In [90]:

```
%matplotlib inline
import pandas as pd
import numpy as np
from sklearn import preprocessing
import matplotlib.pyplot as plt
plt.rc("font", size=14)
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
import seaborn as sns

df = pd.read_csv("./data/bank-additional-full.csv", header=0,nrows =3999)
df = df.dropna()
print(df.shape)
print(list(df.columns))

df.head()
```

```
(3999, 21)
['age', 'job', 'marital', 'education', 'default', 'housing', 'loan', 'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pday
s', 'previous', 'poutcome', 'emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed', 'y']
```

Out[90]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	1.1
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	nonexistent	1.1

5 rows × 21 columns

In [91]:

```
df=df.sample(n=3999)
```

In [92]:

```
df
```

Out[92]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome
631	30	admin.	single	university.degree	no	no	no	telephone	may	tue	...	1	999	0	nonexistent
1805	34	admin.	married	high.school	no	yes	yes	telephone	may	fri	...	1	999	0	nonexistent
44	44	admin.	married	university.degree	unknown	yes	no	telephone	may	mon	...	1	999	0	nonexistent
3230	30	management	married	university.degree	no	no	no	telephone	may	thu	...	2	999	0	nonexistent
1350	50	unemployed	married	professional.course	no	no	no	telephone	may	thu	...	1	999	0	nonexistent
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
458	54	housemaid	married	basic.4y	no	no	no	telephone	may	tue	...	2	999	0	nonexistent
896	33	blue-collar	married	basic.9y	no	yes	no	telephone	may	wed	...	1	999	0	nonexistent
3375	28	student	single	basic.4y	no	no	no	telephone	may	thu	...	8	999	0	nonexistent
329	36	admin.	married	high.school	no	no	no	telephone	may	mon	...	3	999	0	nonexistent
2154	45	blue-collar	married	basic.4y	unknown	yes	no	telephone	may	mon	...	2	999	0	nonexistent

3999 rows × 21 columns

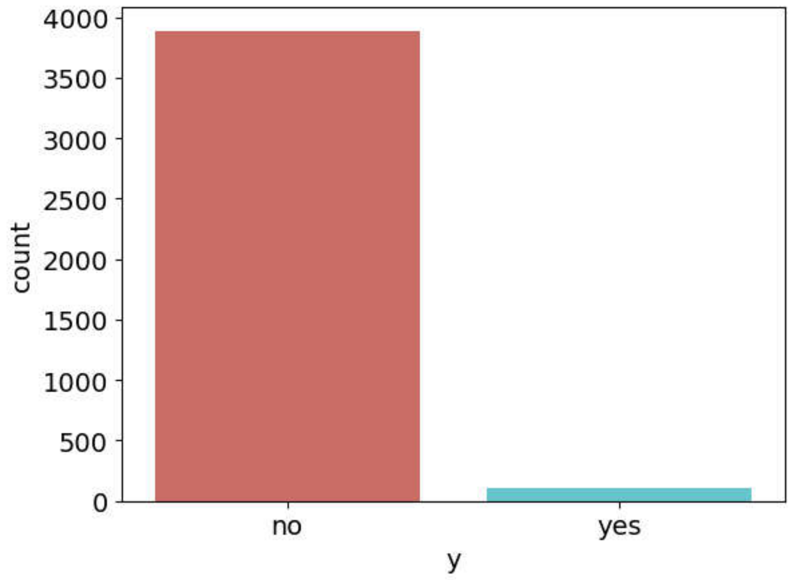
In [93]:

```
print(df['y'].value_counts())
print(df['y'].value_counts()/len(df))
```

```
no     3888
yes     111
Name: y, dtype: int64
no     0.972243
yes    0.027757
Name: y, dtype: float64
```

In [94]:

```
sns.countplot(x='y',data=df, palette='hls')
plt.show()
```



In [95]:

```
df.groupby('y').mean()
```

Out[95]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
y										
no	40.761317	249.978138	2.248457	999.0	0.0	1.1	93.994	-36.4	4.85713	5191.0
yes	40.027027	989.207207	2.108108	999.0	0.0	1.1	93.994	-36.4	4.85745	5191.0

In [96]:

```
df
```

Out[96]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome
631	30	admin.	single	university.degree	no	no	no	telephone	may	tue	...	1	999	0	nonexistent
1805	34	admin.	married	high.school	no	yes	yes	telephone	may	fri	...	1	999	0	nonexistent
44	44	admin.	married	university.degree	unknown	yes	no	telephone	may	mon	...	1	999	0	nonexistent
3230	30	management	married	university.degree	no	no	no	telephone	may	thu	...	2	999	0	nonexistent
1350	50	unemployed	married	professional.course	no	no	no	telephone	may	thu	...	1	999	0	nonexistent
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
458	54	housemaid	married	basic.4y	no	no	no	telephone	may	tue	...	2	999	0	nonexistent
896	33	blue-collar	married	basic.9y	no	yes	no	telephone	may	wed	...	1	999	0	nonexistent
3375	28	student	single	basic.4y	no	no	no	telephone	may	thu	...	8	999	0	nonexistent
329	36	admin.	married	high.school	no	no	no	telephone	may	mon	...	3	999	0	nonexistent
2154	45	blue-collar	married	basic.4y	unknown	yes	no	telephone	may	mon	...	2	999	0	nonexistent

3999 rows × 21 columns

In [97]:

```
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
label = LabelEncoder()
for dataset in [df]:
    dataset['job_Code'] = label.fit_transform(dataset['job'])
    dataset['marital_Code'] = label.fit_transform(dataset['marital'])
    dataset['education_Code'] = label.fit_transform(dataset['education'])
    dataset['default_Code'] = label.fit_transform(dataset['default'])
    dataset['housing_Code'] = label.fit_transform(dataset['housing'])
    dataset['loan_Code'] = label.fit_transform(dataset['loan'])
    dataset['contact_Code'] = label.fit_transform(dataset['contact'])
    dataset['month_Code'] = label.fit_transform(dataset['month'])
    dataset['day_of_week_Code'] = label.fit_transform(dataset['day_of_week'])
```

In [98]:

```
dataset
```

Out[98]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	y	job_Code	marital_Code	education_C
631	30	admin.	single	university.degree	no	no	no	telephone	may	tue	...	no	0	2	
1805	34	admin.	married	high.school	no	yes	yes	telephone	may	fri	...	no	0	1	
44	44	admin.	married	university.degree	unknown	yes	no	telephone	may	mon	...	no	0	1	
3230	30	management	married	university.degree	no	no	no	telephone	may	thu	...	no	4	1	
1350	50	unemployed	married	professional.course	no	no	no	telephone	may	thu	...	no	10	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
458	54	housemaid	married	basic.4y	no	no	no	telephone	may	tue	...	no	3	1	
896	33	blue-collar	married	basic.9y	no	yes	no	telephone	may	wed	...	no	1	1	
3375	28	student	single	basic.4y	no	no	no	telephone	may	thu	...	no	8	2	
329	36	admin.	married	high.school	no	no	no	telephone	may	mon	...	no	0	1	
2154	45	blue-collar	married	basic.4y	unknown	yes	no	telephone	may	mon	...	no	1	1	

3999 rows × 30 columns

In [99]:

```
columns_train_data_x = [
    'age', 'job_Code', 'marital_Code', 'education_Code',
    'default_Code', 'housing_Code', 'loan_Code',
    'contact_Code', 'month_Code', 'day_of_week_Code',
    'duration','campaign','pdays','previous','poutcome','emp.var.rate','cons.price.idx','cons.conf.idx','euribor3m','nr.employed'

]
train_data_x = dataset[columns_train_data_x]
train_data_y = dataset['y']
# train_data_x = pd.get_dummies(train_data_x , columns=columns_train_data_x)
```

In [100]:

```
train_data_y
```

Out[100]:

```
631     no
1805    no
44      no
3230    no
1350    no
        ..
458     no
896     no
3375    no
329     no
2154    no
Name: y, Length: 3999, dtype: object
```

In [101]:

```
train_data_y=train_data_y.map(dict(yes=1, no=0))
```

In [102]:

```
train_data_x
```

Out[102]:

	age	job_Code	marital_Code	education_Code	default_Code	housing_Code	loan_Code	contact_Code	month_Code	day_of_week_Code	duration
631	30	0	2	5	0	0	0	0	0	3	246
1805	34	0	1	3	0	2	2	0	0	0	20
44	44	0	1	5	1	2	0	0	0	1	188
3230	30	4	1	5	0	0	0	0	0	2	221
1350	50	10	1	4	0	0	0	0	0	2	189
...	...	...	...	...	...	...	...	...	...	...	...
458	54	3	1	0	0	0	0	0	0	3	262
896	33	1	1	2	0	2	0	0	0	4	67
3375	28	8	2	0	0	0	0	0	0	2	484
329	36	0	1	3	0	0	0	0	0	1	669
2154	45	1	1	0	1	2	0	0	0	1	170

3999 rows × 20 columns

In [103]:

```
train_data_x=train_data_x.fillna(0)
train_data_x=train_data_x.replace({"poutcome":{'nonexistent':0}})
```

In [ ]:

In [104]:

```
from sklearn.preprocessing import StandardScaler

ss = StandardScaler() ##
#用测试集训练并标准化
ss.fit(train_data_x)
train_data_x = ss.transform(train_data_x)
```

In [105]:

```
train_data_x
```

Out[105]:

```
array([[-1.21441756e+00, -1.03202635e+00,  1.59799302e+00, ...,
        -1.42108547e-14, -8.20892835e-02,  0.00000000e+00],
       [-7.62159901e-01, -1.03202635e+00, -1.73174964e-01, ...,
        -1.42108547e-14, -1.26719775e+00,  0.00000000e+00],
       [ 3.68484252e-01, -1.03202635e+00, -1.73174964e-01, ...,
        -1.42108547e-14, -8.20892835e-02,  0.00000000e+00],
       ...,
       [-1.44054639e+00,  1.25681851e+00,  1.59799302e+00, ...,
        -1.42108547e-14,  1.69557341e+00,  0.00000000e+00],
       [-5.36031070e-01, -1.03202635e+00, -1.73174964e-01, ...,
        -1.42108547e-14, -8.20892835e-02,  0.00000000e+00],
       [ 4.81548667e-01, -7.45920745e-01, -1.73174964e-01, ...,
        -1.42108547e-14, -8.20892835e-02,  0.00000000e+00]])
```

In [ ]:

In [119]:

```
X_train, X_test, y_train, y_test = train_test_split(train_data_x,train_data_y, test_size=0.3, random_state=0)
```

In [121]:

```
#逻辑回归
lg = LogisticRegression(penalty='l2',C=0.01,class_weight='balanced',fit_intercept=True,solver = 'sag',max_iter=300)
lg.fit(X_train,y_train)
```

Out[121]:

```
LogisticRegression
LogisticRegression(C=0.01, class_weight='balanced', max_iter=300, solver='sag')
```

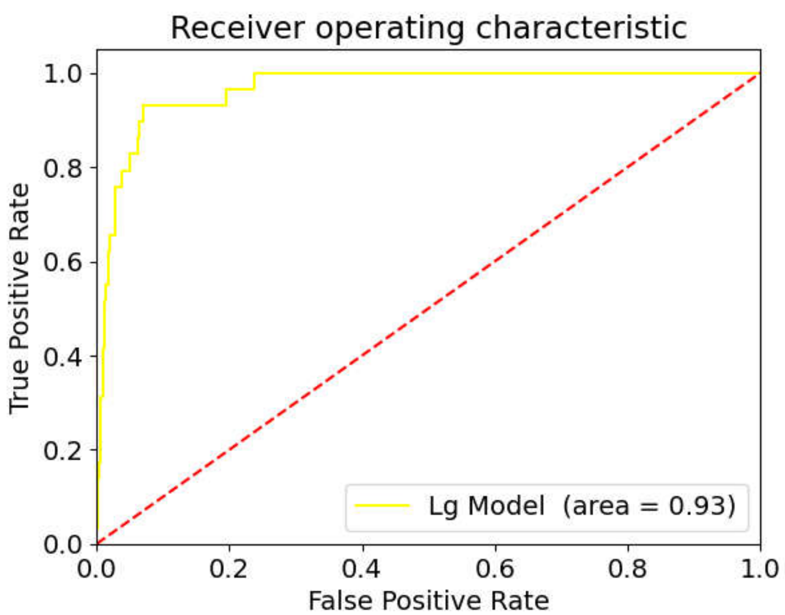
In [122]:

```
# logreg = LogisticRegression(solver='liblinear')
# logreg.fit(X_train, y_train)

# #在测试集上进行预测
# y_pred = logreg.predict(X_test)
# print('在测试集上预测的准确率: {:.2f}'.format(logreg.score(X_test, y_test)))
```

In [124]:

```
print('在测试数据集上面的预测准确率: {:.2f}'.format(lg.score(X_test, y_test)))
print ("\n\n ---逻辑回归---")
rf_roc_auc = roc_auc_score(y_test, lg.predict(X_test))
print ("逻辑回归 AUC = %2.2f" % rf_roc_auc)
print(classification_report(y_test, lg.predict(X_test)))


#绘制Roc曲线观察模型的性能
fprl_gnb, tprl_gnb, thresholdsl_gnb = roc_curve(y_test, lg.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fprl_gnb, tprl_gnb, color = 'yellow',label='Lg Model  (area = %0.2f)' % rf_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Lg_ROC')
plt.show()
```

```
在测试数据集上面的预测准确率: 0.93


 ---逻辑回归---
逻辑回归 AUC = 0.93
              precision    recall  f1-score   support

           0       1.00      0.93      0.96      1171
           1       0.24      0.93      0.38        29

    accuracy                           0.93      1200
   macro avg       0.62      0.93      0.67      1200
weighted avg       0.98      0.93      0.95      1200
```



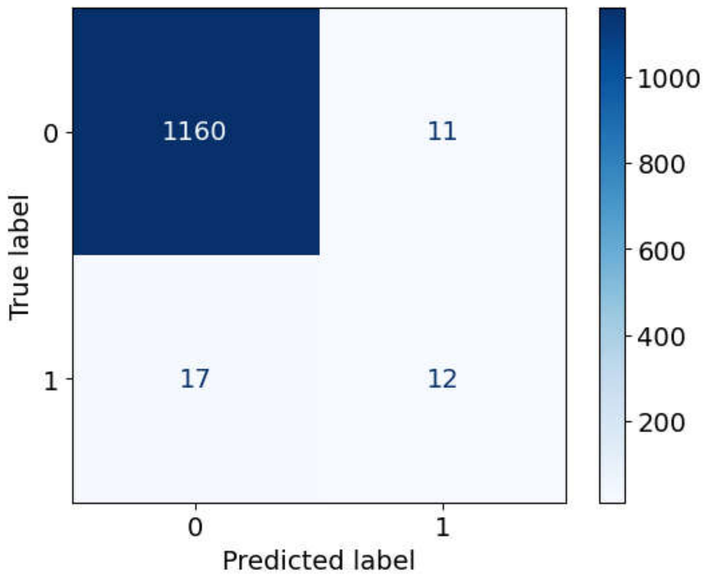
In [ ]:

In [125]:

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay

y_predict = logreg.predict(X_test)
cm = confusion_matrix(y_test, y_predict)
ConfusionMatrixDisplay(cm).plot(cmap='Blues')
```

Out[125]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x161e12e1df0>
```



In [127]:

```
import pickle
pickle.dump(lg,open("case1_lr.pickle.dat","wb"))
```

In [128]:

```
loaded_model=pickle.load(open("case1_lr.pickle.dat","rb"))
```

In [129]:

```
y_pred=loaded_model.predict(X_test)
```

In [ ]:

In [130]:

```
from xgboost.sklearn import XGBClassifier as xgb
```

In [131]:

```
import xgboost as xgb
```

In [132]:

```
#XGBoost训练预测得分
xg_classifier = xgb.XGBClassifier(objective ='binary:logistic', colsample_bytree = 0.3, learning_rate = 0.1,
                max_depth = 6, alpha = 10, n_estimators = 10)

xg_classifier.fit(X_train, y_train)
xg_classifier.score(X_test, y_test)
```

```
[17:14:07] WARNING: C:\Windows\Temp\abs_557yfx631l\croots\recipe\xgboost-split_1659548953302\work\src\learner.cc:1115: Starting in XGBo
ost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly se
t eval_metric if you'd like to restore the old behavior.

D:\Program\anaconda3\envs\graduate\lib\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is
deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False wh
en 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)
```

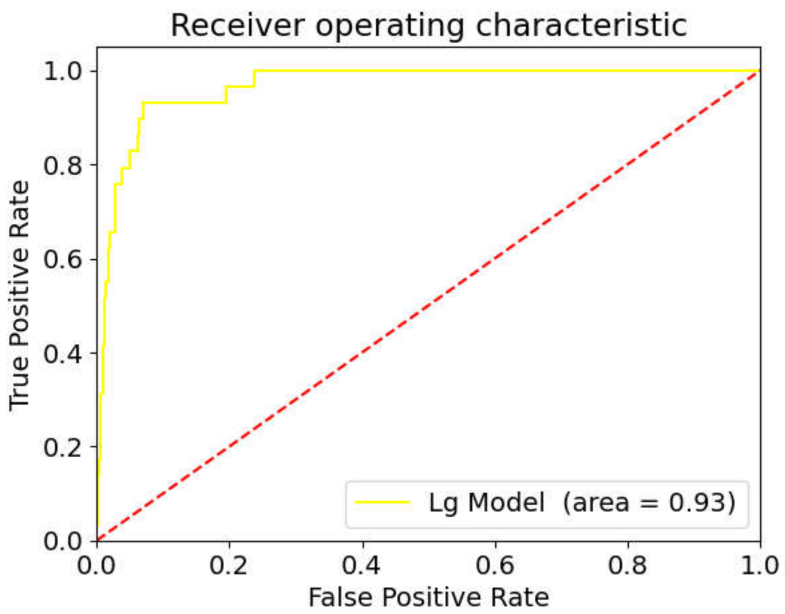
Out[132]:

```
0.9758333333333333
```

In [134]:

```
print('在测试数据集上面的预测准确率: {:.2f}'.format(lg.score(X_test, y_test)))
print ("\n\n ---XGB---")
rf_roc_auc = roc_auc_score(y_test, lg.predict(X_test))
print ("逻辑回归 AUC = %2.2f" % rf_roc_auc)
print(classification_report(y_test, lg.predict(X_test)))


#绘制Roc曲线观察模型的性能
fprl_gnb, tprl_gnb, thresholdsl_gnb = roc_curve(y_test, lg.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fprl_gnb, tprl_gnb, color = 'yellow',label='Lg Model  (area = %0.2f)' % rf_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Lg_ROC')
plt.show()
```

```
在测试数据集上面的预测准确率: 0.93


 ---XGB---
逻辑回归 AUC = 0.93
              precision    recall  f1-score   support

           0       1.00      0.93      0.96      1171
           1       0.24      0.93      0.38        29

    accuracy                           0.93      1200
   macro avg       0.62      0.93      0.67      1200
weighted avg       0.98      0.93      0.95      1200
```



In [185]:

```
#在测试集上进行预测
y_pred = xg_classifier.predict(X_test)
print('在测试集上预测的准确率: {:.2f}'.format(xg_classifier.score(X_test, y_test)))
```

```
在测试集上预测的准确率: 0.97
```

In [186]:

```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

```
[[1167    0]
 [  33    0]]
```

In [187]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

          no       0.97      1.00      0.99      1167
         yes       0.00      0.00      0.00        33

    accuracy                           0.97      1200
   macro avg       0.49      0.50      0.49      1200
weighted avg       0.95      0.97      0.96      1200

D:\Program\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill-d
efined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
D:\Program\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill-d
efined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
D:\Program\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill-d
efined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]: