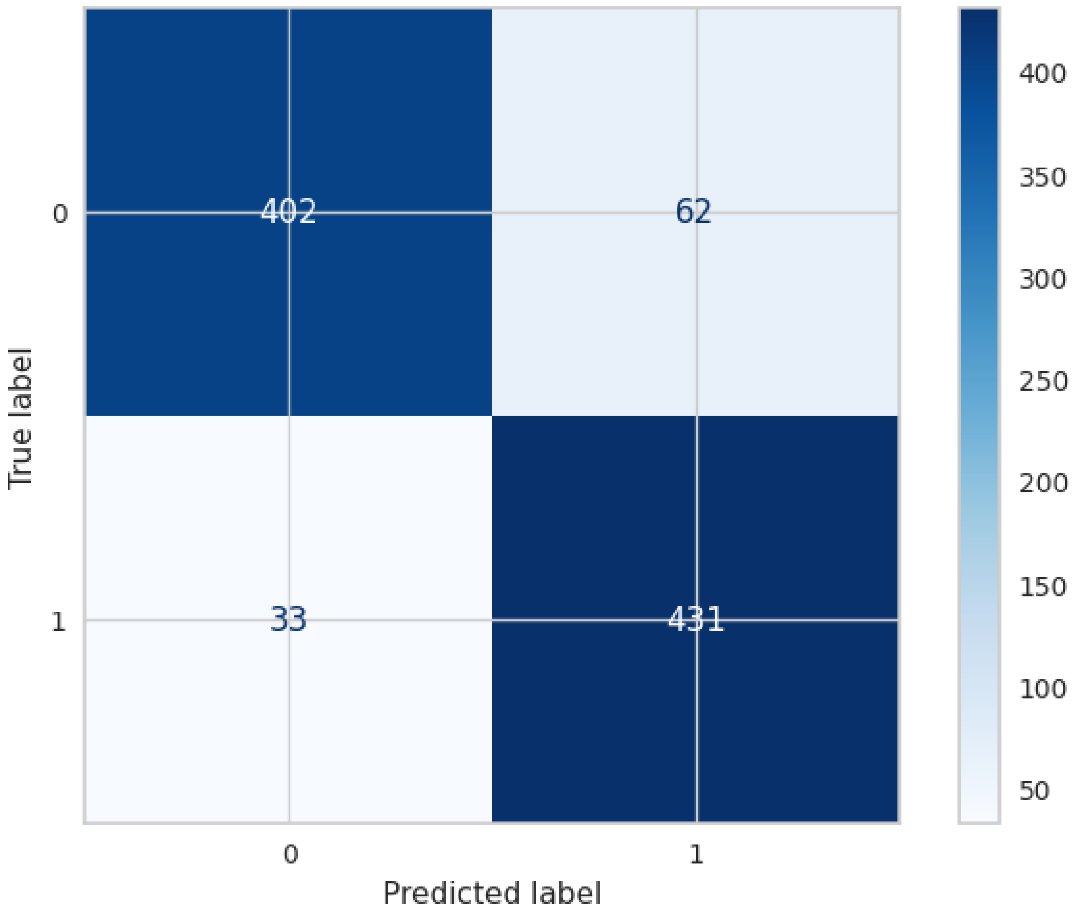
```
   macro avg       0.89      0.89      0.89       928
weighted avg       0.89      0.89      0.89       928

Confusion Matrix is :
 [[398  66]
 [ 37 427]]
```

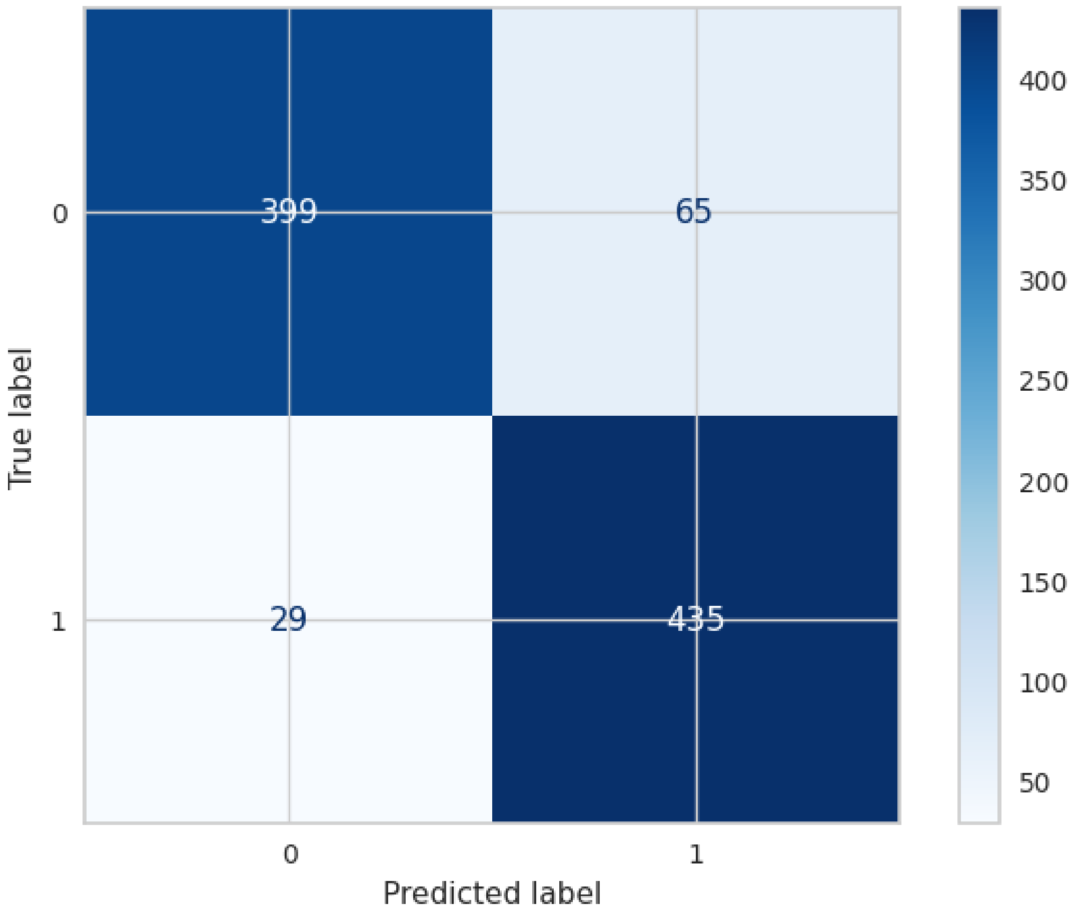


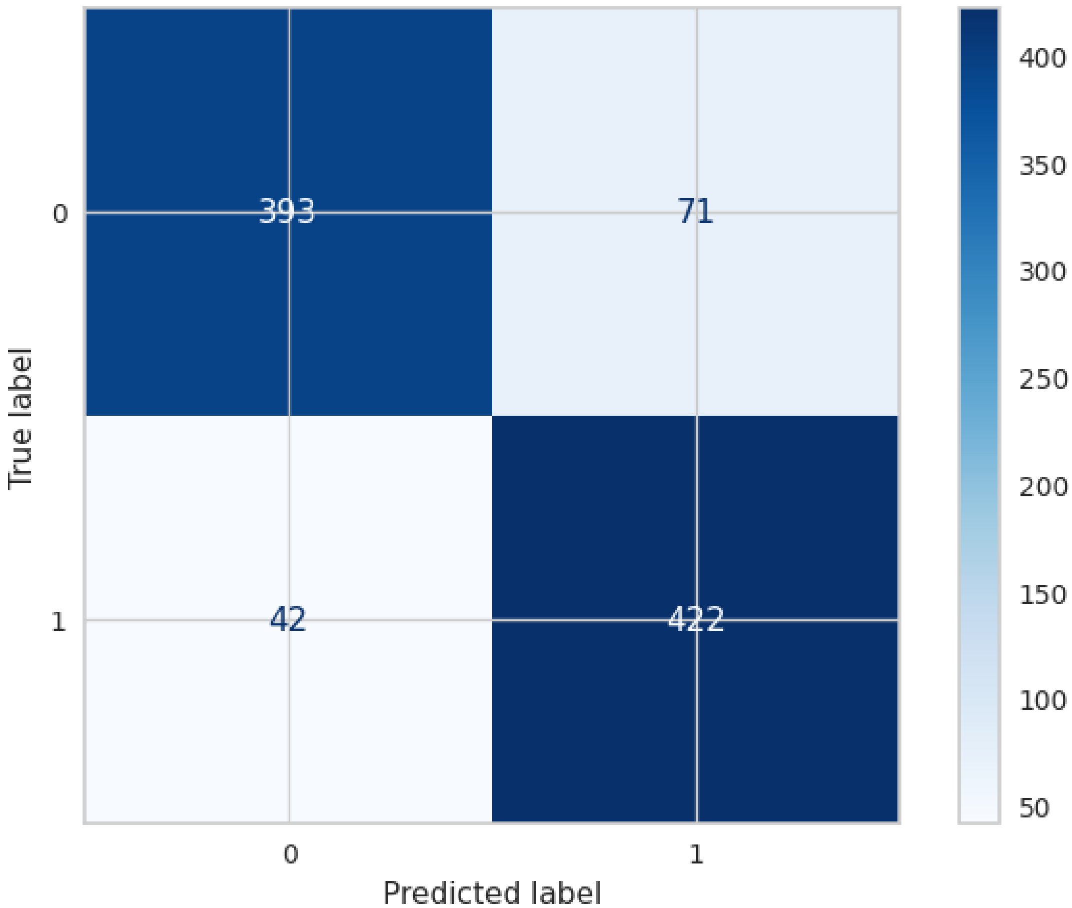


394
70
30
434
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50

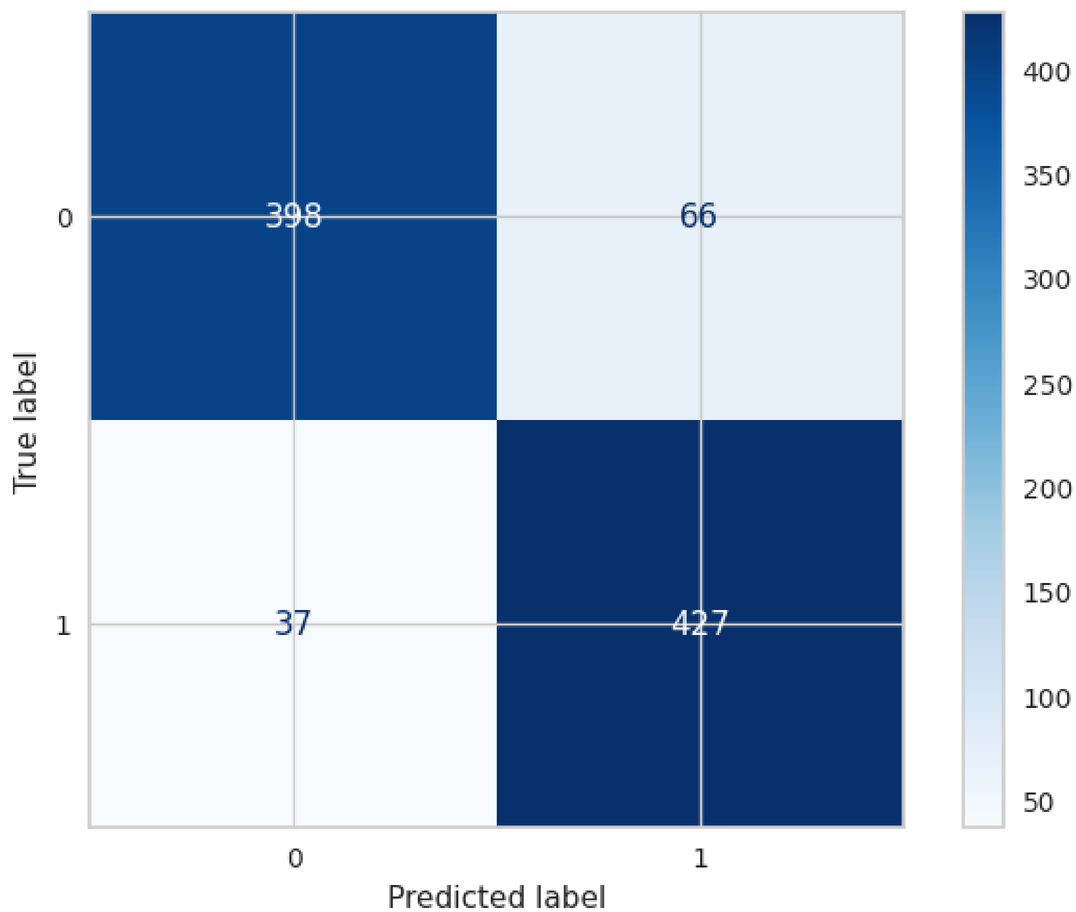

402
62
31
433
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50


402
62
33
431
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50


399
65
29
435
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50



0
1
393
71
42
422
True label
Predicted label
400
350
300
250
200
150
100
50



[328]:
```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['Gradient Under','Gradient Under With Feature','Gradient Under␣
↪Scaling','Gradient Under With Normalize','Gradient Under With PCA'
                ,'Gradient Under With PCA and Scaling',
                'Gradient Under With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[328]:
```
                                       Train Accuracy  Test Accuracy  \
Models
Gradient Under                               0.928623       0.899784
Gradient Under With Feature                  0.919641       0.892241
Gradient Under Scaling                       0.928623       0.899784
Gradient Under With Normalize                0.928144       0.897629
Gradient Under With PCA                      0.933533       0.898707
Gradient Under With PCA and Scaling          0.934132       0.878233
Gradient Under With PCA and Normalize        0.934251       0.889009
```

```
                                           Test F1  Test Recall  Test Precision  \
Models
Gradient Under                            0.903024     0.933190        0.874747
Gradient Under With Feature               0.896694     0.935345        0.861111
Gradient Under Scaling                    0.903024     0.933190        0.874747
Gradient Under With Normalize             0.900731     0.928879        0.874239
Gradient Under With PCA                   0.902490     0.937500        0.870000
Gradient Under With PCA and Scaling       0.881923     0.909483        0.855984
Gradient Under With PCA and Normalize     0.892372     0.920259        0.866126

                                               AUC
Models
Gradient Under                            0.899784
Gradient Under With Feature               0.892241
Gradient Under Scaling                    0.899784
Gradient Under With Normalize             0.897629
Gradient Under With PCA                   0.898707
Gradient Under With PCA and Scaling       0.878233
Gradient Under With PCA and Normalize     0.889009
```

[329]:

```
models_draw(df)
```

SGDClassifier

[330]:

```
X_train,y_train,X_test,y_test=Split(X_classification,y_classification)
```

```
X_train shape is  (37056, 20)
X_test shape is  (4118, 20)
y_train shape is  (37056,)
y_test shape is  (4118,)
```

[331]:

```
cross_validation(SGDClassifier(penalty='l2'),X_train,y_train)
```

```
Train Score Value :  [0.88864526 0.90632484 0.90878732 0.82030697 0.90527914]
Mean 0.885868703965303
Test Score Value :  [0.88869401 0.90460127 0.90635542 0.8160842  0.90918904]
Mean 0.8849847876397325
```

[332]:

```
Values = Models(SGDClassifier(penalty='l2'),X_train,y_train,X_test,y_test)
```

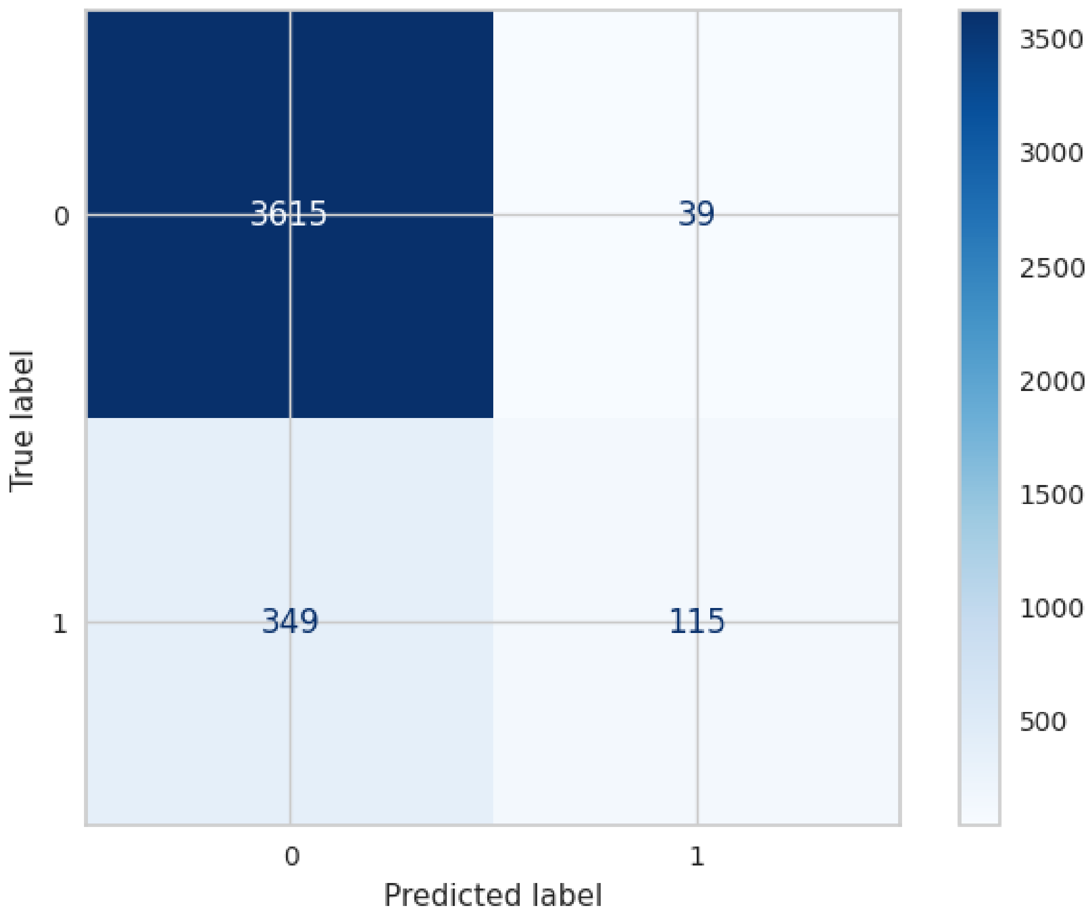
```
Apply Model With Normal Data :

Model Train Score is :  0.9041450777202072
Model Test Score is :  0.9057795046138902
F1 Score is :  0.37216828478964403
Recall Score is :  0.2478448275862069
Precision Score is :  0.7467532467532467
AUC Value  :  0.6185857963875205
```

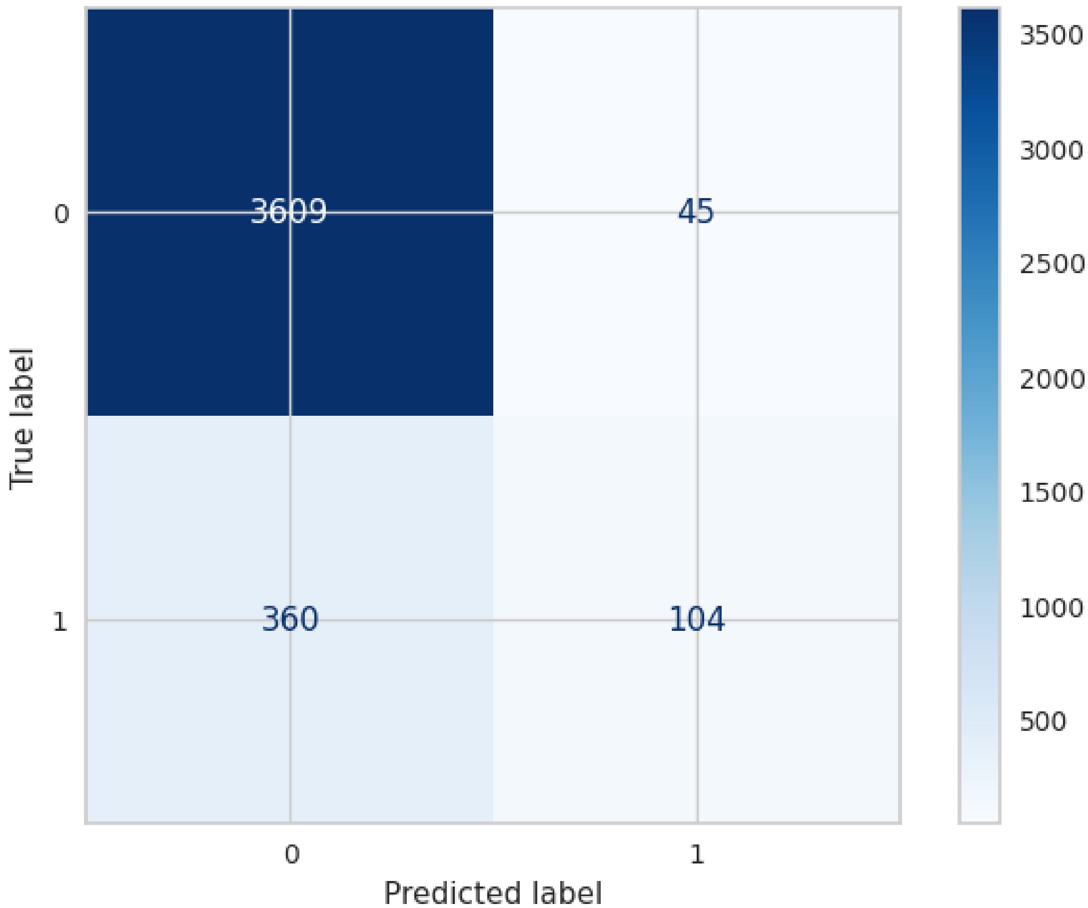
```
Classification Report is :                precision    recall  f1-score
support

           0       0.91      0.99      0.95      3654
           1       0.75      0.25      0.37       464

    accuracy                           0.91      4118
   macro avg       0.83      0.62      0.66      4118
weighted avg       0.89      0.91      0.88      4118

Confusion Matrix is :
 [[3615   39]
 [ 349  115]]


 Apply Model With Feature Selection :

Model Train Score is :  0.9012035837651122
Model Test Score is :  0.90165128703254
F1 Score is :  0.33931484502446985
Recall Score is :  0.22413793103448276
Precision Score is :  0.697986577181208
AUC Value  :  0.605911330049261
Classification Report is :                precision    recall  f1-score
support

           0       0.91      0.99      0.95      3654
           1       0.70      0.22      0.34       464

    accuracy                           0.90      4118
   macro avg       0.80      0.61      0.64      4118
weighted avg       0.89      0.90      0.88      4118

Confusion Matrix is :
 [[3609   45]
 [ 360  104]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.9061960276338514
Model Test Score is :  0.9052938319572608
F1 Score is :  0.39999999999999997
Recall Score is :  0.2801724137931034
Precision Score is :  0.6989247311827957
AUC Value  :  0.6324233716475096
Classification Report is :                precision    recall  f1-score
```

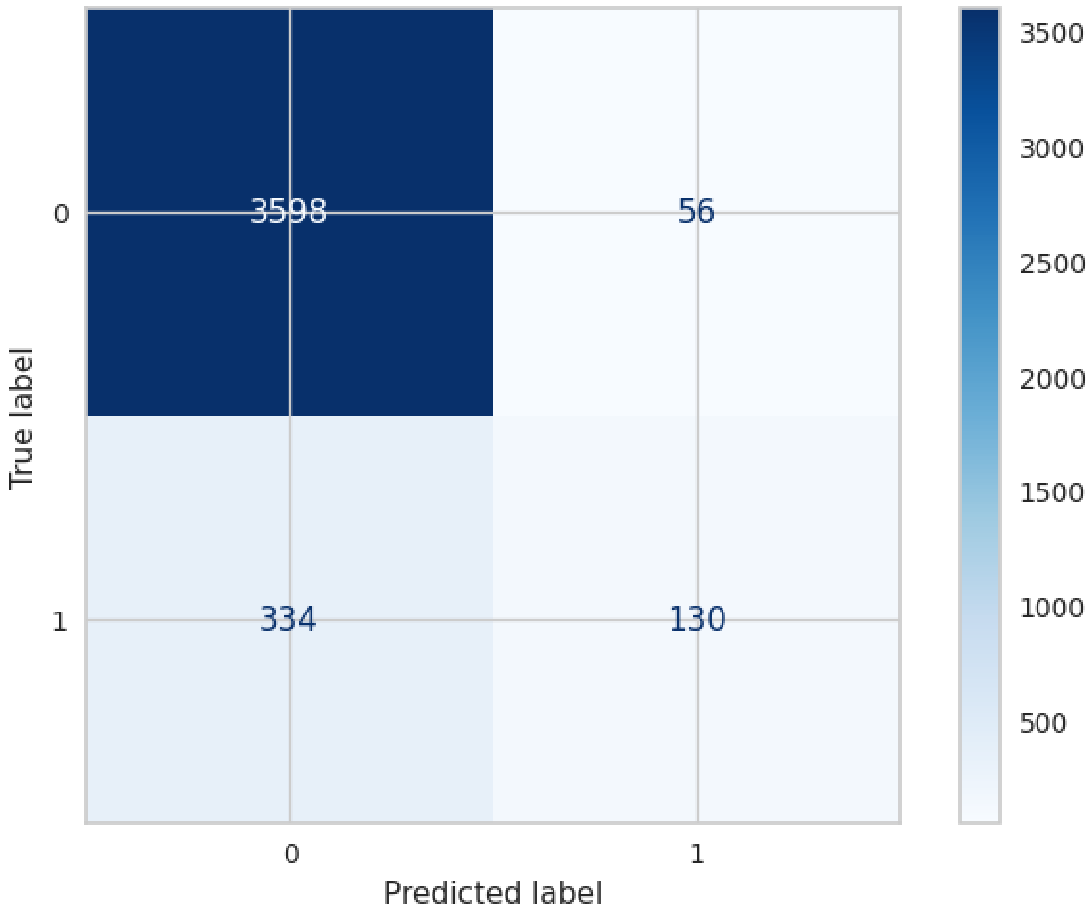
```
support

           0       0.92      0.98      0.95      3654
           1       0.70      0.28      0.40       464

    accuracy                           0.91      4118
   macro avg       0.81      0.63      0.67      4118
weighted avg       0.89      0.91      0.89      4118

Confusion Matrix is :
 [[3598   56]
 [ 334  130]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.8958333333333334
Model Test Score is :  0.8984944147644488
F1 Score is :  0.25089605734767023
Recall Score is :  0.15086206896551724
Precision Score is :  0.7446808510638298
AUC Value  :  0.5721469622331692

Classification Report is :                precision    recall  f1-score
support

           0       0.90      0.99      0.95      3654
           1       0.74      0.15      0.25       464

    accuracy                           0.90      4118
   macro avg       0.82      0.57      0.60      4118
weighted avg       0.88      0.90      0.87      4118

Confusion Matrix is :
 [[3630   24]
 [ 394   70]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.9064389032815199
Model Test Score is :  0.9074793589120933
F1 Score is :  0.4744827586206896
Recall Score is :  0.3706896551724138
Precision Score is :  0.6590038314176245
AUC Value  :  0.6731663929939792

Classification Report is :                precision    recall  f1-score
support
```

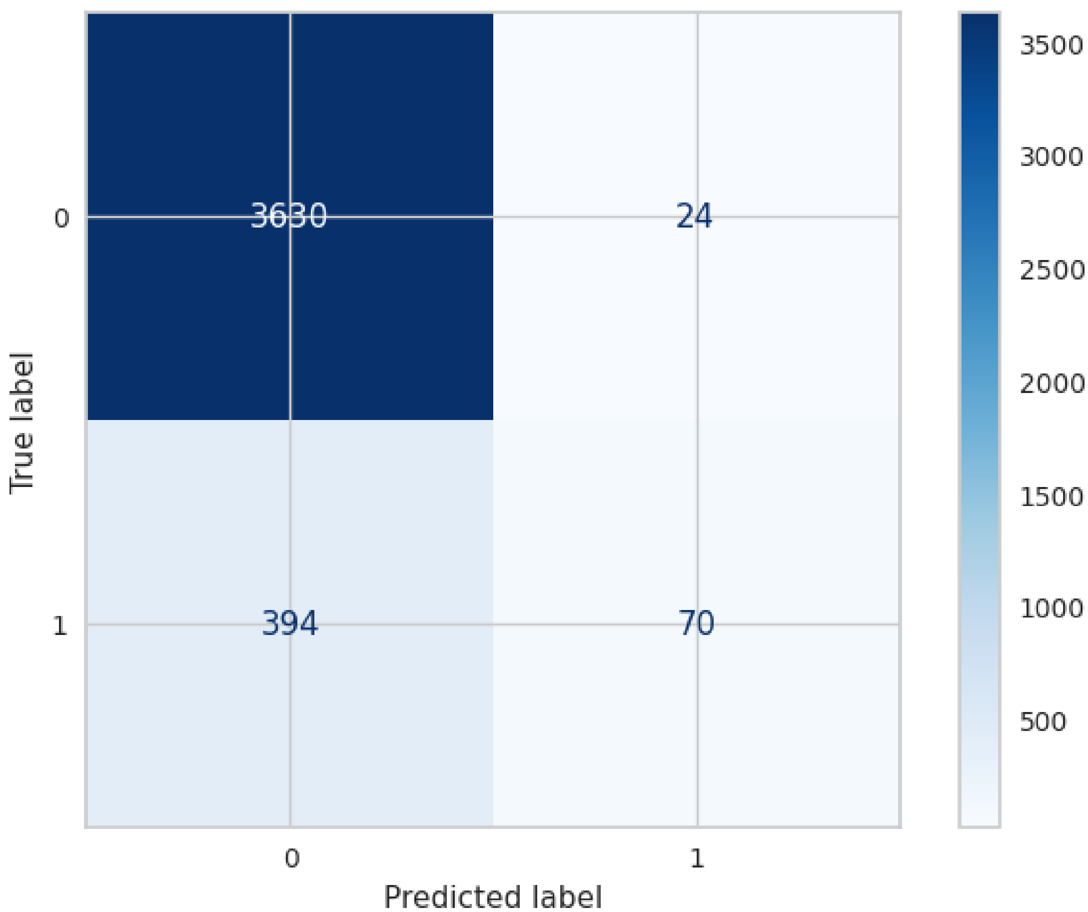
```
           0       0.92      0.98      0.95      3654
           1       0.66      0.37      0.47       464

    accuracy                           0.91      4118
   macro avg       0.79      0.67      0.71      4118
weighted avg       0.89      0.91      0.90      4118

Confusion Matrix is :
 [[3565   89]
 [ 292  172]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  0.9048197322970639
Model Test Score is :  0.9045653229723166
F1 Score is :  0.37120000000000003
Recall Score is :  0.25
Precision Score is :  0.7204968944099379
AUC Value  :  0.6188423645320197

Classification Report is :                precision    recall  f1-score
support

           0       0.91      0.99      0.95      3654
           1       0.72      0.25      0.37       464

    accuracy                           0.90      4118
   macro avg       0.82      0.62      0.66      4118
weighted avg       0.89      0.90      0.88      4118

Confusion Matrix is :
 [[3609   45]
 [ 348  116]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  0.8950777202072538
Model Test Score is :  0.8975230694511899
F1 Score is :  0.23550724637681159
Recall Score is :  0.1400862068965517
Precision Score is :  0.7386363636363636
AUC Value  :  0.5668958675424193

Classification Report is :                precision    recall  f1-score
support
```

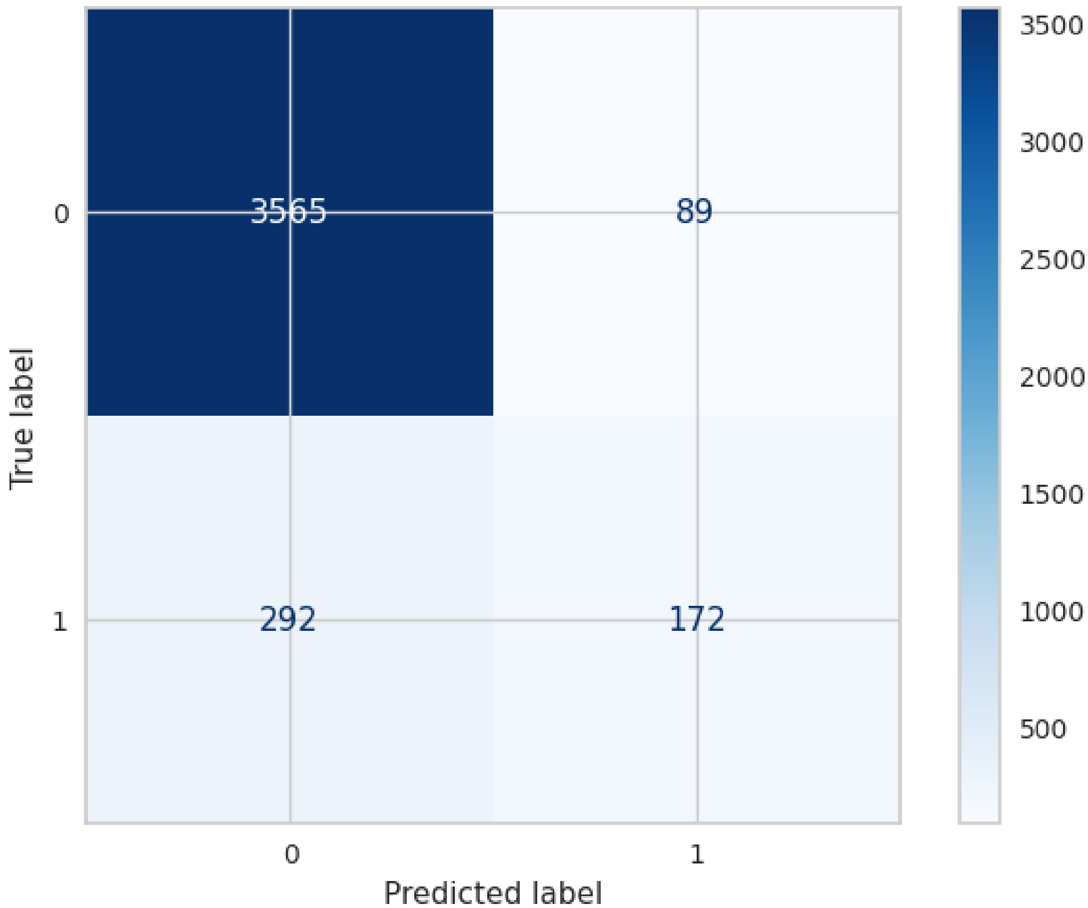
```
           0       0.90      0.99      0.95      3654
           1       0.74      0.14      0.24       464

    accuracy                           0.90      4118
   macro avg       0.82      0.57      0.59      4118
weighted avg       0.88      0.90      0.87      4118

Confusion Matrix is :
 [[3631   23]
 [ 399   65]]
```

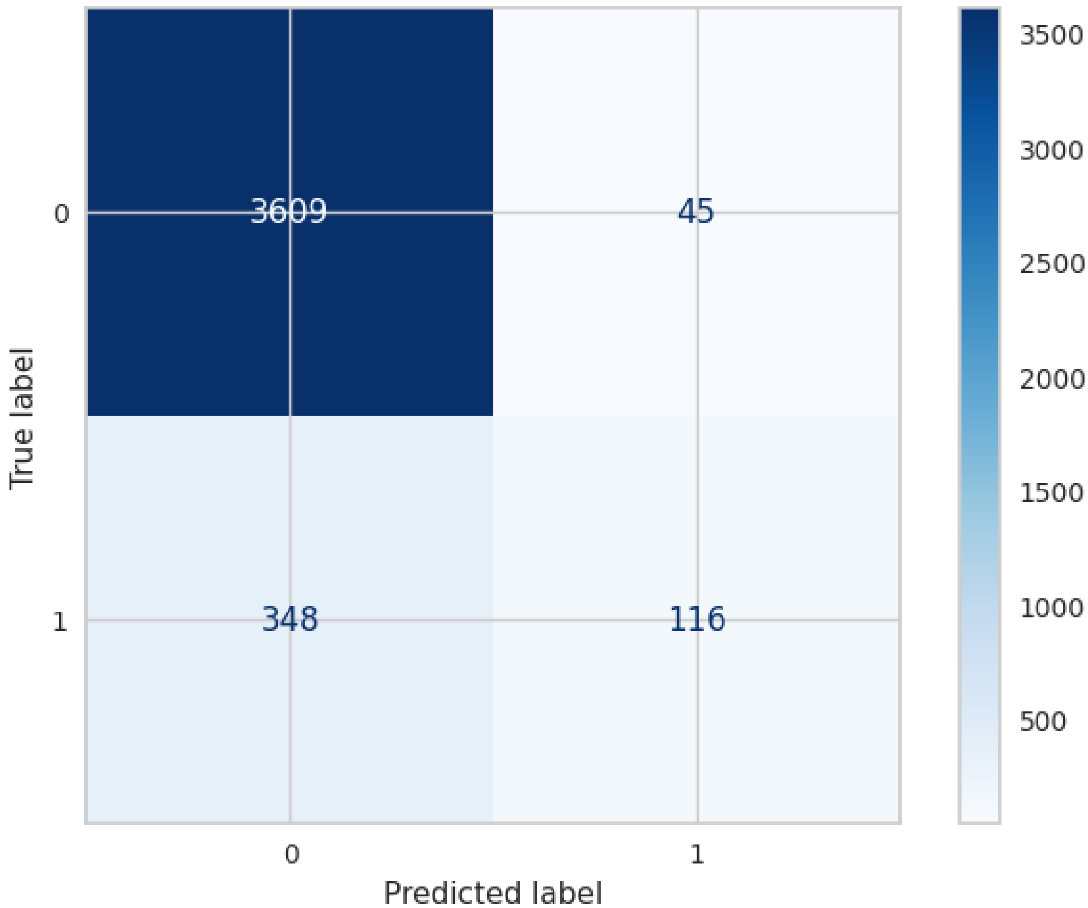



3609
45
360
104
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500

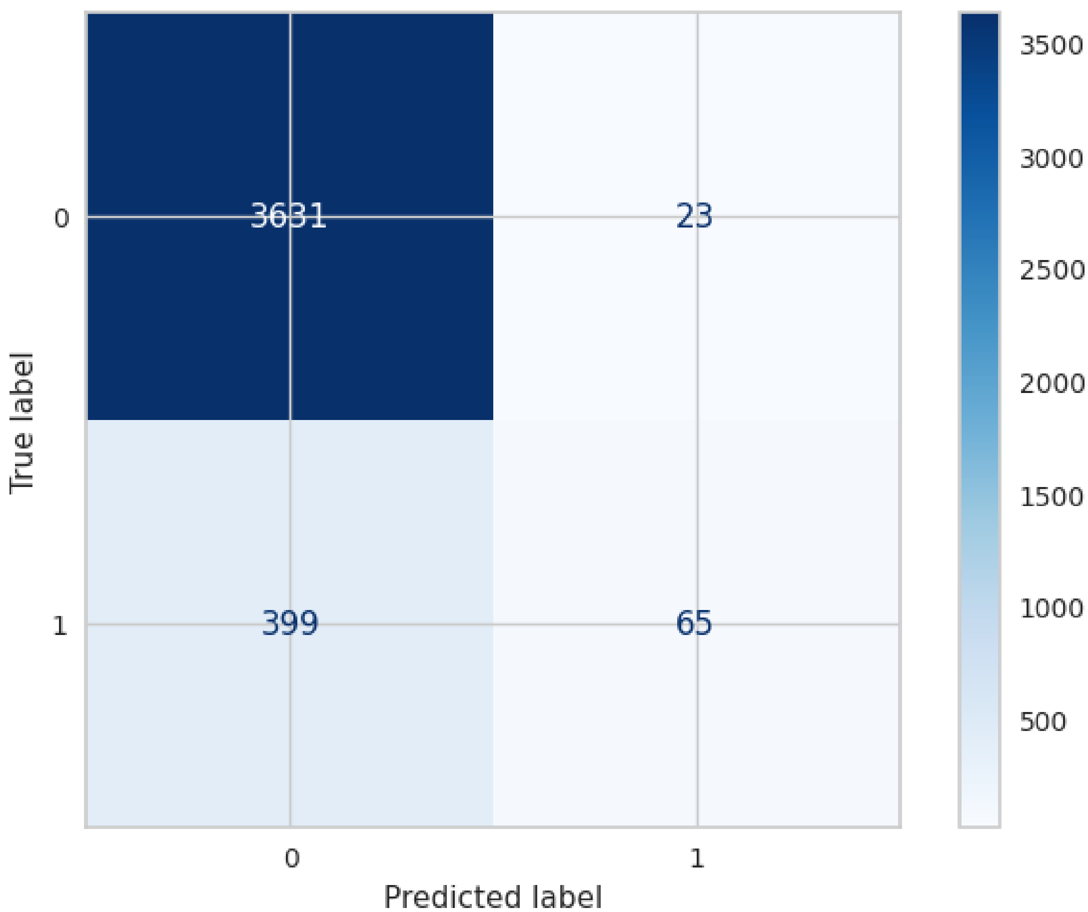

3598
56
334
130
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500


True label
Predicted label
0
1
0
1
3630
24
394
70
3500
3000
2500
2000
1500
1000
500


3565
89
292
172
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500



3609
45
348
116
0
1
True label
0
1
Predicted label
3500
3000
2500
2000
1500
1000
500



[333]:
```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['SGD','SGD With Feature','SGD Scaling','SGD With␣
↪Normalize','SGD With PCA'
                ,'SGD With PCA and Scaling',
                'SGD With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[333]:
```
                            Train Accuracy  Test Accuracy   Test F1  \
Models
SGD                               0.904145       0.905780  0.372168
SGD With Feature                  0.901204       0.901651  0.339315
SGD Scaling                       0.906196       0.905294  0.400000
SGD With Normalize                0.895833       0.898494  0.250896
SGD With PCA                      0.906439       0.907479  0.474483
SGD With PCA and Scaling          0.904820       0.904565  0.371200
SGD With PCA and Normalize        0.895078       0.897523  0.235507
```

```
                              Test Recall  Test Precision       AUC
 Models
 SGD                             0.247845        0.746753  0.618586
 SGD With Feature                0.224138        0.697987  0.605911
 SGD Scaling                     0.280172        0.698925  0.632423
 SGD With Normalize              0.150862        0.744681  0.572147
 SGD With PCA                    0.370690        0.659004  0.673166
 SGD With PCA and Scaling        0.250000        0.720497  0.618842
 SGD With PCA and Normalize      0.140086        0.738636  0.566896
```

[334]:
```
models_draw(df)
```

RandomOverSampler

[335]:
```
X_train,y_train,X_test,y_test=Split(X_classification_over,y_classification_over)
```

```
X_train shape is  (65763, 20)
X_test shape is  (7307, 20)
y_train shape is  (65763,)
y_test shape is  (7307,)
```

[336]:
```
cross_validation(SGDClassifier(penalty='l2'),X_train,y_train)
```

```
Train Score Value :  [0.85246151 0.58165748 0.84092378 0.85054456 0.85280645]
Mean 0.7956787555559389
Test Score Value :  [0.85410173 0.58435338 0.8424694  0.85302616 0.84876825]
Mean 0.7965437815616802
```

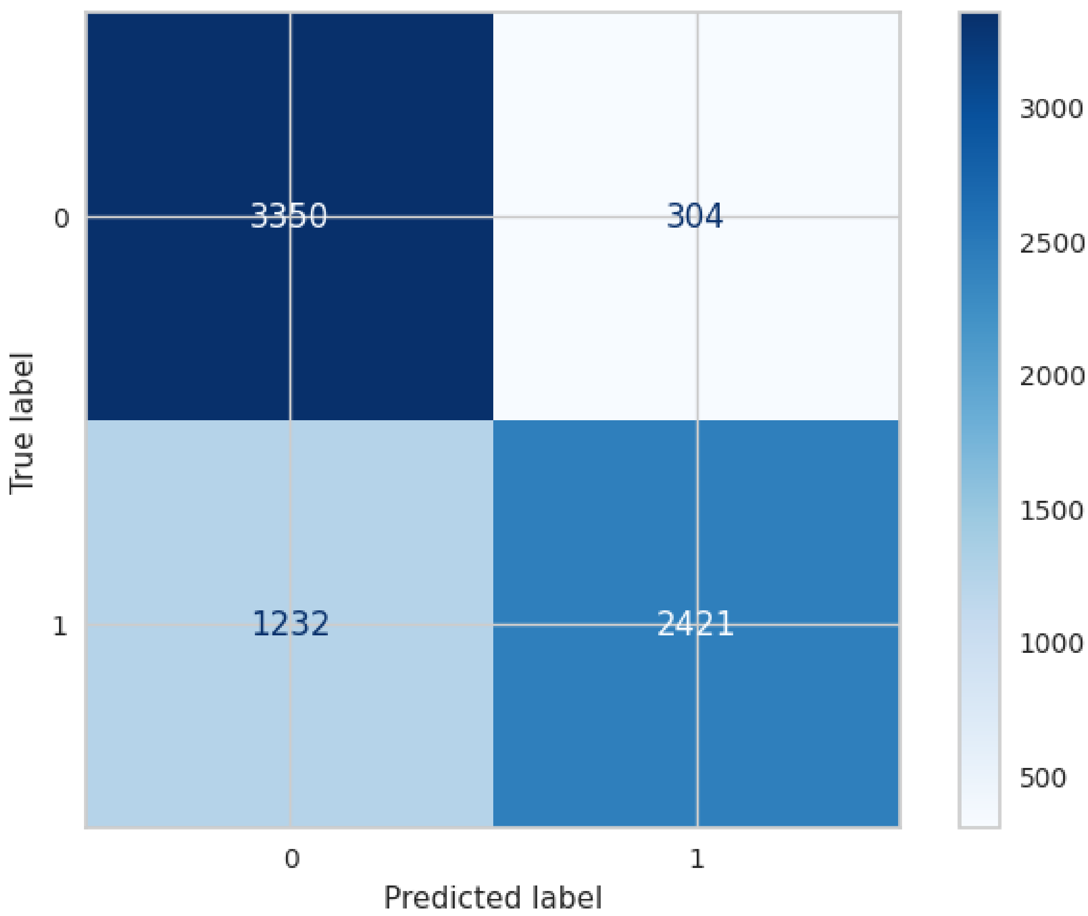
[337]:
```
Values = Models(SGDClassifier(penalty='l2'),X_train,y_train,X_test,y_test)
```

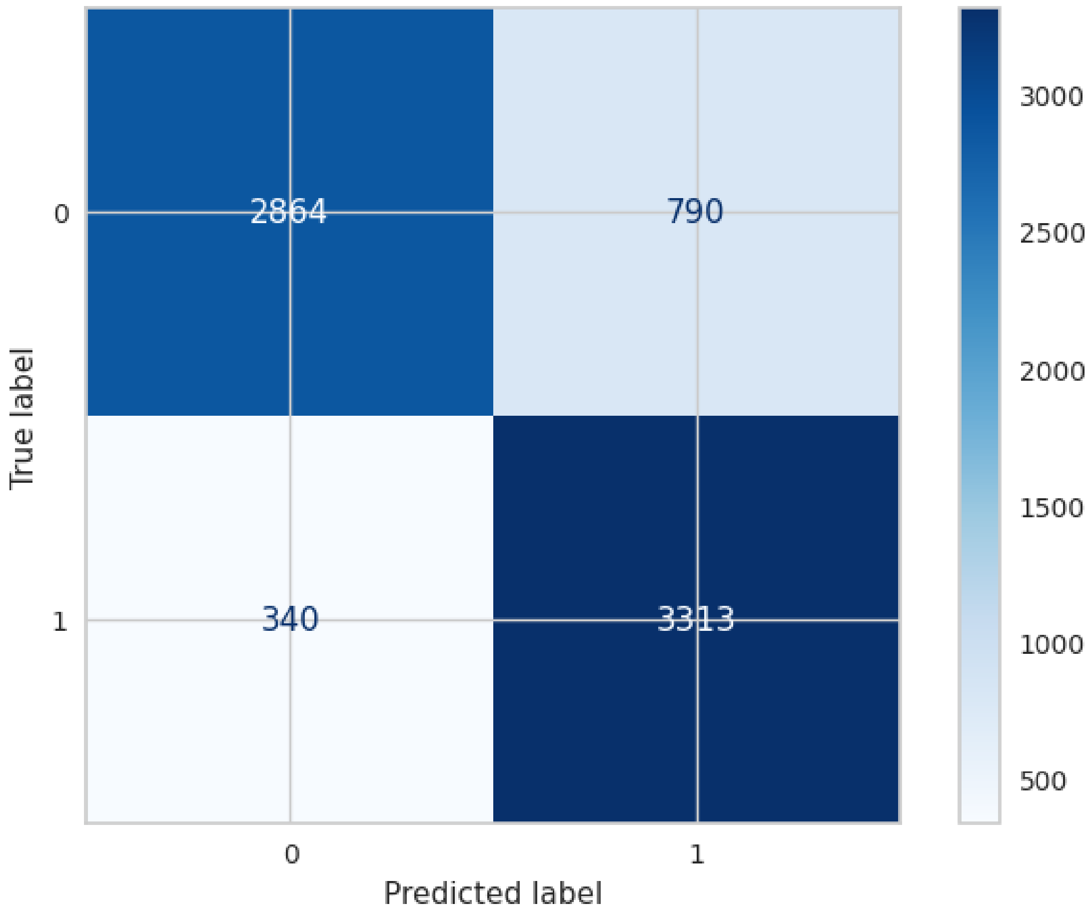
```
Apply Model With Normal Data :

Model Train Score is :  0.7794200386235421
Model Test Score is :  0.789790611742165
F1 Score is :  0.7591721542803387
Recall Score is :  0.6627429509991788
Precision Score is :  0.8884403669724771
AUC Value  :  0.7897732270047891

Classification Report is :                precision    recall  f1-score
support

           0       0.73      0.92      0.81      3654
           1       0.89      0.66      0.76      3653

    accuracy                           0.79      7307
   macro avg       0.81      0.79      0.79      7307
weighted avg       0.81      0.79      0.79      7307
```

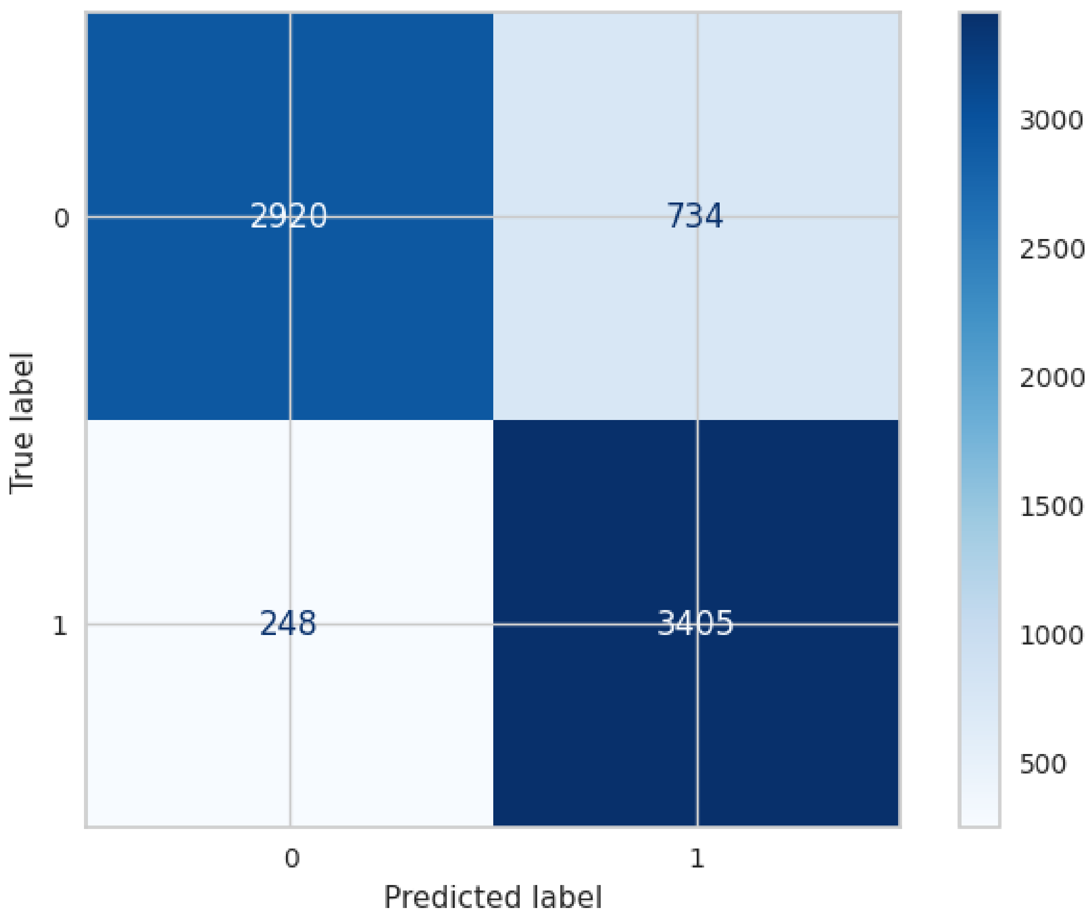
```
Confusion Matrix is :
 [[3350  304]
 [1232 2421]]


 Apply Model With Feature Selection :

Model Train Score is :  0.8419932180709517
Model Test Score is :  0.8453537703571917
F1 Score is :  0.8543063434760186
Recall Score is :  0.906925814399124
Precision Score is :  0.8074579575920059
AUC Value  :  0.8453621956505746

Classification Report is :                precision    recall  f1-score
support

           0       0.89      0.78      0.84      3654
           1       0.81      0.91      0.85      3653

    accuracy                           0.85      7307
   macro avg       0.85      0.85      0.84      7307
weighted avg       0.85      0.85      0.84      7307

Confusion Matrix is :
 [[2864  790]
 [ 340 3313]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.8625062725240636
Model Test Score is :  0.8656083207882852
F1 Score is :  0.8739733059548256
Recall Score is :  0.9321105940323022
Precision Score is :  0.822662478859628
AUC Value  :  0.8656174207162058

Classification Report is :                precision    recall  f1-score
support

           0       0.92      0.80      0.86      3654
           1       0.82      0.93      0.87      3653

    accuracy                           0.87      7307
   macro avg       0.87      0.87      0.87      7307
weighted avg       0.87      0.87      0.87      7307

Confusion Matrix is :
```

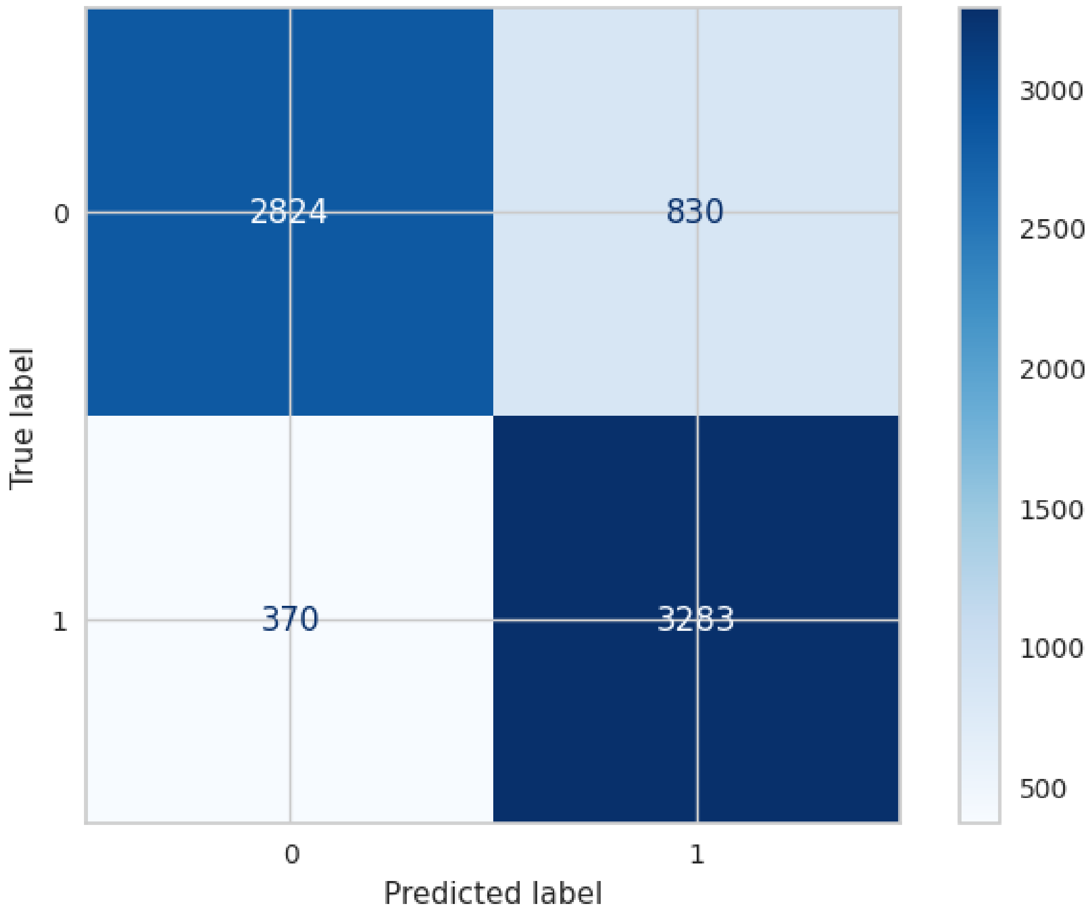
```
 [[2920  734]
 [ 248 3405]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.8340708301020331
Model Test Score is :  0.8357739154235665
F1 Score is :  0.845480298738089
Recall Score is :  0.8987133862578702
Precision Score is :  0.7982008266472161
AUC Value  :  0.8357825278306319
Classification Report is :                precision    recall  f1-score
support

           0       0.88      0.77      0.82      3654
           1       0.80      0.90      0.85      3653

    accuracy                           0.84      7307
   macro avg       0.84      0.84      0.84      7307
weighted avg       0.84      0.84      0.84      7307

Confusion Matrix is :
 [[2824  830]
 [ 370 3283]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.863160135638581
Model Test Score is :  0.8623237990967565
F1 Score is :  0.868668407310705
Recall Score is :  0.9107582808650424
Precision Score is :  0.8302969802845022
AUC Value  :  0.8623304266941523
Classification Report is :                precision    recall  f1-score
support

           0       0.90      0.81      0.86      3654
           1       0.83      0.91      0.87      3653

    accuracy                           0.86      7307
   macro avg       0.87      0.86      0.86      7307
weighted avg       0.87      0.86      0.86      7307

Confusion Matrix is :
 [[2974  680]
```

```
 [ 326 3327]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  0.86322096011435
Model Test Score is :  0.8651977555768441
F1 Score is :  0.8709212423011401
Recall Score is :  0.9096632904462086
Precision Score is :  0.8353443941679236
AUC Value  :  0.8652038400780577
Classification Report is :                precision    recall  f1-score
support

           0       0.90      0.82      0.86      3654
           1       0.84      0.91      0.87      3653

    accuracy                           0.87      7307
   macro avg       0.87      0.87      0.86      7307
weighted avg       0.87      0.87      0.86      7307

Confusion Matrix is :
 [[2999  655]
 [ 330 3323]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  0.8354089685689522
Model Test Score is :  0.8360476255645272
F1 Score is :  0.8440104166666668
Recall Score is :  0.8872159868601149
Precision Score is :  0.8048174819965235
AUC Value  :  0.8360546272560017
Classification Report is :                precision    recall  f1-score
support

           0       0.87      0.78      0.83      3654
           1       0.80      0.89      0.84      3653

    accuracy                           0.84      7307
   macro avg       0.84      0.84      0.84      7307
weighted avg       0.84      0.84      0.84      7307

Confusion Matrix is :
 [[2868  786]
 [ 412 3241]]
```

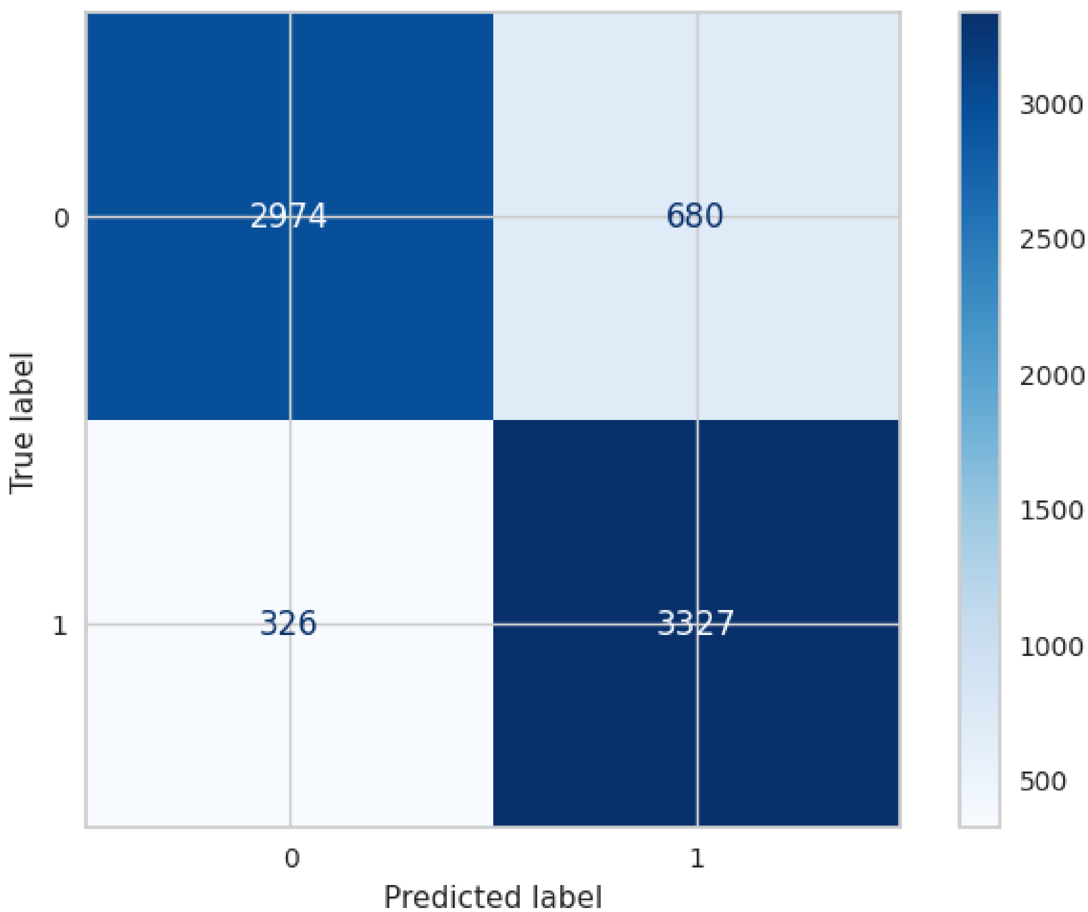

3350
304
1232
2421
0
1
True label
0
1
Predicted label
3000
2500
2000
1500
1000
500


2864
790
340
3313
True label
Predicted label
0
1
0
1
3000
2500
2000
1500
1000
500

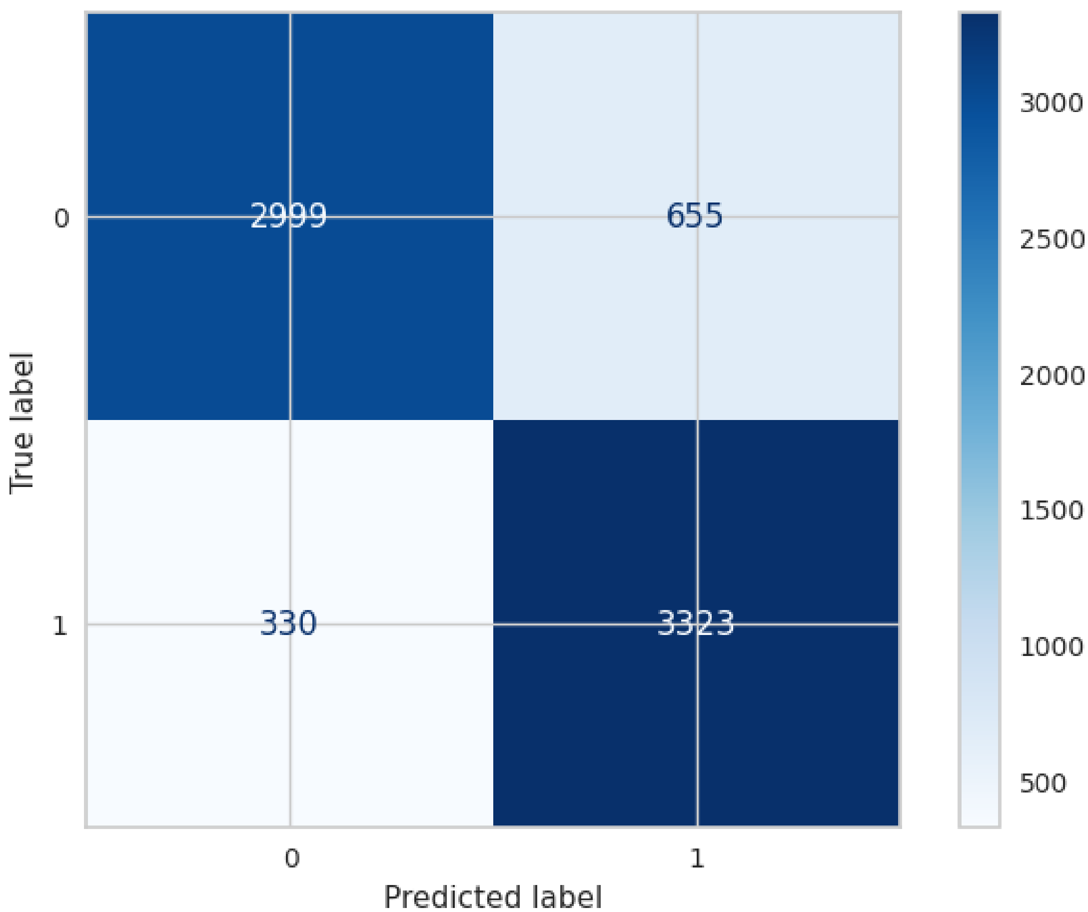

2920
734
248
3405
0
1
True label
0
1
Predicted label
3000
2500
2000
1500
1000
500

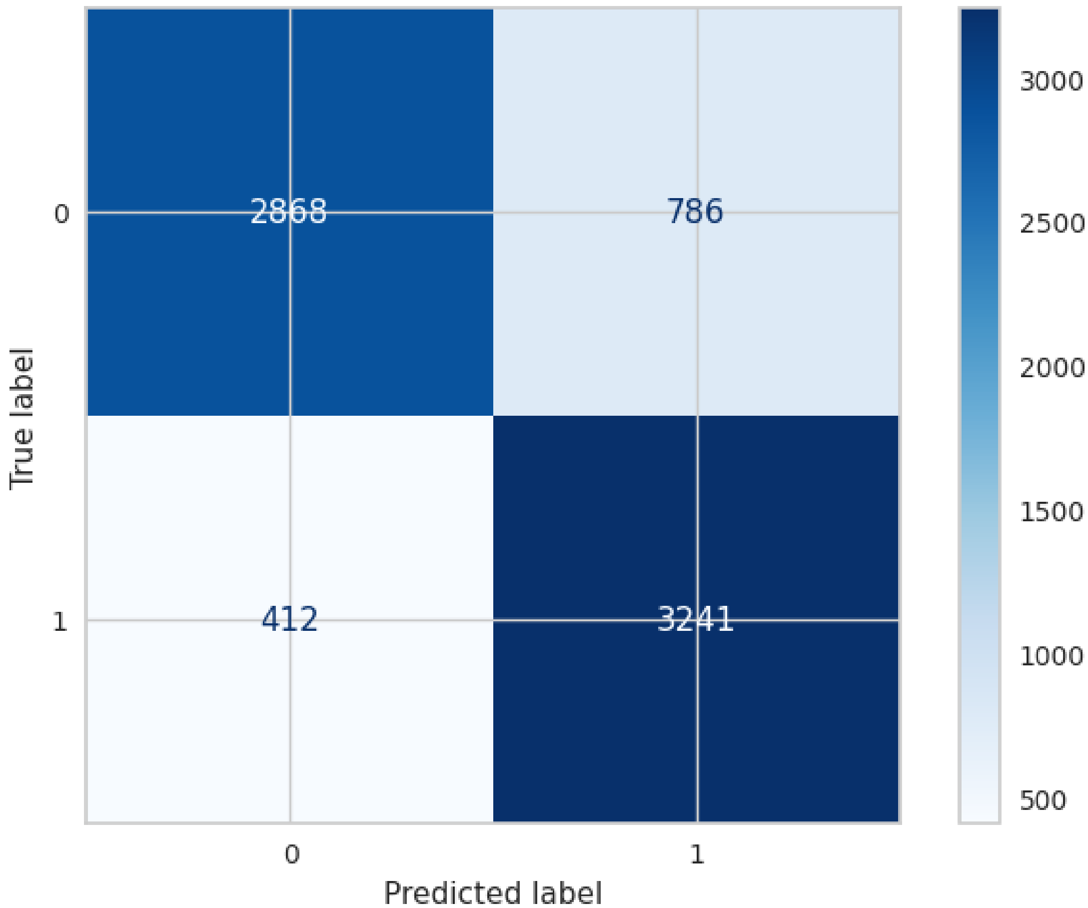


2824
830
370
3283
0
1
True label
Predicted label
3000
2500
2000
1500
1000
500



2974
680
326
3327
0
1
True label
0
1
Predicted label
3000
2500
2000
1500
1000
500


2999
655
330
3323
0
1
True label
0
1
Predicted label
3000
2500
2000
1500
1000
500



[338]:
```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['SGD Over','SGD Over With Feature','SGD Over Scaling','SGD Over␣
↪With Normalize','SGD Over With PCA'
                ,'SGD Over With PCA and Scaling',
                'SGD Over With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[338]:

Models	Train Accuracy	Test Accuracy	Test F1	\
SGD Over	0.779420	0.789791	0.759172	
SGD Over With Feature	0.841993	0.845354	0.854306	
SGD Over Scaling	0.862506	0.865608	0.873973	
SGD Over With Normalize	0.834071	0.835774	0.845480	
SGD Over With PCA	0.863160	0.862324	0.868668	
SGD Over With PCA and Scaling	0.863221	0.865198	0.870921	
SGD Over With PCA and Normalize	0.835409	0.836048	0.844010	

```
                                Test Recall  Test Precision       AUC
Models
SGD Over                           0.662743        0.888440  0.789773
SGD Over With Feature              0.906926        0.807458  0.845362
SGD Over Scaling                   0.932111        0.822662  0.865617
SGD Over With Normalize            0.898713        0.798201  0.835783
SGD Over With PCA                  0.910758        0.830297  0.862330
SGD Over With PCA and Scaling      0.909663        0.835344  0.865204
SGD Over With PCA and Normalize    0.887216        0.804817  0.836055
```

[339]:
```
models_draw(df)
```

RandomUnderSampler

[340]:
```
X_train,y_train,X_test,y_test=Split(X_classification_under,y_classification_under)
```

```
X_train shape is  (8350, 20)
X_test shape is  (928, 20)
y_train shape is  (8350,)
y_test shape is  (928,)
```

[341]:
```
cross_validation(SGDClassifier(penalty='l2'),X_train,y_train)
```

```
Train Score Value :  [0.78248503 0.76317365 0.83143713 0.84730539 0.69191617]
Mean 0.7832634730538921
Test Score Value :  [0.78083832 0.75449102 0.83592814 0.83293413 0.68682635]
Mean 0.7782035928143712
```

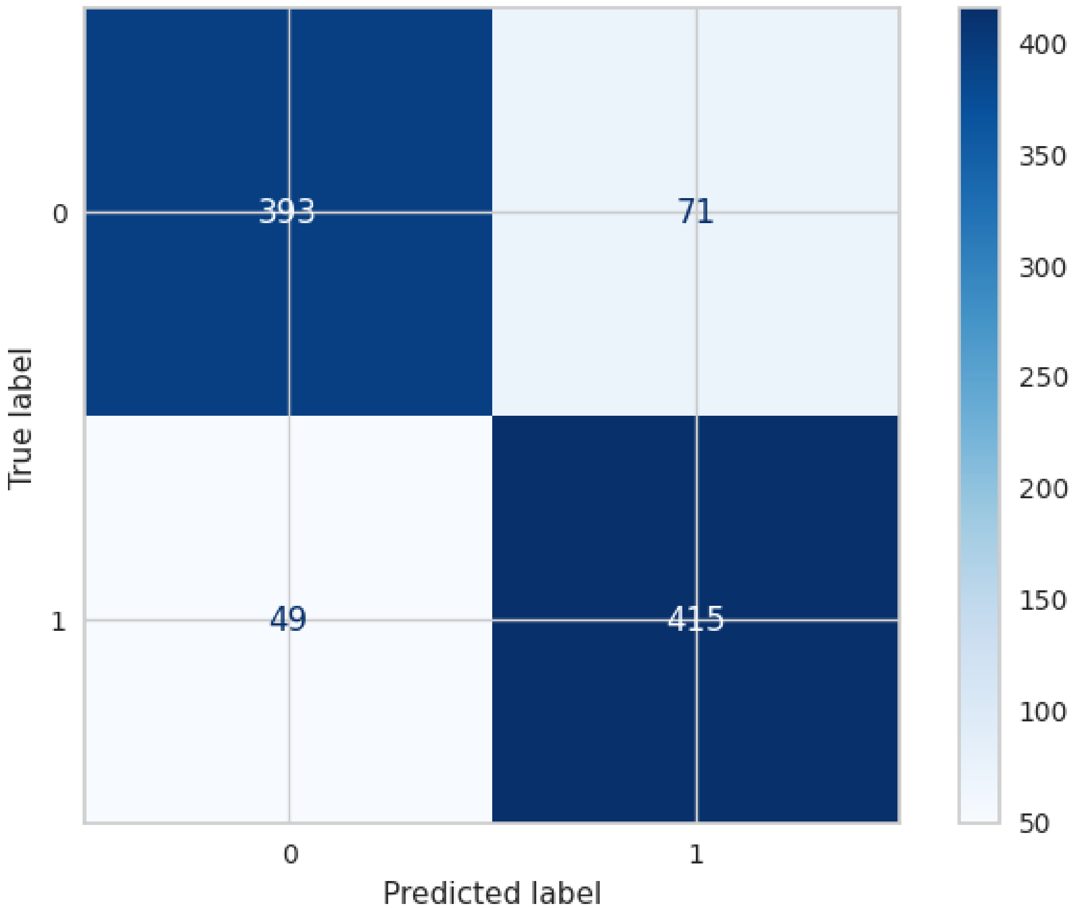
[342]:
```
Values = Models(SGDClassifier(penalty='l2'),X_train,y_train,X_test,y_test)
```

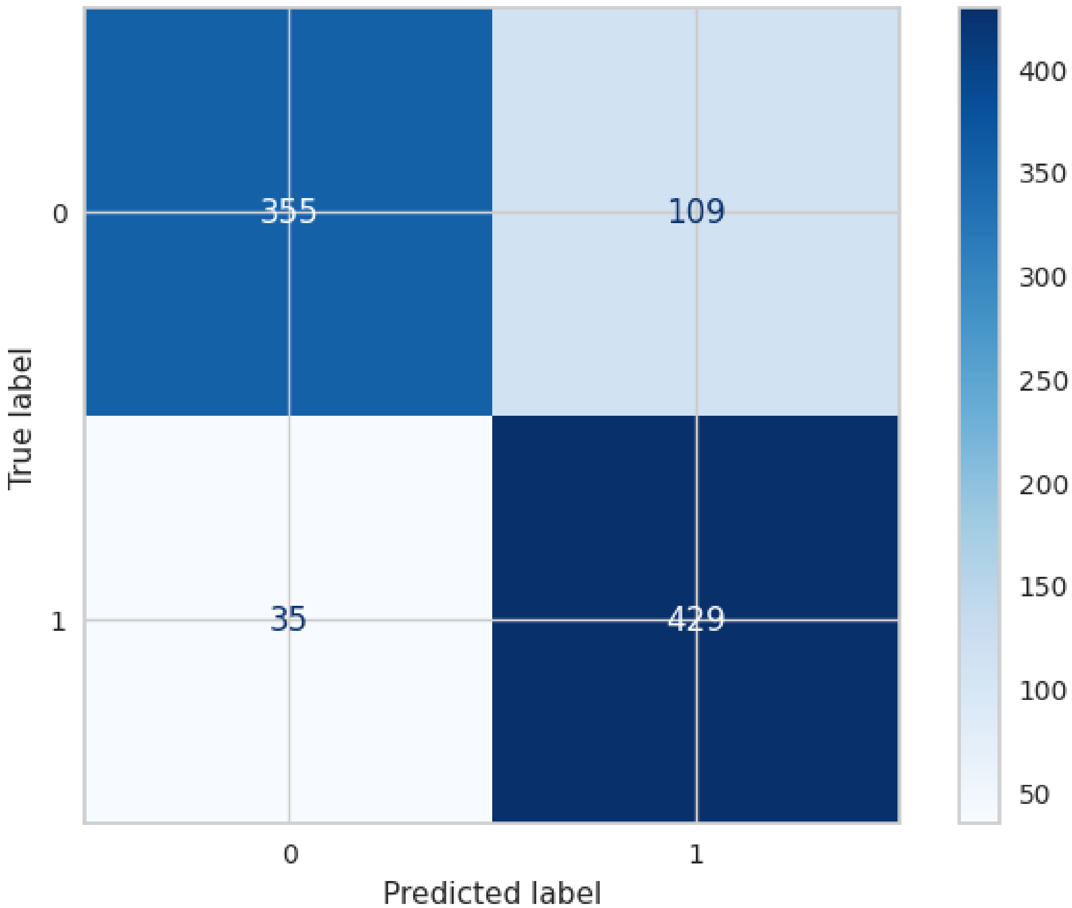
```
Apply Model With Normal Data :

Model Train Score is :  0.859880239520958
Model Test Score is :  0.8706896551724138
F1 Score is :  0.873684210526316
Recall Score is :  0.8943965517241379
Precision Score is :  0.8539094650205762
AUC Value  :  0.8706896551724139

Classification Report is :                precision    recall  f1-score
support

           0       0.89      0.85      0.87       464
           1       0.85      0.89      0.87       464

    accuracy                           0.87       928
   macro avg       0.87      0.87      0.87       928
weighted avg       0.87      0.87      0.87       928
```

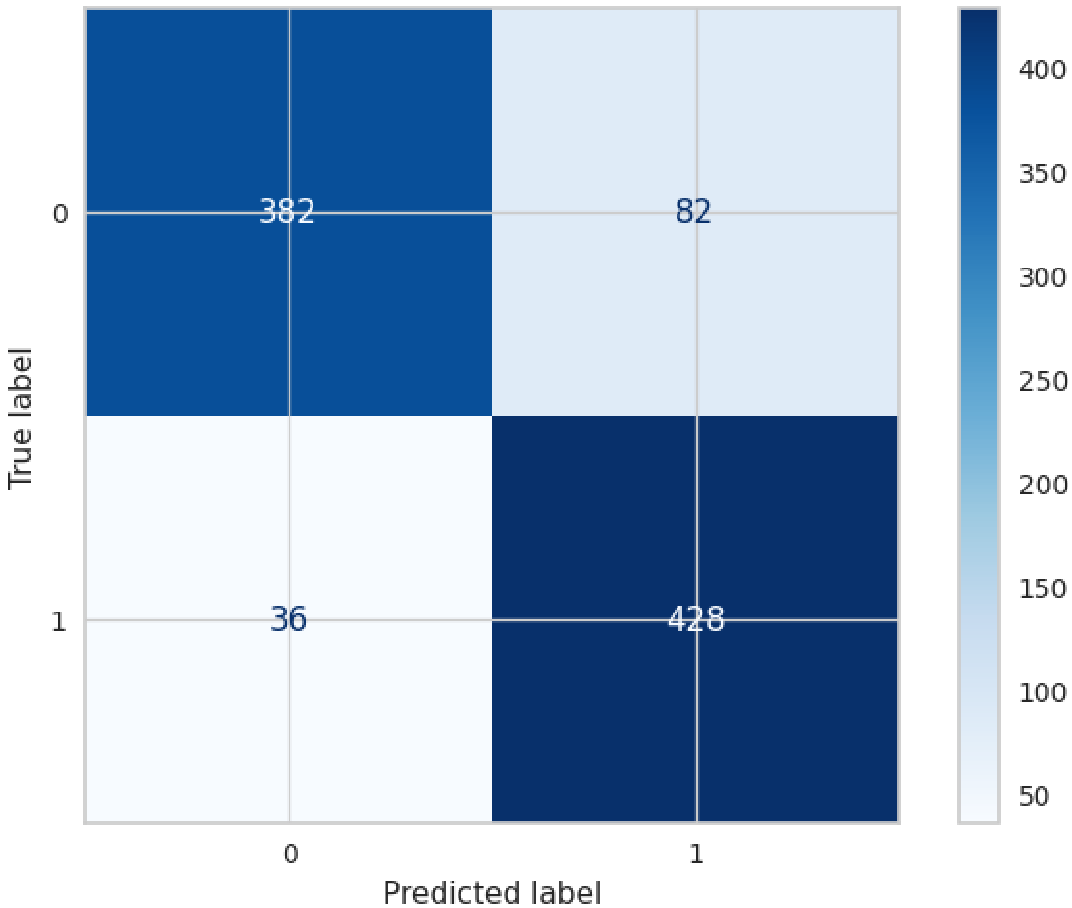
```
Confusion Matrix is :
 [[393  71]
 [ 49 415]]


 Apply Model With Feature Selection :

Model Train Score is :  0.8459880239520958
Model Test Score is :  0.8448275862068966
F1 Score is :  0.8562874251497006
Recall Score is :  0.9245689655172413
Precision Score is :  0.7973977695167286
AUC Value  :  0.8448275862068966
Classification Report is :                precision    recall  f1-score
support

           0       0.91      0.77      0.83       464
           1       0.80      0.92      0.86       464

    accuracy                           0.84       928
   macro avg       0.85      0.84      0.84       928
weighted avg       0.85      0.84      0.84       928

Confusion Matrix is :
 [[355 109]
 [ 35 429]]


 Apply Model With Normal Data With Scaling :

Model Train Score is :  0.86562874251497
Model Test Score is :  0.8728448275862069
F1 Score is :  0.8788501026694046
Recall Score is :  0.9224137931034483
Precision Score is :  0.8392156862745098
AUC Value  :  0.872844827586207
Classification Report is :                precision    recall  f1-score
support

           0       0.91      0.82      0.87       464
           1       0.84      0.92      0.88       464

    accuracy                           0.87       928
   macro avg       0.88      0.87      0.87       928
weighted avg       0.88      0.87      0.87       928

Confusion Matrix is :
```

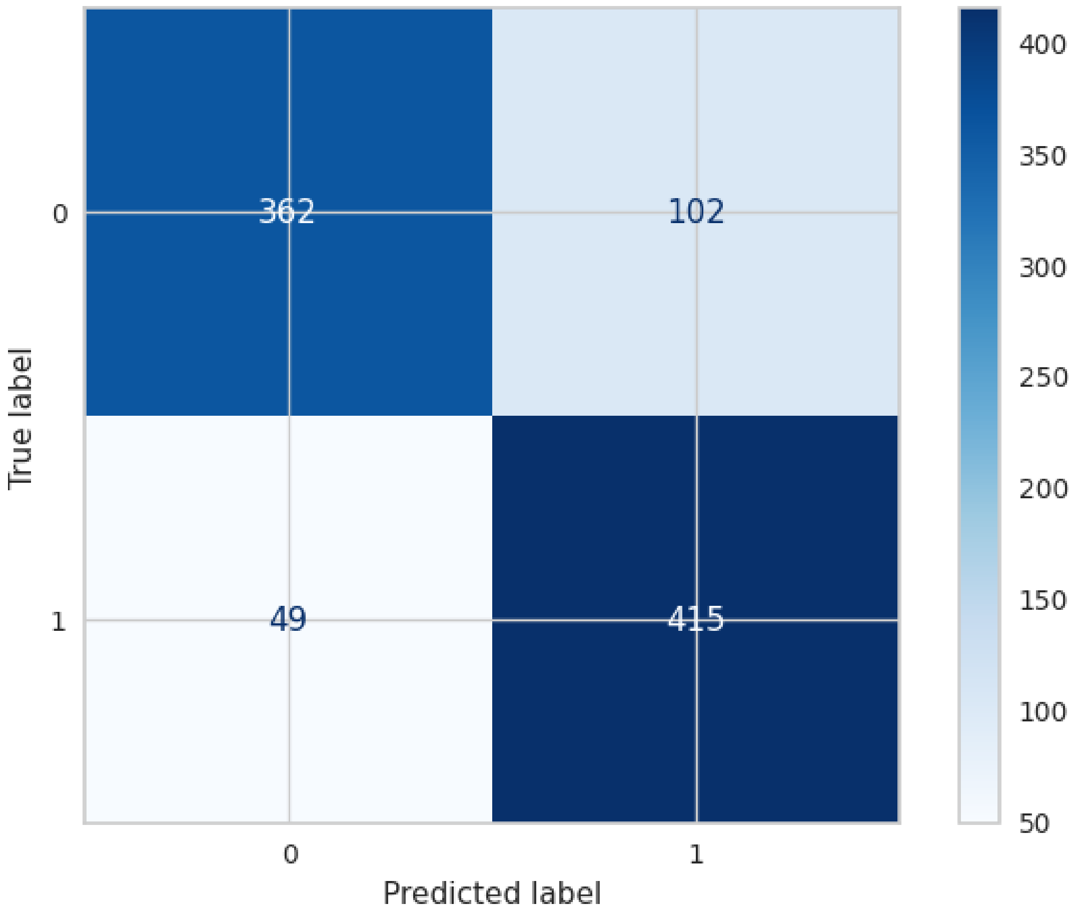
```
 [[382  82]
 [ 36 428]]


 Apply Model With Normal Data With Normalize :

Model Train Score is :  0.8338922155688623
Model Test Score is :  0.8372844827586207
F1 Score is :  0.8460754332313966
Recall Score is :  0.8943965517241379
Precision Score is :  0.8027079303675049
AUC Value  :  0.8372844827586207
Classification Report is :               precision    recall  f1-score
support

           0       0.88      0.78      0.83       464
           1       0.80      0.89      0.85       464

    accuracy                           0.84       928
   macro avg       0.84      0.84      0.84       928
weighted avg       0.84      0.84      0.84       928

Confusion Matrix is :
 [[362 102]
 [ 49 415]]


 Apply Model With Normal Data With PCA :

Model Train Score is :  0.7566467065868263
Model Test Score is :  0.7521551724137931
F1 Score is :  0.7362385321100916
Recall Score is :  0.6918103448275862
Precision Score is :  0.7867647058823529
AUC Value  :  0.7521551724137931
Classification Report is :               precision    recall  f1-score
support

           0       0.72      0.81      0.77       464
           1       0.79      0.69      0.74       464

    accuracy                           0.75       928
   macro avg       0.76      0.75      0.75       928
weighted avg       0.76      0.75      0.75       928

Confusion Matrix is :
 [[377  87]
```

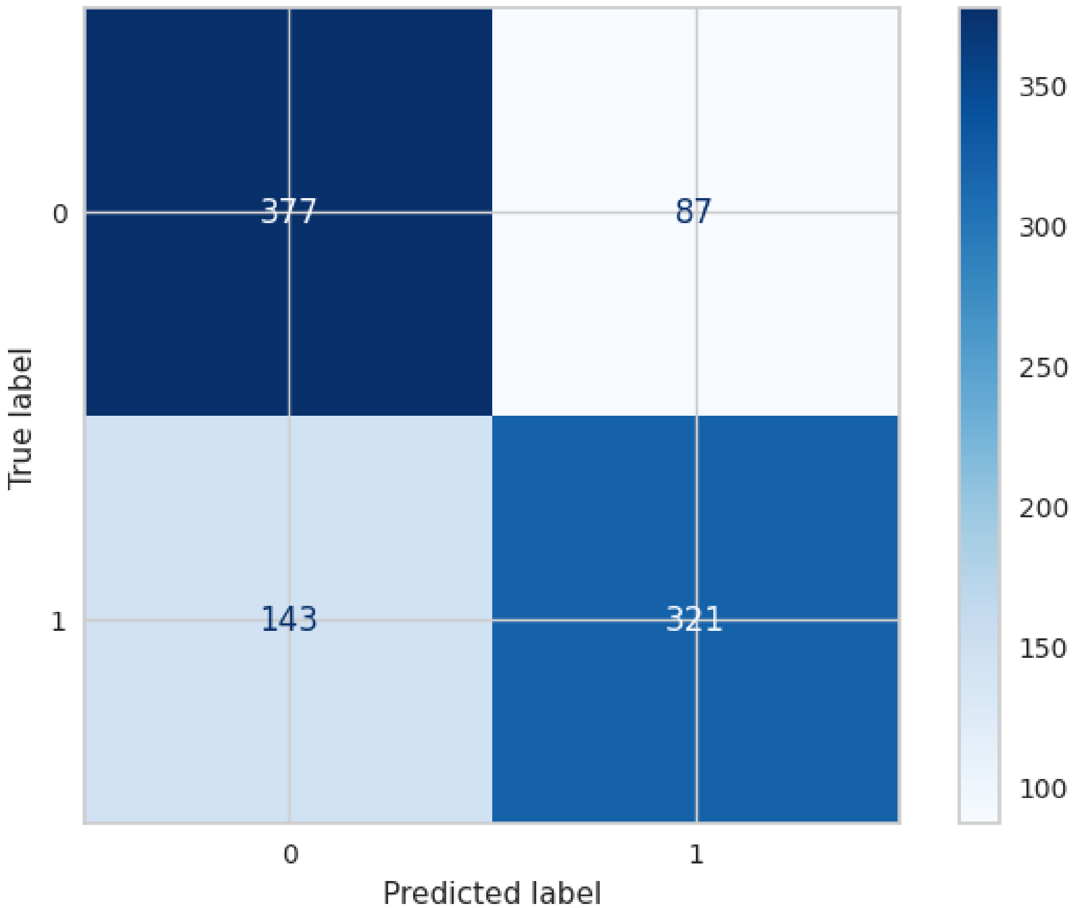
```
 [143 321]]


 Apply Model With Normal Data With PCA and Scaling :

Model Train Score is :  0.8645508982035928
Model Test Score is :  0.8803879310344828
F1 Score is :  0.8847352024922118
Recall Score is :  0.9181034482758621
Precision Score is :  0.8537074148296593
AUC Value  :  0.8803879310344828
Classification Report is :                precision    recall  f1-score
support

           0       0.91      0.84      0.88       464
           1       0.85      0.92      0.88       464

    accuracy                           0.88       928
   macro avg       0.88      0.88      0.88       928
weighted avg       0.88      0.88      0.88       928

Confusion Matrix is :
 [[391  73]
 [ 38 426]]


 Apply Model With Normal Data With PCA and Normalize :

Model Train Score is :  0.8347305389221557
Model Test Score is :  0.8318965517241379
F1 Score is :  0.8311688311688311
Recall Score is :  0.8275862068965517
Precision Score is :  0.8347826086956521
AUC Value  :  0.8318965517241379
Classification Report is :                precision    recall  f1-score
support

           0       0.83      0.84      0.83       464
           1       0.83      0.83      0.83       464

    accuracy                           0.83       928
   macro avg       0.83      0.83      0.83       928
weighted avg       0.83      0.83      0.83       928

Confusion Matrix is :
 [[388  76]
 [ 80 384]]
```

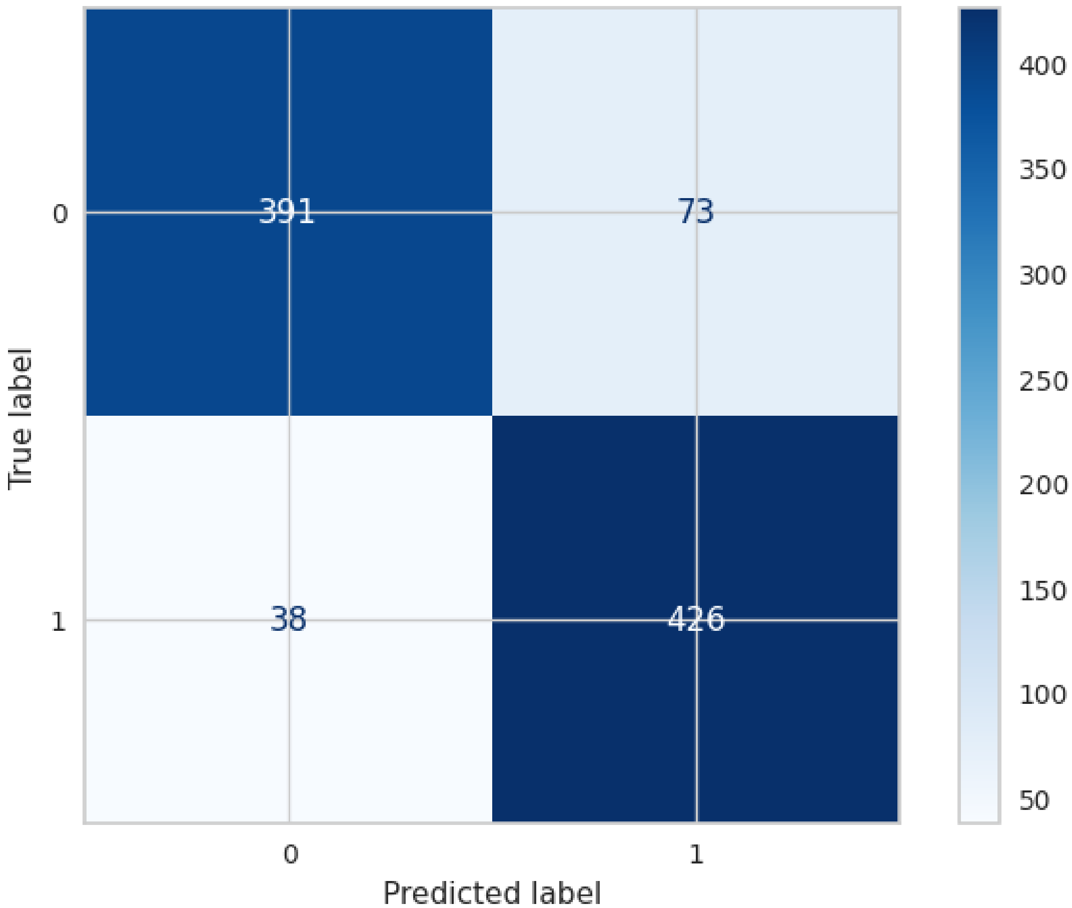

393
71
49
415
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50

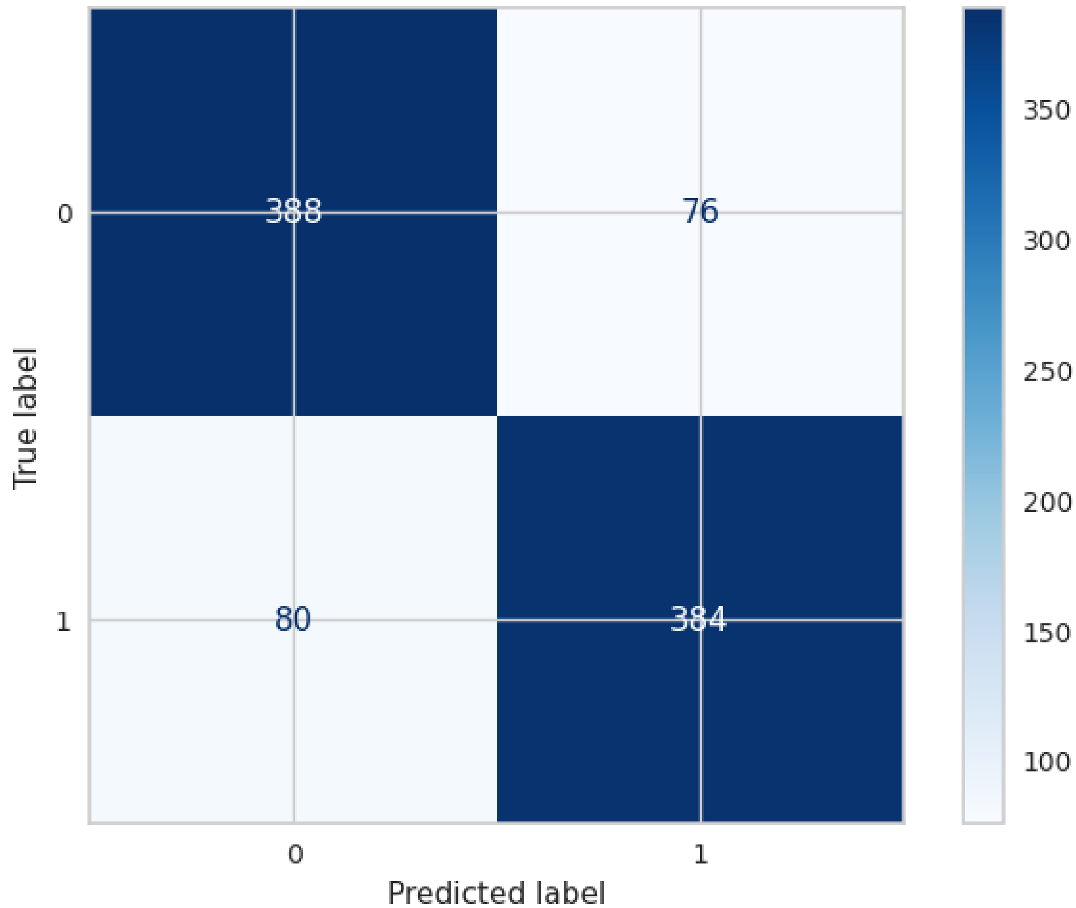

355
109
35
429
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50


0
1
True label
382
82
36
428
0
1
Predicted label
400
350
300
250
200
150
100
50


362
102
49
415
0
1
True label
0
1
Predicted label
400
350
300
250
200
150
100
50


377
87
143
321
0
1
True label
0
1
Predicted label
350
300
250
200
150
100


391
73
38
426
0
1
0
1
True label
Predicted label
400
350
300
250
200
150
100
50



[343]:
```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','Test␣
 ↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['SGD Under','SGD Under With Feature','SGD Under Scaling','SGD␣
 ↪Under With Normalize','SGD Under With PCA'
                ,'SGD Under With PCA and Scaling',
                'SGD Under With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[343]:

Models	Train Accuracy	Test Accuracy	Test F1 \
SGD Under	0.859880	0.870690	0.873684
SGD Under With Feature	0.845988	0.844828	0.856287
SGD Under Scaling	0.865629	0.872845	0.878850
SGD Under With Normalize	0.833892	0.837284	0.846075
SGD Under With PCA	0.756647	0.752155	0.736239
SGD Under With PCA and Scaling	0.864551	0.880388	0.884735
SGD Under With PCA and Normalize	0.834731	0.831897	0.831169

```
                                   Test Recall  Test Precision       AUC
Models
SGD Under                             0.894397        0.853909  0.870690
SGD Under With Feature                0.924569        0.797398  0.844828
SGD Under Scaling                     0.922414        0.839216  0.872845
SGD Under With Normalize              0.894397        0.802708  0.837284
SGD Under With PCA                    0.691810        0.786765  0.752155
SGD Under With PCA and Scaling        0.918103        0.853707  0.880388
SGD Under With PCA and Normalize      0.827586        0.834783  0.831897
```

[344]:

```
models_draw(df)
```

Regression

LinearRegression

[345]:

```
def Check_R(model,X_train,y_train,X_test,y_test):
    y_pred = model.predict(X_test)
    print('R2 Score Train :',r2_score(y_train,model.predict(X_train)))
    print('R2 Score Test :',r2_score(y_test,y_pred))
    MAEValue = mean_absolute_error(y_test, y_pred)
    print('Mean Absolute Error Value is : ', MAEValue)
    MSEValue = mean_squared_error(y_test, y_pred)
    print('Mean Squared Error Value is : ', MSEValue)
    MdSEValue = median_absolute_error(y_test, y_pred)
    print('Median Absolute Error Value is : ', MdSEValue )
    return [r2_score(y_train,model.
 ↪predict(X_train)),r2_score(y_test,y_pred),MAEValue,MSEValue,MdSEValue]
def PipeLine2(model):
    steps = [
    ('poly',PolynomialFeatures(degree=3)),
    ('scaling', MinMaxScaler()),
    ('model', model)
    ]
    return Pipeline(steps).fit(X_train,y_train)
def Models(models, X_train, y_train, X_test, y_test):
    print('Apply Model With Normal Data : \n')
    model = PipeLine(models, X_train, y_train)
    value1 = Check_R(model, X_train, y_train, X_test, y_test)
    print("\n\n Apply Model With Feature Selection :\n")
    try:
        feature = SelectFeature(model, X_train, y_train)
    except:
        feature = SelectFeature(RandomForestRegressor(max_depth=20), X_train,␣
 ↪y_train)
    X_train1 = X_train.loc[:, feature]
    X_test1 = X_test.loc[:, feature]
```

```
    model = PipeLine(models, X_train1, y_train, flage=1)
    value2 = Check_R(model, X_train1, y_train, X_test1, y_test)
    print("\n\n Apply Model With Normal Data With Scaling :\n")
    model = PipeLine(models, X_train, y_train, flage=1)
    value3 = Check_R(model, X_train, y_train, X_test, y_test)
    print("\n\n Apply Model With Normal Data With Normalize :\n")
    model = PipeLine(models, X_train, y_train, flage=2)
    value4 = Check_R(model, X_train, y_train, X_test, y_test)
    print("\n\n Apply Model With Normal Data With PCA :\n")
    model = PipeLine(models, X_train, y_train, flage=3)
    value5 = Check_R(model, X_train, y_train, X_test, y_test)
    print("\n\n Apply Model With Normal Data With PCA and Scaling :\n")
    model = PipeLine(models, X_train, y_train, flage=4)
    value6 = Check_R(model, X_train, y_train, X_test, y_test)
    print("\n\n Apply Model With Normal Data With PCA and Normalize :\n")
    model = PipeLine(models, X_train, y_train, flage=5)
    value7 = Check_R(model, X_train, y_train, X_test, y_test)
    return [value1, value2, value3, value4, value5, value6, value7]
```

[346]:
```
X_train,y_train,X_test,y_test=Split(X_regression,y_regression,classification=0)
```

```
X_train shape is  (37056, 20)
X_test shape is  (4118, 20)
y_train shape is  (37056,)
y_test shape is  (4118,)
```

[347]:
```
cross_validation(LinearRegression(),X_train,y_train)
```

```
Train Score Value :  [0.17496898 0.17965153 0.17770749 0.18636847 0.18140432]
Mean 0.18002015861264875
Test Score Value :  [0.19872269 0.18039782 0.18837726 0.15336561 0.17303233]
Mean 0.17877914309362813
```

[348]:
```
Values = Models(LinearRegression(),X_train,y_train,X_test,y_test)
```

```
Apply Model With Normal Data :

R2 Score Train : 0.1799131533956465
R2 Score Test : 0.17228606829267468
Mean Absolute Error Value is :  0.1961972520465943
Mean Squared Error Value is :  0.06186401164417418
Median Absolute Error Value is :  0.16809573941518413



 Apply Model With Feature Selection :

R2 Score Train : 0.17615493090143175
R2 Score Test : 0.1686980538859444
Mean Absolute Error Value is :  0.19667992701103926
```

```
Mean Squared Error Value is :  0.06213218275563485
Median Absolute Error Value is :  0.16843801909472544


 Apply Model With Normal Data With Scaling :

R2 Score Train : 0.1799131533956465
R2 Score Test : 0.17228606829267468
Mean Absolute Error Value is :  0.19619725204659433
Mean Squared Error Value is :  0.06186401164417418
Median Absolute Error Value is :  0.16809573941518413


 Apply Model With Normal Data With Normalize :

R2 Score Train : 0.07600027203446813
R2 Score Test : 0.07535690410276052
Mean Absolute Error Value is :  0.2096233618591825
Mean Squared Error Value is :  0.06910857611554426
Median Absolute Error Value is :  0.18087912412052315


 Apply Model With Normal Data With PCA :

R2 Score Train : 0.1799131533956465
R2 Score Test : 0.17228606829267468
Mean Absolute Error Value is :  0.19619725204659433
Mean Squared Error Value is :  0.06186401164417419
Median Absolute Error Value is :  0.16809573941518416


 Apply Model With Normal Data With PCA and Scaling :

R2 Score Train : 0.1799131533956465
R2 Score Test : 0.17228606829267468
Mean Absolute Error Value is :  0.19619725204659433
Mean Squared Error Value is :  0.06186401164417418
Median Absolute Error Value is :  0.168095739415184


 Apply Model With Normal Data With PCA and Normalize :

R2 Score Train : 0.07600027203446813
R2 Score Test : 0.07535690410276052
Mean Absolute Error Value is :  0.20962336185918254
Mean Squared Error Value is :  0.06910857611554426
Median Absolute Error Value is :  0.18087912412052315
```

[349]:
```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test␣
↪Accuracy','MAE','MSE','MdSE'])
df['Models'] = ['Linear','Linear With Feature','Linear Scaling','Linear With␣
↪Normalize','Linear With PCA'
                ,'Linear With PCA and Scaling',
                'Linear With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[349]:
```
                               Train Accuracy  Test Accuracy       MAE  \
Models
Linear                               0.179913       0.172286  0.196197
Linear With Feature                  0.176155       0.168698  0.196680
Linear Scaling                       0.179913       0.172286  0.196197
Linear With Normalize                0.076000       0.075357  0.209623
Linear With PCA                      0.179913       0.172286  0.196197
Linear With PCA and Scaling          0.179913       0.172286  0.196197
Linear With PCA and Normalize        0.076000       0.075357  0.209623

                                    MSE      MdSE
Models
Linear                         0.061864  0.168096
Linear With Feature            0.062132  0.168438
Linear Scaling                 0.061864  0.168096
Linear With Normalize          0.069109  0.180879
Linear With PCA                0.061864  0.168096
Linear With PCA and Scaling    0.061864  0.168096
Linear With PCA and Normalize  0.069109  0.180879
```

[350]:
```
models_draw(df)
```

```
RandomForestRegressor
```

[351]:
```
Search(RandomForestRegressor(max_depth=20),{'max_depth':
↪[20,25,30,35,40]},X_train,y_train)
```

[351]:
```
RandomForestRegressor(max_depth=20)
```

[352]:
```
cross_validation(RandomForestRegressor(max_depth=20),X_train,y_train)
```

```
Train Score Value :  [0.74385447 0.73562174 0.73484362 0.7424116  0.73988351]
Mean 0.7393229876878407
Test Score Value :  [0.22238412 0.20073913 0.1963973  0.17856974 0.20199656]
Mean 0.20001736830944444
```

[353]:
```
Values =␣
↪Models(RandomForestRegressor(max_depth=20),X_train,y_train,X_test,y_test)
```

```
Apply Model With Normal Data :
```

```
R2 Score Train : 0.7165890702571098
R2 Score Test : 0.17018661573812754
Mean Absolute Error Value is :  0.1936706765944035
Mean Squared Error Value is :  0.06202092643357853
Median Absolute Error Value is :  0.1584039066677834


 Apply Model With Feature Selection :

R2 Score Train : 0.6932570983662554
R2 Score Test : 0.12648478292557563
Mean Absolute Error Value is :  0.19742999926651494
Mean Squared Error Value is :  0.06528723691890624
Median Absolute Error Value is :  0.1597497728028882


 Apply Model With Normal Data With Scaling :

R2 Score Train : 0.7195803856306913
R2 Score Test : 0.16869874930823747
Mean Absolute Error Value is :  0.1937758235941574
Mean Squared Error Value is :  0.062132130779207734
Median Absolute Error Value is :  0.1586568818167362


 Apply Model With Normal Data With Normalize :

R2 Score Train : 0.6442268530132812
R2 Score Test : 0.1892121434868358
Mean Absolute Error Value is :  0.19006213417931722
Mean Squared Error Value is :  0.060598943034368545
Median Absolute Error Value is :  0.15418867749122317


 Apply Model With Normal Data With PCA :

R2 Score Train : 0.6161050998156575
R2 Score Test : 0.20163035699086174
Mean Absolute Error Value is :  0.18901098939638958
Mean Squared Error Value is :  0.059670795669217576
Median Absolute Error Value is :  0.15621819258772557


 Apply Model With Normal Data With PCA and Scaling :

R2 Score Train : 0.5485881308093272
R2 Score Test : 0.20884627762250152
```

```
Mean Absolute Error Value is :  0.1881573994135784
Mean Squared Error Value is :  0.059131471899399664
Median Absolute Error Value is :  0.1548600122830105


 Apply Model With Normal Data With PCA and Normalize :

R2 Score Train : 0.5952830886884306
R2 Score Test : 0.19930319168869937
Mean Absolute Error Value is :  0.19064306754583685
Mean Squared Error Value is :  0.05984472989435974
Median Absolute Error Value is :  0.15871482923974903
```

[354]:
```python
df = pd.DataFrame(Values,columns=['Train Accuracy','Test␣
↪Accuracy','MAE','MSE','MdSE'])
df['Models'] = ['Random','Random With Feature','Random Scaling','Random With␣
↪Normalize','Random With PCA'
                ,'Random With PCA and Scaling',
                'Random With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[354]:
```
                               Train Accuracy  Test Accuracy       MAE  \
Models
Random                               0.716589       0.170187  0.193671
Random With Feature                  0.693257       0.126485  0.197430
Random Scaling                       0.719580       0.168699  0.193776
Random With Normalize                0.644227       0.189212  0.190062
Random With PCA                      0.616105       0.201630  0.189011
Random With PCA and Scaling          0.548588       0.208846  0.188157
Random With PCA and Normalize        0.595283       0.199303  0.190643

                                    MSE      MdSE
Models
Random                         0.062021  0.158404
Random With Feature            0.065287  0.159750
Random Scaling                 0.062132  0.158657
Random With Normalize          0.060599  0.154189
Random With PCA                0.059671  0.156218
Random With PCA and Scaling    0.059131  0.154860
Random With PCA and Normalize  0.059845  0.158715
```

[355]:
```python
models_draw(df)
```

Ridge

[356]:
```python
Search(Ridge(alpha=1.0),{'alpha':[1,2,.5,5,10,15,40]},X_train,y_train)
```

[356]: Ridge(alpha=0.5)

[357]:
```
cross_validation(Ridge(alpha=.5),X_train,y_train)
```

```
Train Score Value :  [0.17496744 0.17965065 0.17770661 0.1863676  0.18140344]
Mean 0.18001914741418806
Test Score Value :  [0.19872663 0.18036918 0.18837539 0.15336924 0.17303511]
Mean 0.17877511144388875
```

[358]:
```
Values = Models(Ridge(alpha=.5),X_train,y_train,X_test,y_test)
```

```
Apply Model With Normal Data :

R2 Score Train : 0.17991244273646645
R2 Score Test : 0.1722874119661726
Mean Absolute Error Value is :  0.1961980560308896
Mean Squared Error Value is :  0.06186391121692543
Median Absolute Error Value is :  0.16809458644419703


 Apply Model With Feature Selection :

R2 Score Train : 0.17615492417052525
R2 Score Test : 0.16869974303902646
Mean Absolute Error Value is :  0.19668147742933956
Mean Squared Error Value is :  0.06213205650696132
Median Absolute Error Value is :  0.16843067734002226


 Apply Model With Normal Data With Scaling :

R2 Score Train : 0.179912390913616
R2 Score Test : 0.1722951668216467
Mean Absolute Error Value is :  0.1961969724879249
Mean Squared Error Value is :  0.06186333161272768
Median Absolute Error Value is :  0.16809538243749322


 Apply Model With Normal Data With Normalize :

R2 Score Train : 0.040785827125842666
R2 Score Test : 0.042500044556645562
Mean Absolute Error Value is :  0.21354464332853368
Mean Squared Error Value is :  0.07156432448910473
Median Absolute Error Value is :  0.18612895660766393


 Apply Model With Normal Data With PCA :
```

```
R2 Score Train : 0.17991244273646645
R2 Score Test : 0.1722874119661726
Mean Absolute Error Value is :  0.19619805603088963
Mean Squared Error Value is :  0.06186391121692542
Median Absolute Error Value is :  0.16809458644419706


 Apply Model With Normal Data With PCA and Scaling :

R2 Score Train : 0.179912390913616
R2 Score Test : 0.17229516682164658
Mean Absolute Error Value is :  0.19619697248792495
Mean Squared Error Value is :  0.061863331612727696
Median Absolute Error Value is :  0.1680953824374919


 Apply Model With Normal Data With PCA and Normalize :

R2 Score Train : 0.040785827125842666
R2 Score Test : 0.04250004455645562
Mean Absolute Error Value is :  0.21354464332853365
Mean Squared Error Value is :  0.07156432448910473
Median Absolute Error Value is :  0.186128956607664
```

[359]:

```python
df = pd.DataFrame(Values,columns=['Train Accuracy','Test Accuracy','MAE','MSE','MdSE'])
df['Models'] = ['Ridge','Ridge With Feature','Ridge Scaling','Ridge With Normalize','Ridge With PCA'
                ,'Ridge With PCA and Scaling',
                'Ridge With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[359]:

```
                              Train Accuracy  Test Accuracy       MAE  \
Models
Ridge                               0.179912       0.172287  0.196198
Ridge With Feature                  0.176155       0.168700  0.196681
Ridge Scaling                       0.179912       0.172295  0.196197
Ridge With Normalize                0.040786       0.042500  0.213545
Ridge With PCA                      0.179912       0.172287  0.196198
Ridge With PCA and Scaling          0.179912       0.172295  0.196197
Ridge With PCA and Normalize        0.040786       0.042500  0.213545

                                   MSE      MdSE
Models
Ridge                         0.061864  0.168095
```

```
Ridge With Feature           0.062132  0.168431
Ridge Scaling                0.061863  0.168095
Ridge With Normalize         0.071564  0.186129
Ridge With PCA               0.061864  0.168095
Ridge With PCA and Scaling   0.061863  0.168095
Ridge With PCA and Normalize 0.071564  0.186129
```

[360]:
```
models_draw(df)
```

```
DecisionTreeRegressor
```

[361]:
```
Search(DecisionTreeRegressor(max_depth=20),{'max_depth':
↪[20,25,30,35,40]},X_train,y_train)
```

[361]:
```
DecisionTreeRegressor(max_depth=20)
```

[362]:
```
cross_validation(DecisionTreeRegressor(max_depth=20),X_train,y_train)
```

```
Train Score Value :  [0.70625674 0.70679501 0.71844419 0.75654502 0.66250076]
Mean 0.7101083429582381
Test Score Value :  [-0.21417602 -0.27588778 -0.24344511 -0.31790342
-0.22034145]        Mean -0.25435075537412927
```

[363]:
```
Values =␣
↪Models(DecisionTreeRegressor(max_depth=20),X_train,y_train,X_test,y_test)
```

```
Apply Model With Normal Data :

R2 Score Train : 0.7034967259817382
R2 Score Test : -0.36671689827710496
Mean Absolute Error Value is :  0.2360284931274082
Mean Squared Error Value is :  0.10214953122137496
Median Absolute Error Value is :  0.16912335143522111


 Apply Model With Feature Selection :

R2 Score Train : 0.6402090241432501
R2 Score Test : -0.2108979085324858
Mean Absolute Error Value is :  0.22423061974948807
Mean Squared Error Value is :  0.0905034933492553
Median Absolute Error Value is :  0.16665025813895804


 Apply Model With Normal Data With Scaling :

R2 Score Train : 0.7034971183049442
R2 Score Test : -0.3662310630826626
Mean Absolute Error Value is :  0.2362778560360926
```

```
Mean Squared Error Value is :  0.10211321950427707
Median Absolute Error Value is :  0.17145073700543056


 Apply Model With Normal Data With Normalize :

R2 Score Train : 0.6399606848491695
R2 Score Test : -0.146686876560284
Mean Absolute Error Value is :  0.21506593643894678
Mean Squared Error Value is :  0.08570430865821238
Median Absolute Error Value is :  0.15671062839410393


 Apply Model With Normal Data With PCA :

R2 Score Train : 0.572426575253734
R2 Score Test : -0.1260461869978382
Mean Absolute Error Value is :  0.2119297909374955
Mean Squared Error Value is :  0.08416160675297679
Median Absolute Error Value is :  0.1534690638158505


 Apply Model With Normal Data With PCA and Scaling :

R2 Score Train : 0.523811269809662
R2 Score Test : -0.06528195190350794
Mean Absolute Error Value is :  0.20801975199738776
Mean Squared Error Value is :  0.07962003846057045
Median Absolute Error Value is :  0.15499852424108748


 Apply Model With Normal Data With PCA and Normalize :

R2 Score Train : 0.563824358241946
R2 Score Test : -0.16690360484561628
Mean Absolute Error Value is :  0.21881587108931683
Mean Squared Error Value is :  0.08721532335319415
Median Absolute Error Value is :  0.16291698991466252
```

[364]:

```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test␣
↪Accuracy','MAE','MSE','MdSE'])
df['Models'] = ['Decision','Decision With Feature','Decision Scaling','Decision␣
↪With Normalize','Decision With PCA'
                ,'Decision With PCA and Scaling',
                'Decision With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[364]:
```
                                  Train Accuracy  Test Accuracy       MAE  \
Models
Decision                                0.703497      -0.366717  0.236028
Decision With Feature                   0.640209      -0.210898  0.224231
Decision Scaling                        0.703497      -0.366231  0.236278
Decision With Normalize                 0.639961      -0.146687  0.215066
Decision With PCA                       0.572427      -0.126046  0.211930
Decision With PCA and Scaling           0.523811      -0.065282  0.208020
Decision With PCA and Normalize         0.563824      -0.166904  0.218816

                                       MSE      MdSE
Models
Decision                          0.102150  0.169123
Decision With Feature             0.090503  0.166650
Decision Scaling                  0.102113  0.171451
Decision With Normalize           0.085704  0.156711
Decision With PCA                 0.084162  0.153469
Decision With PCA and Scaling     0.079620  0.154999
Decision With PCA and Normalize   0.087215  0.162917
```

[365]:
```
models_draw(df)
```

KNeighborsRegressor

[366]:
```
Search(KNeighborsRegressor(n_neighbors = 5),{'n_neighbors':
 ↪[3,5,7,9,11]},X_train,y_train)
```

[366]:
```
KNeighborsRegressor(n_neighbors=11)
```

[367]:
```
cross_validation(KNeighborsRegressor(n_neighbors = 11),X_train,y_train)
```

```
Train Score Value :  [0.15036278 0.14235686 0.14552232 0.15015352 0.14530772]
Mean 0.14674064002252052
Test Score Value :  [-0.03321914 -0.01232461 -0.02066934 -0.02904956
-0.03249353]        Mean -0.02555123549454876
```

[368]:
```
Values = Models(KNeighborsRegressor(n_neighbors =␣
 ↪11),X_train,y_train,X_test,y_test)
```

```
Apply Model With Normal Data :

R2 Score Train : 0.151576976882271
R2 Score Test : -0.04295235954272725
Mean Absolute Error Value is :  0.2196140746901452
Mean Squared Error Value is :  0.07795110658821741
Median Absolute Error Value is :  0.17857394738698074


 Apply Model With Feature Selection :
```

```
R2 Score Train : 0.29509380492304615
R2 Score Test : 0.11695556130499152
Mean Absolute Error Value is :  0.19917175668440434
Mean Squared Error Value is :  0.0659994586838338
Median Absolute Error Value is :  0.1629875167501234


 Apply Model With Normal Data With Scaling :

R2 Score Train : 0.31052763722849397
R2 Score Test : 0.15694460327245263
Mean Absolute Error Value is :  0.19560891838722974
Mean Squared Error Value is :  0.06301064520233121
Median Absolute Error Value is :  0.16200014105367094


 Apply Model With Normal Data With Normalize :

R2 Score Train : 0.14975918695460533
R2 Score Test : -0.027713133850314486
Mean Absolute Error Value is :  0.21768707191726566
Mean Squared Error Value is :  0.07681211448047434
Median Absolute Error Value is :  0.1797376401720855


 Apply Model With Normal Data With PCA :

R2 Score Train : 0.1508920082287054
R2 Score Test : -0.04054927937603181
Mean Absolute Error Value is :  0.2197171760863265
Mean Squared Error Value is :  0.07777149842443105
Median Absolute Error Value is :  0.17875026447563297


 Apply Model With Normal Data With PCA and Scaling :

R2 Score Train : 0.31062715045430844
R2 Score Test : 0.1563959359608823
Mean Absolute Error Value is :  0.19568107223807887
Mean Squared Error Value is :  0.06305165304290453
Median Absolute Error Value is :  0.1621059313068623


 Apply Model With Normal Data With PCA and Normalize :

R2 Score Train : 0.1497937231272204
R2 Score Test : -0.028090993636208816
```

```
Mean Absolute Error Value is :  0.21774175676078844
Mean Squared Error Value is :  0.07684035602782419
Median Absolute Error Value is :  0.18009027434938993
```

[369]:
```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test␣
↪Accuracy','MAE','MSE','MdSE'])
df['Models'] = ['KNN','KNN With Feature','KNN Scaling','KNN With␣
↪Normalize','KNN With PCA'
                ,'KNN With PCA and Scaling',
                'KNN With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[369]:
```
                            Train Accuracy  Test Accuracy       MAE       MSE  \
Models
KNN                               0.151577      -0.042952  0.219614  0.077951
KNN With Feature                  0.295094       0.116956  0.199172  0.065999
KNN Scaling                       0.310528       0.156945  0.195609  0.063011
KNN With Normalize                0.149759      -0.027713  0.217687  0.076812
KNN With PCA                      0.150892      -0.040549  0.219717  0.077771
KNN With PCA and Scaling          0.310627       0.156396  0.195681  0.063052
KNN With PCA and Normalize        0.149794      -0.028091  0.217742  0.076840

                                MdSE
Models
KNN                         0.178574
KNN With Feature            0.162988
KNN Scaling                 0.162000
KNN With Normalize          0.179738
KNN With PCA                0.178750
KNN With PCA and Scaling    0.162106
KNN With PCA and Normalize  0.180090
```

[370]:
```
models_draw(df)
```

SVR

[371]:
```
Search(SVR(C = 1.0),{'C':[1,.5,2,3,5,10]},X_train,y_train)
```

[371]:
```
SVR(C=10)
```

[372]:
```
cross_validation(SVR(C = 10),X_train,y_train)
```

```
Train Score Value :  [0.13397628 0.13917285 0.13878077 0.14855674 0.14015961]
Mean 0.1401292522035146
Test Score Value :  [0.14612083 0.13909309 0.14292087 0.12060939 0.14106212]
Mean 0.13796125964321812
```

[373]:
```
Values = Models(SVR(C = 10),X_train,y_train,X_test,y_test)
```

```
Apply Model With Normal Data :

R2 Score Train : 0.15259411734317463
R2 Score Test : 0.14929750423378996
Mean Absolute Error Value is :  0.1871221322467224
Mean Squared Error Value is :  0.06358219559655516
Median Absolute Error Value is :  0.14017116730696383


 Apply Model With Feature Selection :

R2 Score Train : 0.201141199937231
R2 Score Test : 0.16572061948467964
Mean Absolute Error Value is :  0.18188834951725455
Mean Squared Error Value is :  0.06235471862148607
Median Absolute Error Value is :  0.13168278289255814


 Apply Model With Normal Data With Scaling :

R2 Score Train : 0.2930056884734801
R2 Score Test : 0.17098762253983169
Mean Absolute Error Value is :  0.18155693735828612
Mean Squared Error Value is :  0.061961058534526166
Median Absolute Error Value is :  0.13470236083799295


 Apply Model With Normal Data With Normalize :

R2 Score Train : 0.09120751932123217
R2 Score Test : 0.09169581781502456
Mean Absolute Error Value is :  0.1930102426595158
Mean Squared Error Value is :  0.06788739243187265
Median Absolute Error Value is :  0.14308272327110252


 Apply Model With Normal Data With PCA :

R2 Score Train : 0.2057614856034412
R2 Score Test : 0.19027129627658657
Mean Absolute Error Value is :  0.17869072138648756
Mean Squared Error Value is :  0.06051978109446625
Median Absolute Error Value is :  0.13049148173228795


 Apply Model With Normal Data With PCA and Scaling :

R2 Score Train : 0.3530544541866525
```

```
R2 Score Test : 0.12914175346149526
Mean Absolute Error Value is :  0.18718274561788223
Mean Squared Error Value is :  0.06508865278267778
Median Absolute Error Value is :  0.1383467359189316


 Apply Model With Normal Data With PCA and Normalize :

R2 Score Train : 0.2252028794203068
R2 Score Test : 0.1394880601456845
Mean Absolute Error Value is :  0.18437929052981045
Mean Squared Error Value is :  0.0643153614163423
Median Absolute Error Value is :  0.13418502758890166
```

[374]:
```python
df = pd.DataFrame(Values,columns=['Train Accuracy','Test␣
 ↪Accuracy','MAE','MSE','MdSE'])
df['Models'] = ['SVR','SVR With Feature','SVR Scaling','SVR With␣
 ↪Normalize','SVR With PCA'
                ,'SVR With PCA and Scaling',
                'SVR With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[374]:
```
                            Train Accuracy  Test Accuracy       MAE       MSE  \
Models
SVR                               0.152594       0.149298  0.187122  0.063582
SVR With Feature                  0.201141       0.165721  0.181888  0.062355
SVR Scaling                       0.293006       0.170988  0.181557  0.061961
SVR With Normalize                0.091208       0.091696  0.193010  0.067887
SVR With PCA                      0.205761       0.190271  0.178691  0.060520
SVR With PCA and Scaling          0.353054       0.129142  0.187183  0.065089
SVR With PCA and Normalize        0.225203       0.139488  0.184379  0.064315

                                MdSE
Models
SVR                         0.140171
SVR With Feature            0.131683
SVR Scaling                 0.134702
SVR With Normalize          0.143083
SVR With PCA                0.130491
SVR With PCA and Scaling    0.138347
SVR With PCA and Normalize  0.134185
```

[375]:
```python
models_draw(df)
```

SGDRegressor

[376]:
```python
Search(SGDRegressor(alpha=0.1),{'alpha':[.1,1,.5,2,3,5,10]},X_train,y_train)
```

[376]: SGDRegressor(alpha=10)

[377]:
```
cross_validation(SGDRegressor(alpha=.5),X_train,y_train)
```

```
Train Score Value :  [-2.00977472e+27 -1.07579961e+27 -1.24805106e+26
-1.80037691e+27
 -8.79456723e+26]        Mean -1.1780426143526495e+27
Test Score Value :  [-1.97530839e+27 -1.03990760e+27 -1.25957414e+26
-1.80248108e+27
 -9.16790554e+26]        Mean -1.1720890068111142e+27
```

[378]:
```
Values = Models(SGDRegressor(alpha=.5),X_train,y_train,X_test,y_test)
```

```
Apply Model With Normal Data :

R2 Score Train : -5.9887800791151424e+26
R2 Score Test : -5.992822159328728e+26
Mean Absolute Error Value is :  6414599043826.015
Mean Squared Error Value is :  4.479083964207926e+25
Median Absolute Error Value is :  7362597988538.242


 Apply Model With Feature Selection :

R2 Score Train : 0.050573834820634 82
R2 Score Test : 0.0503666948264576 14
Mean Absolute Error Value is :  0.21245826849427316
Mean Squared Error Value is :  0.07097636465749936
Median Absolute Error Value is :  0.1826241373347367


 Apply Model With Normal Data With Scaling :

R2 Score Train : 0.05146061835190807
R2 Score Test : 0.05148195233980457
Mean Absolute Error Value is :  0.21154457546318536
Mean Squared Error Value is :  0.07089300940497917
Median Absolute Error Value is :  0.180471079816994


 Apply Model With Normal Data With Normalize :

R2 Score Train : -0.0007011205746128013
R2 Score Test : -7.75366388405807e-06
Mean Absolute Error Value is :  0.21653926442906063
Mean Squared Error Value is :  0.07474139185904376
Median Absolute Error Value is :  0.18367079022795793
```

```
 Apply Model With Normal Data With PCA :

R2 Score Train : -6.237331587836604e+21
R2 Score Test : -6.642599754373229e+21
Mean Absolute Error Value is :  17404847576.16833
Mean Squared Error Value is :  4.964733017172884e+20
Median Absolute Error Value is :  14174237548.082249


 Apply Model With Normal Data With PCA and Scaling :

R2 Score Train : 0.05440495465863948
R2 Score Test : 0.05390186323346979
Mean Absolute Error Value is :  0.21372826565343855
Mean Squared Error Value is :  0.07071214329898676
Median Absolute Error Value is :  0.18650587388274523


 Apply Model With Normal Data With PCA and Normalize :

R2 Score Train : -0.0001351814996795042
R2 Score Test : -0.0006241363085452978
Mean Absolute Error Value is :  0.22014329512025335
Mean Squared Error Value is :  0.07478746079862042
Median Absolute Error Value is :  0.19156657443523614
```

[379]:
```python
df = pd.DataFrame(Values,columns=['Train Accuracy','Test␣
↪Accuracy','MAE','MSE','MdSE'])
df['Models'] = ['SGD','SGD With Feature','SGD Scaling','SGD With␣
↪Normalize','SGD With PCA'
                ,'SGD With PCA and Scaling',
                'SGD With PCA and Normalize']
df.set_index('Models', inplace=True)
df
```

[379]:
```
                            Train Accuracy  Test Accuracy           MAE  \
Models
SGD                          -5.988780e+26  -5.992822e+26  6.414599e+12
SGD With Feature              5.057383e-02   5.036669e-02  2.124583e-01
SGD Scaling                   5.146062e-02   5.148195e-02  2.115446e-01
SGD With Normalize           -7.011206e-04  -7.753664e-06  2.165393e-01
SGD With PCA                 -6.237332e+21  -6.642600e+21  1.740485e+10
SGD With PCA and Scaling      5.440495e-02   5.390186e-02  2.137283e-01
SGD With PCA and Normalize   -1.351815e-04  -6.241363e-04  2.201433e-01

                                     MSE          MdSE
```

```
Models
SGD                         4.479084e+25  7.362598e+12
SGD With Feature            7.097636e-02  1.826241e-01
SGD Scaling                 7.089301e-02  1.804711e-01
SGD With Normalize          7.474139e-02  1.836708e-01
SGD With PCA                4.964733e+20  1.417424e+10
SGD With PCA and Scaling    7.071214e-02  1.865059e-01
SGD With PCA and Normalize  7.478746e-02  1.915666e-01
```

[380]:
```
models_draw(df)
```

GradientBoostingRegressor

[381]:
```
Search(GradientBoostingRegressor(max_depth=2),{'max_depth':
↪[5,10,15,20,25,30,35,40]},X_train,y_train)
```

[381]:
```
GradientBoostingRegressor(max_depth=5)
```

[382]:
```
cross_validation(GradientBoostingRegressor(max_depth=5),X_train,y_train)
```

```
Train Score Value :  [0.28806379 0.29342476 0.29203183 0.30102895 0.29137174]
Mean 0.29318421585473053
Test Score Value :  [0.26966994 0.25176698 0.2499494  0.2233259  0.24889817]
Mean 0.24872207800073584
```

[383]:
```
Values =␣
↪Models(GradientBoostingRegressor(max_depth=5),X_train,y_train,X_test,y_test)
```

```
Apply Model With Normal Data :

R2 Score Train : 0.28789871932049527
R2 Score Test : 0.23590908422212797
Mean Absolute Error Value is :  0.18470457322041237
Mean Squared Error Value is :  0.05710877574983755
Median Absolute Error Value is :  0.15128630612180166


 Apply Model With Feature Selection :

R2 Score Train : 0.2722153116530235
R2 Score Test : 0.23062753800691382
Mean Absolute Error Value is :  0.1852020821806564
Mean Squared Error Value is :  0.05750352280439454
Median Absolute Error Value is :  0.15223484188551334


 Apply Model With Normal Data With Scaling :

R2 Score Train : 0.28690302247955135
```

```
R2 Score Test : 0.2361327559256945
Mean Absolute Error Value is :  0.18462488570218816
Mean Squared Error Value is :  0.057092058345014636
Median Absolute Error Value is :  0.1520219342727312


 Apply Model With Normal Data With Normalize :

R2 Score Train : 0.30498481102426067
R2 Score Test : 0.237201992953589
Mean Absolute Error Value is :  0.18467917869667758
Mean Squared Error Value is :  0.05701214270096162
Median Absolute Error Value is :  0.15175297814966276


 Apply Model With Normal Data With PCA :

R2 Score Train : 0.3050864461207925
R2 Score Test : 0.23266945759972635
Mean Absolute Error Value is :  0.18552039447276308
Mean Squared Error Value is :  0.0573509080752868
Median Absolute Error Value is :  0.15367857374546637


 Apply Model With Normal Data With PCA and Scaling :

R2 Score Train : 0.29721818185547266
R2 Score Test : 0.22197600626916114
Mean Absolute Error Value is :  0.18711213841909594
Mean Squared Error Value is :  0.058150145314493265
Median Absolute Error Value is :  0.15584658428232567


 Apply Model With Normal Data With PCA and Normalize :

R2 Score Train : 0.3039890198899693
R2 Score Test : 0.21909651452733037
Mean Absolute Error Value is :  0.18799955883023536
Mean Squared Error Value is :  0.058365360866415264
Median Absolute Error Value is :  0.15725123736073815
```

[384]:
```
df = pd.DataFrame(Values,columns=['Train Accuracy','Test␣
 ↪Accuracy','MAE','MSE','MdSE'])
df['Models'] = ['Gradient','Gradient With Feature','Gradient Scaling','Gradient␣
 ↪With Normalize','Gradient With PCA'
                 ,'Gradient With PCA and Scaling',
                 'Gradient With PCA and Normalize']
```

```python
df.set_index('Models', inplace=True)
df
```

[384]:

```
                                Train Accuracy  Test Accuracy       MAE  \
Models
Gradient                              0.287899       0.235909  0.184705
Gradient With Feature                 0.272215       0.230628  0.185202
Gradient Scaling                      0.286903       0.236133  0.184625
Gradient With Normalize               0.304985       0.237202  0.184679
Gradient With PCA                     0.305086       0.232669  0.185520
Gradient With PCA and Scaling         0.297218       0.221976  0.187112
Gradient With PCA and Normalize       0.303989       0.219097  0.188000

                                      MSE      MdSE
Models
Gradient                         0.057109  0.151286
Gradient With Feature            0.057504  0.152235
Gradient Scaling                 0.057092  0.152022
Gradient With Normalize          0.057012  0.151753
Gradient With PCA                0.057351  0.153679
Gradient With PCA and Scaling    0.058150  0.155847
Gradient With PCA and Normalize  0.058365  0.157251
```

[385]:

```python
models_draw(df)
```

Clustering

Feature Scaling

[386]:

```python
Columns = X_cluster.columns
```

[387]:

```python
MS = MinMaxScaler()
X_cluster = MS.fit_transform(X_cluster)
```

[388]:

```python
X_cluster = pd.DataFrame(X_cluster,columns=Columns)
X_cluster.head()
```

[388]:

```
        age  job  marital  education  default  housing  loan  contact  \
0  0.735849  0.3      0.5   0.000000      0.0      0.0   0.0      1.0
1  0.754717  0.7      0.5   0.500000      0.0      0.0   0.0      1.0
2  0.377358  0.7      0.5   0.500000      0.0      1.0   0.0      1.0
3  0.433962  0.0      0.5   0.166667      0.0      0.0   0.0      1.0
4  0.735849  0.7      0.5   0.500000      0.0      0.0   1.0      1.0

      month  day_of_week  …  campaign  pdays  previous  poutcome  \
0  0.666667         0.25  …       0.0    1.0       0.0       0.5
1  0.666667         0.25  …       0.0    1.0       0.0       0.5
2  0.666667         0.25  …       0.0    1.0       0.0       0.5
```

```
3  0.666667          0.25  …        0.0     1.0        0.0        0.5
4  0.666667          0.25  …        0.0     1.0        0.0        0.5

   emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  nr.employed    y
0      0.888889            0.72           0.64   0.911111          0.8  0.0
1      0.888889            0.72           0.64   0.911111          0.8  0.0
2      0.888889            0.72           0.64   0.911111          0.8  0.0
3      0.888889            0.72           0.64   0.911111          0.8  0.0
4      0.888889            0.72           0.64   0.911111          0.8  0.0

[5 rows x 21 columns]
```

[389]:
```
X_train,X_test=Split(X_cluster,classification=2)
```

```
X_train shape is  (37056, 21)
X_test shape is  (4118, 21)
```

[390]:
```
X_train.y = X_train.y.astype(int)
X_test.y = X_test.y.astype(int)
y_train=X_train.iloc[:,-1]
y_test=X_test.iloc[:,-1]
```

PCA

[391]:
```
PCAModel = PCA(n_components=2, svd_solver='auto')
PCAModel.fit(X_train.iloc[:,:-1])
print('PCAModel Explained Variance is : ' , PCAModel.explained_variance_)
print('PCAModel Explained Variance ratio is : ' , PCAModel.
 ↪explained_variance_ratio_)
```

```
PCAModel Explained Variance is :  [0.3236848  0.24030423]
PCAModel Explained Variance ratio is :  [0.18894046 0.14026977]
```

[392]:
```
X_train_pca = PCAModel.transform(X_train.iloc[:,:-1])
X_test_pca = PCAModel.transform(X_test.iloc[:,:-1])
X_train_pca = pd.DataFrame(X_train_pca,columns=['Feature1','Feature2'])
X_test_pca = pd.DataFrame(X_test_pca,columns=['Feature1','Feature2'])
X_train_pca.head()
```

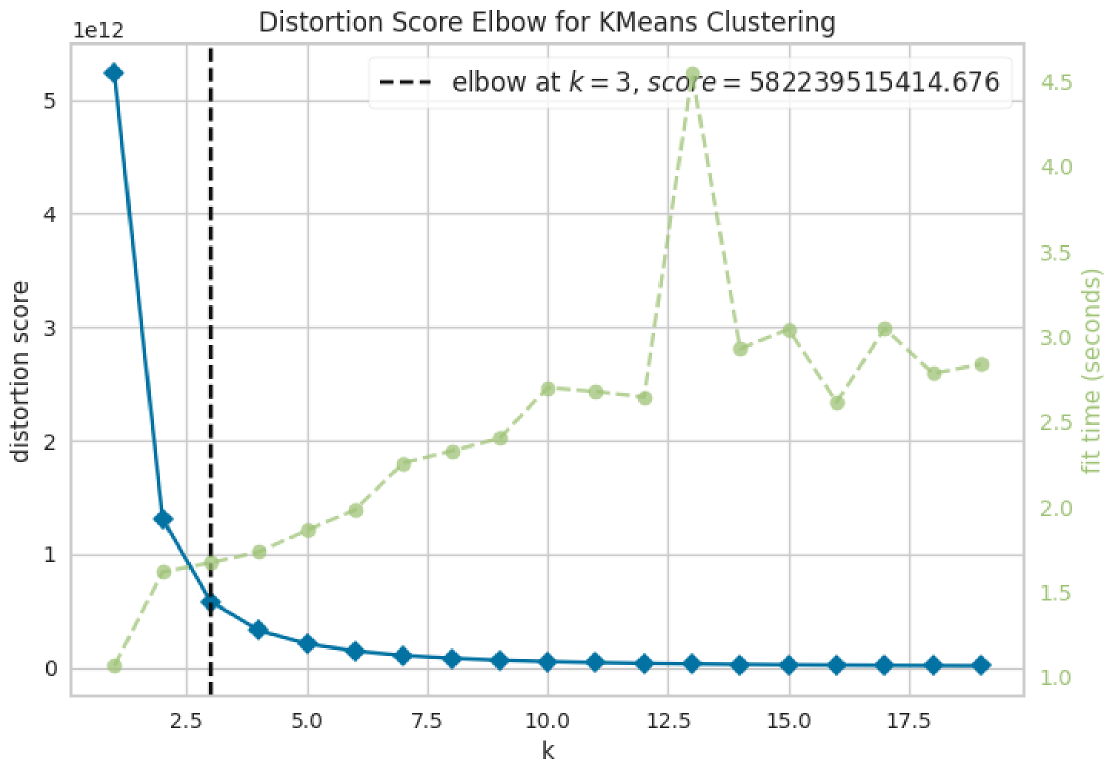
[392]:
```
   Feature1  Feature2
0  0.956731 -0.200738
1  0.231606 -0.408428
2 -0.806488  0.276320
3 -0.434795 -0.580570
4  0.416362  0.718087
```

[393]:
```
X_train.reset_index(inplace=True)
X_test.reset_index(inplace=True)
```

[394]:
```python
fig = go.Figure()
for color,y_ in zip(['red','orange'],X_train.y.unique()):
    pca = X_train_pca[X_train.y==y_]
    scatter_trace = go.Scatter(
        x=pca['Feature1'],
        y=pca['Feature2'],
        mode='markers',
        marker=dict(
            size=20,
            color=color,
            symbol='circle',
            opacity=0.8
        ),
        name=f'Train Cluster {y_}'
    )
    fig.add_trace(scatter_trace)
for color,y_ in zip(['green','blue'],X_test.y.unique()):
    pca = X_test_pca[X_test.y==y_]
    scatter_trace = go.Scatter(
        x=pca['Feature1'],
        y=pca['Feature2'],
        mode='markers',
        marker=dict(
            size=10,
            color=color,
            symbol='circle',
            opacity=0.8
        ),
        name=f'Test Cluster {y_}'
    )
    fig.add_trace(scatter_trace)
fig.update_layout(
    title_text='PCA',
    title_x=0.5,
    title_font=dict(size=20),
    xaxis_title='Feature1',
    yaxis_title='Feature2',
    font=dict(size=15),
    width=1000,
    height=700,
    xaxis=dict(tickangle=-90),
    template='plotly_dark'
)
fig.update_annotations(font=dict(size=20))
fig.show()
```

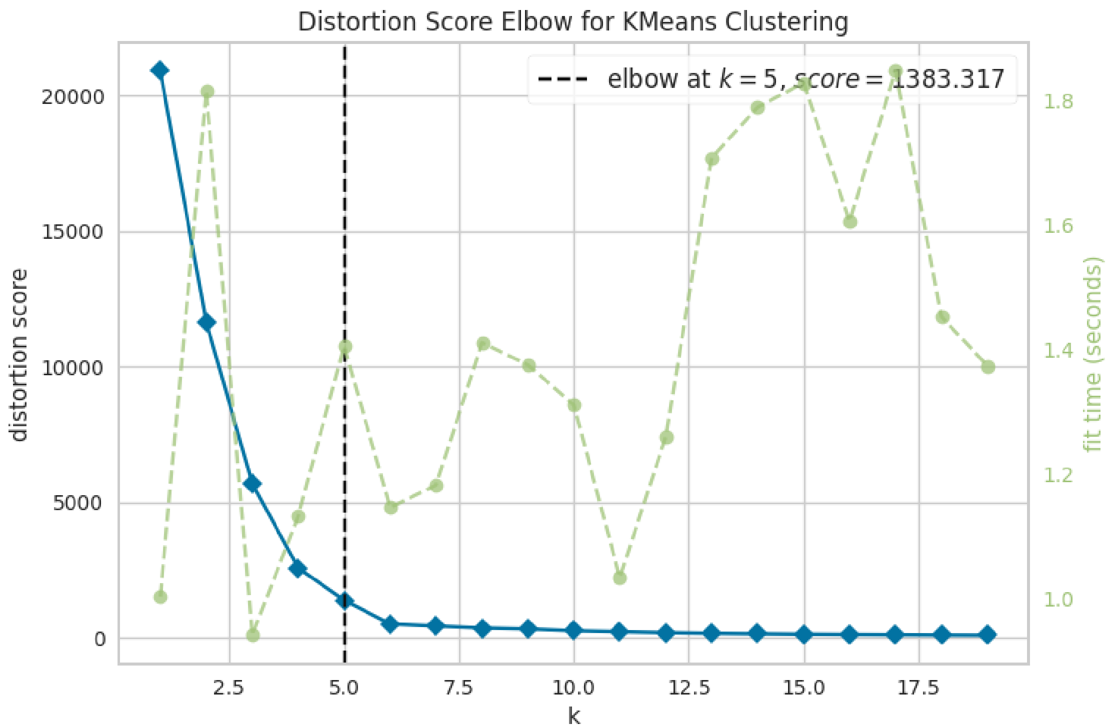
```
Elbow
```

[395]:
```
kmeans = KMeans(init='k-means++', random_state=44)
visualizer = KElbowVisualizer(kmeans, k=(1,20))
visualizer.fit(X_train.iloc[:,:-1])
visualizer.show()
```



[395]: <Axes: title={'center': 'Distortion Score Elbow for KMeans Clustering'}, xlabel='k', ylabel='distortion score'>

[396]:
```
kmeans = KMeans(init='k-means++', random_state=44)
visualizer = KElbowVisualizer(kmeans, k=(1,20))
visualizer.fit(X_train_pca)
visualizer.show()
```



[396]: <Axes: title={'center': 'Distortion Score Elbow for KMeans Clustering'}, xlabel='k', ylabel='distortion score'>

K-Means model with two clusters PCA

[397]:
```
kmeans1 = KMeans(n_clusters=2,random_state=44)
kmeans1.fit(X_train_pca)
```

[397]: KMeans(n_clusters=2, random_state=44)

[398]:
```
kmeans1.cluster_centers_
```

[398]:
```
array([[-0.21870605,  0.506627  ],
       [ 0.17940622, -0.41558994]])
```

[399]:
```
kmeans1.inertia_
```

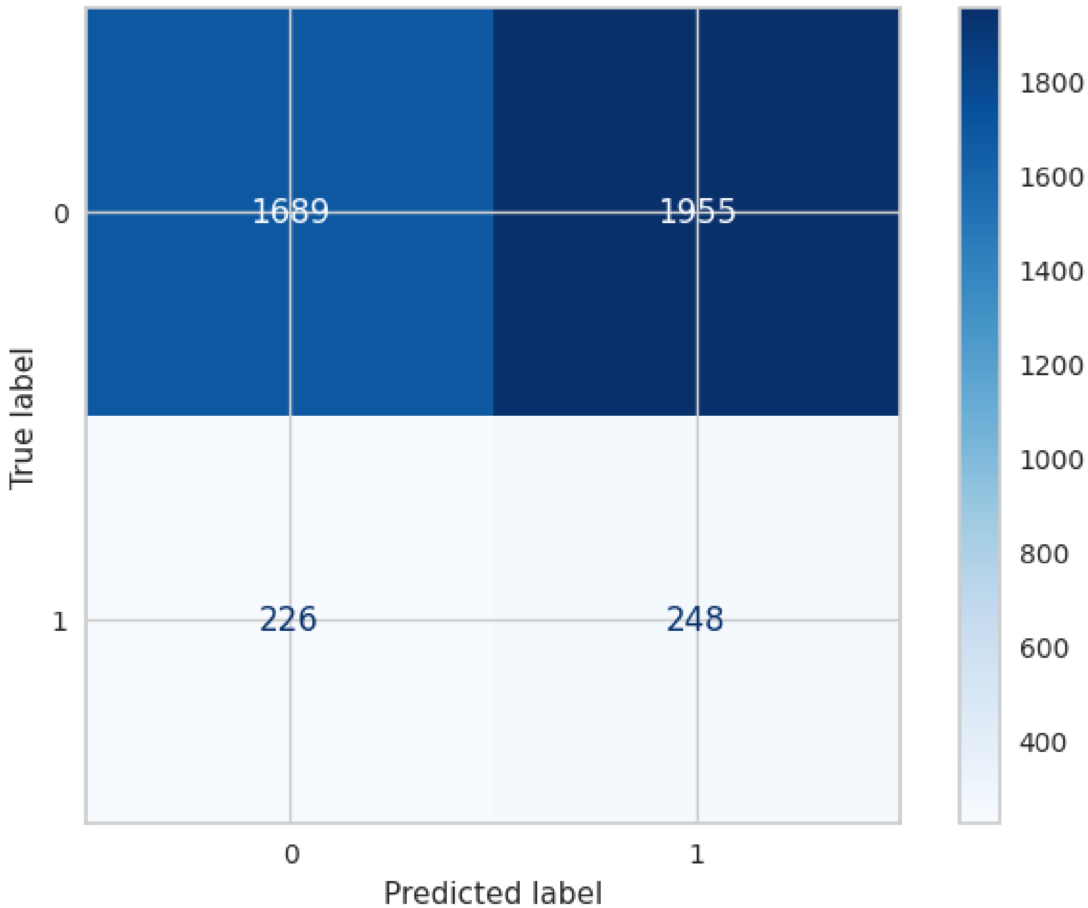
[399]: 11642.53353421653

Evaluation

[400]:
```
y_train_pred = kmeans1.predict(X_train_pca)
y_pred = kmeans1.predict(X_test_pca)
```

[401]:
```
value_kmeans_pca =
    Check(cluster=1,y_train2=y_train,y_train_pred=y_train_pred,y_test2=y_test,y_pred=y_pred)
```

```
Model Train Score is :  0.46594343696027635
Model Test Score is :  0.47037396794560465
F1 Score is :  0.18528203212551367
Recall Score is :  0.5232067510548524
Precision Score is :  0.11257376305038584
AUC Value  :  0.4933541987985568

Classification Report is :                precision    recall  f1-score
support

           0       0.88      0.46      0.61      3644
           1       0.11      0.52      0.19       474

    accuracy                           0.47      4118
   macro avg       0.50      0.49      0.40      4118
weighted avg       0.79      0.47      0.56      4118

Confusion Matrix is :
 [[1689 1955]
 [ 226  248]]
```



[402]:
```
kmeans = KMeans(n_clusters=5,random_state=44)
kmeans.fit(X_train_pca)
```

[402]:
```
KMeans(n_clusters=5, random_state=44)
```

[403]:
```
kmeans.cluster_centers_
```

[403]:
```
array([[ 0.16768806,  0.62574404],
       [ 0.23082439, -0.41251487],
       [-0.83283842,  0.31730313],
       [-0.52876893, -0.63461834],
       [ 0.77951772, -0.21431489]])
```

[404]:
```
kmeans.inertia_
```

[404]:
```
1383.3170484291445
```

[405]:
```python
fig = go.Figure()
for color,y_ in zip(['red','orange'],[0,1]):
    pca = X_train_pca[y_train_pred==y_]
    scatter_trace = go.Scatter(
        x=pca['Feature1'],
        y=pca['Feature2'],
        mode='markers',
        marker=dict(
            size=20,
            color=color,
            symbol='circle',
            opacity=0.8
        ),
        name=f'Train Cluster {y_}'
    )
    fig.add_trace(scatter_trace)
for color,y_ in zip(['green','blue'],[0,1]):
    pca = X_test_pca[y_pred==y_]
    scatter_trace = go.Scatter(
        x=pca['Feature1'],
        y=pca['Feature2'],
        mode='markers',
        marker=dict(
            size=10,
            color=color,
            symbol='circle',
            opacity=0.8
        ),
        name=f'Test Cluster {y_}'
    )
    fig.add_trace(scatter_trace)
fig.update_layout(
    title_text='PCA',
    title_x=0.5,
    title_font=dict(size=20),
    xaxis_title='Feature1',
    yaxis_title='Feature2',
    font=dict(size=15),
    width=1000,
    height=700,
    xaxis=dict(tickangle=-90),
    template='plotly_dark'
)
fig.update_annotations(font=dict(size=20))
fig.show()
```

K-Means model with two clusters

[406]:
```
kmeans2 = KMeans(n_clusters=2,random_state=44)
kmeans2.fit(X_train.iloc[:,:-1])
```

[406]:
```
KMeans(n_clusters=2, random_state=44)
```

[407]:
```
kmeans2.cluster_centers_
```

[407]:
```
array([[ 3.08556103e+04,  4.32962071e-01,  3.77643750e-01,
         6.04478823e-01,  6.60554263e-01,  1.62276194e-04,
         5.92037648e-01,  1.54757397e-01,  9.24433386e-02,
         4.76484977e-01,  4.96740953e-01,  3.63398632e-01,
         2.23984421e-01,  9.44268530e-01,  4.94092374e-02,
         4.28841889e-01,  5.06974871e-01,  4.03940066e-01,
         3.80094120e-01,  6.77451937e-01,  6.45394061e-01],
       [ 1.02762860e+04,  4.33140241e-01,  3.48317088e-01,
         5.64543056e-01,  5.74335362e-01, -2.71050543e-19,
         5.06866283e-01,  1.50573537e-01,  6.39237439e-01,
         4.64023073e-01,  5.06098874e-01,  3.66534504e-01,
         2.86122031e-01,  1.00000000e+00, -1.54043445e-15,
         5.00000000e-01,  9.57922703e-01,  7.32142819e-01,
         4.45999246e-01,  9.46602077e-01,  9.24260865e-01]])
```

[408]:
```
kmeans2.inertia_
```

[408]:
```
1309740502783.2498
```

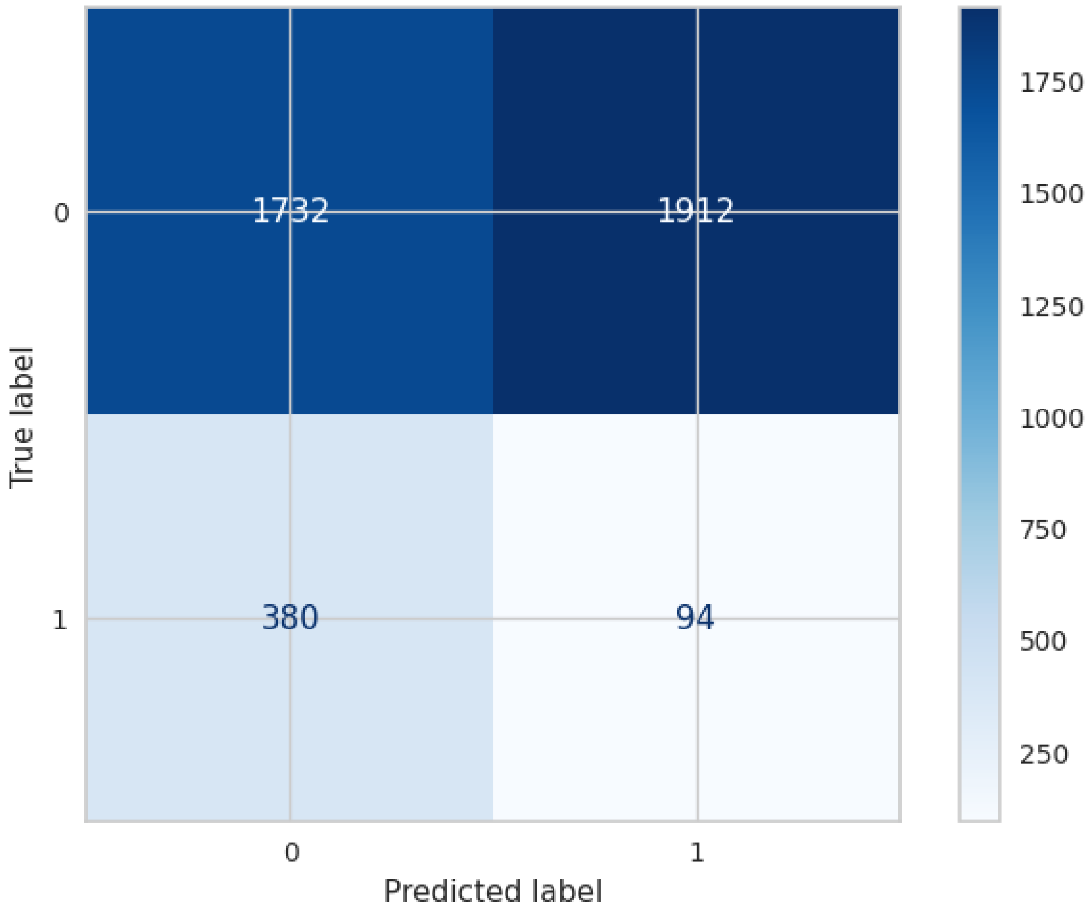
Evaluation

[409]:
```
y_train_pred = kmeans2.predict(X_train.iloc[:,:-1])
y_pred = kmeans2.predict(X_test.iloc[:,:-1])
```

[410]:
```
value_kmeans =␣
↪Check(cluster=1,y_train2=y_train,y_train_pred=y_train_pred,y_test2=y_test,y_pred=y_pred)
```

```
Model Train Score is :  0.4335330310880829
Model Test Score is :  0.4434191355026712
F1 Score is :  0.07580645161290323
Recall Score is :  0.19831223628691982
Precision Score is :  0.04685942173479561
AUC Value  :  0.3368070511840746

Classification Report is :                precision    recall  f1-score
support

           0       0.82      0.48      0.60      3644
           1       0.05      0.20      0.08       474

    accuracy                           0.44      4118
   macro avg       0.43      0.34      0.34      4118
```

```
weighted avg       0.73      0.44      0.54      4118

Confusion Matrix is :
 [[1732 1912]
 [ 380   94]]
```



- The lesser the model inertia, the better the model fit.
- We can see that the model has very high inertia. So, this is not a good model fit to the data.

Use elbow method to find optimal number of clusters

[411]:
```
inertia_values = []
k_range =  np.arange(1, 11)

for k in k_range:
    kmeans = KMeans(n_clusters=k, init='k-means++', max_iter=600, n_init=10,␣
 ↪random_state=44)
    kmeans.fit(X_cluster)
    inertia_values.append(kmeans.inertia_)
```

```python
fig = go.Figure()
fig.add_trace(go.Scatter(x=k_range, y=inertia_values, mode='lines+markers'))
fig.update_layout(
    title='The Elbow Method for Optimal Number of Clusters',
    xaxis=dict(title='Number of Clusters'),
    yaxis=dict(title='Inertia'),
    template='plotly_dark'
)
fig.show()
```

- By the above plot, we can see that there is a kink at k=3.
- Hence k=3 can be considered a good number of the cluster to cluster this data.

K-Means model with different clusters

K-Means model with 3 clusters

[412]:
```python
kmeans = KMeans(n_clusters=3,random_state=44)
kmeans.fit(X_train.iloc[:,:-1])
```

[412]:
```
KMeans(n_clusters=3, random_state=44)
```

[413]:
```python
kmeans.cluster_centers_
```

[413]:
```
array([[ 6.83742978e+03,  4.34689478e-01,  3.41394635e-01,
         5.52844215e-01,  5.50083495e-01, -2.57498016e-19,
         4.84890110e-01,  1.43018746e-01,  9.12330317e-01,
         5.57934712e-01,  5.08140756e-01,  3.64231140e-01,
         2.66402715e-01,  1.00000000e+00, -1.14491749e-15,
         5.00000000e-01,  9.36867055e-01,  7.78765352e-01,
         4.92624434e-01,  9.32633566e-01,  8.86360698e-01],
       [ 3.42898556e+04,  4.24729872e-01,  3.69404501e-01,
         6.25274190e-01,  6.32328648e-01, -2.57498016e-19,
         5.90381022e-01,  1.53871151e-01,  9.97644000e-02,
         4.90806185e-01,  4.73007555e-01,  3.83575731e-01,
         2.08432854e-01,  9.18274308e-01,  6.68848578e-02,
         4.16362012e-01,  3.15685903e-01,  3.65088959e-01,
         3.30051182e-01,  5.57602838e-01,  4.92688277e-01],
       [ 2.05766839e+04,  4.39692342e-01,  3.78085846e-01,
         5.75499151e-01,  6.69738367e-01,  2.42502627e-04,
         5.73033708e-01,  1.61102579e-01,  8.56842616e-02,
         3.62047441e-01,  5.22997332e-01,  3.47196787e-01,
         2.90291811e-01,  9.98031998e-01,  7.28662656e-03,
         4.76881416e-01,  9.44116617e-01,  5.60255436e-01,
         4.16234743e-01,  9.45412787e-01,  9.74852478e-01]])
```

[414]:
```python
kmeans.inertia_
```

[414]: 582242281750.3159

K-Means model with 6 clusters

[415]:
```
kmeans = KMeans(n_clusters=6,random_state=44)
kmeans.fit(X_train.iloc[:,:-1])
```

[415]:
```
KMeans(n_clusters=6, random_state=44)
```

[416]:
```
kmeans.cluster_centers_
```

[416]:
```
array([[ 1.71108019e+04,  4.29488175e-01,  3.60936239e-01,
         5.88297014e-01,  6.22195319e-01, -2.84603070e-19,
         5.49636804e-01,  1.65456013e-01,  9.49152542e-02,
         2.77553583e-01,  5.00686037e-01,  3.70920266e-01,
         3.23066990e-01,  1.00000000e+00,  4.16333634e-17,
         5.00000000e-01,  1.00000000e+00,  6.39838579e-01,
         3.50443906e-01,  9.74465839e-01,  1.00000000e+00],
       [ 3.08666946e+04,  4.08326927e-01,  3.46029957e-01,
         6.07183121e-01,  5.76126054e-01, -2.57498016e-19,
         5.94942825e-01,  1.61217587e-01,  8.02061524e-02,
         4.26370323e-01,  4.96255436e-01,  3.80644304e-01,
         2.17361894e-01,  9.81428943e-01,  4.58320871e-02,
         3.72765341e-01,  3.49635833e-01,  3.51283621e-01,
         1.45240780e-01,  6.92780321e-01,  6.10822999e-01],
       [ 3.40819509e+03,  4.41887584e-01,  3.40293690e-01,
         5.47200258e-01,  5.43461890e-01, -2.57498016e-19,
         4.89269001e-01,  1.48297563e-01,  1.00000000e+00,
         6.66666667e-01,  5.45505890e-01,  3.78761064e-01,
         2.37857028e-01,  1.00000000e+00,  4.16333634e-17,
         5.00000000e-01,  8.88888889e-01,  7.20000000e-01,
         6.40000000e-01,  9.11485076e-01,  8.00000000e-01],
       [ 2.39843280e+04,  4.49240651e-01,  3.94250646e-01,
         5.62903747e-01,  7.16489018e-01,  4.84496124e-04,
         5.96091731e-01,  1.56169251e-01,  7.67118863e-02,
         4.45144272e-01,  5.46794251e-01,  3.24522069e-01,
         2.55943152e-01,  9.96068128e-01,  1.45118125e-02,
         4.53972868e-01,  8.89032443e-01,  4.81679587e-01,
         4.81479328e-01,  9.16810734e-01,  9.50064599e-01],
       [ 3.77420802e+04,  4.41553836e-01,  3.93087106e-01,
         6.43814349e-01,  6.89791360e-01, -2.71050543e-19,
         5.85389770e-01,  1.46429155e-01,  1.20281092e-01,
         5.57080859e-01,  4.47949011e-01,  3.86045525e-01,
         2.00163425e-01,  8.54444542e-01,  8.80862886e-02,
         4.60696192e-01,  2.82671460e-01,  3.79329956e-01,
         5.17424416e-01,  4.21337650e-01,  3.74080732e-01],
       [ 1.02571554e+04,  4.27933987e-01,  3.43359375e-01,
         5.57861328e-01,  5.56586372e-01, -2.57498016e-19,
```

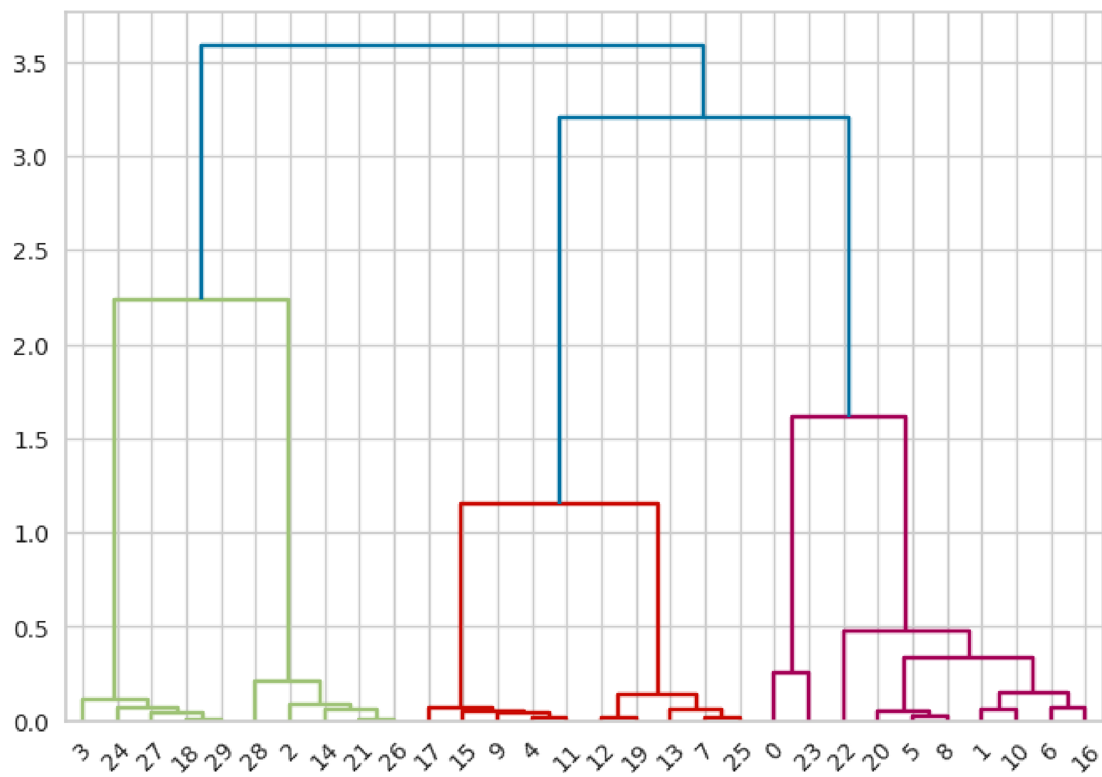
```
4.80631510e-01,  1.38183594e-01,  8.27636719e-01,
4.49544271e-01,  4.70499674e-01,  3.48994572e-01,
2.96093750e-01,  1.00000000e+00,  6.24500451e-17,
5.00000000e-01,  9.84899450e-01,  8.38600260e-01,
3.45416667e-01,  9.53725405e-01,  9.72819010e-01]])
```

[417]:
```
kmeans.inertia_
```

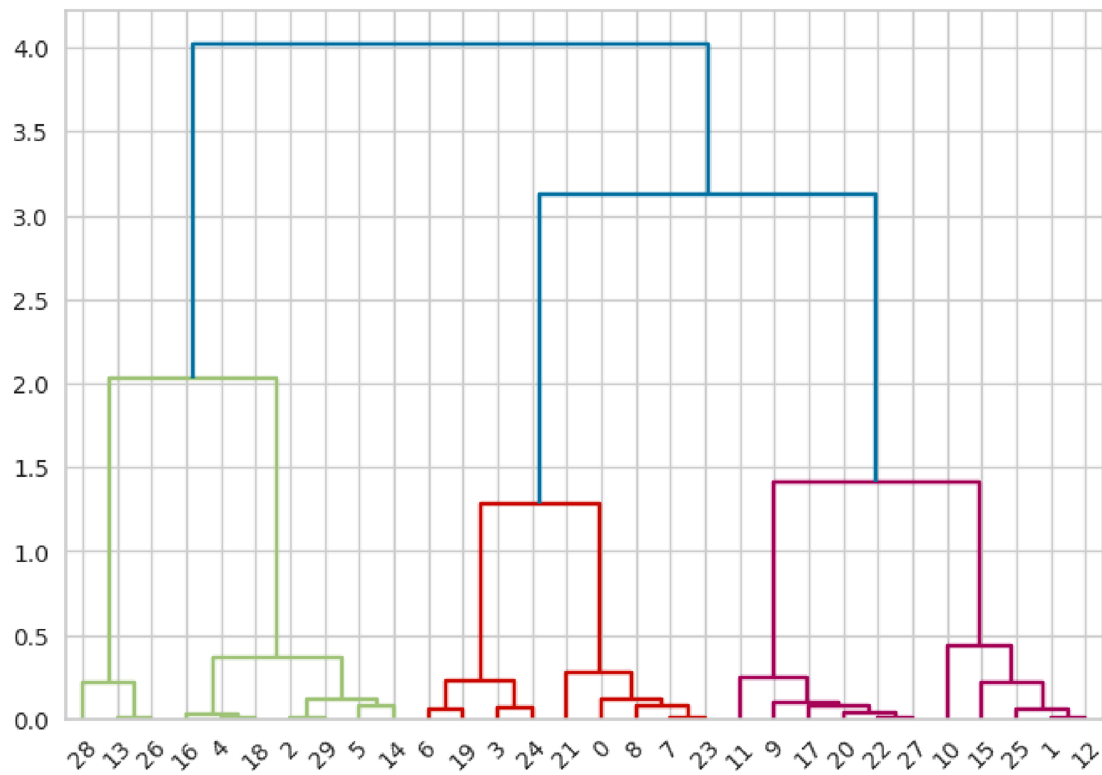
[417]:
```
145524897092.83975
```

AgglomerativeClustering

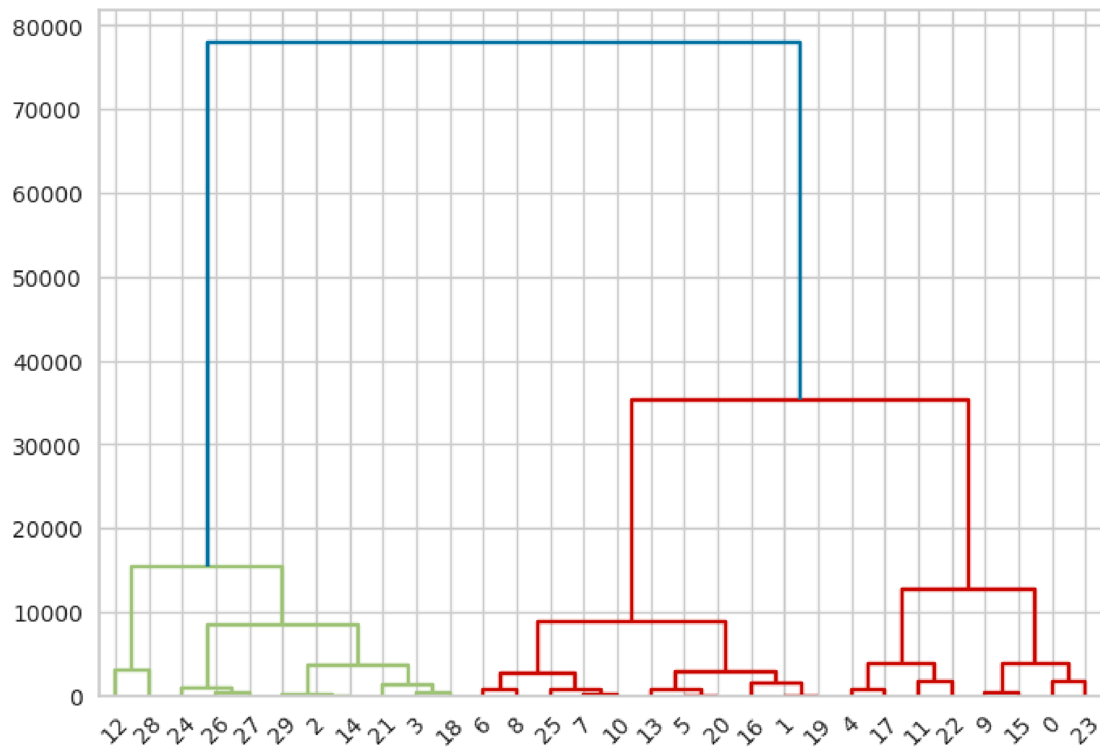
[418]:
```
den = sch.dendrogram(sch.linkage(X_train_pca.iloc[: 30], method = 'ward'))
```



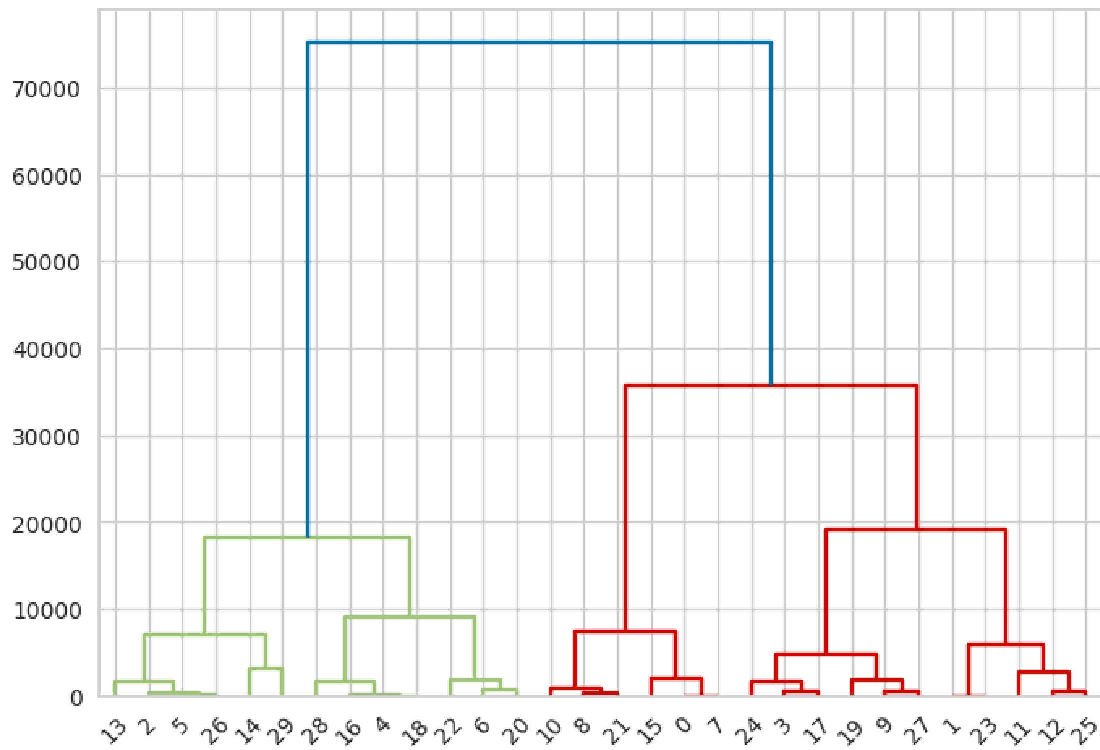
[419]:
```
den = sch.dendrogram(sch.linkage(X_test_pca.iloc[: 30], method = 'ward'))
```



[420]:
```
den = sch.dendrogram(sch.linkage(X_train.iloc[: 30,:-1], method = 'ward'))
```



[421]:
```
den = sch.dendrogram(sch.linkage(X_test.iloc[: 30,:-1], method = 'ward'))
```

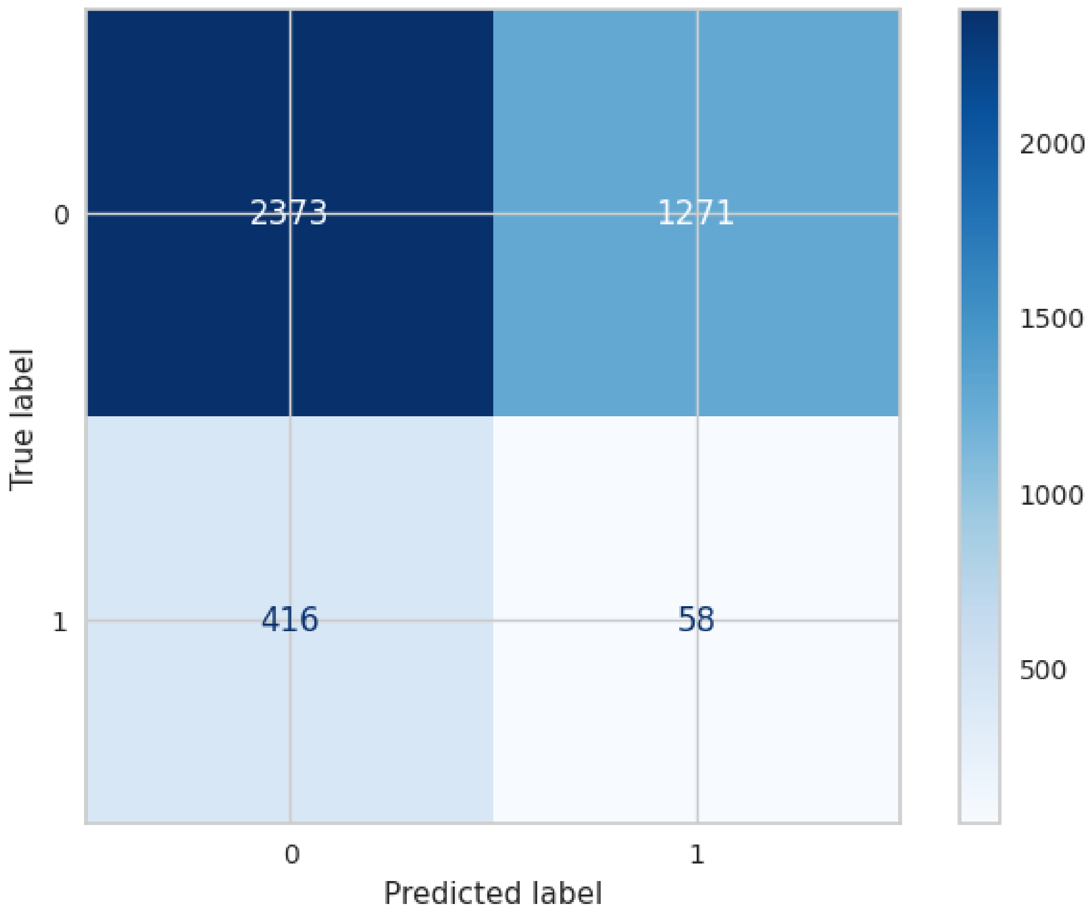


[422]:
```
AggClusteringModel =
  ↪AgglomerativeClustering(n_clusters=2,affinity='euclidean',linkage='ward')
y_train_pred = AggClusteringModel.fit_predict(X_train_pca)
y_pred = AggClusteringModel.fit_predict(X_test_pca)
```

[423]:
```
value_agg_pca =
  ↪Check(cluster=1,y_train2=y_train,y_train_pred=y_train_pred,y_test2=y_test,y_pred=y_pred)
```

```
Model Train Score is :  0.5795822538860104
Model Test Score is :  0.5903351141330743
F1 Score is :  0.06433721575152523
Recall Score is :  0.12236286919831224
Precision Score is :  0.0436418359668924
AUC Value  :  0.386785166761615

Classification Report is :                precision    recall  f1-score
support

           0       0.85      0.65      0.74      3644
           1       0.04      0.12      0.06       474

    accuracy                           0.59      4118
   macro avg       0.45      0.39      0.40      4118
weighted avg       0.76      0.59      0.66      4118
```

```
Confusion Matrix is :
 [[2373 1271]
 [ 416   58]]
```




```
[424]: fig = go.Figure()
for color,y_ in zip(['red','orange'],[0,1]):
    pca = X_train_pca[y_train_pred==y_]
    scatter_trace = go.Scatter(
        x=pca['Feature1'],
        y=pca['Feature2'],
        mode='markers',
        marker=dict(
            size=20,
            color=color,
            symbol='circle',
            opacity=0.8
        ),
```

```
        name=f'Train Cluster {y_}'
    )
    fig.add_trace(scatter_trace)
for color,y_ in zip(['green','blue'],[0,1]):
    pca = X_test_pca[y_pred==y_]
    scatter_trace = go.Scatter(
        x=pca['Feature1'],
        y=pca['Feature2'],
        mode='markers',
        marker=dict(
            size=10,
            color=color,
            symbol='circle',
            opacity=0.8
        ),
        name=f'Test Cluster {y_}'
    )
    fig.add_trace(scatter_trace)
fig.update_layout(
    title_text='PCA',
    title_x=0.5,
    title_font=dict(size=20),
    xaxis_title='Feature1',
    yaxis_title='Feature2',
    font=dict(size=15),
    width=1000,
    height=700,
    xaxis=dict(tickangle=-90),
    template='plotly_dark'
)
fig.update_annotations(font=dict(size=20))
fig.show()
```

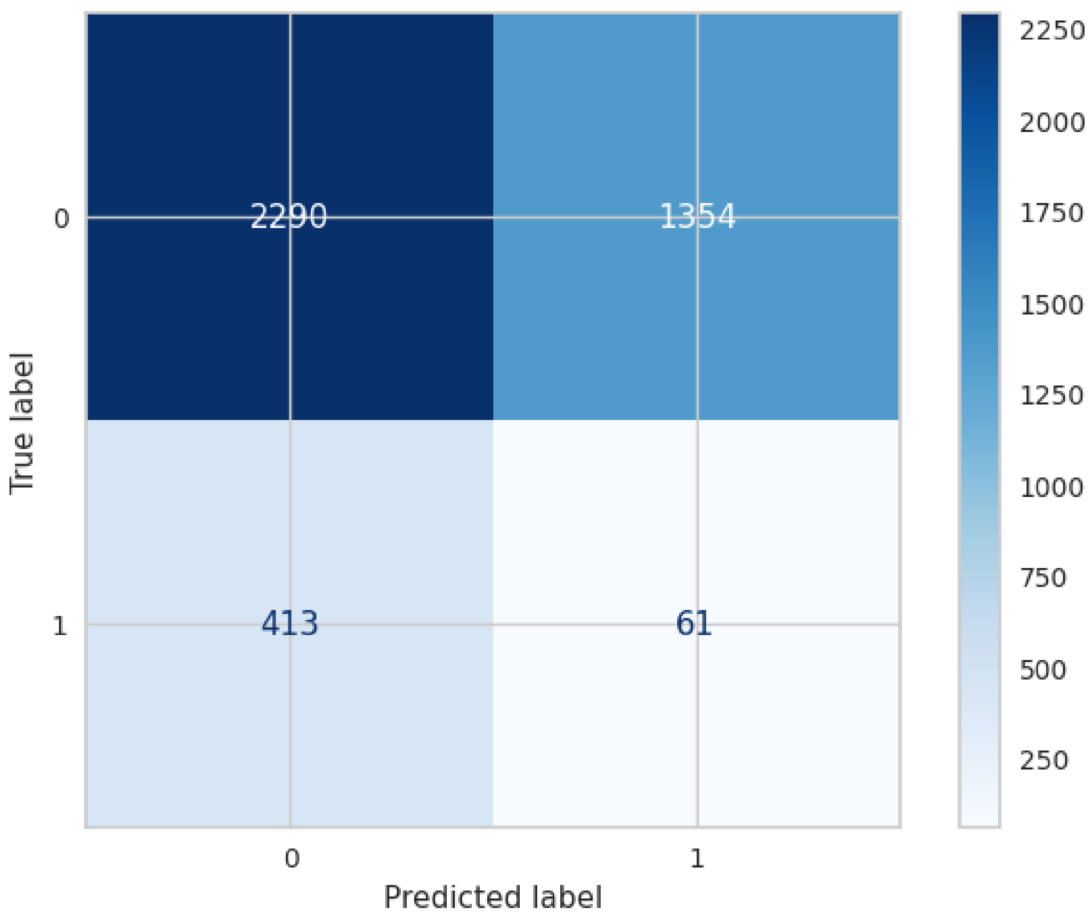
[425]:
```
AggClusteringModel =␣
 ↪AgglomerativeClustering(n_clusters=2,affinity='euclidean',linkage='ward')
y_train_pred = AggClusteringModel.fit_predict(X_train.iloc[:,:-1])
y_pred = AggClusteringModel.fit_predict(X_test.iloc[:,:-1])
```

[426]:
```
y_train = X_train.iloc[:,-1]
y_test = X_test.iloc[:,-1]
```

[427]:
```
value_agg =␣
 ↪Check(cluster=1,y_train2=y_train,y_train_pred=y_train_pred,y_test2=y_test,y_pred=y_pred)
```

```
Model Train Score is :  0.7206660189982729
Model Test Score is :  0.5709082078678971
F1 Score is :  0.06458443620963472
Recall Score is :  0.12869198312236288
```

```
Precision Score is :  0.0431095406360424
AUC Value  :  0.37856113974998495
Classification Report is :                precision    recall  f1-score
support

           0       0.85      0.63      0.72      3644
           1       0.04      0.13      0.06       474

    accuracy                           0.57      4118
   macro avg       0.45      0.38      0.39      4118
weighted avg       0.75      0.57      0.65      4118

Confusion Matrix is :
 [[2290 1354]
 [ 413   61]]
```



[428]:
```
list = [value_kmeans,value_kmeans_pca,value_agg,value_agg_pca]
df = pd.DataFrame(list,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['Kmeans','Kmeans␣
↪PCA','AgglomerativeClustering','AgglomerativeClustering PCA']
df.set_index('Models', inplace=True)
df
```

[428]:
```
                             Train Accuracy  Test Accuracy   Test F1  \
Models
Kmeans                             0.433533       0.443419  0.075806
Kmeans PCA                         0.465943       0.470374  0.185282
AgglomerativeClustering            0.720666       0.570908  0.064584
AgglomerativeClustering PCA        0.579582       0.590335  0.064337

                             Test Recall  Test Precision       AUC
Models
Kmeans                          0.198312        0.046859  0.336807
Kmeans PCA                      0.523207        0.112574  0.493354
AgglomerativeClustering         0.128692        0.043110  0.378561
AgglomerativeClustering PCA     0.122363        0.043642  0.386785
```

[429]:
```
models_draw(df)
```

** #

DL Models

Tabel of Contents

Deep Learning Models

[430]:
```
X_train_c,y_train_c,X_test_c,y_test_c=Split(X_classification,y_classification)
X_train_r,y_train_r,X_test_r,y_test_r=Split(X_regression,y_regression,classification=0)
```

```
X_train shape is  (37056, 20)
X_test shape is  (4118, 20)
y_train shape is  (37056,)
y_test shape is  (4118,)
X_train shape is  (37056, 20)
X_test shape is  (4118, 20)
y_train shape is  (37056,)
y_test shape is  (4118,)
```

[431]:
```
classification_Input = keras.Input(shape=(X_classification.shape[1],))
regression_Input = keras.Input(shape=(X_regression.shape[1],))

dense_layer1 = keras.layers.Dense(128, activation='relu', name='Dense_Layer1')
dense_layer2 = keras.layers.Dense(256, activation='relu', name='Dense_Layer2')
```

```
batch_norm1 = keras.layers.BatchNormalization(name='BatchNorm1')
dropout1 = keras.layers.Dropout(0.5, name='Dropout1')

batch_norm2 = keras.layers.BatchNormalization(name='BatchNorm2')
dropout2 = keras.layers.Dropout(0.5, name='Dropout2')

classification_output = dense_layer1(classification_Input)
classification_output = batch_norm1(classification_output)
classification_output = dropout1(classification_output)

regression_output = dense_layer1(regression_Input)
regression_output = batch_norm1(regression_output)
regression_output = dropout1(regression_output)

layer = dense_layer2(classification_output)
layer2 = dense_layer2(regression_output)

layer = batch_norm2(layer)
layer = dropout2(layer)

layer2 = batch_norm2(layer2)
layer2 = dropout2(layer2)

layer_C = keras.layers.Dense(1, activation='sigmoid',␣
↪name='Dense_Layer3')(layer)
layer_R = keras.layers.Dense(1, name='Dense_Layer4')(layer2)

model = keras.Model(inputs=[classification_Input, regression_Input],␣
↪outputs=[layer_C, layer_R])
```

[432]:
```
model.summary()
```

Model: "functional_1"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 20)	0	-
input_layer_1 (InputLayer)	(None, 20)	0	-
Dense_Layer1 (Dense)	(None, 128)	2,688	input_layer[0][0… input_layer_1[0]…
BatchNorm1	(None, 128)	512	Dense_Layer1[0][…

(BatchNormalizatio…			Dense_Layer1[1][…
Dropout1 (Dropout)	(None, 128)	0	BatchNorm1[0][0], BatchNorm1[1][0]
Dense_Layer2 (Dense)	(None, 256)	33,024	Dropout1[0][0], Dropout1[1][0]
BatchNorm2 (BatchNormalizatio…	(None, 256)	1,024	Dense_Layer2[0][… Dense_Layer2[1][…
Dropout2 (Dropout)	(None, 256)	0	BatchNorm2[0][0], BatchNorm2[1][0]
Dense_Layer3 (Dense)	(None, 1)	257	Dropout2[0][0]
Dense_Layer4 (Dense)	(None, 1)	257	Dropout2[1][0]

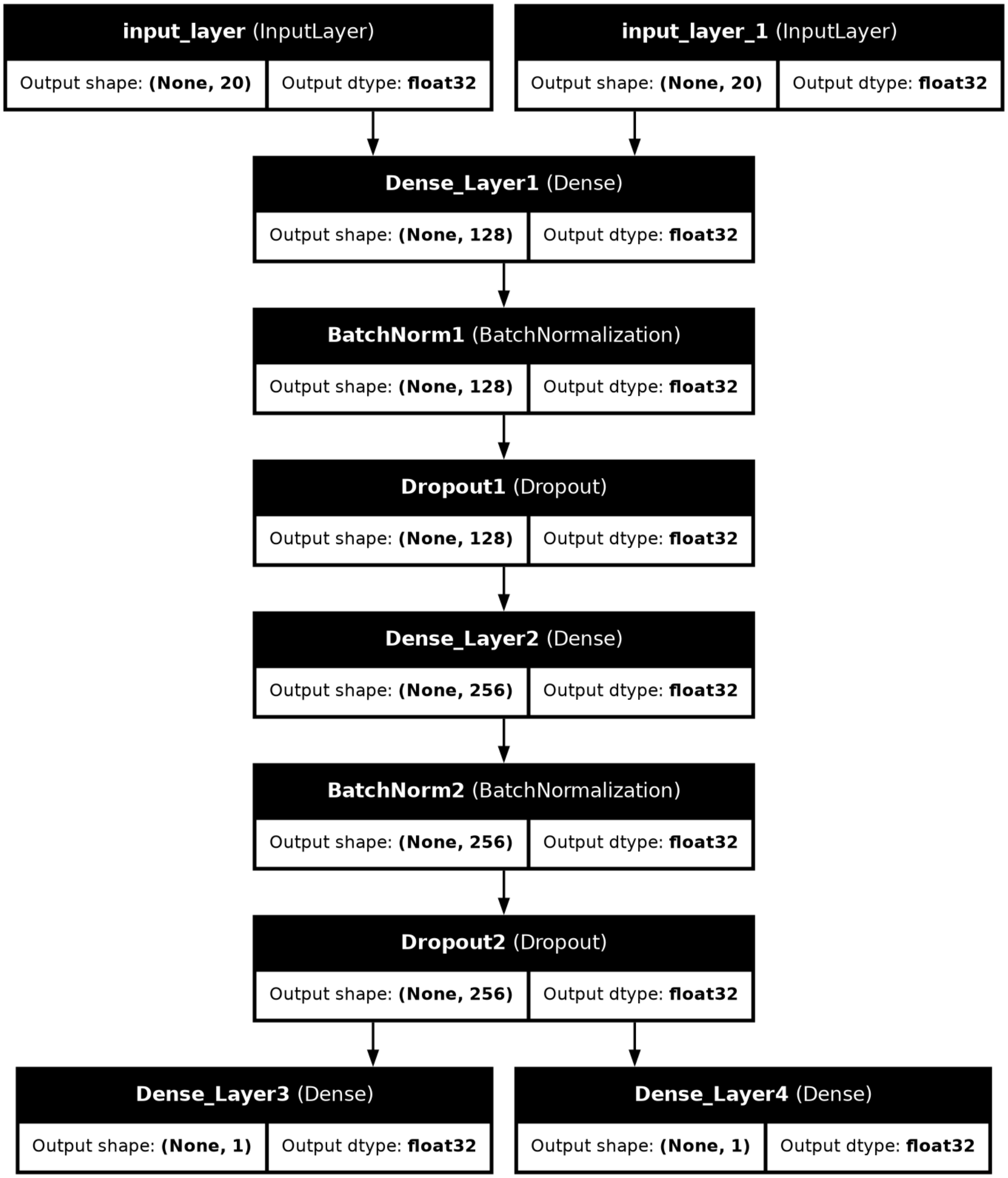
Total params: 37,762 (147.51 KB)

Trainable params: 36,994 (144.51 KB)

Non-trainable params: 768 (3.00 KB)

[433]:
```
keras.utils.plot_model(model, to_file='model.png', show_shapes=True,␣
↪show_layer_names=True,show_dtype=True,dpi=120)
```

[433]:



[434]:

```
model.compile(optimizer='adam',
              loss={'Dense_Layer3': 'binary_crossentropy', 'Dense_Layer4':
 ↪'mse'},
              metrics={'Dense_Layer3': 'accuracy', 'Dense_Layer4': 'mae'})
checkpoint_cb = keras.callbacks.ModelCheckpoint("my_keras_model.keras",
 ↪save_best_only=True)
```

```python
early_stopping_cb = keras.callbacks.EarlyStopping(patience=10,␣
↪restore_best_weights=True)
hist = model.fit([X_train_c, X_train_r], [y_train_c, y_train_r],
          epochs=50,
          batch_size=32,validation_split=.1,
          callbacks=[checkpoint_cb, early_stopping_cb])
```

```
Epoch 1/50
  63/1043              2s 2ms/step -
Dense_Layer3_accuracy: 0.6417 - Dense_Layer4_mae: 1.3646 - loss: 4.2058
```

```
WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
I0000 00:00:1715726207.054039      86 device_compiler.h:186] Compiled cluster
using XLA!  This line is logged at most once for the lifetime of the process.
W0000 00:00:1715726207.075109      86 graph_launch.cc:671] Fallback to op-by-op
mode because memset node breaks graph update
```

```
1043/1043              0s 10ms/step -
Dense_Layer3_accuracy: 0.8102 - Dense_Layer4_mae: 0.7718 - loss: 1.8273
```

```
W0000 00:00:1715726218.517338      85 graph_launch.cc:671] Fallback to op-by-op
mode because memset node breaks graph update
```

```
1043/1043              24s 12ms/step -
Dense_Layer3_accuracy: 0.8102 - Dense_Layer4_mae: 0.7715 - loss: 1.8265 -
val_Dense_Layer3_accuracy: 0.9074 - val_Dense_Layer4_mae: 0.2324 - val_loss:
0.3345
Epoch 2/50
1043/1043              3s 3ms/step -
Dense_Layer3_accuracy: 0.8937 - Dense_Layer4_mae: 0.2281 - loss: 0.3382 -
val_Dense_Layer3_accuracy: 0.9064 - val_Dense_Layer4_mae: 0.2221 - val_loss:
0.3869
Epoch 3/50
1043/1043              3s 3ms/step -
Dense_Layer3_accuracy: 0.8957 - Dense_Layer4_mae: 0.2256 - loss: 0.3199 -
val_Dense_Layer3_accuracy: 0.9034 - val_Dense_Layer4_mae: 0.2092 - val_loss:
0.3576
Epoch 4/50
1043/1043              3s 3ms/step -
Dense_Layer3_accuracy: 0.8985 - Dense_Layer4_mae: 0.2268 - loss: 0.3068 -
val_Dense_Layer3_accuracy: 0.9110 - val_Dense_Layer4_mae: 0.2041 - val_loss:
0.3758
Epoch 5/50
1043/1043              3s 3ms/step -
Dense_Layer3_accuracy: 0.9011 - Dense_Layer4_mae: 0.2295 - loss: 0.3059 -
val_Dense_Layer3_accuracy: 0.8988 - val_Dense_Layer4_mae: 0.2106 - val_loss:
0.4149
Epoch 6/50
1043/1043              3s 3ms/step -
```

```
Dense_Layer3_accuracy: 0.8967 - Dense_Layer4_mae: 0.2279 - loss: 0.3024 -
val_Dense_Layer3_accuracy: 0.9037 - val_Dense_Layer4_mae: 0.2095 - val_loss:
0.4204
Epoch 7/50
1043/1043          3s 3ms/step -
Dense_Layer3_accuracy: 0.9008 - Dense_Layer4_mae: 0.2297 - loss: 0.3035 -
val_Dense_Layer3_accuracy: 0.9072 - val_Dense_Layer4_mae: 0.2383 - val_loss:
0.4417
Epoch 8/50
1043/1043          3s 3ms/step -
Dense_Layer3_accuracy: 0.8961 - Dense_Layer4_mae: 0.2273 - loss: 0.3021 -
val_Dense_Layer3_accuracy: 0.9042 - val_Dense_Layer4_mae: 0.2209 - val_loss:
0.4648
Epoch 9/50
1043/1043          3s 3ms/step -
Dense_Layer3_accuracy: 0.8981 - Dense_Layer4_mae: 0.2205 - loss: 0.2967 -
val_Dense_Layer3_accuracy: 0.9069 - val_Dense_Layer4_mae: 0.4269 - val_loss:
0.7116
Epoch 10/50
1043/1043          3s 3ms/step -
Dense_Layer3_accuracy: 0.8987 - Dense_Layer4_mae: 0.2223 - loss: 0.2952 -
val_Dense_Layer3_accuracy: 0.8934 - val_Dense_Layer4_mae: 0.2054 - val_loss:
0.5030
Epoch 11/50
1043/1043          3s 2ms/step -
Dense_Layer3_accuracy: 0.9037 - Dense_Layer4_mae: 0.2155 - loss: 0.2874 -
val_Dense_Layer3_accuracy: 0.8980 - val_Dense_Layer4_mae: 0.2630 - val_loss:
0.5691
```

[435]:
```
model.evaluate([X_test_c, X_test_r], [y_test_c, y_test_r])
```

```
129/129          1s 5ms/step -
Dense_Layer3_accuracy: 0.9022 - Dense_Layer4_mae: 0.2347 - loss: 0.3416
```

[435]:
```
[0.33887195587158203, 0.902865469455719, 0.23253843188285828]
```

[436]:
```
hist_=pd.DataFrame(hist.history)
hist_
```

[436]:
```
   Dense_Layer3_accuracy  Dense_Layer4_mae      loss  \
0               0.859550          0.495564  0.981496
1               0.893493          0.225321  0.331328
2               0.895682          0.225232  0.315929
3               0.896732          0.227821  0.309192
4               0.900990          0.228293  0.303696
5               0.899100          0.228310  0.300975
6               0.899730          0.228815  0.301962
7               0.901020          0.226401  0.297765
```

```
8                0.900540         0.220134  0.292269
9                0.899910         0.221390  0.292300
10               0.901409         0.214900  0.286633

    val_Dense_Layer3_accuracy  val_Dense_Layer4_mae  val_loss
0                    0.907447              0.232434  0.334514
1                    0.906368              0.222118  0.386918
2                    0.903400              0.209225  0.357563
3                    0.910955              0.204098  0.375783
4                    0.898813              0.210603  0.414932
5                    0.903670              0.209537  0.420378
6                    0.907178              0.238337  0.441722
7                    0.904209              0.220949  0.464843
8                    0.906908              0.426935  0.711648
9                    0.893416              0.205392  0.503035
10                   0.898003              0.263031  0.569103
```

[437]:

```
def summary_plot():
    fig = make_subplots(rows=2, cols=2, subplot_titles=("Total Loss",'',␣
↪"Classification Accuracy", "Regression MAE"))
    fig.add_trace(go.Scatter(x=hist_.index, y=hist_['loss'], mode='lines',␣
↪name='Total Loss', line=dict(color='blue')), row=1, col=1)
    fig.add_trace(go.Scatter(x=hist_.index, y=hist_['val_loss'], mode='lines',␣
↪name='Validation Loss', line=dict(color='orange')), row=1, col=1)
    fig.add_trace(go.Scatter(x=hist_.index, y=hist_['Dense_Layer3_accuracy'],␣
↪mode='lines', name='Train Classification Accuracy', line=dict(color='red')),␣
↪row=2, col=1)
    fig.add_trace(go.Scatter(x=hist_.index,␣
↪y=hist_['val_Dense_Layer3_accuracy'], mode='lines', name='Validation␣
↪Classification Accuracy', line=dict(color='red')), row=2, col=1)
    fig.add_trace(go.Scatter(x=hist_.index, y=hist_['Dense_Layer4_mae'],␣
↪mode='lines', name='Train Regression MAE', line=dict(color='purple')),␣
↪row=2, col=2)
    fig.add_trace(go.Scatter(x=hist_.index, y=hist_['val_Dense_Layer4_mae'],␣
↪mode='lines', name='Validation Regression MAE', line=dict(color='purple')),␣
↪row=2, col=2)
    fig.update_layout(
        title_text="Training Summary",
        title_x=0.5,
        title_font=dict(size=20),
        font=dict(size=15),
        width=1100,
        height=1000,
        template='plotly_dark'
    )
    fig.update_annotations(font=dict(size=20))
```

```
    fig.show()
```

[438]:
```
summary_plot()
```

[439]:
```
predictions = model.predict([X_test_c,X_test_r])
```

```
 88/129              0s 2ms/step

W0000 00:00:1715726308.818039      87 graph_launch.cc:671] Fallback to op-by-op
mode because memset node breaks graph update

129/129              1s 5ms/step
```

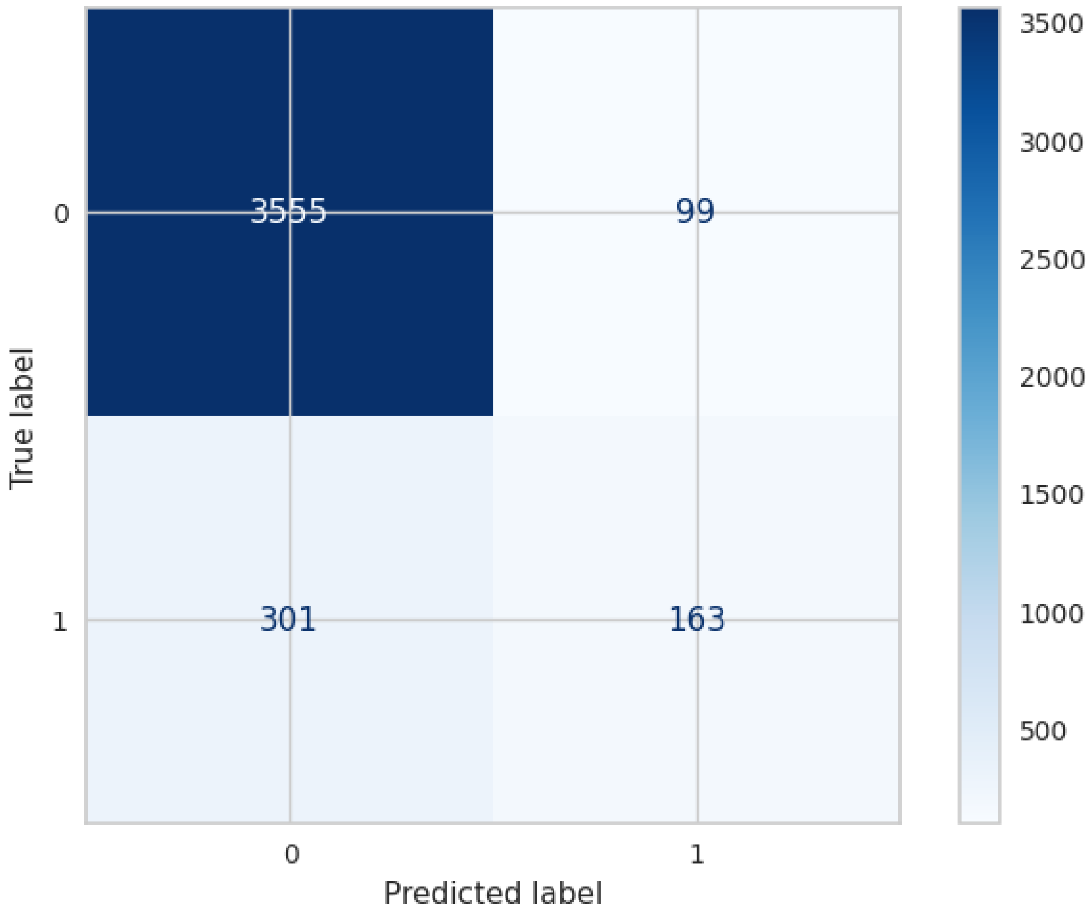
[440]:
```
classification_predictions = np.where(predictions[0]>=.5,1,0)
regression_predictions = predictions[1]
```

[441]:
```
def Check(model_22 = 1):
    if model_22:
        train = accuracy_score(y_train_c,np.where(model.
 ↪predict([X_train_c,X_train_r])[0]>=.5,1,0))
    else:
        train = accuracy_score(y_train_c,np.where(model2.predict(X_train_c)>=.
 ↪5,1,0))
    y_pred=classification_predictions
    test = accuracy_score(y_test_c,y_pred)
    print('Model Train Score is : ' , train)
    print('Model Test Score is : ' , test)
    F1Score = f1_score(y_test_c, y_pred)
    print('F1 Score is : ', F1Score)
    RecallScore = recall_score(y_test_c, y_pred)
    print('Recall Score is : ', RecallScore)
    PrecisionScore = precision_score(y_test_c, y_pred)
    print('Precision Score is : ', PrecisionScore)
    fprValue2, tprValue2, thresholdsValue2 = roc_curve(y_test_c,y_pred)
    AUCValue = auc(fprValue2, tprValue2)
    print('AUC Value  : ', AUCValue)
    Area(fprValue2,tprValue2,AUCValue)
    ClassificationReport = classification_report(y_test_c,y_pred)
    print('Classification Report is : ', ClassificationReport)
    CM = confusion_matrix(y_test_c, y_pred)
    print('Confusion Matrix is : \n', CM)
    disp = ConfusionMatrixDisplay(confusion_matrix=CM, display_labels=[0,1])
    disp.plot(cmap='Blues')
    values=[train,test,F1Score,RecallScore,PrecisionScore,AUCValue]
    return values
def Check_R():
    y_pred = regression_predictions
```

```
    print('R2 Score Train :',r2_score(y_train_c,model.
↪predict([X_train_c,X_train_r])[1]))
    print('R2 Score Test :',r2_score(y_test_c,y_pred))
    MAEValue = mean_absolute_error(y_test_c, y_pred)
    print('Mean Absolute Error Value is : ', MAEValue)
    MSEValue = mean_squared_error(y_test_c, y_pred)
    print('Mean Squared Error Value is : ', MSEValue)
    MdSEValue = median_absolute_error(y_test_c, y_pred)
    print('Median Absolute Error Value is : ', MdSEValue )
```

[442]:
```
values_d = Check()
```

```
1158/1158            2s 2ms/step
Model Train Score is :  0.9028767271157168
Model Test Score is :  0.9028654686741137
F1 Score is :  0.4490358126721763
Recall Score is :  0.35129310344827586
Precision Score is :  0.6221374045801527
AUC Value  :  0.6620997536945812

Classification Report is :                precision    recall  f1-score
support

           0       0.92      0.97      0.95      3654
           1       0.62      0.35      0.45       464

    accuracy                           0.90      4118
   macro avg       0.77      0.66      0.70      4118
weighted avg       0.89      0.90      0.89      4118

Confusion Matrix is :
 [[3555   99]
 [ 301  163]]
```



[443]:
```
Check_R()
```

```
1158/1158              2s 2ms/step
R2 Score Train : -0.10568157381122889
R2 Score Test : -0.10942585308922137
Mean Absolute Error Value is :  0.2526771167203632
Mean Squared Error Value is :  0.1109205772578789
Median Absolute Error Value is :  0.15276914089918137
```

RandomOverSampler

[444]:
```
X_train_c,y_train_c,X_test_c,y_test_c=Split(X_classification_over,y_classification_over)
```

```
X_train shape is  (65763, 20)
X_test shape is  (7307, 20)
y_train shape is  (65763,)
y_test shape is  (7307,)
```

[445]:
```
model2 = keras.Model(inputs=[classification_Input], outputs=[layer_C])
model2.summary()
```

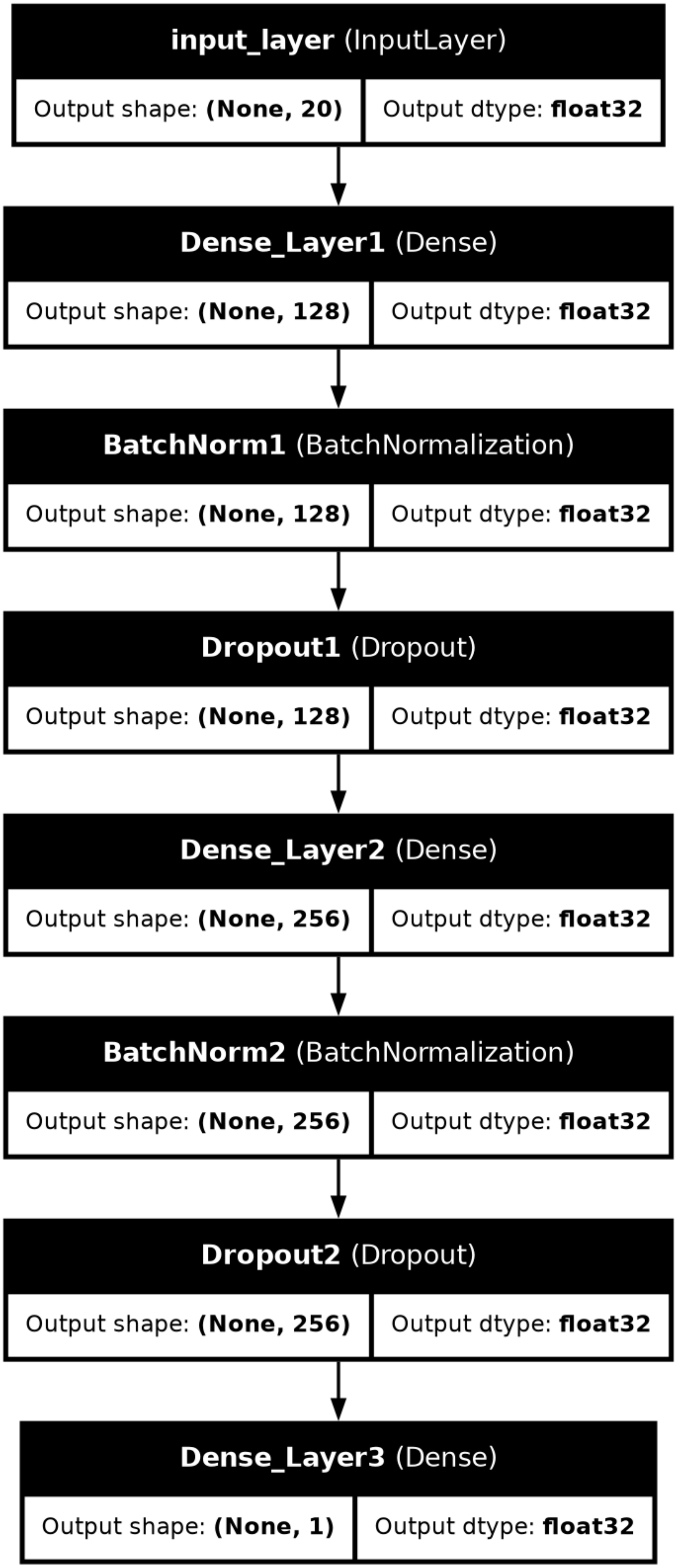
```
Model: "functional_3"
```

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 20)	0
Dense_Layer1 (Dense)	(None, 128)	2,688
BatchNorm1 (BatchNormalization)	(None, 128)	512
Dropout1 (Dropout)	(None, 128)	0
Dense_Layer2 (Dense)	(None, 256)	33,024
BatchNorm2 (BatchNormalization)	(None, 256)	1,024
Dropout2 (Dropout)	(None, 256)	0
Dense_Layer3 (Dense)	(None, 1)	257

```
Total params: 37,505 (146.50 KB)

Trainable params: 36,737 (143.50 KB)

Non-trainable params: 768 (3.00 KB)
```

[446]:
```
keras.utils.plot_model(model2, to_file='model.png', show_shapes=True,␣
↪show_layer_names=True,show_dtype=True,dpi=120)
```

[446]:


input_layer (InputLayer)
Output shape: (None, 20)
Output dtype: float32
Dense_Layer1 (Dense)
Output shape: (None, 128)
Output dtype: float32
BatchNorm1 (BatchNormalization)
Output shape: (None, 128)
Output dtype: float32
Dropout1 (Dropout)
Output shape: (None, 128)
Output dtype: float32
Dense_Layer2 (Dense)
Output shape: (None, 256)
Output dtype: float32
BatchNorm2 (BatchNormalization)
Output shape: (None, 256)
Output dtype: float32
Dropout2 (Dropout)
Output shape: (None, 256)
Output dtype: float32
Dense_Layer3 (Dense)
Output shape: (None, 1)
Output dtype: float32

[447]:
```python
model2.compile(optimizer='adam',
              loss={'Dense_Layer3': 'binary_crossentropy'},
              metrics={'Dense_Layer3': 'accuracy'})
hist = model2.fit(X_train_c,y_train_c,
          epochs=50,
          batch_size=32,validation_split=.1,
          callbacks=[checkpoint_cb, early_stopping_cb])
```

```
Epoch 1/50
1850/1850          15s 5ms/step -
accuracy: 0.8263 - loss: 0.4199 - val_accuracy: 0.8486 - val_loss: 0.3542
Epoch 2/50
1850/1850          4s 2ms/step -
accuracy: 0.8507 - loss: 0.3573 - val_accuracy: 0.8616 - val_loss: 0.3353
Epoch 3/50
1850/1850          4s 2ms/step -
accuracy: 0.8552 - loss: 0.3524 - val_accuracy: 0.8639 - val_loss: 0.3285
Epoch 4/50
1850/1850          4s 2ms/step -
accuracy: 0.8568 - loss: 0.3492 - val_accuracy: 0.8682 - val_loss: 0.3266
Epoch 5/50
1850/1850          4s 2ms/step -
accuracy: 0.8603 - loss: 0.3455 - val_accuracy: 0.8657 - val_loss: 0.3259
Epoch 6/50
1850/1850          4s 2ms/step -
accuracy: 0.8622 - loss: 0.3398 - val_accuracy: 0.8629 - val_loss: 0.3306
Epoch 7/50
1850/1850          4s 2ms/step -
accuracy: 0.8597 - loss: 0.3466 - val_accuracy: 0.8746 - val_loss: 0.3141
Epoch 8/50
1850/1850          4s 2ms/step -
accuracy: 0.8596 - loss: 0.3419 - val_accuracy: 0.8575 - val_loss: 0.3216
Epoch 9/50
1850/1850          4s 2ms/step -
accuracy: 0.8633 - loss: 0.3398 - val_accuracy: 0.8733 - val_loss: 0.3252
Epoch 10/50
1850/1850          4s 2ms/step -
accuracy: 0.8653 - loss: 0.3379 - val_accuracy: 0.8694 - val_loss: 0.3340
Epoch 11/50
1850/1850          4s 2ms/step -
accuracy: 0.8658 - loss: 0.3378 - val_accuracy: 0.8612 - val_loss: 0.3309
Epoch 12/50
1850/1850          4s 2ms/step -
accuracy: 0.8608 - loss: 0.3441 - val_accuracy: 0.8715 - val_loss: 0.3178
Epoch 13/50
```

```
1850/1850             4s 2ms/step -
accuracy: 0.8652 - loss: 0.3399 - val_accuracy: 0.8686 - val_loss: 0.3172
Epoch 14/50
1850/1850             4s 2ms/step -
accuracy: 0.8647 - loss: 0.3383 - val_accuracy: 0.8702 - val_loss: 0.3155
Epoch 15/50
1850/1850             4s 2ms/step -
accuracy: 0.8663 - loss: 0.3340 - val_accuracy: 0.8768 - val_loss: 0.3154
Epoch 16/50
1850/1850             4s 2ms/step -
accuracy: 0.8631 - loss: 0.3407 - val_accuracy: 0.8694 - val_loss: 0.3188
Epoch 17/50
1850/1850             4s 2ms/step -
accuracy: 0.8651 - loss: 0.3385 - val_accuracy: 0.8730 - val_loss: 0.3203
```

[448]:
```
model2.evaluate(X_test_c,y_test_c)
```

```
229/229             1s 3ms/step -
accuracy: 0.8803 - loss: 0.3024
```

[448]: [0.29813024401664734, 0.8810729384422302]

[449]:
```
hist_=pd.DataFrame(hist.history)
hist_
```

[449]:

	accuracy	loss	val_accuracy	val_loss
0	0.842446	0.380350	0.848563	0.354220
1	0.853040	0.354296	0.861639	0.335327
2	0.856334	0.352005	0.863920	0.328529
3	0.860051	0.346525	0.868177	0.326622
4	0.861809	0.343959	0.865744	0.325919
5	0.861352	0.342711	0.862855	0.330645
6	0.861927	0.342001	0.874563	0.314120
7	0.860862	0.341381	0.857534	0.321634
8	0.863397	0.340060	0.873347	0.325238
9	0.863431	0.339565	0.869393	0.333986
10	0.864799	0.339636	0.861183	0.330941
11	0.861437	0.343253	0.871522	0.317807
12	0.864968	0.338657	0.868633	0.317178
13	0.865644	0.337947	0.870154	0.315507
14	0.866218	0.335680	0.876844	0.315375
15	0.863718	0.338923	0.869393	0.318811
16	0.867046	0.336024	0.873042	0.320266

[450]:
```
def summary_plot2():
    fig = make_subplots(rows=1, cols=2, subplot_titles=("Total␣
↪Loss","Classification Accuracy"))
```

```
    fig.add_trace(go.Scatter(x=hist_.index, y=hist_['loss'], mode='lines',␣
↪name='Train Loss', line=dict(color='blue')), row=1, col=1)
    fig.add_trace(go.Scatter(x=hist_.index, y=hist_['val_loss'], mode='lines',␣
↪name='Validation Loss', line=dict(color='blue')), row=1, col=1)
    fig.add_trace(go.Scatter(x=hist_.index, y=hist_['accuracy'], mode='lines',␣
↪name='Train Accuracy', line=dict(color='red')), row=1, col=2)
    fig.add_trace(go.Scatter(x=hist_.index, y=hist_['val_accuracy'],␣
↪mode='lines', name='Validation Accuracy', line=dict(color='red')), row=1,␣
↪col=2)
    fig.update_layout(
        title_text="Training Summary",
        title_x=0.5,
        title_font=dict(size=20),
        font=dict(size=15),
        width=1100,
        height=600,
        template='plotly_dark'
    )
    fig.update_annotations(font=dict(size=20))
    fig.show()
```

[451]:
```
summary_plot2()
```

[452]:
```
predictions = model2.predict(X_test_c)
```

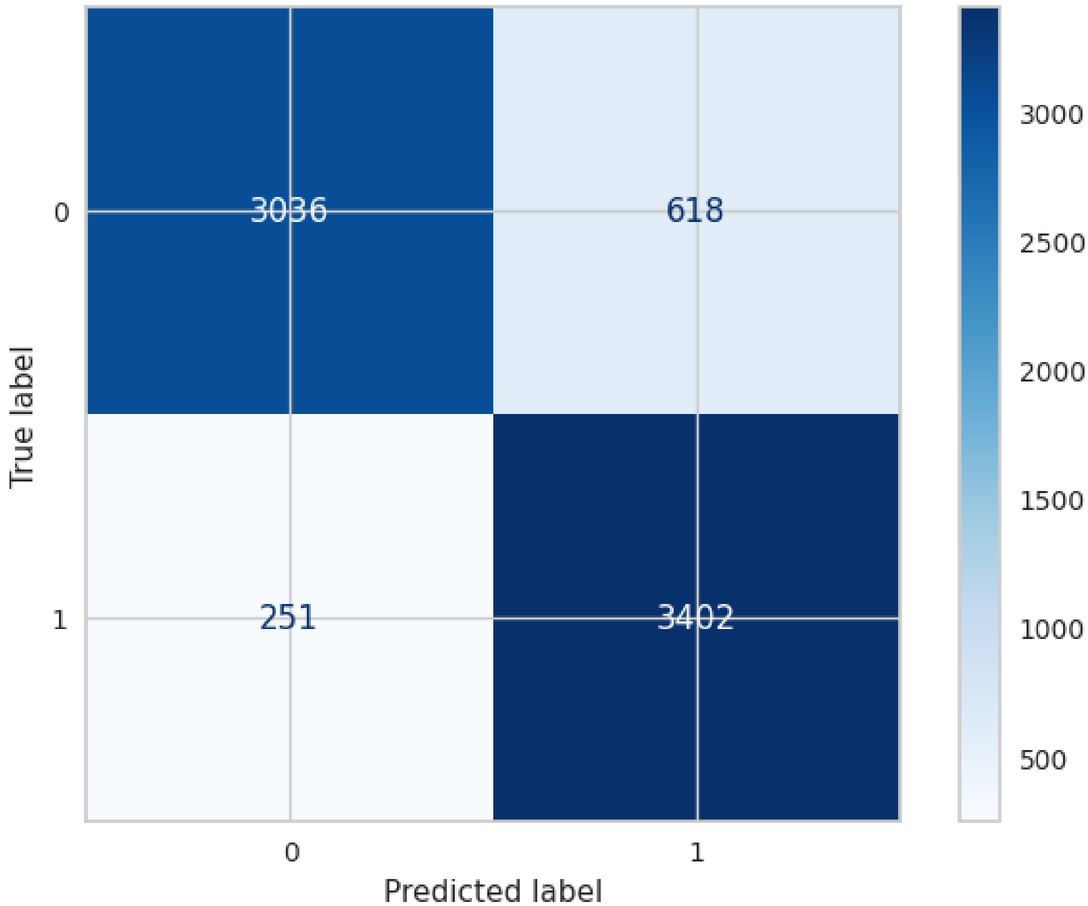
```
229/229              1s 3ms/step
```

[453]:
```
classification_predictions = np.where(predictions>=.5,1,0)
```

[454]:
```
value_d_over = Check(model_22=0)
```

```
2056/2056              3s 1ms/step
Model Train Score is :  0.8779861016072867
Model Test Score is :  0.8810729437525661
F1 Score is :  0.8867457317867848
Recall Score is :  0.9312893512181768
Precision Score is :  0.8462686567164179
AUC Value  :  0.881079815182159

Classification Report is :               precision    recall  f1-score
support

           0       0.92      0.83      0.87      3654
           1       0.85      0.93      0.89      3653

    accuracy                           0.88      7307
   macro avg       0.88      0.88      0.88      7307
weighted avg       0.88      0.88      0.88      7307
```

```
Confusion Matrix is :
 [[3036  618]
 [ 251 3402]]
```




```
RandomUnderSampler
```

[455]:
```python
X_train_c,y_train_c,X_test_c,y_test_c=Split(X_classification_under,y_classification_under)
```

```
X_train shape is  (8350, 20)
X_test shape is  (928, 20)
y_train shape is  (8350,)
y_test shape is  (928,)
```

[456]:
```python
hist = model2.fit(X_train_c,y_train_c,
          epochs=50,
          batch_size=32,validation_split=.1,
          callbacks=[checkpoint_cb, early_stopping_cb])
```

```
Epoch 1/50
235/235            5s 22ms/step -
```

```
accuracy: 0.8625 - loss: 0.3389 - val_accuracy: 0.8814 - val_loss: 0.2894
Epoch 2/50
235/235          1s 2ms/step -
accuracy: 0.8604 - loss: 0.3367 - val_accuracy: 0.8778 - val_loss: 0.2943
Epoch 3/50
235/235          1s 2ms/step -
accuracy: 0.8611 - loss: 0.3488 - val_accuracy: 0.8874 - val_loss: 0.2899
Epoch 4/50
235/235          1s 3ms/step -
accuracy: 0.8614 - loss: 0.3397 - val_accuracy: 0.8862 - val_loss: 0.2872
Epoch 5/50
235/235          1s 2ms/step -
accuracy: 0.8634 - loss: 0.3357 - val_accuracy: 0.8695 - val_loss: 0.3034
Epoch 6/50
235/235          1s 2ms/step -
accuracy: 0.8619 - loss: 0.3376 - val_accuracy: 0.8778 - val_loss: 0.2957
Epoch 7/50
235/235          1s 2ms/step -
accuracy: 0.8652 - loss: 0.3449 - val_accuracy: 0.8766 - val_loss: 0.2977
Epoch 8/50
235/235          1s 2ms/step -
accuracy: 0.8651 - loss: 0.3440 - val_accuracy: 0.8886 - val_loss: 0.2927
Epoch 9/50
235/235          1s 2ms/step -
accuracy: 0.8677 - loss: 0.3363 - val_accuracy: 0.8886 - val_loss: 0.2939
Epoch 10/50
235/235          1s 3ms/step -
accuracy: 0.8738 - loss: 0.3223 - val_accuracy: 0.8850 - val_loss: 0.2943
Epoch 11/50
235/235          1s 2ms/step -
accuracy: 0.8627 - loss: 0.3357 - val_accuracy: 0.8862 - val_loss: 0.2937
Epoch 12/50
235/235          1s 2ms/step -
accuracy: 0.8662 - loss: 0.3285 - val_accuracy: 0.8814 - val_loss: 0.2891
Epoch 13/50
235/235          1s 2ms/step -
accuracy: 0.8684 - loss: 0.3349 - val_accuracy: 0.8754 - val_loss: 0.3076
Epoch 14/50
235/235          1s 2ms/step -
accuracy: 0.8675 - loss: 0.3367 - val_accuracy: 0.8671 - val_loss: 0.3230
```

[457]:
```
model2.evaluate(X_test_c,y_test_c)
```

```
29/29          0s 2ms/step -
accuracy: 0.9002 - loss: 0.2556
```

[457]:
```
[0.283086359500885, 0.8836206793785095]
```

[458]:
```
hist_=pd.DataFrame(hist.history)
hist_
```

[458]:

	accuracy	loss	val_accuracy	val_loss
0	0.861876	0.343294	0.881437	0.289369
1	0.862009	0.340091	0.877844	0.294330
2	0.864138	0.343844	0.887425	0.289905
3	0.864671	0.337534	0.886228	0.287241
4	0.862409	0.337157	0.869461	0.303442
5	0.861743	0.342958	0.877844	0.295726
6	0.867066	0.336165	0.876647	0.297679
7	0.863473	0.344374	0.888623	0.292698
8	0.865602	0.338059	0.888623	0.293894
9	0.865868	0.336652	0.885030	0.294290
10	0.860679	0.338004	0.886228	0.293746
11	0.863340	0.335962	0.881437	0.289134
12	0.865469	0.340935	0.875449	0.307633
13	0.866933	0.336156	0.867066	0.322962

[459]:
```
summary_plot2()
```

[460]:
```
predictions = model2.predict(X_test_c)
```

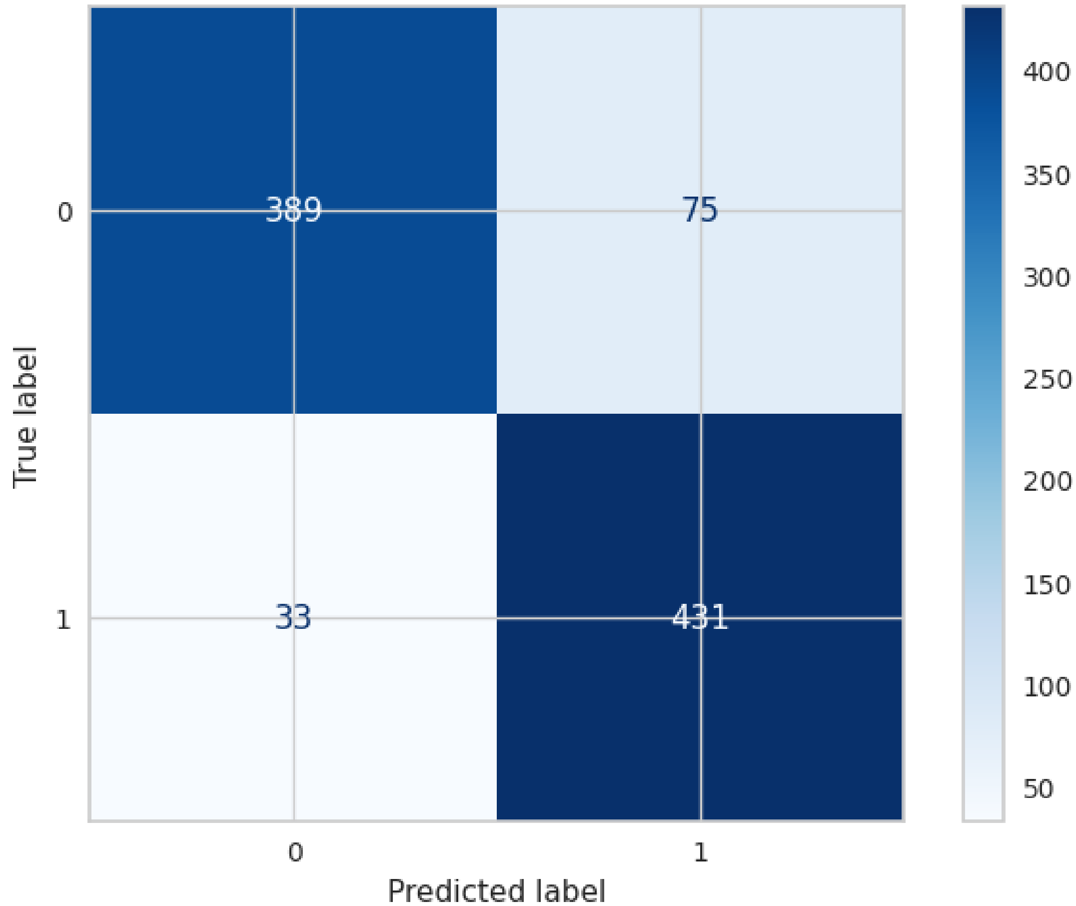
```
29/29              0s 1ms/step
```

[461]:
```
classification_predictions = np.where(predictions>=.5,1,0)
```

[462]:
```
value_d_under = Check(model_22=0)
```

```
261/261              1s 2ms/step
Model Train Score is :  0.8766467065868263
Model Test Score is :  0.8836206896551724
F1 Score is :  0.888659793814433
Recall Score is :  0.9288793103448276
Precision Score is :  0.8517786561264822
AUC Value  :  0.8836206896551724

Classification Report is :                precision    recall  f1-score
support

           0       0.92      0.84      0.88       464
           1       0.85      0.93      0.89       464

    accuracy                           0.88       928
   macro avg       0.89      0.88      0.88       928
weighted avg       0.89      0.88      0.88       928

Confusion Matrix is :
 [[389  75]
```

```
[ 33 431]]
```



[463]:
```python
list = [values_d,value_d_over,value_d_under]
df = pd.DataFrame(list,columns=['Train Accuracy','Test Accuracy','Test␣
↪F1','Test Recall','Test Precision','AUC'])
df['Models'] = ['Deep Learning','Deep Learning With Over','Deep Learning With␣
↪Under']
df.set_index('Models', inplace=True)
df
```

[463]:
```
                           Train Accuracy  Test Accuracy   Test F1  \
Models
Deep Learning                    0.902877       0.902865  0.449036
Deep Learning With Over          0.877986       0.881073  0.886746
Deep Learning With Under         0.876647       0.883621  0.888660

                           Test Recall  Test Precision       AUC
Models
```

```
Deep Learning                 0.351293        0.622137  0.662100
Deep Learning With Over       0.931289        0.846269  0.881080
Deep Learning With Under      0.928879        0.851779  0.883621
```

[464]:
```
models_draw(df)
```