In [1]:

```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
import joblib
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score,confusion_matrix,recall_score
import scikitplot as skplt
```

In [2]:

```
test = pd.read_csv("aps_failure_test_set.csv", na_values='na')
test.head()
```

Out[2]:

	class	aa_000	ab_000	ac_000	ad_000	ae_000	af_000	ag_000	ag_001	ag_002	...	e
0	neg	60	0.0	20.0	12.0	0.0	0.0	0.0	0.0	0.0	...	1
1	neg	82	0.0	68.0	40.0	0.0	0.0	0.0	0.0	0.0	...	1
2	neg	66002	2.0	212.0	112.0	0.0	0.0	0.0	0.0	0.0	...	495
3	neg	59816	NaN	1010.0	936.0	0.0	0.0	0.0	0.0	0.0	...	540
4	neg	1814	NaN	156.0	140.0	0.0	0.0	0.0	0.0	0.0	...	7

5 rows × 171 columns

In [3]:

```
simple_imputer = joblib.load("SimpleImputer.pkl")  # Load the preprossing imputer
model = joblib.load("XGBClassifier.pkl")  # Load the best classifier
important_columns = joblib.load("important_columns.pkl") # Load imp columns
```

Funtion 1

In [4]:

```
def prediction_F1(X):
    X = X.drop(['br_000','bq_000','bp_000','bo_000','ab_000','cr_000','bn_000','bm_000'
],axis=1) # drop the Missing columns
    X = simple_imputer.transform(X[important_columns]) # Load preprocessing models with
imp columns
    X = pd.DataFrame(X,columns=[important_columns])
    return model.predict(X)                                # predict on Test data
```

In [5]:

```
prediction_F1(test[:5])
```

Out[5]:

```
array([0, 0, 0, 0, 0])
```

In [ ]:

```

```

Funtion 2

In [6]:

```
def prediction_F2(X,y):
    y = y.apply(lambda x: 0 if x == 'neg' else 1)  # labeling on Target data
    X = X.drop(['br_000','bq_000','bp_000','bo_000','ab_000','cr_000','bn_000','bm_000'
],axis=1) # drop the missing columns
    X = simple_imputer.transform(X[important_columns])
    X = pd.DataFrame(X,columns=[important_columns])
    pred = model.predict(X)                                 # predict on Test data
    print('-'*40)
    print("Accuracy Score :",accuracy_score(pred,y))  # Calculate the Accuracy Score
    print("Recall Score   :",recall_score(pred,y))    # Calculate the recall Score
    print('-'*40)
    print("Confusion Matrix")
    skplt.metrics.plot_confusion_matrix(pred, y, normalize=False)
    plt.show()
    print('-'*40)
    tn, fp, fn, tp = confusion_matrix(pred,y).ravel()
    print("Cost :",10*fp + 500*fn)                            # cost
    return pred
```

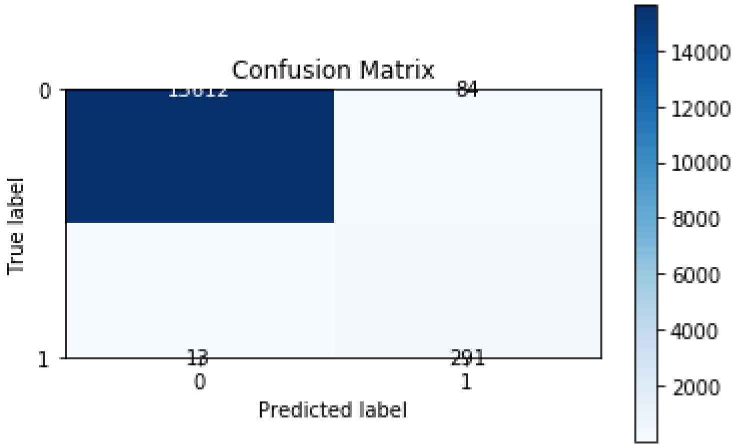
In [7]:

```
y_test = test['class']
x_test = test.drop('class',axis=1)
```

In [8]:

```
pred = prediction_F2(x_test,y_test)
```

```
----------------------------------------
Accuracy Score : 0.9939375
Recall Score   : 0.9572368421052632
----------------------------------------
Confusion Matrix
```



```
----------------------------------------
Cost : 7340
```

In [ ]:

```

```

In [ ]:

```

```