```python
#!usr/bin/python
#-*-encoding:UTF-8-*-
#Date:2021/10/25
#Author:Dasein
#载入包; 设置;载入数据
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style='darkgrid',font_scale=1.3)
plt.rcParams['font.family']='SimHei'
plt.rcParams['axes.unicode_minus']=False
import warnings
warnings.filterwarnings('ignore')
```

```python
df1 = pd.read_csv("E:\\dasein_py\\Data Analysis\\Telecommunication_da\\WA_Fn-UseC_-Telco-Customer-Churn.csv")
```

```python
print(df1.info())
print(df1.shape)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
None
(7043, 21)
```

```python
quantative = [i for i in df1.columns if df1[i].dtype!=object]
quanlitive = [i for i in df1.columns if df1[i].dtype==object]
print("Quantative counts:{}, Quanlitive counts:{}".format(len(quantative),len(quanlitive)))
```

```
Quantative counts:3, Quanlitive counts:18
```

**Data Overall**

- Dtype: float64 & string (Quantative:3, Quanlitive:18)
- Case counts: 7043
- Variable counts: 21

```python
df1.describe()
```

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

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

| | SeniorCitizen | tenure | MonthlyCharges |
|---|---|---|---|
| **count** | 7043.000000 | 7043.000000 | 7043.000000 |
| **mean** | 0.162147 | 32.371149 | 64.761692 |
| **std** | 0.368612 | 24.559481 | 30.090047 |
| **min** | 0.000000 | 0.000000 | 18.250000 |
| **25%** | 0.000000 | 9.000000 | 35.500000 |
| **50%** | 0.000000 | 29.000000 | 70.350000 |
| **75%** | 0.000000 | 55.000000 | 89.850000 |
| **max** | 1.000000 | 72.000000 | 118.750000 |

```
df1.columns
```

```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

**Variable Notes**

- customerID: ID
- gender
- SeniorCitizen: Whether the customer is a senior citizen or not (1, 0)
- Partner: Whether the customer has a partner or not (Yes, No)
- Dependents: Whether the customer has dependents or not (Yes, No)
- tenure: Number of months the customer has stayed with the company
- PhoneService: Whether the customer has a phone service or not (Yes, No)
- MultipleLines: Whether the customer has multiple lines or not (Yes, No, No phone service)
- InternetService: Customer’s internet service provider (DSL, Fiber optic, No)
- OnlineSecurity: Whether the customer has online security or not (Yes, No, No internet service)
- OnlineBackup：Whether the customer has online backup or not (Yes, No, No internet service)
- DeviceProtection：Whether the customer has device protection or not (Yes, No, No internet service)
- TechSupport：Whether the customer has tech support or not (Yes, No, No internet service)
- StreamingTV：Whether the customer has streaming TV or not (Yes, No, No internet service)
- StreamingMovies：Whether the customer has streaming movies or not (Yes, No, No internet service)
- Contract：The contract term of the customer (Month-to-month, One year, Two year)
- PaperlessBilling：Whether the customer has paperless billing or not (Yes, No)
- PaymentMethod：The customer’s payment method (Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic))
- MonthlyCharges：The amount charged to the customer monthly
- TotalCharges：The total amount charged to the customer
- Churn：Whether the customer churned or not (Yes or No)

```
df1.head(3)
```

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

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

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | InternetService | OnlineSecurity | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 7590-VHVEG | Female | 0 | Yes | No | 1 | No | No phone service | DSL | No | ... |
| **1** | 5575-GNVDE | Male | 0 | No | No | 34 | Yes | No | DSL | Yes | ... |
| **2** | 3668-QPYBK | Male | 0 | No | No | 2 | Yes | No | DSL | Yes | ... |

3 rows × 21 columns

**Insights**

- *customerID*可以drop
- Quantative中*SeniorCitizen*是0-1变量
- Quanlitive数据需要重编码

```
df1.drop('customerID',axis=1,inplace=True) #drop colName: CustomerID
```

```
df1.head(3) #double check data after drop
```

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

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

| | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | InternetService | OnlineSecurity | OnlineBackup | De |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Female | 0 | Yes | No | 1 | No | No phone service | DSL | No | Yes | No |
| **1** | Male | 0 | No | No | 34 | Yes | No | DSL | Yes | No | Ye |
| **2** | Male | 0 | No | No | 2 | Yes | No | DSL | Yes | Yes | No |

```
total = df1.isnull().sum()
null_percentage = total/df1.isnull().count()
null_percentage
```

```
gender              0.0
SeniorCitizen       0.0
Partner             0.0
Dependents          0.0
tenure              0.0
PhoneService        0.0
MultipleLines       0.0
InternetService     0.0
OnlineSecurity      0.0
OnlineBackup        0.0
DeviceProtection    0.0
TechSupport         0.0
StreamingTV         0.0
StreamingMovies     0.0
```

```
Contract            0.0
PaperlessBilling    0.0
PaymentMethod       0.0
MonthlyCharges      0.0
TotalCharges        0.0
Churn               0.0
dtype: float64
```

**Insights**

- Hypothesis: probable duplicates.

```
print(df1.duplicated().sum())
df1=df1.drop_duplicates(subset=None, keep='first',inplace=False)
```

```
22
```

```
#double check去重之后data
print(df1.duplicated().sum())
```

```
0
```

```
# TotalCharges应该是数值型，需要强制类型转换
# df1['TotalCharges']=df1['TotalCharges'].astype('float64')
df1['TotalCharges'] = df1['TotalCharges'].apply(pd.to_numeric, errors='coerce')
```

```
df1['TotalCharges'].dtype
```

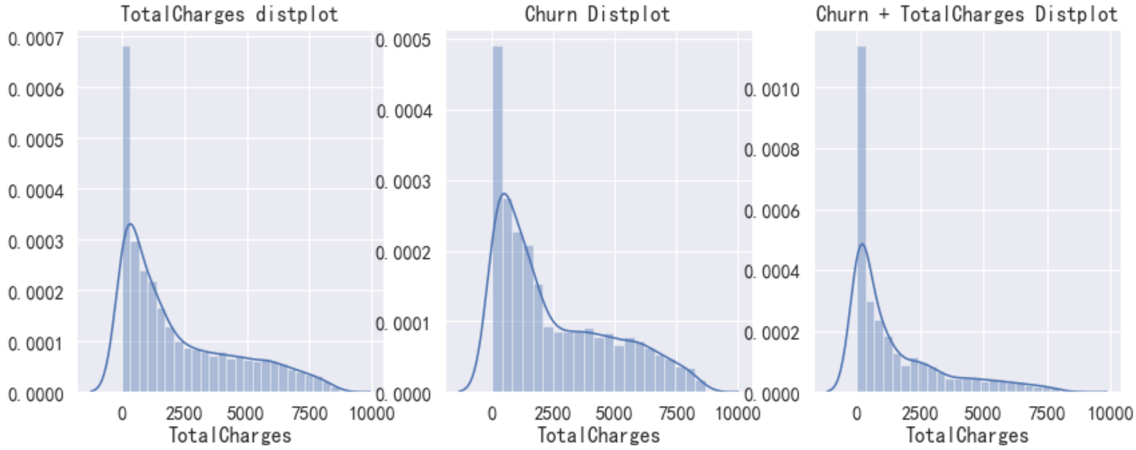
```
dtype('float64')
```

```
df1['TotalCharges'].isnull().sum() #有缺失值
```

```
11
```

```
print('TotalCharges数据分布')
plt.figure(figsize=(14,5))
plt.plot(color='#00338D')
#1
plt.subplot(1,3,1)
plt.title("TotalCharges distplot")
sns.distplot(df1.TotalCharges)
#2
plt.subplot(1,3,2)
plt.title("Churn Distplot")
sns.distplot(df1[df1.Churn=='No']['TotalCharges'])
#2
plt.subplot(1,3,3)
plt.title("Churn + TotalCharges Distplot")
sns.distplot(df1[df1['Churn']=='Yes']['TotalCharges'])
plt.show()
```

```
TotalCharges数据分布
```




**Insights**

- *TotalCharges*偏态分布，需要用中值填充缺失值。

```
df1.fillna({'TotalCharges':df1.TotalCharges.median()},inplace=True)
```

```
df1.TotalCharges.isnull().sum() #已经没有缺失值
```

```
0
```

```
#重编码‘Churn’，定性转定量的哑变量
#df1.Churn=df1.Churn.map({'Yes':1,'No':0})
df1.Churn.replace(to_replace='Yes',value=1,inplace=True)
df1.Churn.replace(to_replace='No',value=0,inplace=True)
df1.Churn.isnull().sum()
```

```
0
```

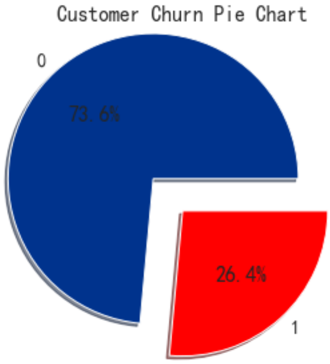
```
df1.Churn.describe()
```

```
count    7021.000000
mean        0.264492
std         0.441094
min         0.000000
25%         0.000000
50%         0.000000
75%         1.000000
max         1.000000
Name: Churn, dtype: float64
```

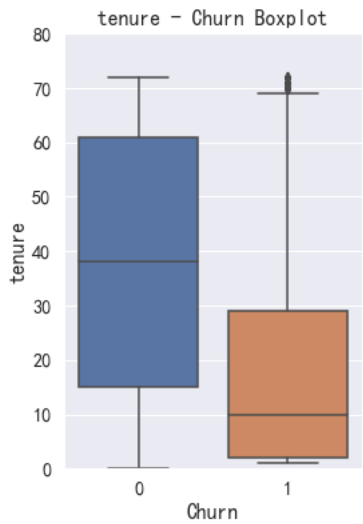
**Insights**

- 平均流失率 26.45%。

```
Churn_Count=df1.Churn.value_counts()
Churn_Lab=df1.Churn.value_counts().index
plt.figure(figsize=(5,5))
plt.pie(Churn_Count,labels=Churn_Lab,
        colors=["#00338D","red"],
        explode=(0.3,0),
        autopct="%1.1f%%",
        shadow=True)
plt.title("Customer Churn Pie Chart")
plt.show()
```




```
plt.figure(figsize=(4,6))
plt.plot(color='#00338D')
fig = sns.boxplot(x="Churn",y="tenure",data=df1)
plt.title("tenure - Churn Boxplot")
fig.axis(ymin=0,ymax=80)
plt.show()
```




- tenure越小流失率越显著

```
df2 = df1.apply(lambda x:pd.factorize(x)[0]) #转换成因子
df2.head(5)
```

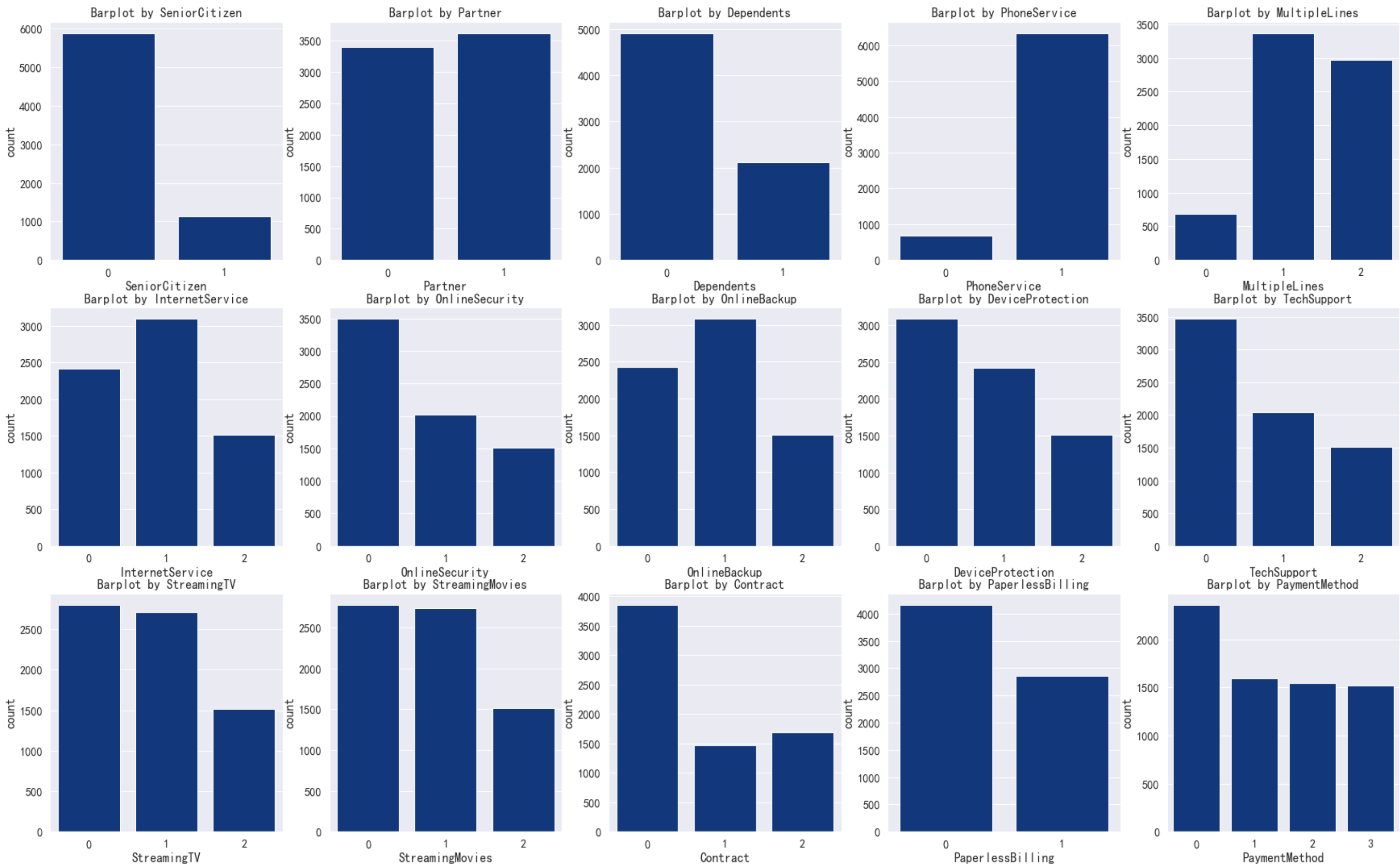
```
.dataframe tbody tr th {
    vertical-align: top;
}

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

| | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | InternetService | OnlineSecurity | OnlineBackup | De |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| **2** | 1 | 0 | 1 | 0 | 2 | 1 | 1 | 0 | 1 | 0 | 0 |
| **3** | 1 | 0 | 1 | 0 | 3 | 0 | 0 | 0 | 1 | 1 | 1 |
| **4** | 0 | 0 | 1 | 0 | 2 | 1 | 1 | 1 | 0 | 1 | 0 |

```
var = list(df2.columns)
var.remove("Churn")
var.remove("tenure")
var.remove("MonthlyCharges")
var.remove("TotalCharges")
plt.figure(figsize=(30,25))
```

```
a=0
for item in var:
    a+=1
    plt.subplot(4,5,a)
    plt.title('Barplot by '+ item)
    sns.countplot(x=item,data=df2,
                 color="#00338D")
#sns.countplot(x=None, y=None,
#hue=None, data=None, order=None,
#hue_order=None, orient=None, color=None,
#palette=None, saturation=0.75, dodge=True, ax=None, **kwargs)
```




**Insights**

- gender对Churn的影响不显著

```
df2.drop("gender",axis=1,inplace=True)
```

```
---------------------------------------------------------------------------

KeyError                                  Traceback (most recent call last)

<ipython-input-28-67322b8776aa> in <module>
----> 1 df2.drop("gender",axis=1,inplace=True)
```

```
D:\anaconda\lib\site-packages\pandas\core\frame.py in drop(self, labels, axis, index, columns, level, inplace, errors)
   3988                 weight  1.0     0.8
   3989         """
-> 3990         return super().drop(
   3991             labels=labels,
   3992             axis=axis,
```

```
D:\anaconda\lib\site-packages\pandas\core\generic.py in drop(self, labels, axis, index, columns, level, inplace, errors)
   3934         for axis, labels in axes.items():
   3935             if labels is not None:
-> 3936                 obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   3937 
   3938         if inplace:
```

```
D:\anaconda\lib\site-packages\pandas\core\generic.py in _drop_axis(self, labels, axis, level, errors)
   3968                 new_axis = axis.drop(labels, level=level, errors=errors)
   3969             else:
-> 3970                 new_axis = axis.drop(labels, errors=errors)
   3971             result = self.reindex(**{axis_name: new_axis})
   3972
```

```
D:\anaconda\lib\site-packages\pandas\core\indexes\base.py in drop(self, labels, errors)
   5016         if mask.any():
   5017             if errors != "ignore":
-> 5018                 raise KeyError(f"{labels[mask]} not found in axis")
   5019             indexer = indexer[~mask]
   5020         return self.delete(indexer)
```

```
KeyError: "['gender'] not found in axis"
```

```
df2.isnull().sum() #转换成因子之后没有缺失值，不需要fillna（TotalCharges已经填充缺失值）
```

```
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

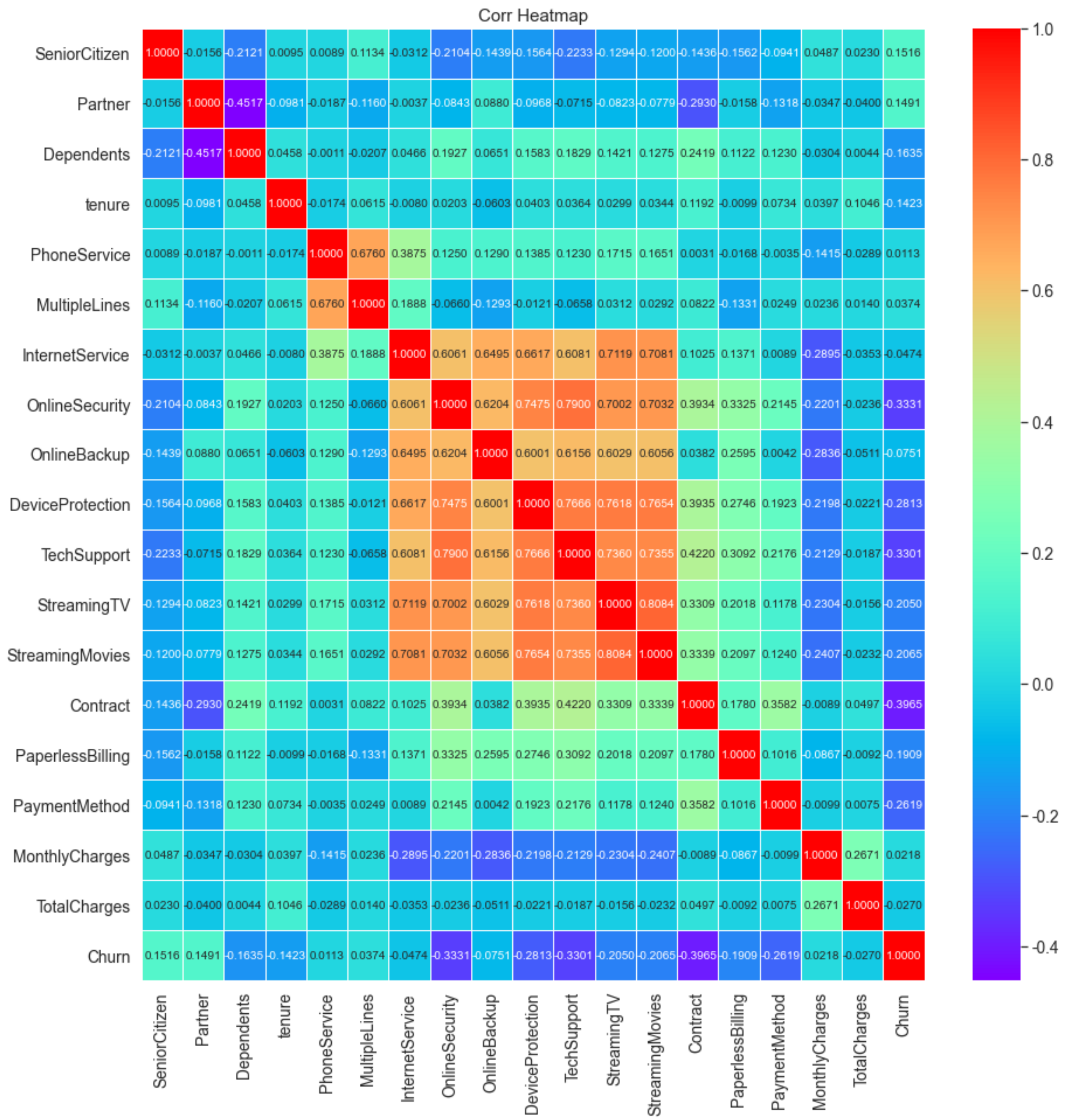
```
corr = df2.corr()
corr
```

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

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

| | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | InternetService | OnlineSecurity | OnlineBack |
|---|---|---|---|---|---|---|---|---|---|
| **SeniorCitizen** | 1.000000 | -0.015553 | -0.212115 | 0.009452 | 0.008909 | 0.113409 | -0.031221 | -0.210370 | -0.143900 |
| **Partner** | -0.015553 | 1.000000 | -0.451659 | -0.098113 | -0.018728 | -0.115992 | -0.003667 | -0.084330 | 0.087952 |
| **Dependents** | -0.212115 | -0.451659 | 1.000000 | 0.045761 | -0.001092 | -0.020715 | 0.046608 | 0.192658 | 0.065093 |
| **tenure** | 0.009452 | -0.098113 | 0.045761 | 1.000000 | -0.017391 | 0.061467 | -0.007970 | 0.020317 | -0.060309 |
| **PhoneService** | 0.008909 | -0.018728 | -0.001092 | -0.017391 | 1.000000 | 0.675973 | 0.387549 | 0.125042 | 0.129032 |
| **MultipleLines** | 0.113409 | -0.115992 | -0.020715 | 0.061467 | 0.675973 | 1.000000 | 0.188826 | -0.065972 | -0.129333 |
| **InternetService** | -0.031221 | -0.003667 | 0.046608 | -0.007970 | 0.387549 | 0.188826 | 1.000000 | 0.606107 | 0.649514 |
| **OnlineSecurity** | -0.210370 | -0.084330 | 0.192658 | 0.020317 | 0.125042 | -0.065972 | 0.606107 | 1.000000 | 0.620365 |
| **OnlineBackup** | -0.143900 | 0.087952 | 0.065093 | -0.060309 | 0.129032 | -0.129333 | 0.649514 | 0.620365 | 1.000000 |
| **DeviceProtection** | -0.156410 | -0.096813 | 0.158328 | 0.040275 | 0.138544 | -0.012102 | 0.661669 | 0.747520 | 0.600141 |
| **TechSupport** | -0.223293 | -0.071483 | 0.182923 | 0.036426 | 0.123035 | -0.065817 | 0.608130 | 0.789952 | 0.615611 |
| **StreamingTV** | -0.129375 | -0.082304 | 0.142145 | 0.029948 | 0.171477 | 0.031247 | 0.711946 | 0.700176 | 0.602861 |
| **StreamingMovies** | -0.120015 | -0.077925 | 0.127508 | 0.034361 | 0.165127 | 0.029227 | 0.708061 | 0.703203 | 0.605631 |
| **Contract** | -0.143624 | -0.293042 | 0.241912 | 0.119246 | 0.003101 | 0.082152 | 0.102456 | 0.393394 | 0.038225 |
| **PaperlessBilling** | -0.156196 | -0.015776 | 0.112220 | -0.009923 | -0.016824 | -0.133094 | 0.137056 | 0.332537 | 0.259546 |
| **PaymentMethod** | -0.094091 | -0.131842 | 0.122957 | 0.073367 | -0.003547 | 0.024891 | 0.008899 | 0.214518 | 0.004219 |
| **MonthlyCharges** | 0.048736 | -0.034681 | -0.030433 | 0.039656 | -0.141515 | 0.023609 | -0.289498 | -0.220075 | -0.283567 |
| **TotalCharges** | 0.022996 | -0.040026 | 0.004450 | 0.104648 | -0.028946 | 0.013971 | -0.035305 | -0.023596 | -0.051101 |
| **Churn** | 0.151619 | 0.149135 | -0.163459 | -0.142337 | 0.011323 | 0.037429 | -0.047366 | -0.333144 | -0.075052 |

```
plt.figure(figsize=(15,15))
sns.set(font_scale=1.25)
ax=sns.heatmap(corr,
               xticklabels=corr.columns,
               yticklabels=corr.columns,
               linewidths=0.6,annot=True,
               cbar=True,cmap="rainbow",fmt='.4f',
               annot_kws={'size': 10})
plt.title("Corr Heatmap")
plt.savefig("Corr Heatmap.png",dpi=100)
plt.show()
```




**Insights**

- 极强相关变量
- **MultipleLines - PhoneService**之间有很强共线性。
- 相关系数矩阵中心的变量之间具有极强的相关性（共线性）
  - *OnlineSecurity / InternetService /OnlineBackup / DeviceProtection / TechSupport / StreamingTV / StreamingMovies*
- 没有与Churn具有极强相关性的变量。
  - *TotalCharges / MonthlyCharges / OnlineBackup / InternetService / MultipleLines / PhoneService* 与Churn相关系数极小。
  - *TotalCharges*与其他变量相关系数均很小。

**热力图效果不是很显著。**

```
#独热编码
df_onehot = pd.get_dummies(df1.iloc[:,:])
df_onehot
```

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

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

| | SeniorCitizen | tenure | MonthlyCharges | TotalCharges | Churn | gender_Female | gender_Male | Partner_No | Partner_Yes | Dependents_N |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 29.85 | 29.85 | 0 | 1 | 0 | 0 | 1 | 1 |
| **1** | 0 | 34 | 56.95 | 1889.50 | 0 | 0 | 1 | 1 | 0 | 1 |
| **2** | 0 | 2 | 53.85 | 108.15 | 1 | 0 | 1 | 1 | 0 | 1 |
| **3** | 0 | 45 | 42.30 | 1840.75 | 0 | 0 | 1 | 1 | 0 | 1 |
| **4** | 0 | 2 | 70.70 | 151.65 | 1 | 1 | 0 | 1 | 0 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **7038** | 0 | 24 | 84.80 | 1990.50 | 0 | 0 | 1 | 0 | 1 | 0 |
| **7039** | 0 | 72 | 103.20 | 7362.90 | 0 | 1 | 0 | 0 | 1 | 0 |
| **7040** | 0 | 11 | 29.60 | 346.45 | 0 | 1 | 0 | 0 | 1 | 0 |
| **7041** | 1 | 4 | 74.40 | 306.60 | 1 | 0 | 1 | 0 | 1 | 1 |
| **7042** | 0 | 66 | 105.65 | 6844.50 | 0 | 0 | 1 | 1 | 0 | 1 |

7021 rows × 46 columns

- 独热编码的相关系数细分将变量数值拆分成子变量，研究自变量和Churn之间的相关性大小，进一步研究变量与Churn的相关性。
- gender和phoneservice不相关，所以继续drop phoneservice，用drop之后的var进行卡方检验频数比较。

```
df1.drop("PhoneService",axis=1,inplace=True)
```

```
var.remove('PhoneService')
```

```
var.remove('gender')
```

```
---------------------------------------------------------------------------

ValueError                                Traceback (most recent call last)

<ipython-input-35-3cb360e98008> in <module>
----> 1 var.remove('gender')
```

```
ValueError: list.remove(x): x not in list
```

```
#交叉分析
print('kf_var与Churn的进行交叉分析','\n')
for item in var:
    print("---------Churn by {}---------".format(item))
    print(pd.crosstab(df2.Churn,df2[item],normalize=0),'\n')
```

```
kf_var与Churn的进行交叉分析

---------Churn by SeniorCitizen---------
SeniorCitizen         0         1
Churn
0              0.871030  0.128970
1              0.744211  0.255789

---------Churn by Partner---------
Partner         0         1
Churn
0        0.529241  0.470759
1        0.360258  0.639742

---------Churn by Dependents---------
Dependents         0         1
Churn
0           0.654531  0.345469
1           0.824448  0.175552

---------Churn by MultipleLines---------
MultipleLines         0         1         2
Churn
0              0.099148  0.490124  0.410728
1              0.091546  0.450727  0.457728
```

```

---------Churn by InternetService---------
InternetService         0         1         2
Churn
0                0.379938  0.348373  0.271689
1                0.246096  0.695207  0.058697

---------Churn by OnlineSecurity---------
OnlineSecurity         0         1         2
Churn
0               0.394462  0.333850  0.271689
1               0.782445  0.158858  0.058697

---------Churn by OnlineBackup---------
OnlineBackup         0         1         2
Churn
0             0.369094  0.359218  0.271689
1             0.281637  0.659666  0.058697

---------Churn by DeviceProtection---------
DeviceProtection         0         1         2
Churn
0                 0.364833  0.363478  0.271689
1                 0.647819  0.293484  0.058697

---------Churn by TechSupport---------
TechSupport         0         1         2
Churn
0            0.392525  0.335786  0.271689
1            0.774367  0.166936  0.058697

---------Churn by StreamingTV---------
StreamingTV         0         1         2
Churn
0            0.361735  0.366576  0.271689
1            0.502962  0.438341  0.058697

---------Churn by StreamingMovies---------
StreamingMovies         0         1         2
Churn
0                0.357668  0.370643  0.271689
1                0.500808  0.440495  0.058697

---------Churn by Contract---------
Contract         0         1         2
Churn
0         0.427963  0.253098  0.318939
1         0.884760  0.089391  0.025848

---------Churn by PaperlessBilling---------
PaperlessBilling         0         1
Churn
0                 0.536406  0.463594
1                 0.749058  0.250942

---------Churn by PaymentMethod---------
PaymentMethod         0         1         2         3
Churn
0              0.250581  0.250581  0.249032  0.249806
1              0.573506  0.162628  0.138934  0.124933
```

- Crosstab中若变量取值对应的Churn - Yes的百分比差异越大，说明该变量对Churn - Yes的影响越显著。
  - SeniorCitizen：在年轻用户流失、留存的占比都很高。
  - Partner：单身越流失。
  - Dependents：经济不独立越流失。
  - StreamingMovies/StreamTvs/Multiplelines：不显著。
  - InternetService：Fiber Optic更易流失。
  - OnlineSecurity/OnlineBackup/DeviceProtection/TechSupport：没开通容易流失。
  - Contract：逐月订阅易流失。
  - Check：电子支票易流失。

```python
from scipy import stats
def ANOVA(x):
    index_list = list(df2['Churn'].value_counts().keys())
    args=[]
    for i in index_list:
```

```
        args.append(df2[df2['Churn']==i][x])
    w,p=stats.levene(*args) #齐性检验
    if p < 0.05:
        print('Churn By {}，p值是{:.2f}，小于0.05，表明方差齐性检验不通过，不可做方差分析。'.format(x,p),'\n')
    else:
        f,p_value = stats.f_oneway(*args)#方差检验
        print('Churn By {},f值是{:.2f}，p值是{:.2f}。'.format(x,f,p_value),'\n')
        if p_value <0.05:
            print("Churn by {}有显著性差异，可进行均值比较。".format(x),'\n')
        else:
            print("Churn by {}没有显著性差异，不可进行均值比较。".format(x),'\n')

```

```
print("MonthlyCharges和TotalCharges齐性检验和方差分析，如下：",'\n')
ANOVA('MonthlyCharges')
ANOVA('TotalCharges')
```

```
MonthlyCharges和TotalCharges齐性检验和方差分析，如下：

Churn By MonthlyCharges,f值是3.34，p值是0.07。

Churn by MonthlyCharges没有显著性差异，不可进行均值比较。

Churn By TotalCharges,f值是5.13，p值是0.02。

Churn by TotalCharges有显著性差异，可进行均值比较。
```

```
df1[["MonthlyCharges","TotalCharges"]]
```

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

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

| | MonthlyCharges | TotalCharges |
|---|---|---|
| **0** | 29.85 | 29.85 |
| **1** | 56.95 | 1889.50 |
| **2** | 53.85 | 108.15 |
| **3** | 42.30 | 1840.75 |
| **4** | 70.70 | 151.65 |
| **...** | ... | ... |
| **7038** | 84.80 | 1990.50 |
| **7039** | 103.20 | 7362.90 |
| **7040** | 29.60 | 346.45 |
| **7041** | 74.40 | 306.60 |
| **7042** | 105.65 | 6844.50 |

7021 rows × 2 columns

- MonthlyCharges & TotalCharges 量纲差异大。
- gender，id，PhoneService对Churn影响不显著，应该drop。

```
#标准化
from sklearn.preprocessing import StandardScaler
#sklearn.preprocessing.StandardScaler(copy=True, with_mean=True, with_std=True)
scaler = StandardScaler(copy=False)
scaler.fit_transform(df1[['MonthlyCharges','TotalCharges']]) #拟合数据
df1[['MonthlyCharges','TotalCharges']]=scaler.transform(df1[['MonthlyCharges','TotalCharges']]) #数据标准化
df1[['MonthlyCharges','TotalCharges']].head()
```

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

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

| | MonthlyCharges | TotalCharges |
|---|---|---|
| **0** | -1.164135 | -0.997334 |
| **1** | -0.262811 | -0.176352 |
| **2** | -0.365914 | -0.962766 |
| **3** | -0.750058 | -0.197874 |
| **4** | 0.194503 | -0.943562 |

```
# 将分类数据转化成整数编码
# 获取分类变量的标签值
def Labs(x):
    print(x,"--",df1[x].unique())
df_obj = df1.select_dtypes(['object'])
print(list(map(Labs,df_obj)))
```

```
gender -- ['Female' 'Male']
Partner -- ['Yes' 'No']
Dependents -- ['No' 'Yes']
MultipleLines -- ['No phone service' 'No' 'Yes']
InternetService -- ['DSL' 'Fiber optic' 'No']
OnlineSecurity -- ['No' 'Yes' 'No internet service']
OnlineBackup -- ['Yes' 'No' 'No internet service']
DeviceProtection -- ['No' 'Yes' 'No internet service']
TechSupport -- ['No' 'Yes' 'No internet service']
StreamingTV -- ['No' 'Yes' 'No internet service']
StreamingMovies -- ['No' 'Yes' 'No internet service']
Contract -- ['Month-to-month' 'One year' 'Two year']
PaperlessBilling -- ['Yes' 'No']
PaymentMethod -- ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
[None, None, None, None, None, None, None, None, None, None, None, None, None, None]
```

- 将No xxx serice合并进No

```
df1.replace(to_replace='No internet service',value = 'No',inplace=True)
df1.replace(to_replace='No phone service',value='No',inplace=True)
df_obj = df1.select_dtypes(['object'])
print(list(map(Labs,df_obj)))
```

```
gender -- ['Female' 'Male']
Partner -- ['Yes' 'No']
Dependents -- ['No' 'Yes']
MultipleLines -- ['No' 'Yes']
InternetService -- ['DSL' 'Fiber optic' 'No']
OnlineSecurity -- ['No' 'Yes']
OnlineBackup -- ['Yes' 'No']
DeviceProtection -- ['No' 'Yes']
TechSupport -- ['No' 'Yes']
StreamingTV -- ['No' 'Yes']
StreamingMovies -- ['No' 'Yes']
Contract -- ['Month-to-month' 'One year' 'Two year']
PaperlessBilling -- ['Yes' 'No']
PaymentMethod -- ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
[None, None, None, None, None, None, None, None, None, None, None, None, None, None]
```

```
import sklearn #特征工程
from sklearn import preprocessing  #数据预处理
from sklearn.preprocessing import LabelEncoder #编码转换
def labelencoder(x):
    df1[x]=LabelEncoder().fit_transform(df1[x])
for i in range (len(df_obj.columns)):
    labelencoder(df_obj.columns[i])
print(list(map(Labs,df_obj.columns)))
```

```
gender -- [0 1]
Partner -- [1 0]
Dependents -- [0 1]
MultipleLines -- [0 1]
InternetService -- [0 1 2]
OnlineSecurity -- [0 1]
OnlineBackup -- [1 0]
DeviceProtection -- [0 1]
TechSupport -- [0 1]
StreamingTV -- [0 1]
StreamingMovies -- [0 1]
Contract -- [0 1 2]
PaperlessBilling -- [1 0]
PaymentMethod -- [2 3 0 1]
[None, None, None, None, None, None, None, None, None, None, None, None, None, None]
```

```
list(map(Labs,df1.columns))
```

```
gender -- [0 1]
SeniorCitizen -- [0 1]
Partner -- [1 0]
Dependents -- [0 1]
tenure -- [ 1 34  2 45  8 22 10 28 62 13 16 58 49 25 69 52 71 21 12 30 47 72 17 27
  5 46 11 70 63 43 15 60 18 66  9  3 31 50 64 56  7 42 35 48 29 65 38 68
 32 55 37 36 41  6  4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26  0
 39]
MultipleLines -- [0 1]
InternetService -- [0 1 2]
OnlineSecurity -- [0 1]
OnlineBackup -- [1 0]
DeviceProtection -- [0 1]
TechSupport -- [0 1]
StreamingTV -- [0 1]
StreamingMovies -- [0 1]
Contract -- [0 1 2]
PaperlessBilling -- [1 0]
PaymentMethod -- [2 3 0 1]
MonthlyCharges -- [-1.16413536 -0.26281076 -0.36591432 ... -0.05826662 -0.68686569
  0.46057706]
TotalCharges -- [-0.99733366 -0.17635202 -0.96276648 ... -0.85756393 -0.87515655
  2.01113704]
Churn -- [0 1]
```

```
[None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None]
```

```
# #处理样本不平衡，分拆变量
# df1.drop("gender",axis=1,inplace=True)
# df1.drop("PhoneService",axis=1,inplace=True)
```

```
x=df1[var]
y=df1['Churn'].values
x
```

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

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

| | SeniorCitizen | Partner | Dependents | MultipleLines | InternetService | OnlineSecurity | OnlineBackup | DeviceProtection | TechSupport |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **1** | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| **2** | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| **3** | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| **4** | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **7038** | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| **7039** | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| **7040** | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| **7041** | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| **7042** | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |

7021 rows × 14 columns

```
from sklearn.model_selection import StratifiedShuffleSplit #分层抽样
from sklearn.model_selection import train_test_split #数据集训练集划分
```

```
#分层抽样stratified random sampling、过抽样、欠抽样，抽样上面多试错
sss=StratifiedShuffleSplit(n_splits=5,test_size=.2,random_state=0)
print(sss)
print(sss.split(x,y))
```

```
StratifiedShuffleSplit(n_splits=5, random_state=0, test_size=0.2,
            train_size=None)
<generator object BaseShuffleSplit.split at 0x000001EF6EA42900>
```

```
print("训练数据和测试数据被分成的份数：",sss.get_n_splits(x,y))
#拆分训练集和测试集
for train_index,test_index,in sss.split(x,y):
    print("train:",train_index,"test:",test_index)
    x_train,x_test=x.iloc[train_index],x.iloc[test_index]
    y_train,y_test= y[train_index],y[test_index]
```

```
训练数据和测试数据被分成的份数： 5
train: [5297 5907 3429 ... 4096 6084 3612] test: [4979 2569 5247 ... 1572 4876 4997]
train: [4203 5971  767 ... 1505  230 4637] test: [3201  692  688 ... 4736 3769 5207]
train: [5070 2818 1921 ... 4575 6509 1607] test: [1213 3852 1396 ... 1855 2852 1846]
train: [1468 2332 1900 ... 6038 5207  943] test: [5733 3682 4429 ... 6390  944 6816]
train: [5861 1463 4124 ... 4026 6659 4286] test: [6811 5731 3968 ...    2 4805 6708]
```

```
print("分层抽样数据特征：",x.shape,"train特征:",x_train_.shape,"test特征：",x_test_.shape)
print("分层抽样数据特征：",y.shape,"train特征:",y_train_.shape,"test特征：",y_test_.shape)
```

```
分层抽样数据特征： (7021, 14) train特征：(5616, 14) test特征： (1405, 14)
分层抽样数据特征： (7021,) train特征：(5616,) test特征： (1405,)
```

```python
# sklearn.linear_model.RidgeCV(alphas=(0.1, 1.0, 10.0), fit_intercept=True, normalize=False, scoring=None, cv=None, gcv_mode=None, store_cv_values=False)
from sklearn.linear_model import RidgeClassifier, RidgeCV # 岭回归
from sklearn.metrics import accuracy_score
rcv = RidgeClassifier()
rcv.fit(x_train,y_train)
pred = rcv.predict(x_test)
print(accuracy_score(y_test,pred))
```

```python
from sklearn.model_selection import train_test_split #数据集训练集划分
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.2,random_state=22)
```

```python
# 训练分类模型
from sklearn import metrics
from sklearn.metrics import recall_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
from sklearn.ensemble import RandomForestClassifier #随机森林
from sklearn.svm import SVC#支持向量机
from sklearn.linear_model import LogisticRegression #逻辑回归
from sklearn.neighbors import KNeighborsClassifier #k邻近算法
from sklearn.naive_bayes import GaussianNB #朴素贝叶斯
from sklearn.tree import DecisionTreeClassifier #决策树
from sklearn.ensemble import AdaBoostClassifier #分类器算法
from sklearn.ensemble import GradientBoostingClassifier #梯度提升
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
from sklearn.linear_model import RidgeClassifier # 岭
from sklearn.neural_network import MLPClassifier #神经网络
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from xgboost import XGBClassifier
import time
```

```python
Classifiers = [["Random Forest",RandomForestClassifier()],
             ["Support Vector Machine",SVC()],
             ["LogisticRegression",LogisticRegression()],
             ["KNeighbor",KNeighborsClassifier(n_neighbors=5)],
             ["Naive Bayes",GaussianNB()],
             ["Decision Tree",DecisionTreeClassifier()],
             ["GradientBoostingClassifier",GradientBoostingClassifier()],
             ["XGB",XGBClassifier()],
             ["CatBoost",CatBoostClassifier(logging_level='Silent')],
              ['RidgeClassifier',RidgeClassifier()],
               ['MLPClassifier',MLPClassifier(solver='lbfgs',activation = 'tanh',
                    max_iter = 50,alpha = 0.001,
                    hidden_layer_sizes = (10,30),
                    random_state = 1,verbose = True)],
               ['SGDClassifier',SGDClassifier()],
               ['XGBClassifier',XGBClassifier()],
               ['BaggingClassifier',BaggingClassifier()],
               ['XGBClassifier',XGBClassifier()]
              ]
```

```python
import time
Classify_result=[]
names=[]
prediction=[]
for name,classifier in Classifiers:
    classifier=classifier
    t1 = time.time()
    classifier.fit(x_train,y_train)
    y_pred=classifier.predict(x_test)
    t2=time.time()
    precision=precision_score(y_test,y_pred)
    f1score = f1_score(y_test, y_pred)
    time_diff = t2 -t1
    class_eva=pd.DataFrame([precision,f1score,time_diff])
    Classify_result.append(class_eva)
    name=pd.Series(name)
    names.append(name)
```

```
    y_pred=pd.Series(y_pred)
    prediction.append(y_pred)
```

```
[16:03:33] WARNING: D:\Build\xgboost\xgboost-1.4.2.git\src\learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[16:03:39] WARNING: D:\Build\xgboost\xgboost-1.4.2.git\src\learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[16:03:39] WARNING: D:\Build\xgboost\xgboost-1.4.2.git\src\learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

```
names = pd.DataFrame(names)
names=names[0].tolist()
names
```

```
['Random Forest',
 'Support Vector Machine',
 'LogisticRegression',
 'KNeighbor',
 'Naive Bayes',
 'Decision Tree',
 'GradientBoostingClassifier',
 'XGB',
 'CatBoost',
 'RidgeClassifier',
 'MLPClassifier',
 'SGDClassifier',
 'XGBClassifier',
 'BaggingClassifier',
 'XGBClassifier']
```

```
result = pd.concat(Classify_result,axis=1)
```

```
result.columns =names
```

```
result.index = ["precision",'f1score',"time_diff"]
```

```
result.T
```

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

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

| | precision | f1score | time_diff |
|---|---|---|---|
| **Random Forest** | 0.546032 | 0.500728 | 0.553662 |
| **Support Vector Machine** | 0.624000 | 0.501608 | 1.780465 |
| **LogisticRegression** | 0.578544 | 0.477093 | 0.029593 |
| **KNeighbor** | 0.506702 | 0.507383 | 0.178499 |
| **Naive Bayes** | 0.507937 | 0.584475 | 0.005983 |
| **Decision Tree** | 0.476584 | 0.470748 | 0.012998 |
| **GradientBoostingClassifier** | 0.596215 | 0.548621 | 0.435386 |
| **XGB** | 0.550152 | 0.516405 | 0.482712 |
| **CatBoost** | 0.579618 | 0.530612 | 4.346667 |
| **RidgeClassifier** | 0.614973 | 0.411449 | 0.010227 |
| **MLPClassifier** | 0.612903 | 0.557185 | 0.662815 |
| **SGDClassifier** | 0.736842 | 0.136585 | 0.026943 |
| **XGBClassifier** | 0.550152 | 0.516405 | 0.475735 |
| **BaggingClassifier** | 0.526946 | 0.498584 | 0.144153 |
| **XGBClassifier** | 0.550152 | 0.516405 | 0.473733 |