log = LogisticRegression(C=2, max\_iter=100)

raf = RandomForestClassifier(min\_samples\_leaf=10, min\_samples\_split=20,random\_state=100)

#Ensemble Technique

from sklearn.ensemble import VotingClassifier

Ens = VotingClassifier( estimators= [('Log',log),('raf',raf)], voting = 'hard')

Ens= Ens.fit(X\_train , y\_train)

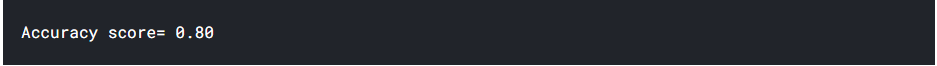
Ens

#Accuracy

y\_pred1 = Ens.predict(X\_test)

en=Ens.score(X\_test, y\_test)

print('Accuracy score= {:.2f}'.format(Ens.score(X\_test, y\_test)))



#Confusion Matrix

from sklearn.metrics import classification\_report, confusion\_matrix

from mlxtend.plotting import plot\_confusion\_matrix

print('\n')

print("confusion matrix")

print('\n')

CR=confusion\_matrix(y\_test, y\_pred1)

print(CR)

print('\n')

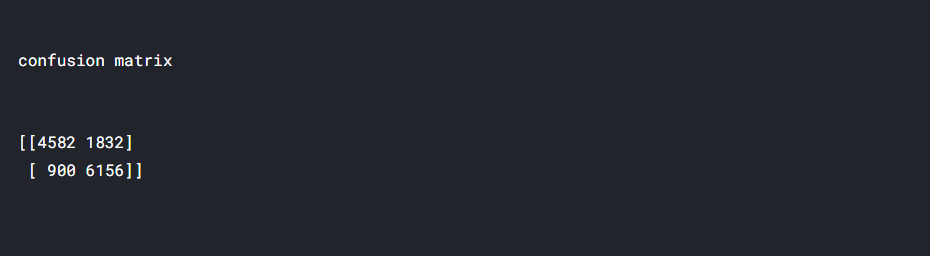
fig, ax = plot\_confusion\_matrix(conf\_mat=CR,figsize=(10, 10),

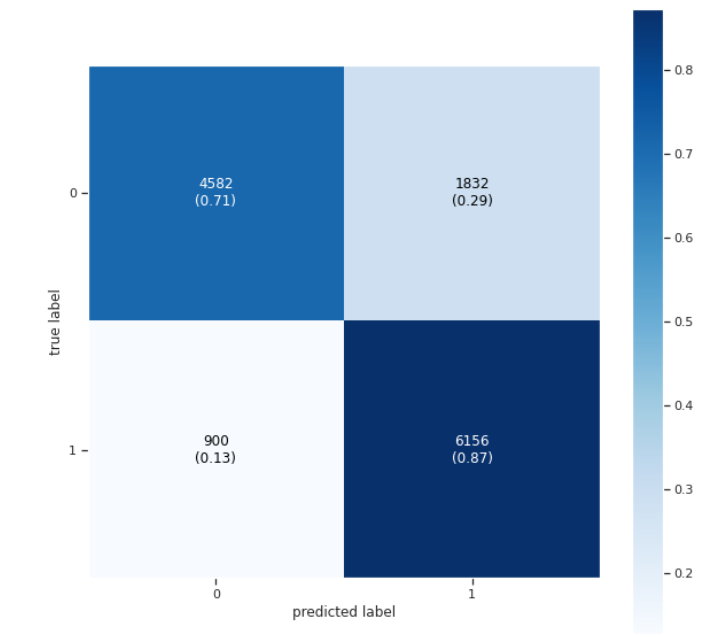
show\_absolute=True,

show\_normed=True,

colorbar=True)

plt.show()





from prettytable import PrettyTable

x = PrettyTable()

print('\n')

print("Comparison of all algorithm results")

x.field\_names = ["Model", "Accuracy"]

x.add\_row(["LogisticRegression Algorithm", round(lr,2)])

x.add\_row(["Random Forest Algorithm", round(rf,2)])

x.add\_row(["Ensemble Modelling", round(en,2)])

print(x)

print('\n')

Comparison of all algorithm results

+------------------------------------+------------------------------- +

| Model | Accuracy |

+------------------------------------+------------------------------- +

| Logistic Regression Algorithm | 0.8 |

| Random Forest Algorithm | 0.79 |

| Ensemble Modelling | 0.8 |

+------------------------------------+------------------------------- +