IBM员工流失预测

- 任务目标：通过分析数据预测员工流失
- 任务输出：预测模型准确率
- 任务方法：属于二分类问题，使用分类算法模型
- 模型评价指标

本项目主要从以下方面进行分析：

- 探索性数据分析
- 数据清洗
- 特征相关性分析
- 对类别数据进行标签编码
- 切分数据集
- 训练模型并预测
- 调整优化

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import sklearn
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import re
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
plt.rcParams['font.sans-serif']=['Microsoft YaHei']
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号
```

```
/Users/orangeli/opt/anaconda3/lib/python3.7/importlib/_bootstrap.py:219: RuntimeWarning: numpy.ufunc size
changed, may indicate binary incompatibility. Expected 192 from C header, got 216 from PyObject
  return f(*args, **kwds)
```

一、加载数据集

```
path = '/Users/orangeli/huxin/WA_Fn-UseC_-HR-Employee-Attrition.csv'
data = pd.read_csv(path, encoding = 'utf-8')
data.head(5)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCou
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1

5 rows × 35 columns

二、探索性数据分析

```
data.isnull().sum()
```

```
Age                         0
Attrition                   0
```

```
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

```
data.duplicated()
```

```
0       False
1       False
2       False
3       False
4       False
5       False
6       False
7       False
8       False
9       False
10      False
11      False
12      False
13      False
14      False
15      False
16      False
17      False
18      False
19      False
20      False
21      False
22      False
23      False
24      False
25      False
26      False
27      False
28      False
29      False
        ...
1440    False
1441    False
1442    False
1443    False
1444    False
1445    False
1446    False
1447    False
1448    False
1449    False
```

```
1450    False
1451    False
1452    False
1453    False
1454    False
1455    False
1456    False
1457    False
1458    False
1459    False
1460    False
1461    False
1462    False
1463    False
1464    False
1465    False
1466    False
1467    False
1468    False
1469    False
Length: 1470, dtype: bool
```

```
data.describe()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	J
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000	1
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156	2
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428	0
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000	1
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000	2
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000	3
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000	3
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000	4

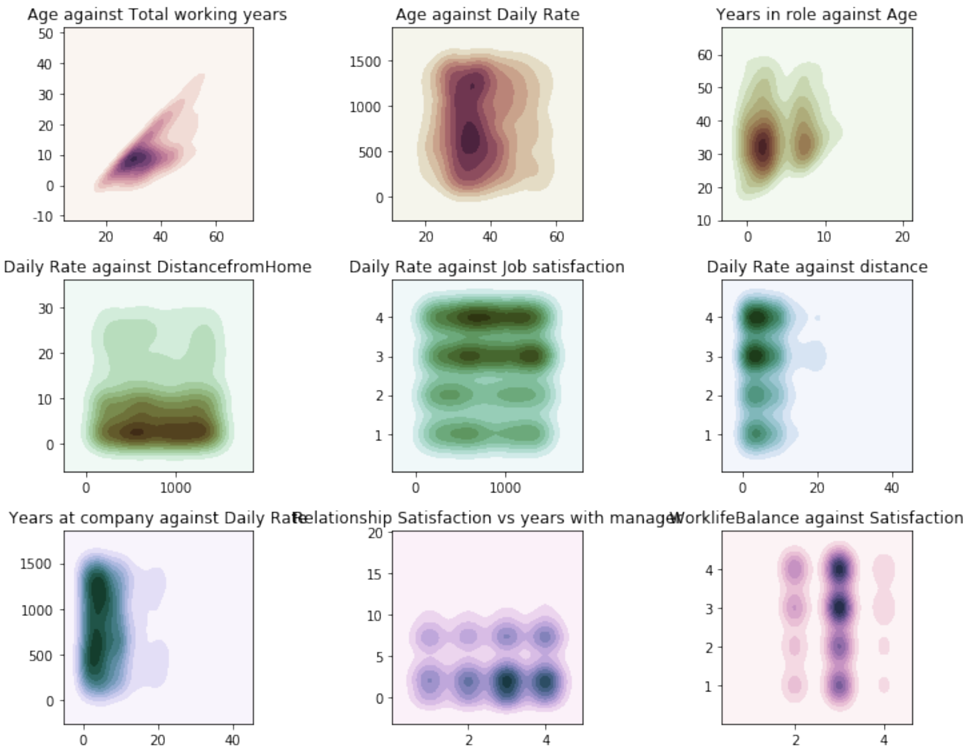
8 rows × 26 columns

```
data.columns
```

```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

```
import seaborn as sns
f, axes = plt.subplots(3, 3, figsize=(10, 8),
                       sharex=False, sharey=False)

# Defining our colormap scheme
s = np.linspace(0, 3, 10)
cmap = sns.cubehelix_palette(start=0.0, light=1, as_cmap=True)
```

```
# Generate and plot
x = data['Age'].values
y = data['TotalWorkingYears'].values
sns.kdeplot(x, y, cmap=cmap, shade=True, cut=5, ax=axes[0,0])
axes[0,0].set( title = 'Age against Total working years')

cmap = sns.cubehelix_palette(start=0.333333333333, light=1, as_cmap=True)
# Generate and plot
x = data['Age'].values
y = data['DailyRate'].values
sns.kdeplot(x, y, cmap=cmap, shade=True, ax=axes[0,1])
axes[0,1].set( title = 'Age against Daily Rate')

cmap = sns.cubehelix_palette(start=0.666666666667, light=1, as_cmap=True)
# Generate and plot
x = data['YearsInCurrentRole'].values
y = data['Age'].values
sns.kdeplot(x, y, cmap=cmap, shade=True, ax=axes[0,2])
axes[0,2].set( title = 'Years in role against Age')

cmap = sns.cubehelix_palette(start=1.0, light=1, as_cmap=True)
# Generate and plot
x = data['DailyRate'].values
y = data['DistanceFromHome'].values
sns.kdeplot(x, y, cmap=cmap, shade=True,  ax=axes[1,0])
axes[1,0].set( title = 'Daily Rate against DistancefromHome')

cmap = sns.cubehelix_palette(start=1.333333333333, light=1, as_cmap=True)
# Generate and plot
x = data['DailyRate'].values
y = data['JobSatisfaction'].values
sns.kdeplot(x, y, cmap=cmap, shade=True,  ax=axes[1,1])
axes[1,1].set( title = 'Daily Rate against Job satisfaction')

cmap = sns.cubehelix_palette(start=1.666666666667, light=1, as_cmap=True)
# Generate and plot
x = data['YearsAtCompany'].values
y = data['JobSatisfaction'].values
sns.kdeplot(x, y, cmap=cmap, shade=True,  ax=axes[1,2])
axes[1,2].set( title = 'Daily Rate against distance')

cmap = sns.cubehelix_palette(start=2.0, light=1, as_cmap=True)
# Generate and plot
x = data['YearsAtCompany'].values
y = data['DailyRate'].values
sns.kdeplot(x, y, cmap=cmap, shade=True,  ax=axes[2,0])
axes[2,0].set( title = 'Years at company against Daily Rate')

cmap = sns.cubehelix_palette(start=2.333333333333, light=1, as_cmap=True)
# Generate and plot
x = data['RelationshipSatisfaction'].values
y = data['YearsWithCurrManager'].values
sns.kdeplot(x, y, cmap=cmap, shade=True,  ax=axes[2,1])
axes[2,1].set( title = 'Relationship Satisfaction vs years with manager')

cmap = sns.cubehelix_palette(start=2.666666666667, light=1, as_cmap=True)
# Generate and plot
x = data['WorkLifeBalance'].values
y = data['JobSatisfaction'].values
sns.kdeplot(x, y, cmap=cmap, shade=True,  ax=axes[2,2])
axes[2,2].set( title = 'WorklifeBalance against Satisfaction')

f.tight_layout()
```



特征相关性分析

```
data_corr = data.corr()
data_corr
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

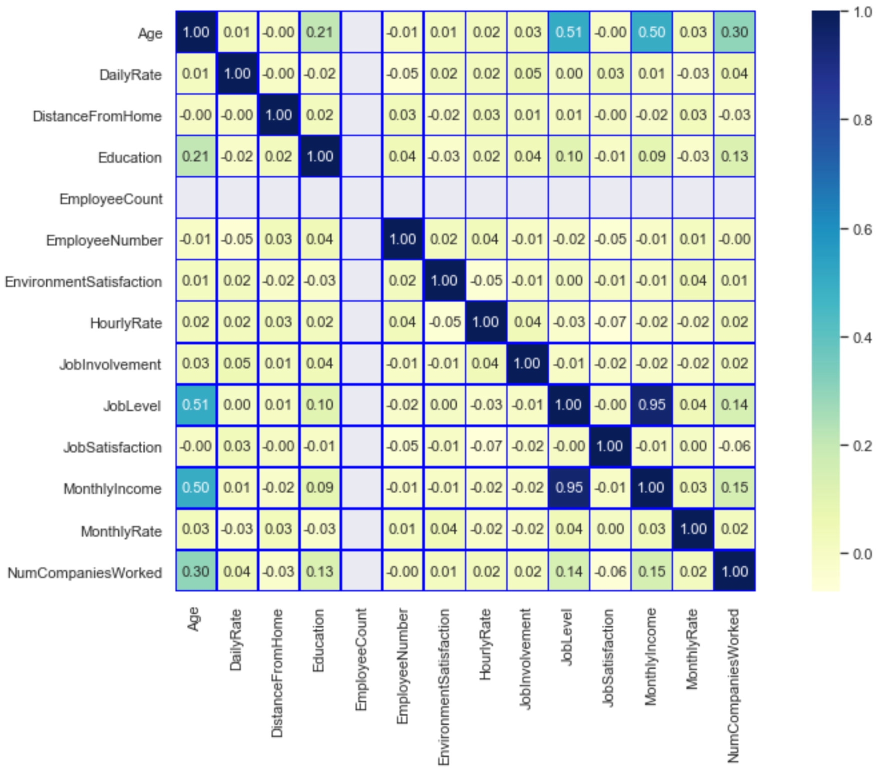
	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	Hou
Age	1.000000	0.010661	-0.001686	0.208034	NaN	-0.010145	0.010146	0.02
DailyRate	0.010661	1.000000	-0.004985	-0.016806	NaN	-0.050990	0.018355	0.02
DistanceFromHome	-0.001686	-0.004985	1.000000	0.021042	NaN	0.032916	-0.016075	0.03
Education	0.208034	-0.016806	0.021042	1.000000	NaN	0.042070	-0.027128	0.01
EmployeeCount	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
EmployeeNumber	-0.010145	-0.050990	0.032916	0.042070	NaN	1.000000	0.017621	0.03
EnvironmentSatisfaction	0.010146	0.018355	-0.016075	-0.027128	NaN	0.017621	1.000000	-0.04
HourlyRate	0.024287	0.023381	0.031131	0.016775	NaN	0.035179	-0.049857	1.00
JobInvolvement	0.029820	0.046135	0.008783	0.042438	NaN	-0.006888	-0.008278	0.04
JobLevel	0.509604	0.002966	0.005303	0.101589	NaN	-0.018519	0.001212	-0.02
JobSatisfaction	-0.004892	0.030571	-0.003669	-0.011296	NaN	-0.046247	-0.006784	-0.07
MonthlyIncome	0.497855	0.007707	-0.017014	0.094961	NaN	-0.014829	-0.006259	-0.01
MonthlyRate	0.028051	-0.032182	0.027473	-0.026084	NaN	0.012648	0.037600	-0.01
NumCompaniesWorked	0.299635	0.038153	-0.029251	0.126317	NaN	-0.001251	0.012594	0.02
PercentSalaryHike	0.003634	0.022704	0.040235	-0.011111	NaN	-0.012944	-0.031701	-0.00
PerformanceRating	0.001904	0.000473	0.027110	-0.024539	NaN	-0.020359	-0.029548	-0.00
RelationshipSatisfaction	0.053535	0.007846	0.006557	-0.009118	NaN	-0.069861	0.007665	0.00
StandardHours	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
StockOptionLevel	0.037510	0.042143	0.044872	0.018422	NaN	0.062227	0.003432	0.05
TotalWorkingYears	0.680381	0.014515	0.004628	0.148280	NaN	-0.014365	-0.002693	-0.00
TrainingTimesLastYear	-0.019621	0.002453	-0.036942	-0.025100	NaN	0.023603	-0.019359	-0.00
WorkLifeBalance	-0.021490	-0.037848	-0.026556	0.009819	NaN	0.010309	0.027627	-0.00
YearsAtCompany	0.311309	-0.034055	0.009508	0.069114	NaN	-0.011240	0.001458	-0.01
YearsInCurrentRole	0.212901	0.009932	0.018845	0.060236	NaN	-0.008416	0.018007	-0.02
YearsSinceLastPromotion	0.216513	-0.033229	0.010029	0.054254	NaN	-0.009019	0.016194	-0.02
YearsWithCurrManager	0.202089	-0.026363	0.014406	0.069065	NaN	-0.009197	-0.004999	-0.02

26 rows × 26 columns

```
h1 = data_corr.loc['Age':'NumCompaniesWorked', 'Age':'NumCompaniesWorked']
h2 = data_corr.loc['PercentSalaryHike':, 'Age':'NumCompaniesWorked']
h3 = data_corr.loc['PercentSalaryHike':, 'PercentSalaryHike':]

sns.set(rc = {'figure.figsize':(15,8)})
sns.heatmap(h1, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size': 11}, cmap="YlGnBu",
            linewidths=0.5, linecolor='blue')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x117850e10>
```

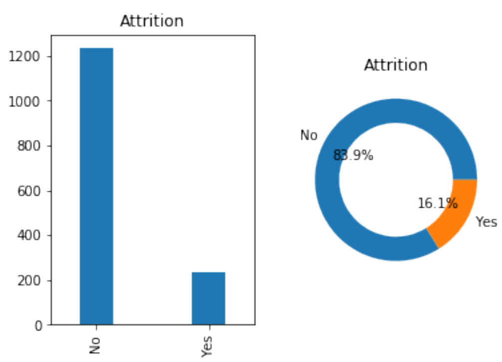
标签编码




```
plt.subplot(1,2,1)
data['Attrition'].value_counts().plot(kind='bar',width =0.3,title = 'Attrition')
plt.subplot(1,2,2)
ratio = data['Attrition'].value_counts()/len(data['Attrition'])
label1 = data['Attrition'].value_counts().index
plt.pie(ratio,labels=label1,autopct='%1.1f%%',wedgeprops={'width':0.3})
plt.title('Attrition')
```

```
Text(0.5, 1.0, 'Attrition')
```

```
findfont: Font family ['sans-serif'] not found. Falling back to DejaVu Sans.
findfont: Font family ['sans-serif'] not found. Falling back to DejaVu Sans.
```



标签编码

```
data["Attrition"] = LabelEncoder().fit_transform(data['Attrition'])
data["BusinessTravel"] = LabelEncoder().fit_transform(data['BusinessTravel'])
data["Department"] = LabelEncoder().fit_transform(data['Department'])
data["EducationField"] = LabelEncoder().fit_transform(data['EducationField'])
data["Gender"] = LabelEncoder().fit_transform(data['Gender'])
data["JobRole"] = LabelEncoder().fit_transform(data['JobRole'])
data["MaritalStatus"] = LabelEncoder().fit_transform(data['MaritalStatus'])
data["Over18"] = LabelEncoder().fit_transform(data['Over18'])
data["OverTime"] = LabelEncoder().fit_transform(data['OverTime'])
data.head(5)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber
0	41	1	2	1102	2	1	2	1	1	1
1	49	0	1	279	1	8	1	1	1	2
2	37	1	2	1373	1	2	2	4	1	4
3	33	0	1	1392	1	3	4	1	1	5
4	27	0	2	591	1	2	1	3	1	7

5 rows × 35 columns

数据归一化

```
from sklearn.preprocessing import StandardScaler
cols = list(data.columns)
cols.remove('Attrition')
cols.remove('EmployeeCount')
cols.remove('StandardHours')
sc = StandardScaler()
data[cols]= sc.fit_transform(data[cols])
data[cols].head(5)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeNumber	EnvironmentSatisfactio
0	0.446350	0.590048	0.742527	1.401512	-1.010909	-0.891688	-0.937414	-1.701283	-0.660531
1	1.322365	-0.913194	-1.297775	-0.493817	-0.147150	-1.868426	-0.937414	-1.699621	0.254625
2	0.008343	0.590048	1.414363	-0.493817	-0.887515	-0.891688	1.316673	-1.696298	1.169781
3	-0.429664	-0.913194	1.461466	-0.493817	-0.764121	1.061787	-0.937414	-1.694636	1.169781
4	-1.086676	0.590048	-0.524295	-0.493817	-0.887515	-1.868426	0.565311	-1.691313	-1.575686

5 rows × 32 columns

切分数据集，对不平衡样本进行SMOTE采样

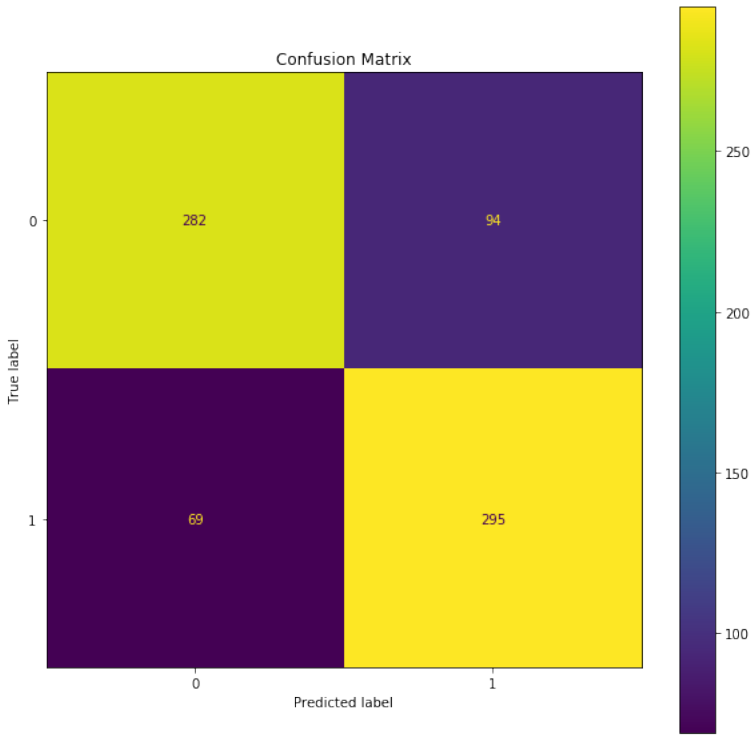
```
from imblearn.over_sampling import SMOTE

oversampler=SMOTE(random_state=0)
smote_train, smote_target = oversampler.fit_sample(data[cols],data['Attrition'])
x_train,x_test,y_train,y_test = train_test_split(smote_train,smote_target,test_size = 0.3,random_state=0,shuffle=True)
print("Train Feature Size : ",len(x_train))
print("Train Label Size : ",len(y_train))
print("Test Feature Size : ",len(x_test))
print("Test Label Size : ",len(y_test))
```

```
Train Feature Size :  1726
Train Label Size :  1726
Test Feature Size :  740
Test Label Size :  740
```

使用逻辑回归模型进行训练并预测

```
from sklearn.metrics import ConfusionMatrixDisplay
logistic_model = LogisticRegression(solver='liblinear',random_state=0).fit(x_train,y_train)
print("Train Accuracy : {:.2f} %".format(accuracy_score(logistic_model.predict(x_train),y_train)))
print("Test Accuracy : {:.2f} %".format(accuracy_score(logistic_model.predict(x_test),y_test)))

cm = confusion_matrix(y_test,logistic_model.predict(x_test))
classes = ['0','1']
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=classes)
fig, ax = plt.subplots(figsize=(10,10))
plt.title("Confusion Matrix")
disp = disp.plot(ax=ax)
plt.show()
```

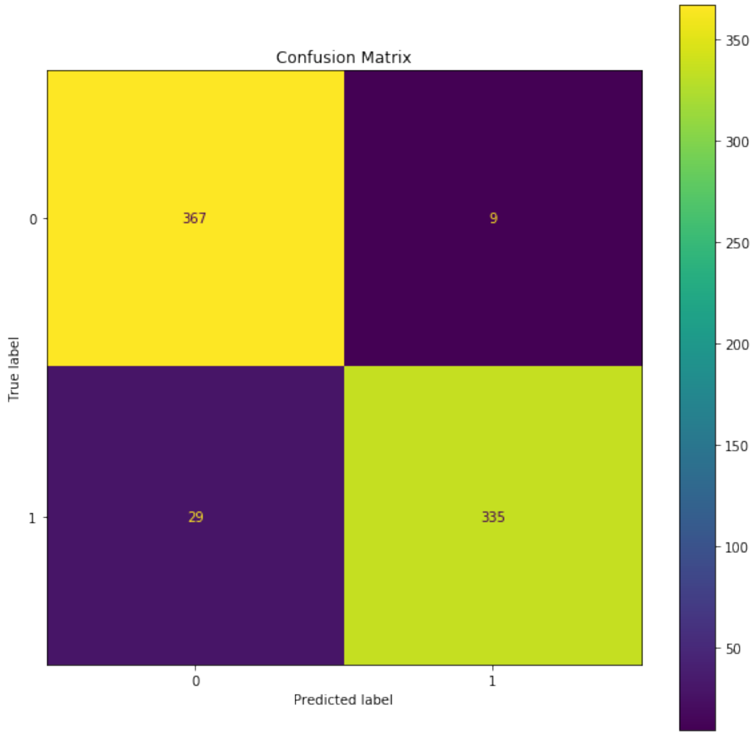
```
Train Accuracy : 0.79 %
Test Accuracy : 0.78 %
```



使用随机森林模型进行训练并预测

```
random_forest = RandomForestClassifier(n_estimators=590,
                                       random_state=0).fit(x_train,y_train)
print("Train Accuracy : {:.2f} %".format(accuracy_score(random_forest.predict(x_train),y_train)))
print("Test Accuracy : {:.2f} %".format(accuracy_score(random_forest.predict(x_test),y_test)))

cm = confusion_matrix(y_test,random_forest.predict(x_test))
classes = ["0","1"]
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=classes)
fig, ax = plt.subplots(figsize=(10,10))
plt.title("Confusion Matrix")
disp = disp.plot(ax=ax)
plt.show()
```

```
Train Accuracy : 1.00 %
Test Accuracy : 0.95 %
```




```
#随机森林可以查看特征重要性feature_importances_
import numpy as np
import seaborn as sns
feature_importance = random_forest.feature_importances_
sorted_idx = np.argsort(feature_importance)
data.columns[sorted_idx]
```

```
Index(['MonthlyRate', 'OverTime', 'EmployeeNumber', 'DailyRate', 'Attrition',
       'PercentSalaryHike', 'JobInvolvement', 'StockOptionLevel',
       'StandardHours', 'DistanceFromHome', 'Gender', 'Over18', 'Education',
       'WorkLifeBalance', 'MaritalStatus', 'EducationField', 'Department',
       'BusinessTravel', 'EnvironmentSatisfaction', 'MonthlyIncome',
       'EmployeeCount', 'TrainingTimesLastYear', 'TotalWorkingYears',
       'RelationshipSatisfaction', 'Age', 'YearsAtCompany', 'JobSatisfaction',
       'JobRole', 'JobLevel', 'HourlyRate', 'PerformanceRating',
       'NumCompaniesWorked'],
      dtype='object')
```

调整优化