Performance testing

Date	18-NOV-2022
Project Name	Developing a flight delay prediction using machine learning
Team ID	PNT2022TMID43039
Marks	10marks

1.Metrics:

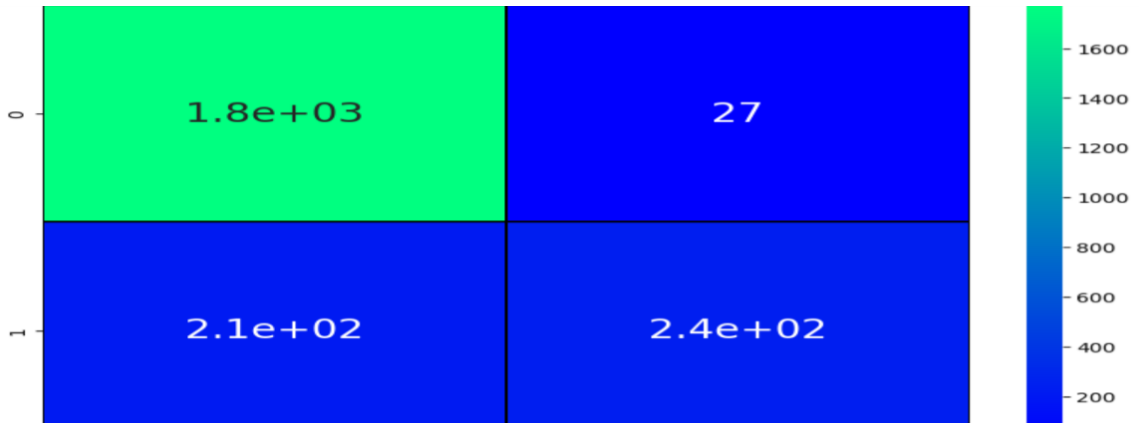
Regression Model:

MAE-, MSE-, RMSE-, R2 Score

RANDOM FOREST:

In [48]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
pred=rf.predict(x_test)
cm=confusion_matrix(y_test, pred)
plt.figure(figsize=(10,6))
sns.heatmap(cm, annot=True,cmap='winter',linewidths=0.3, linecolor='black',annot_kws={"size": 20})
TP=cm[0][0]
TN=cm[1][1]
FN=cm[1][0]
FP=cm[0][1]
#print(round(accuracy_score(prediction3,y_test)*100,2))
#print('Testing Accuracy for knn',(TP+TN)/(TP+TN+FN+FP))
print('Testing Sensitivity for Random Forest',(TP/(TP+FN)))
print('Testing Specificity for Random Forest',(TN/(TN+FP)))
print('Testing Precision for Random Forest',(TP/(TP+FP)))
print('Testing accuracy for Random Forest',accuracy_score(y_test, pred))
```

```
Testing Sensitivity for Random Forest 0.8942065491183879
Testing Specificity for Random Forest 0.8969465648854962
Testing Precision for Random Forest 0.9850166481687015
Testing accuracy for Random Forest 0.8945260347129506
```



Classification Model:

Confusion Matrix-, Accurancy score- & Classification report-

```
In [49]: print(classification_report(y_test,pred))
                  precision    recall  f1-score   support

               0       0.89      0.99      0.94      1802
               1       0.90      0.53      0.66       445

        accuracy                           0.89      2247
       macro avg       0.90      0.76      0.80      2247
    weighted avg       0.89      0.89      0.88      2247
```

2.TUNE THE MODEL:

Hyperparameters tuning-

Validation Method

```
In [58]: #HYPERPARAMETER TUNING
         grid.fit(X_train, y_train)
```

Out[58]:

GridSearchCV

```
GridSearchCV(cv=5,
             estimator=GradientBoostingClassifier(learning_rate=0.7,
                                                  max_depth=4),
             param_grid={'max_features': array([1, 2, 3, 4, 5]),
                         'n_estimators': array([ 10,  20,  30,  40,  50,  60,  70,  80,  90, 100, 110, 120, 130,
       140, 150, 160, 170, 180, 190, 200])})
```

estimator: GradientBoostingClassifier

```
GradientBoostingClassifier(learning_rate=0.7, max_depth=4)
```

GradientBoostingClassifier

```
GradientBoostingClassifier(learning_rate=0.7, max_depth=4)
```

```
In [59]: print("The best parameters are %s with a score of %0.2f"
               % (grid.best_params_, grid.best_score_))

         The best parameters are {'max_features': 5, 'n_estimators': 200} with a score of 0.97
```