通信客户流失预警模型案例分析

一. 环境介绍

windows 10

conda 4.7.12（Anaconda3-2019.10-Windows-x86_64）

python 3.7.4

PyCharm 2022.3.3

jupyter notebook

二. 业务理解

流失客户是指那些曾经使用过产品或服务，由于对产品失去兴趣等种种原因，不再使用产品或服务的顾客。

电信服务公司、互联网服务提供商、保险公司等经常使用客户流失分析和客户流失率作为他们的关键业务指标之一，因为留住一个老客户的成本远远低于获得一个新客户。

预测分析使用客户流失预测模型，通过评估客户流失的风险倾向来预测客户流失。由于这些模型生成了一个流失概率排序名单，对于潜在的高概率流失客户，他们可以有效地实施客户保留营销计划。

三. 数据理解

此次分析数据来自于 IBM Sample Data Sets，统计自某电信公司一段时间内的消费数据。共有 7043 笔客户资料，每笔客户资料包含 21 个字段，其中 1 个客户 ID 字段，19 个输入字段及 1 个目标字段-Churn

（Yes 代表流失，No 代表未流失），输入字段主要包含以下三个维度指标：用户画像指标、消费产品指标、消费信息指标。字段的具体说明如下：

字段	字段翻译	角色	测量类型	不同值个数
customerID	客户ID	ID	无类型	7043
gender	性别	输入	分类	2
SeniorCitizen	老年人	输入	分类	2
Partner	是否有配偶	输入	分类	2
Dependents	是否经济独立	输入	分类	2
tenure	在网时长	输入	数值	73
PhoneService	是否开通电话服务业务	输入	分类	2
MultipleLines	是否开通多线业务	输入	分类	3
InternetService	是否开通互联网服务	输入	分类	3
OnlineSecurity	是否开通网络安全服务	输入	分类	3
OnlineBackup	是否开通在线备份业务	输入	分类	3
DeviceProtection	是否开通了设备保护业务	输入	分类	3
TechSupport	是否开通了技术支持服务	输入	分类	3
StreamingTV	是否开通网络电视	输入	分类	3
StreamingMovies	是否开通网络电影	输入	分类	3
Contract	签订合同方式	输入	分类	3
PaperlessBilling	是否开通电子账单	输入	分类	2
PaymentMethod	付款方式	输入	分类	4
MonthlyCharges	月费用	输入	数值	1585
TotalCharges	总费用	输入	数值	6531
Churn	是否流失	目标	分类	2

四．数据准备

```
# 数据处理
import pandas as pd
import numpy as np
#读入数据集
df=pd.read_csv("Telco-Customer-Churn.csv")
print(df.head(10).to_string())
```

代码运行如下：

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	Streamin
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes	No	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	No	Yes	No	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	No	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	No	Yes	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	No	No	No	
5	9305-CDSKC	Female	0	No	No	8	Yes	Yes	Fiber optic	No	No	Yes	No	
6	1452-KIOVK	Male	0	No	Yes	22	Yes	Yes	Fiber optic	No	Yes	No	No	
7	6713-OKOMC	Female	0	No	No	10	No	No phone service	DSL	Yes	No	No	No	
8	7892-POOKP	Female	0	Yes	No	28	Yes	Yes	Fiber optic	No	No	Yes	Yes	
9	6388-TABGU	Male	0	No	Yes	62	Yes	No	DSL	Yes	Yes	No	No	

```
# 数据初步清洗
# 首先进行初步的数据清洗工作，包含错误值和异常值处理，并划分类别型和数值型字段类型，
# 其中清洗部分包含：OnlineSecurity、OnlineBackup、DeviceProtection、TechSupport、StreamingTV、StreamingMovies：
# 错误值处理 TotalCharges：异常值处理 tenure：自定义分箱
# 错误值处理
C_olumns=['OnlineSecurity','OnlineBackup','DeviceProtection','TechSupport','StreamingTV','StreamingMovies']
for i in C_olumns:
    df[i]=df[i].replace({ 'No internet service': 'No'})
```

```
#  替换 SeniorCitizen,Yes:1,No:0
df['SeniorCitizen']=df['SeniorCitizen'].replace({1:'Yes',0:'No'})
#  替换 TotalCharges 进而对空值进行删除
df['TotalCharges']=df['TotalCharges'].replace('',np.nan)
df=df.dropna(subset=['TotalCharges'])
#  重置索引
df.reset_index(drop=True,inplace=True)
# print(df.head(100).to_string())
```

代码运行如下：

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	Strea
0	7590-VHVEG	Female	No	Yes	No	1	No	No phone service	DSL	No	Yes	No	No	
1	5575-GNVDE	Male	No	No	No	34	Yes	No	DSL	Yes	No	Yes	No	
2	3668-QPYBK	Male	No	No	No	2	Yes	No	DSL	Yes	Yes	No	No	
3	7795-CFOCW	Male	No	No	No	45	No	No phone service	DSL	Yes	No	Yes	Yes	
4	9237-HQITU	Female	No	No	No	2	Yes	No	Fiber optic	No	No	No	No	
5	9305-CDSKC	Female	No	No	No	8	Yes	Yes	Fiber optic	No	No	Yes	No	
6	1452-KIOVK	Male	No	No	Yes	22	Yes	Yes	Fiber optic	No	Yes	No	No	
7	6713-OKOMC	Female	No	No	No	10	No	No phone service	DSL	Yes	No	No	No	
8	7892-POOKP	Female	No	Yes	No	28	Yes	Yes	Fiber optic	No	No	Yes	Yes	
9	6388-TABGU	Male	No	No	Yes	62	Yes	No	DSL	Yes	Yes	No	No	
10	9763-GRSKD	Male	No	Yes	Yes	13	Yes	No	DSL	Yes	No	No	No	
11	7469-LKBCI	Male	No	No	No	16	Yes	No	No	No	No	No	No	
12	8091-TTVAX	Male	No	Yes	No	58	Yes	Yes	Fiber optic	No	No	Yes	No	
13	0280-XJGEX	Male	No	No	No	49	Yes	Yes	Fiber optic	No	Yes	Yes	No	
14	5129-JLPIS	Male	No	No	No	25	Yes	No	Fiber optic	Yes	No	Yes	Yes	
15	3655-SNQYZ	Female	No	Yes	Yes	69	Yes	Yes	Fiber optic	Yes	Yes	Yes	Yes	
16	8191-XWSZG	Female	No	No	No	52	Yes	No	No	No	No	No	No	
17	9959-WOFKT	Male	No	No	Yes	71	Yes	Yes	Fiber optic	Yes	No	Yes	No	
18	4190-MFLUW	Female	No	Yes	Yes	10	Yes	No	DSL	No	No	Yes	Yes	

```
#  将 TotalCharges 列中的字符串转换为浮点数
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
#  转换 tenure,编写函数
def transform_tenure(x):
    if x<=12:
        return('Tenure_1')
    elif x<=24:
        return('Tenure_2')
    elif x<=36:
        return('Tenure_3')
    elif x<=48:
        return('Tenure_4')
    elif x<=60:
        return('Tenure_5')
    else:
        return('Tenure_over_5')
df['tenure_group']=df.tenure.apply(transform_tenure)
# print(df.head(100).to_string())
```

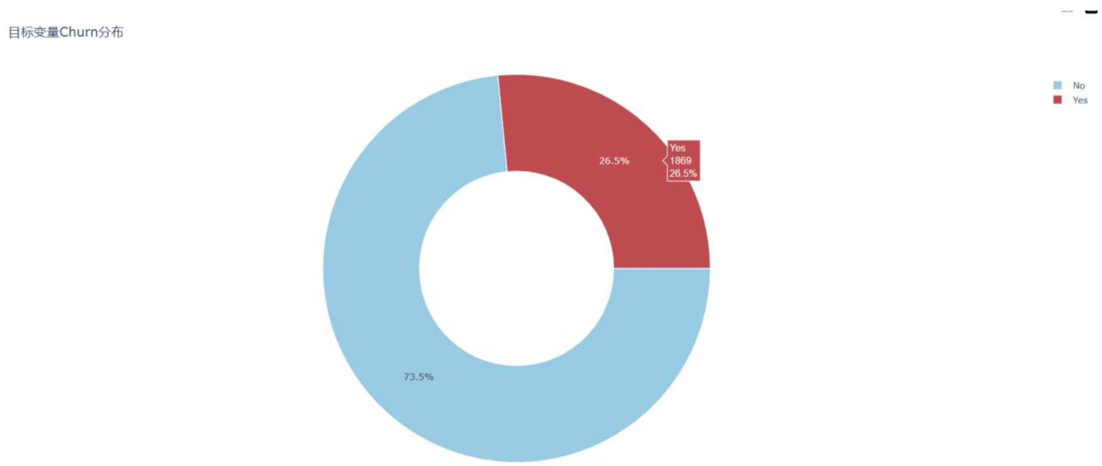
代码运行如下：

PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn	tenure_group
Yes	Electronic check	29.85	29.85	No	Tenure_1
No	Mailed check	56.95	1889.50	No	Tenure_3
Yes	Mailed check	53.85	108.15	Yes	Tenure_1
No	Bank transfer (automatic)	42.30	1840.75	No	Tenure_4
Yes	Electronic check	70.70	151.65	Yes	Tenure_1
Yes	Electronic check	99.65	820.50	Yes	Tenure_1
Yes	Credit card (automatic)	89.10	1949.40	No	Tenure_2
No	Mailed check	29.75	301.90	No	Tenure_1
Yes	Electronic check	104.80	3046.05	Yes	Tenure_3

探索性分析
目标变量 Churn 分布
可视化

```
df['Churn'].value_counts
trace0 = go.Pie(labels=df[ 'Churn'].value_counts().index,
values=df[ 'Churn'].value_counts().values,
hole= 0.5,
rotation= 90,
marker=dict(colors=[ 'rgb(154,203,228)', 'rgb(191,76,81)'],
line=dict(color= 'white', width= 1.3))
)
data = [trace0]
layout = go.Layout(title= '目标变量 Churn 分布')
fig = go.Figure(data=data, layout=layout)
py.offline.plot(fig, filename= '整体流失情况分布.html',auto_open=False)
```

效果图如下：

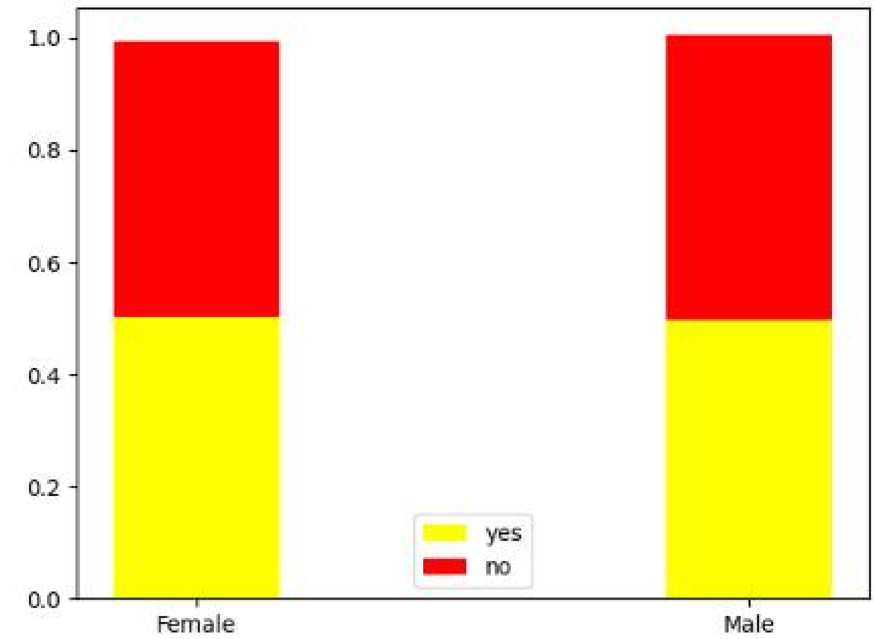


性别与是否流失的关系
男性和女性在客户流失比例上没有显著差异

```
a1 = df[(df['Churn']=='Yes')&(df['gender']=='Female')]['Churn'].count()
a_1 = df[(df['Churn']=='Yes')&(df['gender']=='Male')]['Churn'].count()
a2 = df[(df['Churn']=='No')&(df['gender']=='Female')]['Churn'].count()
a_2 = df[(df['Churn']=='No')&(df['gender']=='Male')]['Churn'].count()
a1_p = a1/(a_1+a1)
a_1_p = a_1/(a_1+a1)
a2_p = a2/(a2+a_2)
a_2_p = a_2/(a2+a_2)
plt.bar(['Female','Male'],height=[a1_p,a_1_p],width=0.3,color='yellow',
data=None,label=u'yes')
plt.bar(['Female','Male'],height=[a2_p,a_2_p],                    width=0.3,
bottom=[a1_p,a_1_p],color='red', data=None,label=u'no')
```

```
plt.legend(loc='best')
plt.show()
```

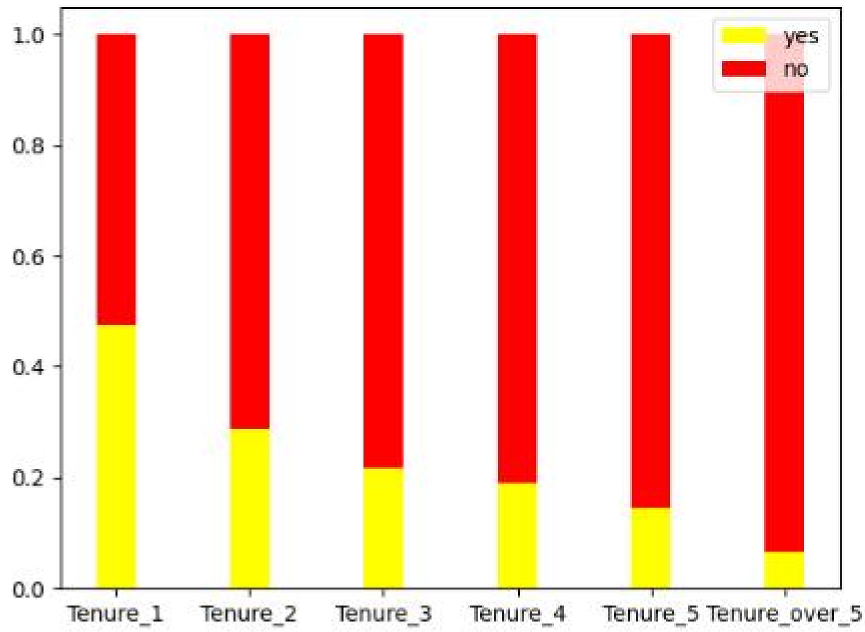
效果图如下：




```
# 在网时长与是否流失的关系
# 用户的在网时长越长，表示用户的忠诚度越高，其流失的概率越低
a1 = df[(df['Churn']=='Yes')&(df['tenure_group']=='Tenure_1')]['Churn'].count()
a_1 = df[(df['Churn']=='No')&(df['tenure_group']=='Tenure_1')]['Churn'].count()
a2 = df[(df['Churn']=='Yes')&(df['tenure_group']=='Tenure_2')]['Churn'].count()
a_2 = df[(df['Churn']=='No')&(df['tenure_group']=='Tenure_2')]['Churn'].count()
a3 = df[(df['Churn']=='Yes')&(df['tenure_group']=='Tenure_3')]['Churn'].count()
a_3 = df[(df['Churn']=='No')&(df['tenure_group']=='Tenure_3')]['Churn'].count()
a4 = df[(df['Churn']=='Yes')&(df['tenure_group']=='Tenure_4')]['Churn'].count()
a_4 = df[(df['Churn']=='No')&(df['tenure_group']=='Tenure_4')]['Churn'].count()
a5 = df[(df['Churn']=='Yes')&(df['tenure_group']=='Tenure_5')]['Churn'].count()
a_5 = df[(df['Churn']=='No')&(df['tenure_group']=='Tenure_5')]['Churn'].count()
a6 = df[(df['Churn']=='Yes')&(df['tenure_group']=='Tenure_over_5')]['Churn'].count()
a_6 = df[(df['Churn']=='No')&(df['tenure_group']=='Tenure_over_5')]['Churn'].count()
a1_p = a1/(a1+a_1)
a_1_p = a_1/(a1+a_1)
a2_p = a2/(a2+a_2)
a_2_p = a_2/(a2+a_2)
a3_p = a3/(a3+a_3)
a_3_p = a_3/(a3+a_3)
a4_p = a4/(a4+a_4)
```

```
a_4_p = a_4/(a4+a_4)
a5_p = a5/(a5+a_5)
a_5_p = a_5/(a5+a_5)
a6_p = a6/(a6+a_6)
a_6_p = a_6/(a6+a_6)
plt.bar(['Tenure_1','Tenure_2','Tenure_3','Tenure_4','Tenure_5','Tenure_over_5'],height=[a1_p,a2_p,a3_p,a4_p,a5_p,a6_p], width=0.3,color='yellow', data=None,label=u'yes')
plt.bar(['Tenure_1','Tenure_2','Tenure_3','Tenure_4','Tenure_5','Tenure_over_5'],height=[a_1_p,a_2_p,a_3_p,a_4_p,a_5_p,a_6_p], width=0.3, bottom=[a1_p,a2_p,a3_p,a4_p,a5_p,a6_p],color='red', data=None,label=u'no')
plt.legend(loc='best')
plt.show()
```

效果图如下：




```
# 对于二分类变量，编码为 0 和 1;
# 对于多分类变量，进行 one_hot 编码；
# 对于数值型变量，部分模型如 KNN、神经网络、Logistic 需要进行标准化处理。
# 建模数据
df_model=df
Id_col=['customerID']
Target_cil=['Churn']
# 分类型
Category_cols=df.nunique()[df.nunique()<10].index.tolist()
# 数值型
num_cols=[i for i in df.columns if i not in Category_cols +Id_col]
```

```
# 二分类型
binary_cols=df_model.nunique()[df_model.nunique()==2].index.tolist()
# 多分类型
multi_cols=[i for i in Category_cols if i not in binary_cols]
# 二分类标签编码
le=LabelEncoder()
for i in binary_cols:
    df_model[i]=le.fit_transform(df_model[i])
#多分类哑变量变换
df_model=df_model.dropna()
df_model=pd.get_dummies(data=df_model,columns=multi_cols)
print(df.head(100).to_string())
```

代码运行如下：

p	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn	tenure_gro
1	0	0	0	0	Month-to-month	1	Electronic check	29.85	29.85	0	Tenure
0	1	0	0	0	One year	0	Mailed check	56.95	1889.50	0	Tenure
1	0	0	0	0	Month-to-month	1	Mailed check	53.85	108.15	1	Tenure
0	1	1	0	0	One year	0	Bank transfer (automatic)	42.30	1840.75	0	Tenure
0	0	0	0	0	Month-to-month	1	Electronic check	70.70	151.65	1	Tenure
0	1	0	1	1	Month-to-month	1	Electronic check	99.65	820.50	1	Tenure
1	0	0	1	0	Month-to-month	1	Credit card (automatic)	89.10	1949.40	0	Tenure
0	0	0	0	0	Month-to-month	0	Mailed check	29.75	301.90	0	Tenure
0	1	1	1	1	Month-to-month	1	Electronic check	104.80	3046.05	1	Tenure

```
# 使用统计检定方式进行特征筛选。
X = df_model.copy().drop(['customerID','Churn'], axis=1)
y = df_model[Target_cil]
fs = SelectKBest(score_func=f_classif, k=20)
y = y.values.ravel()
X_train_fs = fs.fit_transform(X,y)
def SelectName(feature_data, model):
    scores = model.scores_
    indices = np.argsort(scores)[::-1]
    return list(feature_data.columns.values[indices[0:model.k]])
fea_name = [i for i in X.columns if i in SelectName(X,fs)]
X_train = pd.DataFrame(X_train_fs,columns = fea_name)
```

代码运行如下：

7012	1.0	1.0	0.0	63.0	0.0	0.0	1.0	103.50	6479.40	0.0
7013	0.0	1.0	0.0	44.0	1.0	0.0	1.0	84.80	3626.35	0.0
7014	0.0	0.0	0.0	18.0	0.0	1.0	1.0	95.05	1679.40	0.0
7015	0.0	0.0	0.0	9.0	0.0	0.0	1.0	44.20	403.35	1.0
7016	0.0	0.0	0.0	13.0	0.0	1.0	0.0	73.35	931.55	1.0
7017	0.0	1.0	0.0	68.0	0.0	1.0	0.0	64.10	4326.25	1.0
7018	1.0	0.0	0.0	6.0	0.0	0.0	1.0	44.40	263.05	1.0
7019	0.0	0.0	0.0	2.0	0.0	0.0	1.0	20.05	39.25	0.0
7020	1.0	1.0	0.0	55.0	1.0	0.0	0.0	60.00	3316.10	1.0
7021	1.0	0.0	0.0	1.0	0.0	0.0	1.0	75.75	75.75	0.0
7022	0.0	0.0	0.0	38.0	0.0	0.0	1.0	69.50	2625.25	0.0
7023	0.0	0.0	0.0	67.0	1.0	0.0	1.0	102.95	6886.25	0.0
7024	0.0	0.0	0.0	19.0	0.0	0.0	1.0	78.70	1495.10	0.0
7025	0.0	0.0	0.0	12.0	0.0	1.0	0.0	60.65	743.30	1.0
7026	0.0	0.0	0.0	72.0	0.0	0.0	1.0	21.15	1419.40	0.0
7027	0.0	1.0	1.0	24.0	1.0	1.0	1.0	84.80	1990.50	1.0
7028	0.0	1.0	1.0	72.0	0.0	0.0	1.0	103.20	7362.90	0.0
7029	0.0	1.0	1.0	11.0	1.0	0.0	1.0	29.60	346.45	1.0

五．构建模型

```
# 模型建立和评估
# 首先使用分层抽样的方式将数据划分训练集和测试集。
# 重新划分
# 分层抽样
X_train, X_test, y_train, y_test = train_test_split(X_train, y, test_size=0.2, random_state=0, stratify=y)
# print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
X_train = pd.DataFrame(X_train)
X_test = pd.DataFrame(X_test)
y_train = pd.DataFrame(y_train)
y_test = pd.DataFrame(y_test)
#修正索引
for i in[X_train, X_test, y_train, y_test]:
    i.index= range(i.shape[0])
# 保存标准化训练和测试数据
st= StandardScaler()
num_scaled_train=pd.DataFrame(st.fit_transform(X_train[num_cols]),columns=num_cols)
num_scaled_test=pd.DataFrame(st.transform(X_test[num_cols]),columns=num_cols)
X_train_scaled= pd.concat([X_train.drop(num_cols, axis= 1), num_scaled_train], axis= 1)
X_test_scaled= pd.concat([X_test.drop(num_cols, axis= 1), num_scaled_test], axis= 1)
parameters = { 'splitter': ( 'best', 'random'),
'criterion': ( "gini", "entropy"),
"max_depth": [* range( 3, 20)],}
clf = DecisionTreeClassifier(random_state= 25)
GS = GridSearchCV(clf, parameters, scoring= 'f1', cv= 10)
GS.fit(X_train, y_train)
# print(GS.best_params_)
# print(GS.best_score_)
clf = GS.best_estimator_
test_pred = clf.predict(X_test)
print('测试集：n', classification_report(y_test, test_pred))
```

代码运行如下：

```
{'criterion': 'entropy', 'max_depth': 5, 'splitter': 'best'}
0.6014522720158608
测试集：n              precision    recall  f1-score   support

           0       0.84      0.88      0.86      1033
           1       0.62      0.53      0.57       374

    accuracy                           0.79      1407
   macro avg       0.73      0.71      0.71      1407
weighted avg       0.78      0.79      0.78      1407
```

六．评估模型

```
# 输出决策树属性重要性排序
imp = pd.DataFrame(zip(X_train.columns, clf.feature_importances_))
imp.columns = ['feature', 'importances']
imp = imp.sort_values('importances', ascending=False)
imp = imp[imp['importances'] != 0]
table = ff.create_table(np.round(imp, 4))
py.offline.iplot(table)
```

效果图如下：

feature	importances
Contract_Month-to-month	0.5369
tenure	0.1441
InternetService_Fiber optic	0.1075
TotalCharges	0.0726
MonthlyCharges	0.0618
PaymentMethod_Electronic check	0.0174
Contract_Two year	0.0166
InternetService_DSL	0.0165
InternetService_No	0.0111
TechSupport	0.007
Contract_One year	0.0037
PaymentMethod_Credit card (automatic)	0.0034
PaymentMethod_Bank transfer (automatic)	0.0013