Project 2 - Finance

DESCRIPTION

Problem Statement

- Finance Industry is the biggest consumer of Data Scientists. It faces constant attack by fraudsters, who try to trick the system. Correctly identifying fraudulent transactions is often compared with finding needle in a haystack because of the low event rate.
- It is important that credit card companies are able to recognize fraudulent credit card transactions so that the customers are not charged for items that they did not purchase. You are required to try various techniques such as supervised models with oversampling, unsupervised anomaly detection, and heuristics to get good accuracy at fraud detection.

Dataset Snapshot

The datasets contain transactions made by credit cards in September 2013 by European cardholders. This dataset represents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

Project Task: Week 1

Exploratory Data Analysis (EDA):

- Perform an EDA on the Dataset.
  ```
  - Check all the latent features and parameters with their mean and standard deviation. Value are close to 0 centered (mean) with unit standard deviation
  - Find if there is any connection between Time, Amount, and the transaction being fraudulent.
  ```
- Check the class count for each class. It’s a class Imbalance problem.
- Use techniques like undersampling or oversampling before running Naïve Bayes, Logistic Regression or SVM.
  ```
   - Oversampling or undersampling can be used to tackle the class imbalance problem
   - Oversampling increases the prior probability of imbalanced class and in case of other classifiers, error gets multiplied as the low-proportionate class is mimicked multiple times.
  ```
- Following are the matrices for evaluating the model performance: Precision, Recall, F1-Score, AUC-ROC curve. Use F1-Score as the evaluation criteria for this project.

In [1]:
```
!pip install matplotlib
!python -m pip install seaborn
!pip install -U imbalanced-learn
!pip install delayed
!pip install xgboost
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/site-packa
ges (3.4.2)
```

```
Requirement already satisfied: pyparsing>=2.2.1 in /usr/local/lib/python3.7/site
-packages (from matplotlib) (2.4.7)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/site-pac
kages (from matplotlib) (0.10.0)
Requirement already satisfied: numpy>=1.16 in /usr/local/lib/python3.7/site-pack
ages (from matplotlib) (1.18.5)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.7/site-pa
ckages (from matplotlib) (8.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.7/
site-packages (from matplotlib) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/sit
e-packages (from matplotlib) (1.3.1)
Requirement already satisfied: six in /usr/local/lib/python3.7/site-packages (fr
om cycler>=0.10->matplotlib) (1.15.0)
WARNING: Running pip as root will break packages and permissions. You should ins
tall packages reliably by using venv: https://pip.pypa.io/warnings/venv
WARNING: You are using pip version 21.1.1; however, version 21.1.2 is available.
You should consider upgrading via the '/usr/local/bin/python3.7 -m pip install -
-upgrade pip' command.
Requirement already satisfied: seaborn in /usr/local/lib/python3.7/site-packages
(0.11.1)
Requirement already satisfied: matplotlib>=2.2 in /usr/local/lib/python3.7/site-
packages (from seaborn) (3.4.2)
Requirement already satisfied: pandas>=0.23 in /usr/local/lib/python3.7/site-pac
kages (from seaborn) (1.1.0)
Requirement already satisfied: numpy>=1.15 in /usr/local/lib/python3.7/site-pack
ages (from seaborn) (1.18.5)
Requirement already satisfied: scipy>=1.0 in /usr/local/lib/python3.7/site-packa
ges (from seaborn) (1.5.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.7/
site-packages (from matplotlib>=2.2->seaborn) (2.8.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.7/site-pa
ckages (from matplotlib>=2.2->seaborn) (8.2.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/sit
e-packages (from matplotlib>=2.2->seaborn) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/site-pac
kages (from matplotlib>=2.2->seaborn) (0.10.0)
Requirement already satisfied: pyparsing>=2.2.1 in /usr/local/lib/python3.7/site
-packages (from matplotlib>=2.2->seaborn) (2.4.7)
Requirement already satisfied: six in /usr/local/lib/python3.7/site-packages (fr
om cycler>=0.10->matplotlib>=2.2->seaborn) (1.15.0)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/site-pac
kages (from pandas>=0.23->seaborn) (2021.1)
WARNING: Running pip as root will break packages and permissions. You should ins
tall packages reliably by using venv: https://pip.pypa.io/warnings/venv
WARNING: You are using pip version 21.1.1; however, version 21.1.2 is available.
You should consider upgrading via the '/usr/local/bin/python -m pip install --up
grade pip' command.
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.7/site
-packages (0.8.0)
Requirement already satisfied: scipy>=0.19.1 in /usr/local/lib/python3.7/site-pa
ckages (from imbalanced-learn) (1.5.2)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.7/site-pa
ckages (from imbalanced-learn) (1.18.5)
Requirement already satisfied: scikit-learn>=0.24 in /usr/local/lib/python3.7/si
te-packages (from imbalanced-learn) (0.24.2)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/site-pac
kages (from imbalanced-learn) (1.0.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/
site-packages (from scikit-learn>=0.24->imbalanced-learn) (2.1.0)
WARNING: Running pip as root will break packages and permissions. You should ins
tall packages reliably by using venv: https://pip.pypa.io/warnings/venv
WARNING: You are using pip version 21.1.1; however, version 21.1.2 is available.
You should consider upgrading via the '/usr/local/bin/python3.7 -m pip install -
-upgrade pip' command.
```

```
Requirement already satisfied: delayed in /usr/local/lib/python3.7/site-packages
(0.11.0b1)
Requirement already satisfied: redis in /usr/local/lib/python3.7/site-packages
 (from delayed) (3.5.3)
Requirement already satisfied: hiredis in /usr/local/lib/python3.7/site-packages
(from delayed) (2.0.0)
WARNING: Running pip as root will break packages and permissions. You should ins
tall packages reliably by using venv: https://pip.pypa.io/warnings/venv
WARNING: You are using pip version 21.1.1; however, version 21.1.2 is available.
You should consider upgrading via the '/usr/local/bin/python3.7 -m pip install -
-upgrade pip' command.
Requirement already satisfied: xgboost in /usr/local/lib/python3.7/site-packages
(1.4.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/site-packages
 (from xgboost) (1.5.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/site-packages
 (from xgboost) (1.18.5)
WARNING: Running pip as root will break packages and permissions. You should ins
tall packages reliably by using venv: https://pip.pypa.io/warnings/venv
WARNING: You are using pip version 21.1.1; however, version 21.1.2 is available.
You should consider upgrading via the '/usr/local/bin/python3.7 -m pip install -
-upgrade pip' command.
```

In [2]:

```
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import numpy as np
import copy
import time
import sys
from datetime import datetime

from sklearn.preprocessing import RobustScaler

from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler

from sklearn.metrics import confusion_matrix, accuracy_score, classification_rep
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV, cross_val_score, KFold, Randon
from sklearn.pipeline import Pipeline

from sklearn.linear_model import SGDClassifier, LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB, MultinomialNB

from sklearn.ensemble import RandomForestClassifier, BaggingClassifier, Gradient
import xgboost as xgb

import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense,BatchNormalization,Dropout
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier

from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor
```

In [3]:

```
# read data
df_train_original = pd.read_csv('./dataset/train_data.csv')
df_test_original = pd.read_csv('./dataset/test_data_hidden.csv')
# df_test_original = pd.read_csv('./dataset/test_data.csv')
```

In [4]:
```
df_train = copy.deepcopy(df_train_original)
df_test = copy.deepcopy(df_test_original)
```

In [5]:
```
print(df_train.shape)
print(df_test.shape)
```

```
(49820, 31)
(51731, 31)
```

In [6]:
```
df_train.head(5)
```

Out[6]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	
0	38355.0	1.043949	0.318555	1.045810	2.805989	-0.561113	-0.367956	0.032736	-0.042333	-0
1	22555.0	-1.665159	0.808440	1.805627	1.903416	-0.821627	0.934790	-0.824802	0.975890	1
2	2431.0	-0.324096	0.601836	0.865329	-2.138000	0.294663	-1.251553	1.072114	-0.334896	1
3	86773.0	-0.258270	1.217501	-0.585348	-0.875347	1.222481	-0.311027	1.073860	-0.161408	0
4	127202.0	2.142162	-0.494988	-1.936511	-0.818288	-0.025213	-1.027245	-0.151627	-0.305750	-0

5 rows × 31 columns

In [7]:
```
df_test.head(5)
```

Out[7]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	
0	113050.0	0.114697	0.796303	-0.149553	-0.823011	0.878763	-0.553152	0.939259	-0.108502	(
1	26667.0	-0.039318	0.495784	-0.810884	0.546693	1.986257	4.386342	-1.344891	-1.743736	-0
2	159519.0	2.275706	-1.531508	-1.021969	-1.602152	-1.220329	-0.462376	-1.196485	-0.147058	-0
3	137545.0	1.940137	-0.357671	-1.210551	0.382523	0.050823	-0.171322	-0.109124	-0.002115	0
4	63369.0	1.081395	-0.502615	1.075887	-0.543359	-1.472946	-1.065484	-0.443231	-0.143374	1

5 rows × 31 columns

In [8]:
```
# Rows containing train duplicate data
duplicate_rows_df_train = df_train[df_train.duplicated()]
print("number of duplicate train rows: ", duplicate_rows_df_train.shape)
```

```
number of duplicate train rows:  (45, 31)
```

In [9]:
```
# Rows containing test duplicate data
duplicate_rows_df_train = df_test[df_test.duplicated()]
print("number of duplicate test rows: ", duplicate_rows_df_train.shape)
```

```
number of duplicate test rows:  (35, 31)
```

In [10]:
```
# Remove duplicate entires
df_train = df_train.drop_duplicates()
df_test = df_test.drop_duplicates()
```

In [11]:
```
print(df_train.shape)
print(df_test.shape)
```

```
(49775, 31)
(51696, 31)
```

1.a Check all the latent features and parameters with their mean and standard deviation. Value are close to 0 centered (mean)

In [12]:

```
df_train.describe().T
```

Out[12]:

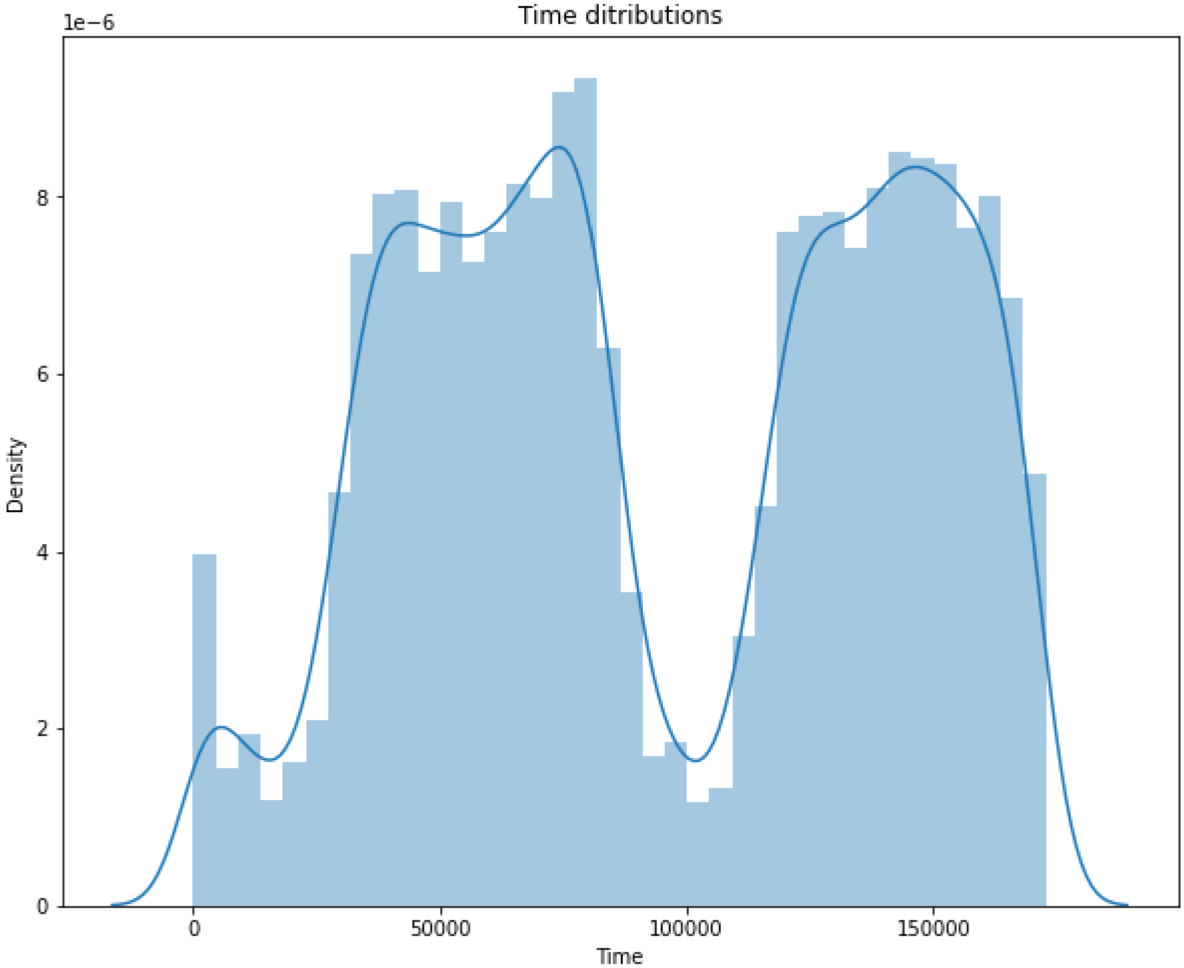
	count	mean	std	min	25%	50%	75%
Time	49775.0	94982.460593	47587.599737	1.000000	54272.500000	84999.000000	139695.500000
V1	49775.0	-0.018686	2.011125	-46.855047	-0.926589	0.003073	1.316281
V2	49775.0	-0.011404	1.709330	-63.344698	-0.606529	0.057337	0.805030
V3	49775.0	-0.005354	1.548371	-31.813586	-0.889742	0.176664	1.032670
V4	49775.0	0.004637	1.432208	-5.416315	-0.857632	-0.017937	0.751129
V5	49775.0	-0.008664	1.382881	-42.147898	-0.701238	-0.062255	0.599577
V6	49775.0	0.006113	1.330202	-20.367836	-0.767659	-0.272083	0.405894
V7	49775.0	-0.008490	1.253202	-43.557242	-0.562308	0.033625	0.564746
V8	49775.0	0.002226	1.249907	-73.216718	-0.207571	0.024914	0.333363
V9	49775.0	0.000042	1.114440	-13.320155	-0.654103	-0.052561	0.603923
V10	49775.0	0.005836	1.113380	-24.588262	-0.534055	-0.086150	0.464591
V11	49775.0	0.002042	1.027091	-4.682931	-0.762247	-0.033749	0.741759
V12	49775.0	-0.010736	1.021204	-18.683715	-0.417415	0.132749	0.611540
V13	49775.0	-0.003147	0.998782	-5.791881	-0.658109	-0.016266	0.655738
V14	49775.0	0.002653	0.984984	-18.493773	-0.419780	0.053819	0.497331
V15	49775.0	-0.000766	0.915765	-4.196620	-0.577175	0.051339	0.649315
V16	49775.0	0.002511	0.892291	-13.256833	-0.463774	0.068545	0.530814
V17	49775.0	-0.003582	0.883525	-22.883999	-0.483061	-0.065740	0.401597
V18	49775.0	0.003795	0.843381	-9.287832	-0.495987	0.005211	0.510037
V19	49775.0	0.002435	0.814511	-7.213527	-0.455023	0.008069	0.460578
V20	49775.0	0.000897	0.803862	-25.222345	-0.211995	-0.061712	0.134693
V21	49775.0	-0.000538	0.770270	-34.830382	-0.227635	-0.027866	0.186964
V22	49775.0	0.002759	0.729673	-10.933144	-0.541486	0.009736	0.532538
V23	49775.0	-0.003600	0.660129	-44.807735	-0.164045	-0.010882	0.149255
V24	49775.0	0.002617	0.602102	-2.790479	-0.351048	0.042591	0.440542
V25	49774.0	-0.002312	0.521638	-10.295397	-0.319345	0.012377	0.347896
V26	49774.0	-0.000412	0.481761	-2.534330	-0.327176	-0.052539	0.241244
V27	49774.0	0.004313	0.395197	-9.543518	-0.071368	0.001224	0.092245
V28	49774.0	0.000452	0.336381	-9.617915	-0.053209	0.011160	0.078886
Amount	49774.0	90.271051	258.760090	0.000000	5.900000	22.000000	78.395000
Class	49774.0	0.001728	0.041531	0.000000	0.000000	0.000000	0.000000

1.b Find if there is any connection between Time, Amount, and the transaction being fraudulent.

In [13]:
```
# visualizations of time
plt.figure(figsize=(10,8))
plt.title("Time ditributions")
sns.distplot(df_train.Time)
```

```
/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2557: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future versio
n. Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[13]:
```
<AxesSubplot:title={'center':'Time ditributions'}, xlabel='Time', ylabel='Densit
y'>
```



In [14]:
```
# The time inseconds need to converts
df_train['Time'] = df_train['Time'] / 3600
# df_train_original['Time'] = df_train_original['Time'] / 24
print(df_train['Time'].min())
print(df_train['Time'].max())
```

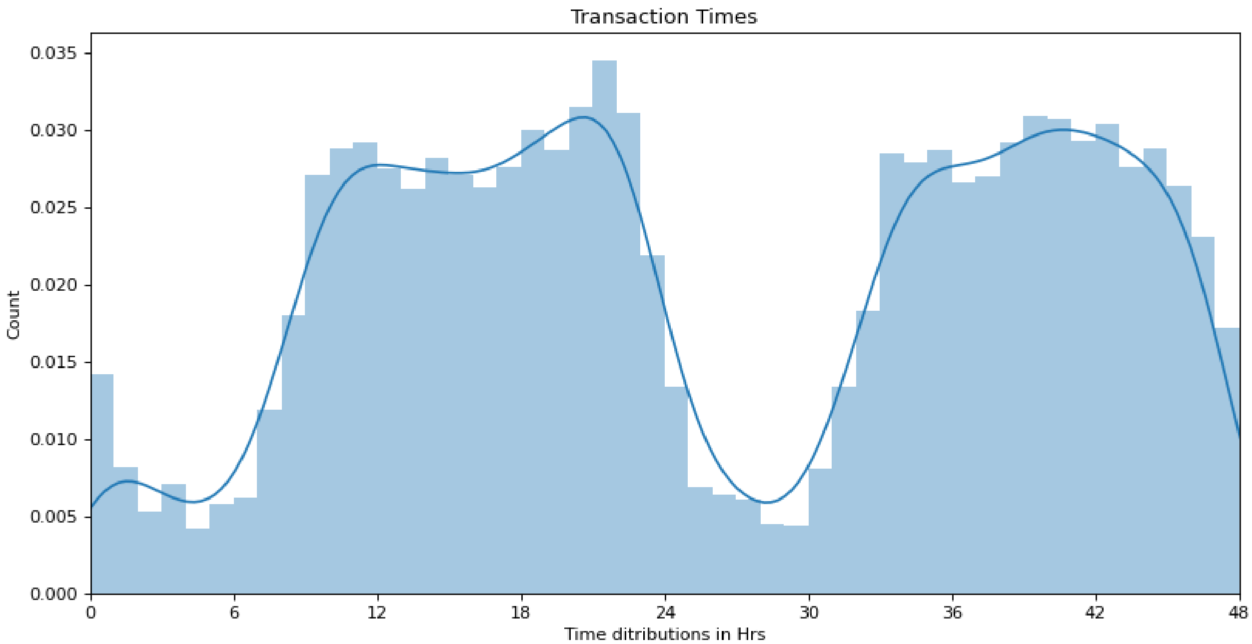
```
0.0002777777777777778
47.99666666666667
```

In [15]:
```
plt.figure(figsize=(12,6), dpi=80)
sns.distplot(df_train.Time, bins=48)
plt.xlim([0,48])
plt.xticks(np.arange(0,54,6))
```

```
plt.xlabel('Time ditributions in Hrs')
plt.ylabel('Count')
plt.title('Transaction Times')
```

```
/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2557: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future versio
n. Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

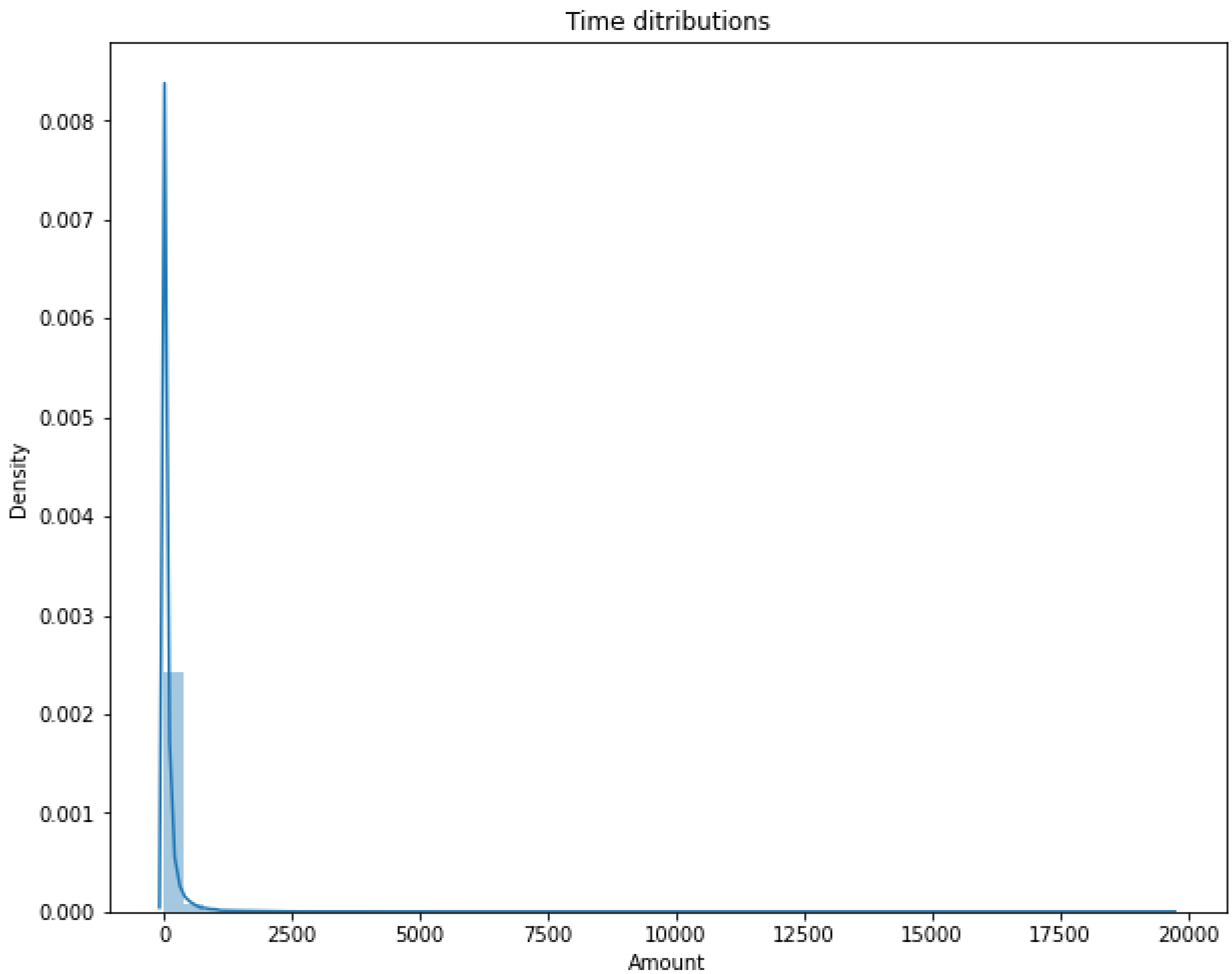
Out[15]: Text(0.5, 1.0, 'Transaction Times')



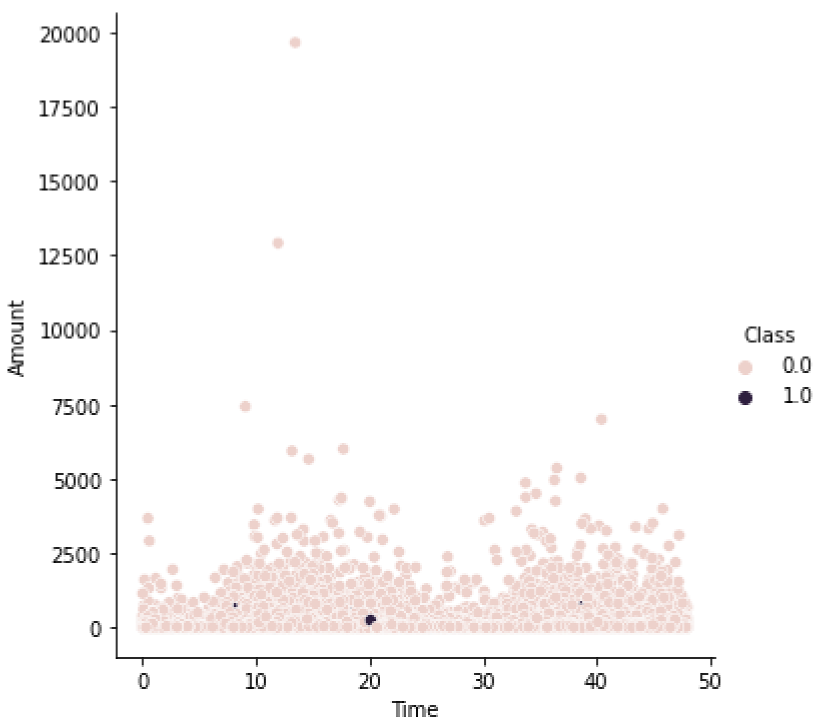
In [16]:
```
# visualizations of amount
plt.figure(figsize=(10,8))
plt.title("Time ditributions")
sns.distplot(df_train.Amount)
```

```
/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2557: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future versio
n. Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

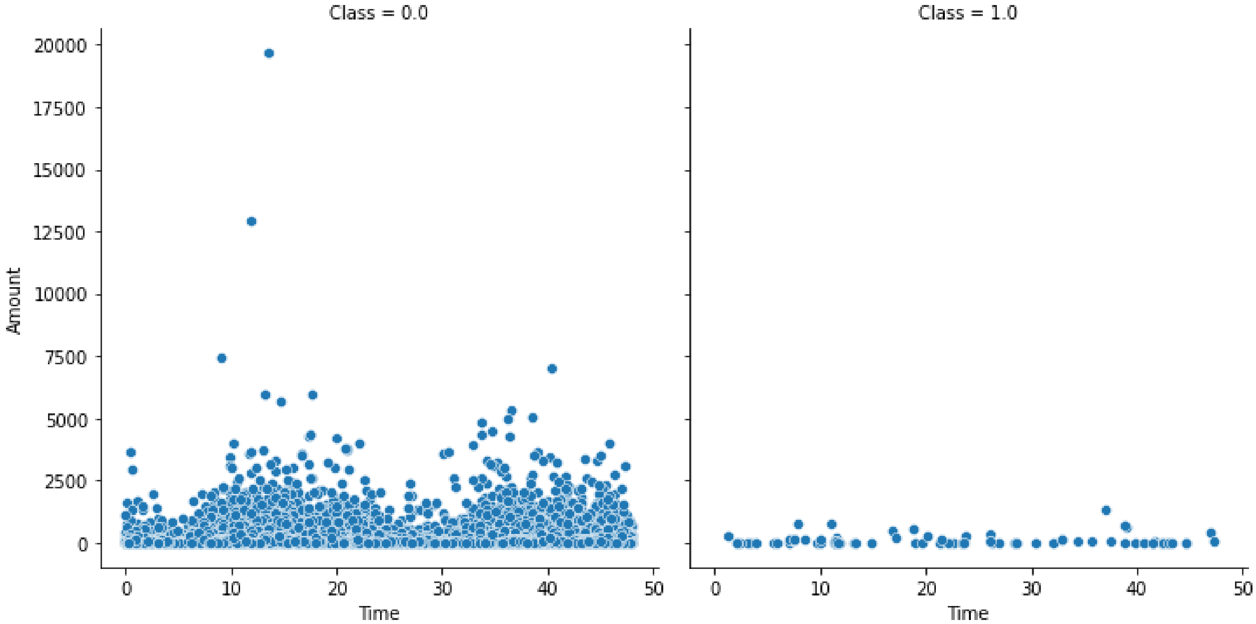
Out[16]: <AxesSubplot:title={'center':'Time ditributions'}, xlabel='Amount', ylabel='Density'>



In [17]:
```
sns.relplot(x="Time", y="Amount", hue="Class", data=df_train);
```



In [18]:
```
sns.relplot(x="Time", y="Amount", col="Class", data=df_train);
```



2.Check the class count for each class. It’s a class Imbalance problem.

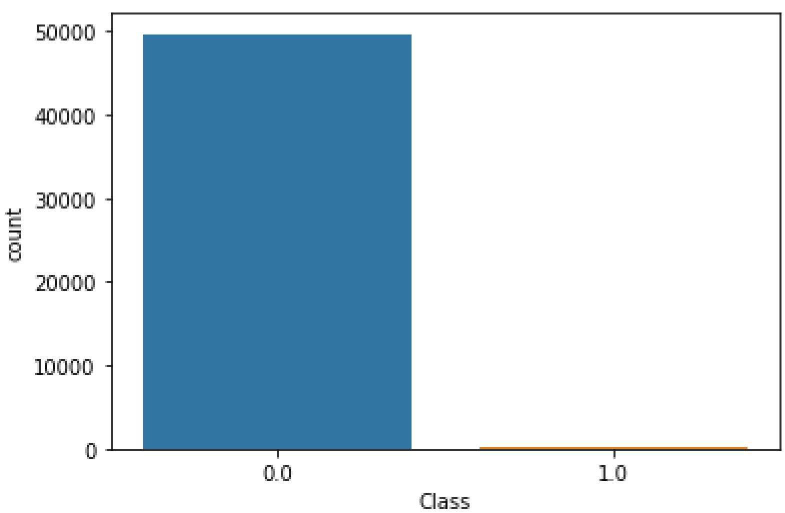
In [19]:
```
df_train['Class'].value_counts()
```

Out[19]:
```
0.0    49688
1.0       86
Name: Class, dtype: int64
```

In [20]:
```
sns.countplot('Class', data=df_train)
```

```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[20]:
```
<AxesSubplot:xlabel='Class', ylabel='count'>
```



3.Use techniques like undersampling or oversampling before running Naïve Bayes, Logistic Regression or SVM.

In [21]:
```
robust_scaler = RobustScaler()
```

```
df_train['scaled_amount'] = robust_scaler.fit_transform(df_train['Amount'].value
df_train['scaled_time'] = robust_scaler.fit_transform(df_train['Time'].values.re

df_test['scaled_amount'] = robust_scaler.fit_transform(df_test['Amount'].values.
df_test['scaled_time'] = robust_scaler.fit_transform(df_test['Time'].values.resh
```

In [22]:

```
# Remove the Time,Amount
df_train.drop(['Time','Amount'], axis=1, inplace=True)
df_test.drop(['Time','Amount'], axis=1, inplace=True)
```

In [23]:

```
df_train.dropna(inplace=True)
df_test.dropna(inplace=True)
```

In [24]:

```
y_train = df_train.pop('Class')
X_train = df_train
```

In [25]:

```
print(y_train.shape)
print(X_train.shape)
```

```
(49774,)
(49774, 30)
```

In [26]:

```
y_test = df_test.pop('Class')
X_test = df_test
print(X_test.shape)
print(y_test.shape)
```

```
(51695, 30)
(51695,)
```

In [27]:

```
X_train.describe().T
```

Out[27]:

	count	mean	std	min	25%	50%	75%	max
V1	49774.0	-0.018659	2.011137	-46.855047	-0.926545	0.003188	1.316292	2.406284
V2	49774.0	-0.011408	1.709347	-63.344698	-0.606529	0.057305	0.805040	22.057729
V3	49774.0	-0.005399	1.548354	-31.813586	-0.889746	0.176599	1.032617	4.029428
V4	49774.0	0.004657	1.432215	-5.416315	-0.857526	-0.017884	0.751157	16.715537
V5	49774.0	-0.008641	1.382885	-42.147898	-0.701170	-0.062236	0.599594	29.162172
V6	49774.0	0.006123	1.330214	-20.367836	-0.767667	-0.272063	0.405904	22.529298
V7	49774.0	-0.008480	1.253212	-43.557242	-0.562314	0.033627	0.564765	36.677268
V8	49774.0	0.002214	1.249916	-73.216718	-0.207577	0.024904	0.333330	17.941363
V9	49774.0	0.000063	1.114442	-13.320155	-0.654096	-0.052557	0.603927	10.392889
V10	49774.0	0.005840	1.113391	-24.588262	-0.534072	-0.086148	0.464616	13.727347
V11	49774.0	0.002065	1.027088	-4.682931	-0.762206	-0.033707	0.741792	11.669205
V12	49774.0	-0.010737	1.021214	-18.683715	-0.417434	0.132813	0.611543	7.848392
V13	49774.0	-0.003157	0.998790	-5.791881	-0.658133	-0.016297	0.655754	4.369998
V14	49774.0	0.002660	0.984993	-18.493773	-0.419793	0.053856	0.497335	10.526766
V15	49774.0	-0.000764	0.915774	-4.196620	-0.577186	0.051361	0.649317	5.825654
V16	49774.0	0.002530	0.892291	-13.256833	-0.463755	0.068552	0.530817	6.351853

	count	mean	std	min	25%	50%	75%	max
V17	49774.0	-0.003580	0.883533	-22.883999	-0.483066	-0.065740	0.401621	7.611862
V18	49774.0	0.003769	0.843369	-9.287832	-0.495990	0.005194	0.510022	5.041069
V19	49774.0	0.002464	0.814495	-7.213527	-0.454976	0.008075	0.460588	4.891062
V20	49774.0	0.000903	0.803869	-25.222345	-0.211983	-0.061706	0.134694	39.420904
V21	49774.0	-0.000532	0.770276	-34.830382	-0.227622	-0.027860	0.186965	27.202839
V22	49774.0	0.002771	0.729675	-10.933144	-0.541472	0.009776	0.532545	10.503090
V23	49774.0	-0.003599	0.660136	-44.807735	-0.164049	-0.010877	0.149256	13.944567
V24	49774.0	0.002610	0.602105	-2.790479	-0.351080	0.042588	0.440575	4.022866
V25	49774.0	-0.002312	0.521638	-10.295397	-0.319345	0.012377	0.347896	4.032572
V26	49774.0	-0.000412	0.481761	-2.534330	-0.327176	-0.052539	0.241244	3.463246
V27	49774.0	0.004313	0.395197	-9.543518	-0.071368	0.001224	0.092245	11.135740
V28	49774.0	0.000452	0.336381	-9.617915	-0.053209	0.011160	0.078886	15.870474
scaled_amount	49774.0	0.941735	3.569351	-0.303469	-0.222084	0.000000	0.777916	270.839782
scaled_time	49774.0	0.116883	0.557080	-0.995025	-0.359687	0.000012	0.640305	1.027697

In [28]:

```
# oversampling tarin data
oversample = RandomOverSampler(sampling_strategy='minority')

# fit and apply the transform
X_train_over, y_train_over = oversample.fit_resample(X_train, y_train)

print(X_train_over.shape)
print(y_train_over.shape)
```

```
(99376, 30)
(99376,)
```

In [29]:

```
# undersampling tarin data
undersample = RandomUnderSampler(sampling_strategy='majority')

# fit and apply the transform
X_train_under, y_train_under = undersample.fit_resample(X_train, y_train)

print(X_train_under.shape)
print(y_train_under.shape)
```

```
(172, 30)
(172,)
```

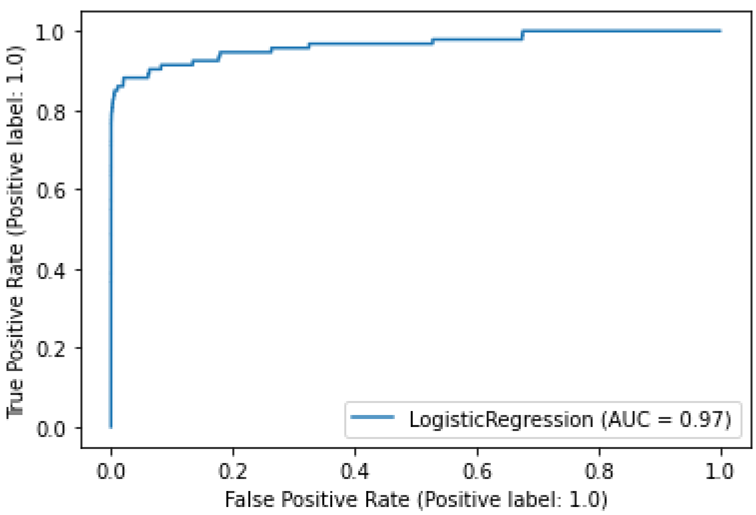
In [30]:

```
def modal_perfomance(model, X_train, Y_train, X_test, Y_test):
    model.fit(X_train, Y_train)
    predicted = model.predict(X_test)
    report = classification_report(Y_test, predicted)
    print("f1_score       : ", f1_score(Y_test, predicted, average="macro"))
    print("precision_score: ", precision_score(Y_test, predicted, average="macro
    print("recall_score   : ", recall_score(Y_test, predicted, average="macro"))
    print("\nAccuracy Score :",accuracy_score(predicted, Y_test))
    print(report)
    plot_roc_curve(model, X_test, Y_test)
```

```
    plt.show()
#     roc_curve_plt(model, X_test, Y_test, predicted)
```

In [31]:
```
lr = LogisticRegression(class_weight ='balanced')
modal_perfomance(lr, X_train_over, y_train_over, X_test, y_test)
```

```
/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:765: Co
nvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressio
n
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
f1_score       :  0.5428535152028675
precision_score:  0.5264447353448544
recall_score   :  0.9266940974614221

Accuracy Score : 0.9715059483509043
              precision    recall  f1-score   support

         0.0       1.00      0.97      0.99     51602
         1.0       0.05      0.88      0.10        93

    accuracy                           0.97     51695
   macro avg       0.53      0.93      0.54     51695
weighted avg       1.00      0.97      0.98     51695
```



Observations: Week 1

- Traing set contains two days transaction details, the rate transaction low at night times
- The amount of transcation is highy skewed distribution, only low number count detected for high amount transcation
- There is no connection between Time, Amount, and the transaction being fraudulent
- All the fraudulent transactions are low amount
- This is highly class Imbalance problem, fraudulent transaction are very rare less than 0.17 %

Project Task: Week 2

Modeling Techniques:

- Try out models like Naive Bayes, Logistic Regression or SVM. Find out which one performs the best
- Use different Tree-based classifiers like Random Forest and XGBoost.
  ```
   a.    Remember Tree-based classifiers work on two ideologies: Bagging
  or Boosting
   b.    Tree-based classifiers have fine-tuning parameters which takes
  care of the imbalanced class. Random-Forest and XGBboost.
  ```
- Compare the results of 1 with 2 and check if there is any incremental gain.

In [32]:
```
df_train = copy.deepcopy(df_train_original)
df_test = copy.deepcopy(df_test_original)

df_train.drop(['Time','Amount'], axis=1, inplace=True)
df_test.drop(['Time','Amount'], axis=1, inplace=True)
df_train.dropna(inplace=True)
df_test.dropna(inplace=True)

y_train = df_train.pop('Class')
X_train = df_train
y_test = df_test.pop('Class')
X_test = df_test
```

In [33]:
```
# Create logistic regression pipeline
lr_pipeline = Pipeline(
    steps=[
        ('scaler', StandardScaler()),
        ('lr', LogisticRegression(n_jobs = -1, verbose=2))
    ]
)
```

In [34]:
```
# Create SVC pipeline
svm_pipeline = Pipeline(
    steps=[
        ('scaler', StandardScaler()),
        ('svm', SVC(verbose=2))
    ]
)
```

In [35]:
```
# Create Gaussian NB pipeline
gnb_pipeline = Pipeline(
    steps=[
        ('scaler', StandardScaler()),
        ('gnb', GaussianNB())
    ]
)
```

In [36]:
```
pipelines = [
    lr_pipeline,
    svm_pipeline,
    gnb_pipeline
]
```

In [37]:
```
pipeline_dict = {
    0 : 'LogisticRegression',
    1 : 'SVM',
```

```
    2 : 'Gaussian NB'
}

for pipe in pipelines:
    pipe.fit(X_train, y_train)
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done   1 out of   1 | elapsed:    1.6s finished
[LibSVM]
```

In [38]:

```
# finding best modal
for i,modal in enumerate(pipelines):
    print("{} Test accuracy {}".format(pipeline_dict[i], modal.score(X_test, y_t
```

```
LogisticRegression Test accuracy 0.9989367871641214
SVM Test accuracy 0.9990721051614151
Gaussian NB Test accuracy 0.9799922675430118
```

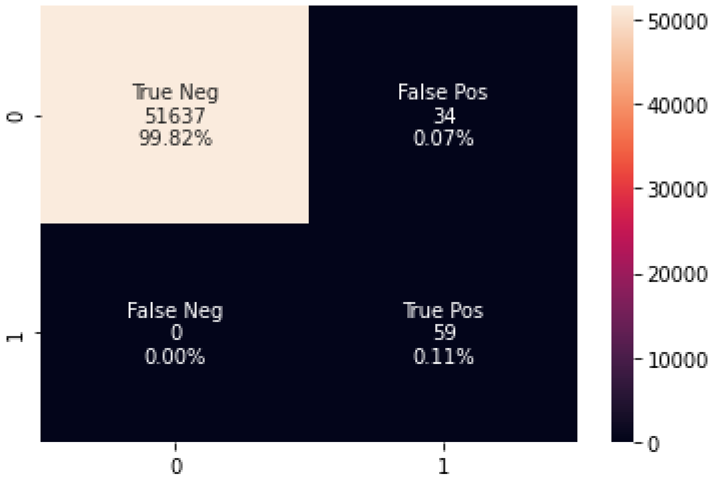
In [39]:

```
def tree_perfomace_tracker(modal, X_test, y_test ):
    predict = modal.predict(X_test)
    cf_matrix = confusion_matrix(predict, y_test)

    group_names = ['True Neg','False Pos','False Neg','True Pos']
    group_counts = ["{0:0.0f}".format(value) for value in cf_matrix.flatten()]
    group_percentages = ["{0:.2%}".format(value) for value in cf_matrix.flatten(
    labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in zip(group_names,group_counts
    labels = np.asarray(labels).reshape(2,2)
    sns.heatmap(cf_matrix, annot=labels, fmt='')

    print("Accuracy Score :",accuracy_score(predict, y_test))
    print("f1_score       : ", f1_score(predict, y_test))
```

In [40]:

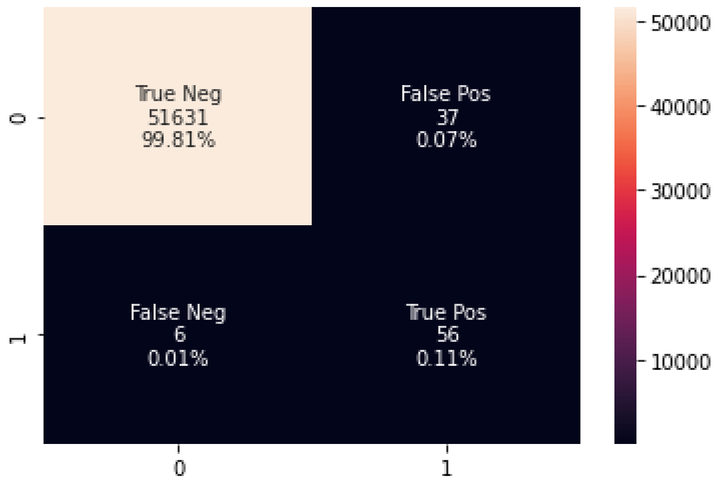
```
# RAndom Forest
rf = RandomForestClassifier(class_weight='balanced_subsample', random_state=42,
rf.fit(X_train, y_train)
tree_perfomace_tracker(rf, X_test, y_test)
```

```
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    2.0s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:    4.5s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    0.1s finished
Accuracy Score : 0.9993427411560023
f1_score       :  0.7763157894736843
```



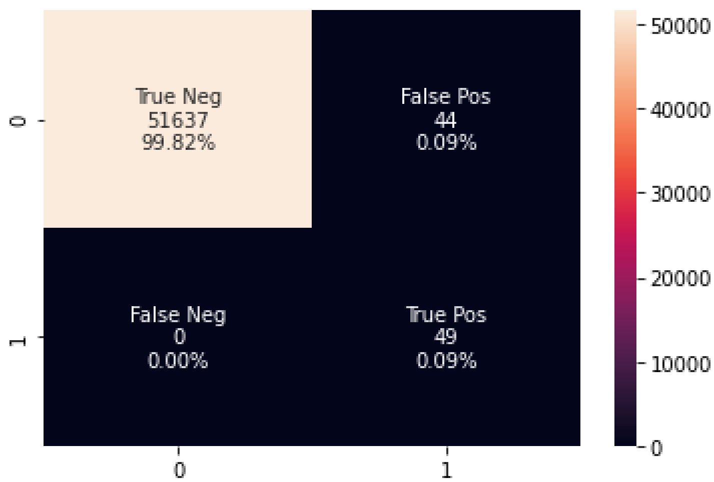
In [41]:
```
# Xg Boost
xb = xgb.XGBClassifier(learning_rate=0.001, n_jobs=-1)
xb.fit(X_train, y_train, eval_metric='logloss', verbose=True)
tree_perfomace_tracker(xb, X_test, y_test)
```

```
/usr/local/lib/python3.7/site-packages/xgboost/sklearn.py:1146: UserWarning: The
use of label encoder in XGBClassifier is deprecated and will be removed in a fut
ure release. To remove this warning, do the following: 1) Pass option use_label_
encoder=False when constructing XGBClassifier object; and 2) Encode your labels
(y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
Accuracy Score : 0.9991687608737676
f1_score       :  0.7225806451612904
```



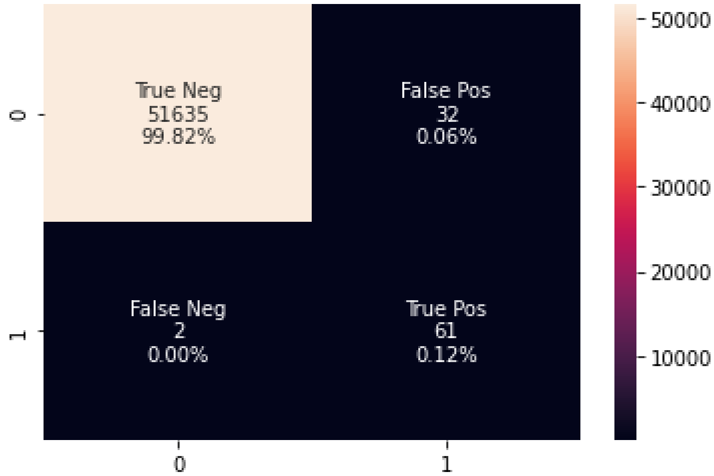
In [42]:
```
# Bagging Classifire
BC_clf = BaggingClassifier(base_estimator=SVC(), random_state=0, n_jobs=-1)
BC_clf.fit(X_train, y_train)
tree_perfomace_tracker(BC_clf, X_test, y_test)
```

```
Accuracy Score : 0.9991494297312972
f1_score       :  0.6901408450704225
```



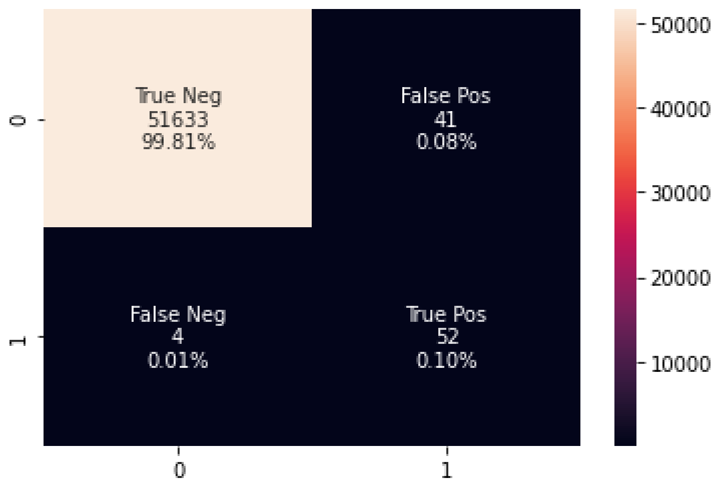
In [43]:
```
# Bagging Classifire
BC_RFC = BaggingClassifier(base_estimator=RandomForestClassifier(class_weight='b
BC_RFC.fit(X_train, y_train)
tree_perfomace_tracker(BC_RFC, X_test, y_test)
```

```
Accuracy Score : 0.9993427411560023
f1_score       :  0.782051282051282
```



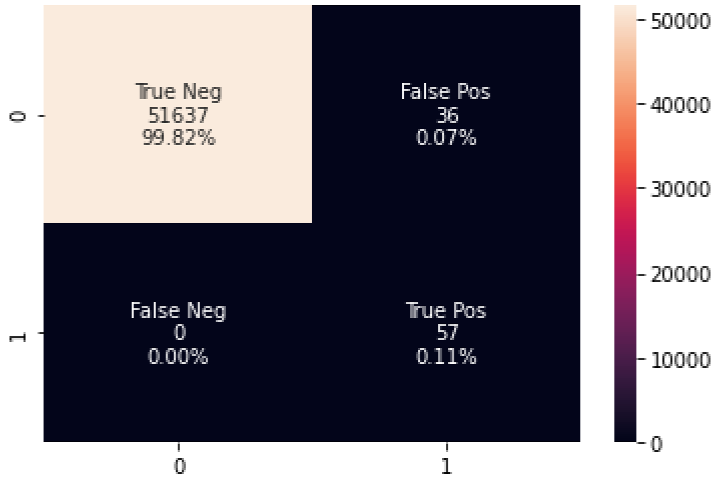
In [44]:
```
# Gradient Classifire
GBC = GradientBoostingClassifier(random_state=0, loss='deviance', learning_rate=
GBC.fit(X_train, y_train)
tree_perfomace_tracker(GBC,  X_test, y_test)
```

```
Accuracy Score : 0.9991300985888266
f1_score       :  0.6979865771812079
```



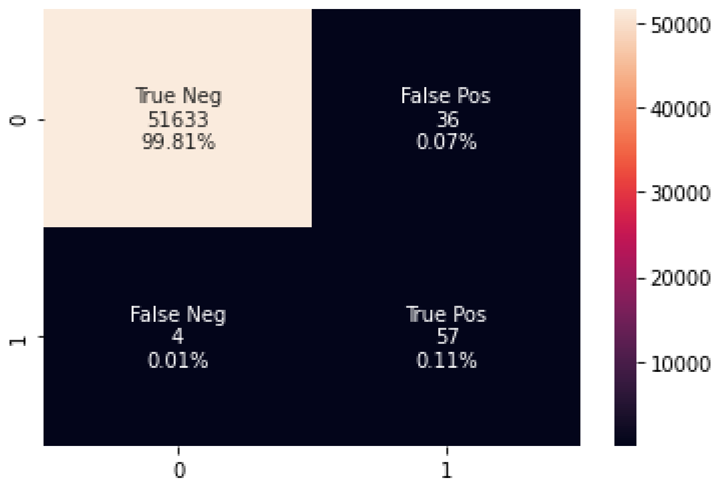
In [45]:
```
# Extra Classifire
ETC = ExtraTreesClassifier(n_estimators=100, random_state=0, class_weight='balar
ETC.fit(X_train, y_train)
tree_perfomace_tracker(ETC, X_test, y_test)
```

```
Accuracy Score : 0.9993040788710613
f1_score       :  0.76
```



In [46]:
```
# Ada boost Classifire
ADC = AdaBoostClassifier(n_estimators=100, random_state=0)
ADC.fit(X_train, y_train)
tree_perfomace_tracker(ADC, X_test, y_test)
```

```
Accuracy Score : 0.9992267543011792
f1_score       :  0.7402597402597403
```



Observations: Week 2

- Compare Logistic, SVM the essembles method had more incremental gain.
- RandomForestClassifier Bagging classifire outperfome compare to other modals
- Computation wise EtraTreeClassifire is the better one and a little downwards

Project Task: Week 3

Applying ANN:

- Use ANN (Artificial Neural Network) to identify fradulent and non-fradulent.
```
 a)    Fine-tune number of layers
 b)    Number of Neurons in each layers
 c)    Experiment in batch-size
 d)    Experiment with number of epochs. Check the observations in
loss and accuracy
 e)    Play with different Learning Rate variants of Gradient Descent
```

```
like Adam, SGD, RMS-prop
 f)    Find out which activation performs best for this use case and
why?
 g)    Check Confusion Matrix, Precision, Recall and F1-Score
```

- Try out Dropout for ANN. How is it performed? Compare model performance with the traditional ML based prediction models from above.
- Find the best setting of neural net that can be best classified as fraudulent and non-fraudulent transactions. Use techniques like Grid Search, Cross-Validation and Random search.

Anomaly Detection:

- Implement anomaly detection algorithms.

```
  a)    Assume that the data is coming from a single or a combination
of multivariate Gaussian
  b)    Formalize a scoring criterion, which gives a scoring
probability for the given data point whether it belongs to the
multivariate Gaussian or Normal Distribution fitted in a)
```

In [47]:

```
df_train.head(5)
```

Out[47]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9
0	1.043949	0.318555	1.045810	2.805989	-0.561113	-0.367956	0.032736	-0.042333	-0.322674
1	-1.665159	0.808440	1.805627	1.903416	-0.821627	0.934790	-0.824802	0.975890	1.747469
2	-0.324096	0.601836	0.865329	-2.138000	0.294663	-1.251553	1.072114	-0.334896	1.071268
3	-0.258270	1.217501	-0.585348	-0.875347	1.222481	-0.311027	1.073860	-0.161408	0.200665
4	2.142162	-0.494988	-1.936511	-0.818288	-0.025213	-1.027245	-0.151627	-0.305750	-0.869482

5 rows × 28 columns

In [48]:

```
def model_building(optimizer='Adam', learn_rate = 0.01, dropout_rate = 0.2):
    model = Sequential()
    model.add(Dense(input_dim = 28, units = 256, activation = 'relu'))
    model.add(BatchNormalization())
    model.add(Dense(units =  512 , activation = 'relu'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(units = 256, activation = 'relu'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(units = 114, activation = 'relu'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(units = 56, activation = 'relu'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(units =1, activation = 'sigmoid'))

    model.compile(optimizer = optimizer, loss='binary_crossentropy', metrics = [
    return model
```

In [49]:

```
model_building().summary()
```

```
[2021-06-21 17:32:56.294 tensorflow-2-3-gpu--ml-g4dn-xlarge-c85184389676cdfa7bdf
06745c9b:187 INFO utils.py:27] RULE_JOB_STOP_SIGNAL_FILENAME: None
[2021-06-21 17:32:56.320 tensorflow-2-3-gpu--ml-g4dn-xlarge-c85184389676cdfa7bdf
06745c9b:187 INFO profiler_config_parser.py:102] Unable to find config at /opt/m
```

```
l/input/config/profilerconfig.json. Profiler is disabled.
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense (Dense)                (None, 256)               7424      
_________________________________________________________________
batch_normalization (BatchNo (None, 256)               1024      
_________________________________________________________________
dense_1 (Dense)              (None, 512)               131584    
_________________________________________________________________
dropout (Dropout)            (None, 512)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328    
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_3 (Dense)              (None, 114)               29298     
_________________________________________________________________
dropout_2 (Dropout)          (None, 114)               0         
_________________________________________________________________
dense_4 (Dense)              (None, 56)                6440      
_________________________________________________________________
dropout_3 (Dropout)          (None, 56)                0         
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 57        
=================================================================
Total params: 307,155
Trainable params: 306,643
Non-trainable params: 512
_________________________________________________________________
```

In [50]:

```
# tf.keras.metrics.AUC(),tf.keras.metrics.Precision(),tf.keras.metrics.Recall()
adam_optimizer = tf.keras.optimizers.Adam(learning_rate=0.01)
model_building().compile(optimizer=adam_optimizer, loss='binary_crossentropy', m
```

In [51]:

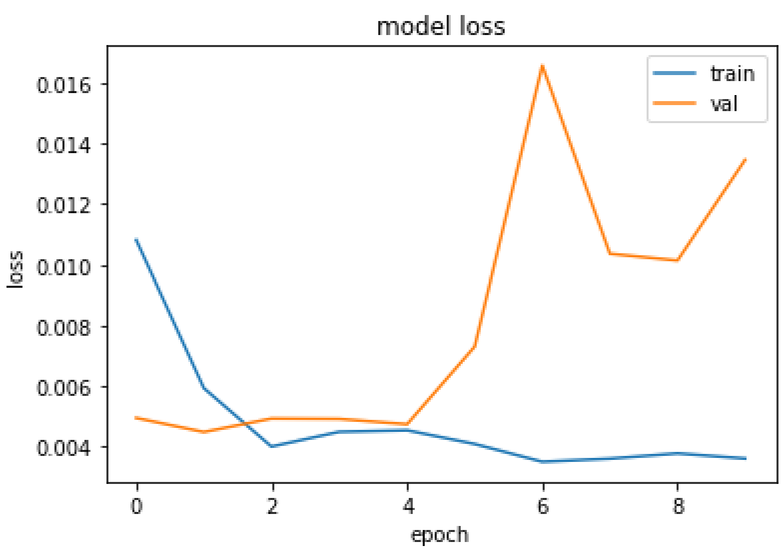
```
# print(os.getcwd())
logdir="logs/fit/"+ datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir)
history = model_building().fit(X_train, y_train, epochs=10, batch_size=45, valid
```

```
Epoch 1/10
1108/1108 [==============================] - 7s 6ms/step - loss: 0.0108 - accura
cy: 0.9981 - auc_2: 0.8776 - val_loss: 0.0050 - val_accuracy: 0.9994 - val_auc_
2: 0.9031
Epoch 2/10
1108/1108 [==============================] - 6s 6ms/step - loss: 0.0059 - accura
cy: 0.9990 - auc_2: 0.9195 - val_loss: 0.0045 - val_accuracy: 0.9994 - val_auc_
2: 0.9188
Epoch 3/10
1108/1108 [==============================] - 6s 6ms/step - loss: 0.0040 - accura
cy: 0.9992 - auc_2: 0.9484 - val_loss: 0.0049 - val_accuracy: 0.9993 - val_auc_
2: 0.9233
Epoch 4/10
1108/1108 [==============================] - 6s 6ms/step - loss: 0.0045 - accura
cy: 0.9991 - auc_2: 0.9370 - val_loss: 0.0049 - val_accuracy: 0.9994 - val_auc_
2: 0.9487
Epoch 5/10
1108/1108 [==============================] - 6s 6ms/step - loss: 0.0045 - accura
cy: 0.9991 - auc_2: 0.9370 - val_loss: 0.0047 - val_accuracy: 0.9984 - val_auc_
2: 0.9349
Epoch 6/10
1108/1108 [==============================] - 6s 6ms/step - loss: 0.0041 - accura
cy: 0.9991 - auc_2: 0.9595 - val_loss: 0.0073 - val_accuracy: 0.9993 - val_auc_
```

```
2: 0.9137
Epoch 7/10
1108/1108 [==============================] - 6s 6ms/step - loss: 0.0035 - accura
cy: 0.9990 - auc_2: 0.9657 - val_loss: 0.0166 - val_accuracy: 0.9993 - val_auc_
2: 0.8870
Epoch 8/10
1108/1108 [==============================] - 6s 6ms/step - loss: 0.0036 - accura
cy: 0.9992 - auc_2: 0.9545 - val_loss: 0.0104 - val_accuracy: 0.9993 - val_auc_
2: 0.9188
Epoch 9/10
1108/1108 [==============================] - 7s 6ms/step - loss: 0.0038 - accura
cy: 0.9991 - auc_2: 0.9600 - val_loss: 0.0101 - val_accuracy: 0.9993 - val_auc_
2: 0.9135
Epoch 10/10
1108/1108 [==============================] - 6s 6ms/step - loss: 0.0036 - accura
cy: 0.9992 - auc_2: 0.9714 - val_loss: 0.0135 - val_accuracy: 0.9994 - val_auc_
2: 0.9083
```

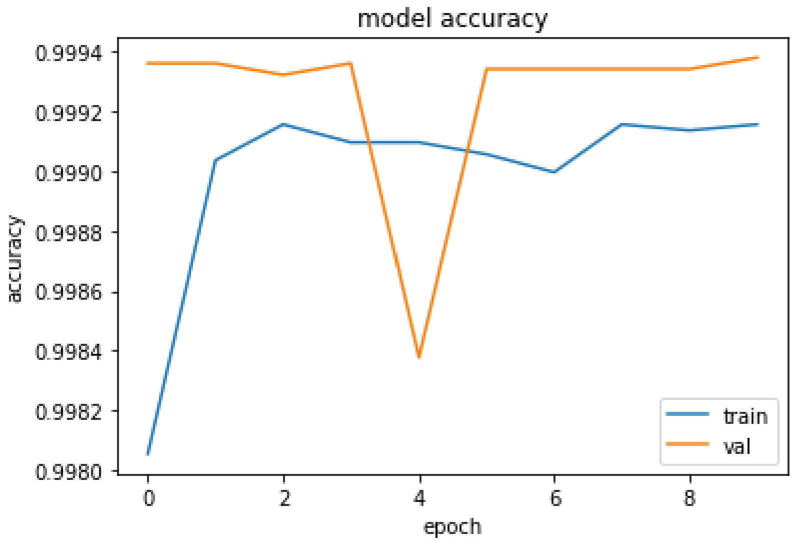
In [52]:

```
# Experiment with number of epochs. Check the observations in loss and accuracy
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper right')
plt.show()
```



In [53]:

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='lower right')
plt.show()
```



In [58]:
```
ann_modal = KerasClassifier(build_fn=model_building, epochs=30)
batch_size = [10, 20, 40, 60, 80, 100]
epochs = [10, 50, 100]
learn_rate = [0.01, 0.02, 0.2]
dropout_rate = [0.0, 0.1, 0.2, 0.4]
param_grid = dict(batch_size=batch_size, epochs=epochs,learn_rate=learn_rate, dr
```

RandamSearch

In [59]:
```
grid = RandomizedSearchCV(estimator=ann_modal, param_distributions =param_grid,
grid_result = grid.fit(X_train, y_train, validation_data=(X_test, y_test))
```

```
/usr/local/lib/python3.7/site-packages/sklearn/model_selection/_search.py:925: U
serWarning: One or more of the test scores are non-finite: [nan nan nan nan nan
nan nan nan nan nan]
  category=UserWarning
Epoch 1/50
1246/1246 [==============================] - 8s 6ms/step - loss: 0.0164 - accura
cy: 0.9971 - auc_5: 0.8042 - val_loss: 0.0053 - val_accuracy: 0.9993 - val_auc_
5: 0.9026
Epoch 2/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0077 - accura
cy: 0.9988 - auc_5: 0.9190 - val_loss: 0.0058 - val_accuracy: 0.9993 - val_auc_
5: 0.8868
Epoch 3/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0064 - accura
cy: 0.9990 - auc_5: 0.9255 - val_loss: 0.0063 - val_accuracy: 0.9994 - val_auc_
5: 0.8869
Epoch 4/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0053 - accura
cy: 0.9991 - auc_5: 0.9309 - val_loss: 0.0046 - val_accuracy: 0.9993 - val_auc_
5: 0.9347
Epoch 5/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0054 - accura
cy: 0.9991 - auc_5: 0.9369 - val_loss: 0.0053 - val_accuracy: 0.9994 - val_auc_
5: 0.9182
Epoch 6/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0049 - accura
cy: 0.9989 - auc_5: 0.9315 - val_loss: 0.0068 - val_accuracy: 0.9992 - val_auc_
5: 0.9241
Epoch 7/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0036 - accura
cy: 0.9989 - auc_5: 0.9654 - val_loss: 0.0105 - val_accuracy: 0.9994 - val_auc_
5: 0.8816
```

```
Epoch 8/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0047 - accura
cy: 0.9991 - auc_5: 0.9599 - val_loss: 0.0091 - val_accuracy: 0.9994 - val_auc_
5: 0.8920
Epoch 9/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0061 - accura
cy: 0.9991 - auc_5: 0.9372 - val_loss: 0.0066 - val_accuracy: 0.9994 - val_auc_
5: 0.9133
Epoch 10/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0045 - accura
cy: 0.9991 - auc_5: 0.9544 - val_loss: 0.0085 - val_accuracy: 0.9991 - val_auc_
5: 0.9232
Epoch 11/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0048 - accura
cy: 0.9992 - auc_5: 0.9597 - val_loss: 0.0094 - val_accuracy: 0.9993 - val_auc_
5: 0.9081
Epoch 12/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0048 - accura
cy: 0.9989 - auc_5: 0.9598 - val_loss: 0.0119 - val_accuracy: 0.9993 - val_auc_
5: 0.9401
Epoch 13/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0035 - accura
cy: 0.9991 - auc_5: 0.9769 - val_loss: 0.0243 - val_accuracy: 0.9994 - val_auc_
5: 0.8870
Epoch 14/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0052 - accura
cy: 0.9989 - auc_5: 0.9430 - val_loss: 0.0211 - val_accuracy: 0.9994 - val_auc_
5: 0.9290
Epoch 15/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0041 - accura
cy: 0.9991 - auc_5: 0.9600 - val_loss: 0.0137 - val_accuracy: 0.9994 - val_auc_
5: 0.9232
Epoch 16/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0045 - accura
cy: 0.9989 - auc_5: 0.9768 - val_loss: 0.0199 - val_accuracy: 0.9994 - val_auc_
5: 0.9350
Epoch 17/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0028 - accura
cy: 0.9991 - auc_5: 0.9772 - val_loss: 0.0241 - val_accuracy: 0.9993 - val_auc_
5: 0.9191
Epoch 18/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0039 - accura
cy: 0.9988 - auc_5: 0.9657 - val_loss: 0.0350 - val_accuracy: 0.9994 - val_auc_
5: 0.8870
Epoch 19/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0058 - accura
cy: 0.9988 - auc_5: 0.9657 - val_loss: 0.0408 - val_accuracy: 0.9987 - val_auc_
5: 0.9031
Epoch 20/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0055 - accura
cy: 0.9991 - auc_5: 0.9544 - val_loss: 0.0191 - val_accuracy: 0.9993 - val_auc_
5: 0.9190
Epoch 21/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0043 - accura
cy: 0.9990 - auc_5: 0.9771 - val_loss: 0.0432 - val_accuracy: 0.9991 - val_auc_
5: 0.8923
Epoch 22/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0047 - accura
cy: 0.9986 - auc_5: 0.9656 - val_loss: 0.0396 - val_accuracy: 0.9989 - val_auc_
5: 0.8975
Epoch 23/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0035 - accura
cy: 0.9989 - auc_5: 0.9770 - val_loss: 0.0305 - val_accuracy: 0.9983 - val_auc_
5: 0.8974
Epoch 24/50
```

```
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0056 - accura
cy: 0.9992 - auc_5: 0.9771 - val_loss: 0.0302 - val_accuracy: 0.9994 - val_auc_
5: 0.9298
Epoch 25/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0061 - accura
cy: 0.9988 - auc_5: 0.9656 - val_loss: 0.0265 - val_accuracy: 0.9994 - val_auc_
5: 0.9243
Epoch 26/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0028 - accura
cy: 0.9991 - auc_5: 0.9884 - val_loss: 0.0407 - val_accuracy: 0.9993 - val_auc_
5: 0.8655
Epoch 27/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0032 - accura
cy: 0.9991 - auc_5: 0.9716 - val_loss: 0.0220 - val_accuracy: 0.9994 - val_auc_
5: 0.9078
Epoch 28/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0031 - accura
cy: 0.9990 - auc_5: 0.9715 - val_loss: 0.0537 - val_accuracy: 0.9993 - val_auc_
5: 0.8494
Epoch 29/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0058 - accura
cy: 0.9989 - auc_5: 0.9714 - val_loss: 0.0670 - val_accuracy: 0.9993 - val_auc_
5: 0.9245
Epoch 30/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0026 - accura
cy: 0.9992 - auc_5: 0.9829 - val_loss: 0.0449 - val_accuracy: 0.9993 - val_auc_
5: 0.9084
Epoch 31/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0039 - accura
cy: 0.9991 - auc_5: 0.9603 - val_loss: 0.0435 - val_accuracy: 0.9993 - val_auc_
5: 0.9031
Epoch 32/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0057 - accura
cy: 0.9992 - auc_5: 0.9772 - val_loss: 0.0522 - val_accuracy: 0.9993 - val_auc_
5: 0.8922
Epoch 33/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0029 - accura
cy: 0.9990 - auc_5: 0.9828 - val_loss: 0.0453 - val_accuracy: 0.9994 - val_auc_
5: 0.8922
Epoch 34/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0027 - accura
cy: 0.9991 - auc_5: 0.9885 - val_loss: 0.0337 - val_accuracy: 0.9993 - val_auc_
5: 0.8976
Epoch 35/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0028 - accura
cy: 0.9989 - auc_5: 0.9940 - val_loss: 0.0558 - val_accuracy: 0.9993 - val_auc_
5: 0.8977
Epoch 36/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0058 - accura
cy: 0.9990 - auc_5: 0.9772 - val_loss: 0.1151 - val_accuracy: 0.9994 - val_auc_
5: 0.9139
Epoch 37/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0032 - accura
cy: 0.9990 - auc_5: 0.9771 - val_loss: 0.0675 - val_accuracy: 0.9993 - val_auc_
5: 0.9189
Epoch 38/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0046 - accura
cy: 0.9989 - auc_5: 0.9826 - val_loss: 0.0709 - val_accuracy: 0.9994 - val_auc_
5: 0.8976
Epoch 39/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0026 - accura
cy: 0.9992 - auc_5: 0.9829 - val_loss: 0.0702 - val_accuracy: 0.9993 - val_auc_
5: 0.8869
Epoch 40/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0026 - accura
```

```
cy: 0.9990 - auc_5: 0.9885 - val_loss: 0.0687 - val_accuracy: 0.9994 - val_auc_
5: 0.9029
Epoch 41/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0033 - accura
cy: 0.9991 - auc_5: 0.9885 - val_loss: 0.1063 - val_accuracy: 0.9993 - val_auc_
5: 0.8923
Epoch 42/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0029 - accura
cy: 0.9991 - auc_5: 0.9885 - val_loss: 0.1604 - val_accuracy: 0.9991 - val_auc_
5: 0.8923
Epoch 43/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0095 - accura
cy: 0.9989 - auc_5: 0.9771 - val_loss: 0.0586 - val_accuracy: 0.9993 - val_auc_
5: 0.9191
Epoch 44/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0034 - accura
cy: 0.9991 - auc_5: 0.9827 - val_loss: 0.0545 - val_accuracy: 0.9994 - val_auc_
5: 0.8976
Epoch 45/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0041 - accura
cy: 0.9990 - auc_5: 0.9715 - val_loss: 0.0582 - val_accuracy: 0.9994 - val_auc_
5: 0.9139
Epoch 46/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0027 - accura
cy: 0.9989 - auc_5: 0.9772 - val_loss: 0.0875 - val_accuracy: 0.9993 - val_auc_
5: 0.9137
Epoch 47/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0031 - accura
cy: 0.9991 - auc_5: 0.9772 - val_loss: 0.0615 - val_accuracy: 0.9994 - val_auc_
5: 0.9085
Epoch 48/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0033 - accura
cy: 0.9991 - auc_5: 0.9773 - val_loss: 0.1010 - val_accuracy: 0.9994 - val_auc_
5: 0.8977
Epoch 49/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0022 - accura
cy: 0.9990 - auc_5: 0.9885 - val_loss: 0.1205 - val_accuracy: 0.9991 - val_auc_
5: 0.9031
Epoch 50/50
1246/1246 [==============================] - 7s 6ms/step - loss: 0.0063 - accura
cy: 0.9987 - auc_5: 0.9658 - val_loss: 0.1527 - val_accuracy: 0.9982 - val_auc_
5: 0.9136
```

In [60]:

```
print("\nBest: %f using %s" % (grid_result.best_score_, grid_result.best_params_
```

```
Best: nan using {'learn_rate': 0.2, 'epochs': 50, 'dropout_rate': 0.4, 'batch_si
ze': 40}
```

Cross-Validation

In [61]:

```
scores = cross_val_score(ann_modal, X_train, y_train, cv=5, scoring='f1')
print("f1 score is: ", scores.mean())
```

```
Epoch 1/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0108 - accura
cy: 0.9977 - auc_6: 0.8787
Epoch 2/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0055 - accura
cy: 0.9990 - auc_6: 0.9162
Epoch 3/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0061 - accura
cy: 0.9991 - auc_6: 0.9022
Epoch 4/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0050 - accura
cy: 0.9990 - auc_6: 0.9230
```

```
Epoch 5/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0048 - accura
cy: 0.9992 - auc_6: 0.9438
Epoch 6/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0060 - accura
cy: 0.9989 - auc_6: 0.9439
Epoch 7/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0057 - accura
cy: 0.9991 - auc_6: 0.9442
Epoch 8/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0047 - accura
cy: 0.9991 - auc_6: 0.9307
Epoch 9/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0038 - accura
cy: 0.9991 - auc_6: 0.9515
Epoch 10/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0039 - accura
cy: 0.9990 - auc_6: 0.9650
Epoch 11/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0048 - accura
cy: 0.9992 - auc_6: 0.9443
Epoch 12/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0022 - accura
cy: 0.9994 - auc_6: 0.9792
Epoch 13/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0045 - accura
cy: 0.9990 - auc_6: 0.9651
Epoch 14/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0040 - accura
cy: 0.9991 - auc_6: 0.9924
Epoch 15/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0035 - accura
cy: 0.9991 - auc_6: 0.9722
Epoch 16/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0032 - accura
cy: 0.9992 - auc_6: 0.9653
Epoch 17/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0037 - accura
cy: 0.9993 - auc_6: 0.9721
Epoch 18/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0041 - accura
cy: 0.9991 - auc_6: 0.9652
Epoch 19/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0030 - accura
cy: 0.9991 - auc_6: 0.9722
Epoch 20/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0032 - accura
cy: 0.9990 - auc_6: 0.9791
Epoch 21/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0027 - accura
cy: 0.9995 - auc_6: 0.9792
Epoch 22/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0055 - accura
cy: 0.9993 - auc_6: 0.9517
Epoch 23/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0031 - accura
cy: 0.9991 - auc_6: 0.9721
Epoch 24/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0037 - accura
cy: 0.9993 - auc_6: 0.9928
Epoch 25/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0029 - accura
cy: 0.9992 - auc_6: 0.9791
Epoch 26/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0033 - accura
```

```
cy: 0.9992 - auc_6: 0.9790
Epoch 27/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0034 - accura
cy: 0.9991 - auc_6: 0.9722
Epoch 28/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0029 - accura
cy: 0.9992 - auc_6: 0.9860
Epoch 29/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0020 - accura
cy: 0.9991 - auc_6: 0.9861
Epoch 30/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0033 - accura
cy: 0.9993 - auc_6: 0.9724
WARNING:tensorflow:From /usr/local/lib/python3.7/site-packages/tensorflow/pytho
n/keras/wrappers/scikit_learn.py:241: Sequential.predict_classes (from tensorflo
w.python.keras.engine.sequential) is deprecated and will be removed after 2021-0
1-01.
Instructions for updating:
Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model doe
s multi-class classification   (e.g. if it uses a `softmax` last-layer activatio
n).* `(model.predict(x) > 0.5).astype("int32")`,   if your model does binary cla
ssification   (e.g. if it uses a `sigmoid` last-layer activation).
```



```
WARNING:tensorflow:From /usr/local/lib/python3.7/site-packages/tensorflow/pytho
n/keras/wrappers/scikit_learn.py:241: Sequential.predict_classes (from tensorflo
w.python.keras.engine.sequential) is deprecated and will be removed after 2021-0
1-01.
Instructions for updating:
Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model doe
s multi-class classification   (e.g. if it uses a `softmax` last-layer activatio
n).* `(model.predict(x) > 0.5).astype("int32")`,   if your model does binary cla
ssification   (e.g. if it uses a `sigmoid` last-layer activation).
```



```
Epoch 1/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0105 - accura
cy: 0.9985 - auc_7: 0.8555
Epoch 2/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0052 - accura
cy: 0.9990 - auc_7: 0.9104
Epoch 3/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0052 - accura
cy: 0.9989 - auc_7: 0.9251
Epoch 4/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0048 - accura
cy: 0.9988 - auc_7: 0.9182
Epoch 5/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0042 - accura
cy: 0.9989 - auc_7: 0.9698
Epoch 6/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0059 - accura
cy: 0.9990 - auc_7: 0.9476
Epoch 7/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0036 - accura
cy: 0.9990 - auc_7: 0.9627
Epoch 8/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0034 - accura
cy: 0.9990 - auc_7: 0.9774
Epoch 9/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0036 - accura
cy: 0.9990 - auc_7: 0.9701
Epoch 10/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0046 - accura
cy: 0.9991 - auc_7: 0.9480
Epoch 11/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0037 - accura
cy: 0.9991 - auc_7: 0.9702
Epoch 12/30
```

```
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0061 - accura
cy: 0.9990 - auc_7: 0.9477
Epoch 13/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0043 - accura
cy: 0.9991 - auc_7: 0.9627
Epoch 14/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0042 - accura
cy: 0.9989 - auc_7: 0.9701
Epoch 15/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0030 - accura
cy: 0.9989 - auc_7: 0.9702
Epoch 16/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0027 - accura
cy: 0.9992 - auc_7: 0.9849
Epoch 17/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0051 - accura
cy: 0.9990 - auc_7: 0.9552
Epoch 18/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0041 - accura
cy: 0.9991 - auc_7: 0.9774
Epoch 19/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0022 - accura
cy: 0.9991 - auc_7: 0.9923
Epoch 20/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0041 - accura
cy: 0.9991 - auc_7: 0.9776
Epoch 21/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0024 - accura
cy: 0.9991 - auc_7: 0.9850
Epoch 22/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0037 - accura
cy: 0.9990 - auc_7: 0.9628
Epoch 23/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0045 - accura
cy: 0.9990 - auc_7: 0.9850
Epoch 24/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0030 - accura
cy: 0.9991 - auc_7: 0.9850
Epoch 25/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0031 - accura
cy: 0.9988 - auc_7: 0.9848
Epoch 26/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0058 - accura
cy: 0.9986 - auc_7: 0.9478
Epoch 27/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0046 - accura
cy: 0.9989 - auc_7: 0.9849
Epoch 28/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0028 - accura
cy: 0.9991 - auc_7: 0.9776
Epoch 29/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0019 - accura
cy: 0.9990 - auc_7: 0.9997
Epoch 30/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0016 - accura
cy: 0.9990 - auc_7: 0.9997
Epoch 1/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0131 - accura
cy: 0.9984 - auc_8: 0.8487
Epoch 2/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0072 - accura
cy: 0.9989 - auc_8: 0.9102
Epoch 3/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0056 - accura
cy: 0.9989 - auc_8: 0.9174
```

```
Epoch 4/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0052 - accura
cy: 0.9990 - auc_8: 0.9250
Epoch 5/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0036 - accura
cy: 0.9992 - auc_8: 0.9550
Epoch 6/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0064 - accura
cy: 0.9989 - auc_8: 0.9328
Epoch 7/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0044 - accura
cy: 0.9989 - auc_8: 0.9547
Epoch 8/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0034 - accura
cy: 0.9989 - auc_8: 0.9625
Epoch 9/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0045 - accura
cy: 0.9990 - auc_8: 0.9475
Epoch 10/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0028 - accura
cy: 0.9989 - auc_8: 0.9774
Epoch 11/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0044 - accura
cy: 0.9988 - auc_8: 0.9623
Epoch 12/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0064 - accura
cy: 0.9989 - auc_8: 0.9476
Epoch 13/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0035 - accura
cy: 0.9988 - auc_8: 0.9699
Epoch 14/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0039 - accura
cy: 0.9990 - auc_8: 0.9774
Epoch 15/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0033 - accura
cy: 0.9990 - auc_8: 0.9775
Epoch 16/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0038 - accura
cy: 0.9988 - auc_8: 0.9699
Epoch 17/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0025 - accura
cy: 0.9991 - auc_8: 0.9776
Epoch 18/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0036 - accura
cy: 0.9990 - auc_8: 0.9775
Epoch 19/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0034 - accura
cy: 0.9992 - auc_8: 0.9701
Epoch 20/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0027 - accura
cy: 0.9990 - auc_8: 0.9850
Epoch 21/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0064 - accura
cy: 0.9986 - auc_8: 0.9549
Epoch 22/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0035 - accura
cy: 0.9992 - auc_8: 0.9850
Epoch 23/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0022 - accura
cy: 0.9991 - auc_8: 0.9850
Epoch 24/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0032 - accura
cy: 0.9991 - auc_8: 0.9777
Epoch 25/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0066 - accura
```

```
cy: 0.9988 - auc_8: 0.9626
Epoch 26/30
1246/1246 [==============================] - 4s 4ms/step - loss: 0.0042 - accura
cy: 0.9990 - auc_8: 0.9776
Epoch 27/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0035 - accura
cy: 0.9989 - auc_8: 0.9700
Epoch 28/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0064 - accura
cy: 0.9988 - auc_8: 0.9847
Epoch 29/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0020 - accura
cy: 0.9990 - auc_8: 0.9923
Epoch 30/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0024 - accura
cy: 0.9992 - auc_8: 0.9924
Epoch 1/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0108 - accura
cy: 0.9984 - auc_9: 0.8741
Epoch 2/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0078 - accura
cy: 0.9989 - auc_9: 0.9164
Epoch 3/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0057 - accura
cy: 0.9991 - auc_9: 0.9310
Epoch 4/30
1246/1246 [==============================] - 4s 4ms/step - loss: 0.0063 - accura
cy: 0.9991 - auc_9: 0.9171
Epoch 5/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0041 - accura
cy: 0.9993 - auc_9: 0.9586
Epoch 6/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0045 - accura
cy: 0.9991 - auc_9: 0.9517
Epoch 7/30
1246/1246 [==============================] - 4s 4ms/step - loss: 0.0040 - accura
cy: 0.9991 - auc_9: 0.9651
Epoch 8/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0039 - accura
cy: 0.9991 - auc_9: 0.9655
Epoch 9/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0041 - accura
cy: 0.9992 - auc_9: 0.9589
Epoch 10/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0035 - accura
cy: 0.9992 - auc_9: 0.9590
Epoch 11/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0050 - accura
cy: 0.9992 - auc_9: 0.9453
Epoch 12/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0034 - accura
cy: 0.9993 - auc_9: 0.9657
Epoch 13/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0034 - accura
cy: 0.9992 - auc_9: 0.9658
Epoch 14/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0037 - accura
cy: 0.9992 - auc_9: 0.9657
Epoch 15/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0028 - accura
cy: 0.9991 - auc_9: 0.9861
Epoch 16/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0025 - accura
cy: 0.9992 - auc_9: 0.9862
Epoch 17/30
```

```
1246/1246 [==============================] - 4s 4ms/step - loss: 0.0042 - accura
cy: 0.9993 - auc_9: 0.9590
Epoch 18/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0032 - accura
cy: 0.9994 - auc_9: 0.9657
Epoch 19/30
1246/1246 [==============================] - 4s 4ms/step - loss: 0.0028 - accura
cy: 0.9993 - auc_9: 0.9862
Epoch 20/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0027 - accura
cy: 0.9993 - auc_9: 0.9794
Epoch 21/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0025 - accura
cy: 0.9992 - auc_9: 0.9930
Epoch 22/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0087 - accura
cy: 0.9993 - auc_9: 0.9794
Epoch 23/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0029 - accura
cy: 0.9991 - auc_9: 0.9929
Epoch 24/30
1246/1246 [==============================] - 4s 4ms/step - loss: 0.0031 - accura
cy: 0.9991 - auc_9: 0.9727
Epoch 25/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0022 - accura
cy: 0.9993 - auc_9: 0.9862
Epoch 26/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0034 - accura
cy: 0.9992 - auc_9: 0.9591
Epoch 27/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0024 - accura
cy: 0.9993 - auc_9: 0.9862
Epoch 28/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0017 - accura
cy: 0.9994 - auc_9: 0.9998
Epoch 29/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0037 - accura
cy: 0.9992 - auc_9: 0.9659
Epoch 30/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0038 - accura
cy: 0.9991 - auc_9: 0.9794
Epoch 1/30
1246/1246 [==============================] - 4s 4ms/step - loss: 0.0098 - accura
cy: 0.9986 - auc_10: 0.8877
Epoch 2/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0066 - accura
cy: 0.9990 - auc_10: 0.9231
Epoch 3/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0052 - accura
cy: 0.9992 - auc_10: 0.9374
Epoch 4/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0048 - accura
cy: 0.9990 - auc_10: 0.9651
Epoch 5/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0040 - accura
cy: 0.9991 - auc_10: 0.9651
Epoch 6/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0048 - accura
cy: 0.9991 - auc_10: 0.9442
Epoch 7/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0030 - accura
cy: 0.9992 - auc_10: 0.9791
Epoch 8/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0036 - accura
cy: 0.9992 - auc_10: 0.9651
```

```
Epoch 9/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0029 - accura
cy: 0.9993 - auc_10: 0.9791
Epoch 10/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0030 - accura
cy: 0.9991 - auc_10: 0.9791
Epoch 11/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0036 - accura
cy: 0.9990 - auc_10: 0.9585
Epoch 12/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0060 - accura
cy: 0.9992 - auc_10: 0.9652
Epoch 13/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0030 - accura
cy: 0.9991 - auc_10: 0.9790
Epoch 14/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0019 - accura
cy: 0.9993 - auc_10: 0.9861
Epoch 15/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0027 - accura
cy: 0.9995 - auc_10: 0.9792
Epoch 16/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0018 - accura
cy: 0.9993 - auc_10: 0.9929
Epoch 17/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0019 - accura
cy: 0.9993 - auc_10: 0.9929
Epoch 18/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0031 - accura
cy: 0.9990 - auc_10: 0.9722
Epoch 19/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0027 - accura
cy: 0.9992 - auc_10: 0.9860
Epoch 20/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0016 - accura
cy: 0.9993 - auc_10: 0.9998
Epoch 21/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0039 - accura
cy: 0.9992 - auc_10: 0.9723
Epoch 22/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0028 - accura
cy: 0.9992 - auc_10: 0.9860
Epoch 23/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0027 - accura
cy: 0.9990 - auc_10: 0.9928
Epoch 24/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0016 - accura
cy: 0.9995 - auc_10: 0.9930
Epoch 25/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0017 - accura
cy: 0.9993 - auc_10: 0.9998
Epoch 26/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0027 - accura
cy: 0.9992 - auc_10: 0.9861
Epoch 27/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0024 - accura
cy: 0.9992 - auc_10: 0.9860
Epoch 28/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0040 - accura
cy: 0.9994 - auc_10: 0.9791
Epoch 29/30
1246/1246 [==============================] - 5s 4ms/step - loss: 0.0025 - accura
cy: 0.9992 - auc_10: 0.9929
Epoch 30/30
1246/1246 [==============================] - 4s 4ms/step - loss: 0.0016 - accura
```

```
cy: 0.9994 - auc_10: 0.9930
f1 score is:  0.7393983391079579
```

In [62]:

```
# Experiment in batch-size
batch_size = [10, 20, 40, 60, 80, 100]
param_grid = dict(batch_size = batch_size)
grid = GridSearchCV(estimator = ann_modal, param_grid = param_grid, n_jobs=-1, c
grid_result = grid.fit(X_train,y_train,validation_data=(X_test, y_test))
print("\nBest: %f using %s" % (grid_result.best_score_, grid_result.best_params_
```

```
/usr/local/lib/python3.7/site-packages/sklearn/model_selection/_search.py:925: U
serWarning: One or more of the test scores are non-finite: [nan nan nan nan nan
nan]
  category=UserWarning
Epoch 1/30
4982/4982 [==============================] - 27s 6ms/step - loss: 0.0147 - accur
acy: 0.9986 - auc_11: 0.8627 - val_loss: 0.0139 - val_accuracy: 0.9982 - val_auc
_11: 0.8118
Epoch 2/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0110 - accur
acy: 0.9988 - auc_11: 0.8913 - val_loss: 0.0125 - val_accuracy: 0.9987 - val_auc
_11: 0.8709
Epoch 3/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0115 - accur
acy: 0.9987 - auc_11: 0.8971 - val_loss: 0.0158 - val_accuracy: 0.9990 - val_auc
_11: 0.8440
Epoch 4/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0097 - accur
acy: 0.9988 - auc_11: 0.8972 - val_loss: 0.0167 - val_accuracy: 0.9993 - val_auc
_11: 0.8655
Epoch 5/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0109 - accur
acy: 0.9986 - auc_11: 0.9028 - val_loss: 0.0314 - val_accuracy: 0.9992 - val_auc
_11: 0.8728
Epoch 6/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0099 - accur
acy: 0.9984 - auc_11: 0.8856 - val_loss: 0.0261 - val_accuracy: 0.9993 - val_auc
_11: 0.8869
Epoch 7/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0101 - accur
acy: 0.9985 - auc_11: 0.9307 - val_loss: 0.0336 - val_accuracy: 0.9982 - val_auc
_11: 0.8869
Epoch 8/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0096 - accur
acy: 0.9983 - auc_11: 0.9082 - val_loss: 0.0441 - val_accuracy: 0.9993 - val_auc
_11: 0.8816
Epoch 9/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0115 - accur
acy: 0.9983 - auc_11: 0.9140 - val_loss: 0.0173 - val_accuracy: 0.9982 - val_auc
_11: 0.8868
Epoch 10/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0106 - accur
acy: 0.9982 - auc_11: 0.9136 - val_loss: 0.0143 - val_accuracy: 0.9982 - val_auc
_11: 0.9414
Epoch 11/30
4982/4982 [==============================] - 26s 5ms/step - loss: 0.0105 - accur
acy: 0.9985 - auc_11: 0.9366 - val_loss: 0.0236 - val_accuracy: 0.9993 - val_auc
_11: 0.9025
Epoch 12/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0071 - accur
acy: 0.9986 - auc_11: 0.9484 - val_loss: 0.0297 - val_accuracy: 0.9982 - val_auc
_11: 0.8760
Epoch 13/30
4982/4982 [==============================] - 26s 5ms/step - loss: 0.0090 - accur
```

```
acy: 0.9983 - auc_11: 0.9309 - val_loss: 0.0347 - val_accuracy: 0.9992 - val_auc
_11: 0.9027
Epoch 14/30
4982/4982 [==============================] - 26s 5ms/step - loss: 0.0083 - accur
acy: 0.9983 - auc_11: 0.9310 - val_loss: 0.0767 - val_accuracy: 0.9982 - val_auc
_11: 0.5753
Epoch 15/30
4982/4982 [==============================] - 26s 5ms/step - loss: 0.0092 - accur
acy: 0.9986 - auc_11: 0.9372 - val_loss: 0.0639 - val_accuracy: 0.9985 - val_auc
_11: 0.8654
Epoch 16/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0107 - accur
acy: 0.9985 - auc_11: 0.9370 - val_loss: 0.0290 - val_accuracy: 0.9989 - val_auc
_11: 0.9124
Epoch 17/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0111 - accur
acy: 0.9987 - auc_11: 0.9600 - val_loss: 0.0313 - val_accuracy: 0.9993 - val_auc
_11: 0.8709
Epoch 18/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0074 - accur
acy: 0.9988 - auc_11: 0.9428 - val_loss: 0.0763 - val_accuracy: 0.9989 - val_auc
_11: 0.8750
Epoch 19/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0090 - accur
acy: 0.9987 - auc_11: 0.9092 - val_loss: 0.1617 - val_accuracy: 0.9984 - val_auc
_11: 0.6984
Epoch 20/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0054 - accur
acy: 0.9989 - auc_11: 0.9430 - val_loss: 0.0303 - val_accuracy: 0.9993 - val_auc
_11: 0.8868
Epoch 21/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0089 - accur
acy: 0.9986 - auc_11: 0.9426 - val_loss: 0.0455 - val_accuracy: 0.9992 - val_auc
_11: 0.8922
Epoch 22/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0069 - accur
acy: 0.9987 - auc_11: 0.9430 - val_loss: 0.0488 - val_accuracy: 0.9987 - val_auc
_11: 0.7952
Epoch 23/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0074 - accur
acy: 0.9987 - auc_11: 0.9258 - val_loss: 0.0967 - val_accuracy: 0.9992 - val_auc
_11: 0.8655
Epoch 24/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0064 - accur
acy: 0.9986 - auc_11: 0.9430 - val_loss: 0.0233 - val_accuracy: 0.9992 - val_auc
_11: 0.8846
Epoch 25/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0070 - accur
acy: 0.9987 - auc_11: 0.9484 - val_loss: 0.0691 - val_accuracy: 0.9992 - val_auc
_11: 0.8544
Epoch 26/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0074 - accur
acy: 0.9985 - auc_11: 0.9317 - val_loss: 0.0196 - val_accuracy: 0.9988 - val_auc
_11: 0.8912
Epoch 27/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0103 - accur
acy: 0.9986 - auc_11: 0.9202 - val_loss: 0.0858 - val_accuracy: 0.9986 - val_auc
_11: 0.7365
Epoch 28/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0070 - accur
acy: 0.9989 - auc_11: 0.9485 - val_loss: 0.0404 - val_accuracy: 0.9989 - val_auc
_11: 0.8751
Epoch 29/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0088 - accur
acy: 0.9985 - auc_11: 0.9541 - val_loss: 0.0748 - val_accuracy: 0.9988 - val_auc
```

```
_11: 0.7352
Epoch 30/30
4982/4982 [==============================] - 27s 5ms/step - loss: 0.0050 - accur
acy: 0.9986 - auc_11: 0.9542 - val_loss: 0.0766 - val_accuracy: 0.9986 - val_auc
_11: 0.7847

Best: nan using {'batch_size': 10}
```

In [63]:

```
#  Play with different Learning Rate variants of Gradient Descent like Adam, SGD
optimizers = ['SGD', 'RMSprop', 'Adam']
param_grid = dict(optimizer=optimizers)
grid = GridSearchCV(estimator = ann_modal, param_grid = param_grid, n_jobs = -1,
grid_result = grid.fit(X_train, y_train, validation_data=(X_test, y_test))
print("Best optimizers With Best score: ",grid_result.best_score_, grid_result.b
```

```
/usr/local/lib/python3.7/site-packages/sklearn/model_selection/_search.py:925: U
serWarning: One or more of the test scores are non-finite: [nan nan nan]
  category=UserWarning
Epoch 1/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0201 - accura
cy: 0.9972 - auc_12: 0.7501 - val_loss: 0.0049 - val_accuracy: 0.9993 - val_auc_
12: 0.8921
Epoch 2/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0053 - accura
cy: 0.9988 - auc_12: 0.9118 - val_loss: 0.0045 - val_accuracy: 0.9993 - val_auc_
12: 0.9083
Epoch 3/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0041 - accura
cy: 0.9991 - auc_12: 0.9303 - val_loss: 0.0043 - val_accuracy: 0.9993 - val_auc_
12: 0.9244
Epoch 4/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0042 - accura
cy: 0.9991 - auc_12: 0.9308 - val_loss: 0.0039 - val_accuracy: 0.9993 - val_auc_
12: 0.9297
Epoch 5/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0040 - accura
cy: 0.9992 - auc_12: 0.9361 - val_loss: 0.0036 - val_accuracy: 0.9993 - val_auc_
12: 0.9350
Epoch 6/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0038 - accura
cy: 0.9992 - auc_12: 0.9255 - val_loss: 0.0035 - val_accuracy: 0.9994 - val_auc_
12: 0.9349
Epoch 7/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0035 - accura
cy: 0.9991 - auc_12: 0.9425 - val_loss: 0.0034 - val_accuracy: 0.9994 - val_auc_
12: 0.9349
Epoch 8/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0039 - accura
cy: 0.9990 - auc_12: 0.9423 - val_loss: 0.0035 - val_accuracy: 0.9994 - val_auc_
12: 0.9350
Epoch 9/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0033 - accura
cy: 0.9992 - auc_12: 0.9424 - val_loss: 0.0036 - val_accuracy: 0.9994 - val_auc_
12: 0.9350
Epoch 10/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0033 - accura
cy: 0.9992 - auc_12: 0.9536 - val_loss: 0.0034 - val_accuracy: 0.9994 - val_auc_
12: 0.9350
Epoch 11/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0033 - accura
cy: 0.9992 - auc_12: 0.9597 - val_loss: 0.0034 - val_accuracy: 0.9994 - val_auc_
12: 0.9349
Epoch 12/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0031 - accura
```

```
cy: 0.9992 - auc_12: 0.9651 - val_loss: 0.0035 - val_accuracy: 0.9994 - val_auc_
12: 0.9350
Epoch 13/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0029 - accura
cy: 0.9993 - auc_12: 0.9653 - val_loss: 0.0034 - val_accuracy: 0.9994 - val_auc_
12: 0.9350
Epoch 14/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0032 - accura
cy: 0.9992 - auc_12: 0.9597 - val_loss: 0.0033 - val_accuracy: 0.9994 - val_auc_
12: 0.9349
Epoch 15/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0029 - accura
cy: 0.9991 - auc_12: 0.9654 - val_loss: 0.0033 - val_accuracy: 0.9994 - val_auc_
12: 0.9350
Epoch 16/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0028 - accura
cy: 0.9993 - auc_12: 0.9543 - val_loss: 0.0034 - val_accuracy: 0.9994 - val_auc_
12: 0.9350
Epoch 17/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0028 - accura
cy: 0.9993 - auc_12: 0.9655 - val_loss: 0.0034 - val_accuracy: 0.9994 - val_auc_
12: 0.9350
Epoch 18/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0025 - accura
cy: 0.9992 - auc_12: 0.9601 - val_loss: 0.0033 - val_accuracy: 0.9994 - val_auc_
12: 0.9349
Epoch 19/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0025 - accura
cy: 0.9993 - auc_12: 0.9713 - val_loss: 0.0034 - val_accuracy: 0.9994 - val_auc_
12: 0.9349
Epoch 20/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0028 - accura
cy: 0.9993 - auc_12: 0.9598 - val_loss: 0.0034 - val_accuracy: 0.9994 - val_auc_
12: 0.9349
Epoch 21/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0023 - accura
cy: 0.9993 - auc_12: 0.9882 - val_loss: 0.0033 - val_accuracy: 0.9994 - val_auc_
12: 0.9350
Epoch 22/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0026 - accura
cy: 0.9992 - auc_12: 0.9656 - val_loss: 0.0034 - val_accuracy: 0.9994 - val_auc_
12: 0.9349
Epoch 23/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0023 - accura
cy: 0.9993 - auc_12: 0.9880 - val_loss: 0.0033 - val_accuracy: 0.9994 - val_auc_
12: 0.9350
Epoch 24/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0023 - accura
cy: 0.9994 - auc_12: 0.9826 - val_loss: 0.0034 - val_accuracy: 0.9994 - val_auc_
12: 0.9350
Epoch 25/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0022 - accura
cy: 0.9993 - auc_12: 0.9882 - val_loss: 0.0033 - val_accuracy: 0.9994 - val_auc_
12: 0.9349
Epoch 26/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0021 - accura
cy: 0.9994 - auc_12: 0.9770 - val_loss: 0.0034 - val_accuracy: 0.9994 - val_auc_
12: 0.9350
Epoch 27/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0023 - accura
cy: 0.9993 - auc_12: 0.9826 - val_loss: 0.0034 - val_accuracy: 0.9994 - val_auc_
12: 0.9349
Epoch 28/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0021 - accura
cy: 0.9993 - auc_12: 0.9882 - val_loss: 0.0032 - val_accuracy: 0.9994 - val_auc_
```

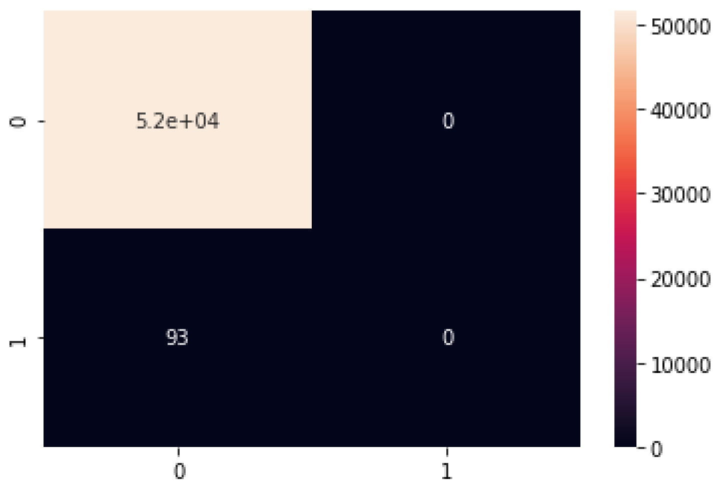
```
12: 0.9350
Epoch 29/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0022 - accura
cy: 0.9994 - auc_12: 0.9770 - val_loss: 0.0032 - val_accuracy: 0.9995 - val_auc_
12: 0.9350
Epoch 30/30
1557/1557 [==============================] - 9s 6ms/step - loss: 0.0022 - accura
cy: 0.9992 - auc_12: 0.9770 - val_loss: 0.0032 - val_accuracy: 0.9994 - val_auc_
12: 0.9350
Best optimizers With Best score:  nan {'optimizer': 'SGD'}
```

Perfomace Evalution

In [64]:

```python
# ann_predictions = model_building().predict(X_test)
ann_predictions = model_building().predict(X_test, verbose=0)
yhat_classes = model_building().predict_classes(X_test, verbose=0)
yhat_probs = ann_predictions[:, 0]
yhat_classes = yhat_classes[:, 0]

accuracy = accuracy_score(y_test, yhat_classes)
f1 = f1_score(y_test, yhat_classes)

cm = tf.math.confusion_matrix(labels=y_test, predictions=ann_predictions, num_cl
sns.heatmap(cm, annot=True)

print("Accuracy Score :", accuracy)
print("f1_score       : ", f1)
```

```
Accuracy Score : 0.8842257877440557
f1_score       :  0.0029965040785749956
```



After Optimization

In [ ]:

```python
df_train = copy.deepcopy(df_train_original)
df_test = copy.deepcopy(df_test_original)

df_train.drop(['Time','Amount'], axis=1, inplace=True)
df_test.drop(['Time','Amount'], axis=1, inplace=True)
df_train.dropna(inplace=True)
df_test.dropna(inplace=True)

y_train = df_train.pop('Class')
X_train = df_train
y_test = df_test.pop('Class')
X_test = df_test
```

```
# oversampling tarin data
oversample = RandomOverSampler(sampling_strategy='minority')

# fit and apply the transform
X_train_over, y_train_over = oversample.fit_resample(X_train, y_train)
```

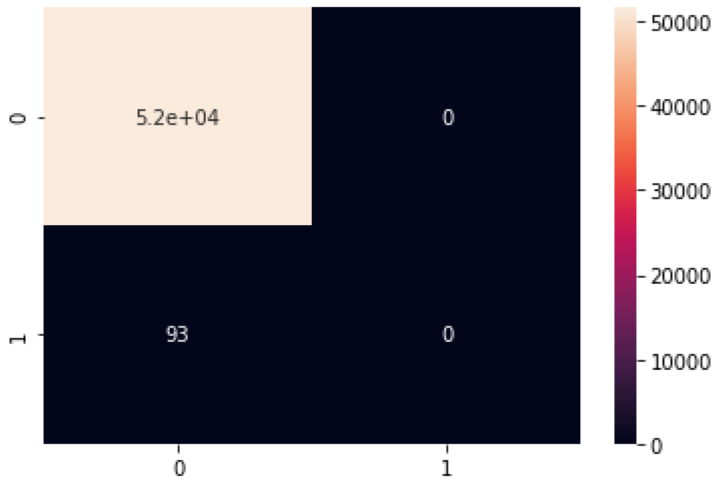
In [77]:
```
model = model_building("SGD", 0.2, 0.4)

logdir="logs/fit/"+ datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir)
history = model.fit(X_train, y_train, epochs=10, batch_size=10, validation_data=
```

```
Epoch 1/10
4982/4982 [==============================] - 26s 5ms/step - loss: 0.0160 - accur
acy: 0.9976 - auc_22: 0.7330 - val_loss: 0.0046 - val_accuracy: 0.9986 - val_auc
_22: 0.8867
Epoch 2/10
4982/4982 [==============================] - 26s 5ms/step - loss: 0.0062 - accur
acy: 0.9985 - auc_22: 0.9271 - val_loss: 0.0043 - val_accuracy: 0.9993 - val_auc
_22: 0.9135
Epoch 3/10
4982/4982 [==============================] - 26s 5ms/step - loss: 0.0058 - accur
acy: 0.9987 - auc_22: 0.9240 - val_loss: 0.0042 - val_accuracy: 0.9993 - val_auc
_22: 0.9349
Epoch 4/10
4982/4982 [==============================] - 26s 5ms/step - loss: 0.0046 - accur
acy: 0.9989 - auc_22: 0.9357 - val_loss: 0.0049 - val_accuracy: 0.9993 - val_auc
_22: 0.9137
Epoch 5/10
4982/4982 [==============================] - 26s 5ms/step - loss: 0.0037 - accur
acy: 0.9989 - auc_22: 0.9536 - val_loss: 0.0050 - val_accuracy: 0.9993 - val_auc
_22: 0.8923
Epoch 6/10
4982/4982 [==============================] - 26s 5ms/step - loss: 0.0038 - accur
acy: 0.9990 - auc_22: 0.9591 - val_loss: 0.0049 - val_accuracy: 0.9993 - val_auc
_22: 0.8977
Epoch 7/10
4982/4982 [==============================] - 26s 5ms/step - loss: 0.0036 - accur
acy: 0.9990 - auc_22: 0.9595 - val_loss: 0.0046 - val_accuracy: 0.9993 - val_auc
_22: 0.9298
Epoch 8/10
4982/4982 [==============================] - 26s 5ms/step - loss: 0.0038 - accur
acy: 0.9991 - auc_22: 0.9483 - val_loss: 0.0043 - val_accuracy: 0.9993 - val_auc
_22: 0.9296
Epoch 9/10
4982/4982 [==============================] - 26s 5ms/step - loss: 0.0032 - accur
acy: 0.9991 - auc_22: 0.9708 - val_loss: 0.0043 - val_accuracy: 0.9993 - val_auc
_22: 0.9298
Epoch 10/10
4982/4982 [==============================] - 26s 5ms/step - loss: 0.0035 - accur
acy: 0.9992 - auc_22: 0.9594 - val_loss: 0.0044 - val_accuracy: 0.9993 - val_auc
_22: 0.9190
```

In [78]:
```
# ann_predictions = model_building().predict(X_test)
ann_predictions = model.predict(X_test, verbose=0)
yhat_classes = model.predict_classes(X_test, verbose=0)
yhat_probs = ann_predictions[:, 0]
yhat_classes = yhat_classes[:, 0]

accuracy = accuracy_score(y_test, yhat_classes)
f1 = f1_score(y_test, yhat_classes)

cm = tf.math.confusion_matrix(labels=y_test, predictions=ann_predictions, num_cl
```

```
sns.heatmap(cm, annot=True)

print("Accuracy Score :", accuracy)
print("f1_score       : ", f1)
```

```
Accuracy Score : 0.9993427411560023
f1_score       :  0.8068181818181818
```



Anomaly Detection:

In [65]:
```
df_train = copy.deepcopy(df_train_original)
df_test = copy.deepcopy(df_test_original)

fraud = df_train[df_train['Class'] == 1]
valid = df_train[df_train['Class'] == 0]
outlier_fraction = len(fraud) / float(len(valid))
n_outliers = len(fraud)

df_train.dropna(inplace=True)
df_test.dropna(inplace=True)

y_train = df_train.pop('Class')
X_train = df_train
```

In [66]:
```
classifiers = {
    "Isolation Forest" : IsolationForest(
        n_estimators = 100,
        max_samples = len(X_train),
        contamination = outlier_fraction,
        random_state = 1,
        verbose = 1,
        n_jobs = -1
    ),
    "Local Outlier Factor" : LocalOutlierFactor(
        n_neighbors=20,
        algorithm='auto',
        leaf_size=30,
        metric='minkowski',
        p=2, metric_params = None,
        contamination = outlier_fraction,
        n_jobs = -1
    )
}
```

In [67]:
```
for i, (clf_name,clf) in enumerate(classifiers.items()):
```

```
    #Fit the data and tag outliers
    if clf_name == "Local Outlier Factor":
        y_pred = clf.fit_predict(X_train)
        scores_prediction = clf.negative_outlier_factor_
    else:
        clf.fit(X_train)
        scores_prediction = clf.decision_function(X_train)
        y_pred = clf.predict(X_train)
    #Reshape the prediction values to 0 for Valid transactions , 1 for Fraud tra
    y_pred[y_pred == 1] = 0
    y_pred[y_pred == -1] = 1
    n_errors = (y_pred != y_train).sum()
    # Run Classification Metrics
    print("{}: {}".format(clf_name, n_errors))
    print("Accuracy Score :")
    print(accuracy_score(y_train, y_pred))
    print("Classification Report :")
    print(classification_report(y_train, y_pred))
```

```
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done   2 out of   4 | elapsed:    0.7s remaining:    0.7s
[Parallel(n_jobs=4)]: Done   4 out of   4 | elapsed:    0.7s finished
Isolation Forest: 113
Accuracy Score :
0.9977317890764568
Classification Report :
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     49730
         1.0       0.37      0.37      0.37        89

    accuracy                           1.00     49819
   macro avg       0.68      0.68      0.68     49819
weighted avg       1.00      1.00      1.00     49819

Local Outlier Factor: 177
Accuracy Score :
0.9964471386418836
Classification Report :
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     49730
         1.0       0.01      0.01      0.01        89

    accuracy                           1.00     49819
   macro avg       0.50      0.50      0.50     49819
weighted avg       1.00      1.00      1.00     49819
```

Observations: Week 3

- After fine tuning this is the best combination of hyperparsm learn_rate: 0.2, epochs: 50, dropout_rate: 0.4, batch_size: 40, optimization: SGD
- ANN oupperfome most, with more than 99% accuracy and 80% of f1 score
- In terms of Anomaly Detection Isolation Forest is better compare to Local Outlier Factor

Project Task: Week 4

- Visualize the scores for Fraudulent and Non-Fraudulent transactions.

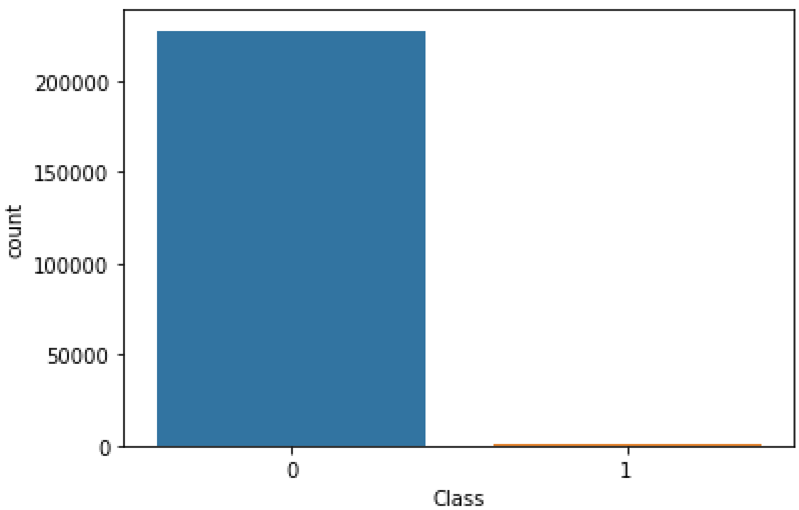
- Find out the threshold value for marking or reporting a transaction as fraudulent in your anomaly detection system.
- Can this score be used as an engineered feature in the models developed previously? Are there any incremental gains in F1-Score? Why or Why not?
- Be as creative as possible in finding other interesting insights.

Visualize the scores for Fraudulent and Non-Fraudulent transactions.

In [2]:
```
sns.countplot('Class', data=df_train_original)
```

```
/home/kalesh/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[2]: <AxesSubplot:xlabel='Class', ylabel='count'>



Find out the threshold value for marking or reporting a transaction as fraudulent in your anomaly detection system.

- the AUC score of ANN traing its more than 91 %,which can use as threshold for Anoamly detection.

Find out the threshold value for marking or reporting a transaction as fraudulent in your anomaly detection system.

- F1 takes both recall and precission, which make more accurate way to messure perfomance of different modals.
- F1 score for neural network model is better than traditional Machine Learning model.

Be as creative as possible in finding other interesting insights.

- The training data contains two days of data, transaction rate day time much more than compare to nights
- The fraudulent transaction had less ammount
- The time and ammount had no correlation in fraudulent transaction