exp 6 -银行潜客挖掘实验

数据清洗

观察总体数据

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  int64
 1   job             41188 non-null  object
 2   marital         41188 non-null  object
 3   education       41188 non-null  object
 4   default         41188 non-null  object
```

检查缺失数据

```
data.isnull().sum().sort_values(ascending=False)
```

```
y                 0
day_of_week       0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
duration          0
```

没有缺失数据，不需要进行删除或填充

删除age异常值

```
query = data.loc[:,'age']>0
data = data.loc[query,:]
```

年龄分箱

```
cutPoint = [0, 20, 30, 40, 50, 60, 70, 100, 200]
data['ageGroup'] = pd.cut(data['age'],cutPoint)
```

提取年龄WOE值作为一个特征

```
woe = {}
    woel = []
    good_t = sum(data['y']=="yes")
    bad_t = sum(data['y']=="no")
    for i, v in data.ageGroup.items():
        good = 0
        bad = 0
        if woe.get(v) != None:
            woel.append(woe[v])
        else:
            for j, vj in data.ageGroup.items():
                if vj == v:
                    if data.y[j] == "yes":
                        good += 1
                    else:
                        bad += 1
```

```
            woel.append(math.log((good/good_t+0.1)/(bad/bad_t+0.1)))
            woe[v] = math.log((good/good_t+0.1)/(bad/bad_t+0.1))
    data["age_woe"] = woel
```

进行one-hot编码

```
x = pd.get_dummies(data)
```

模型训练

KNN

```
data = pd.read_csv('bank-additional-full.csv', sep=';')
x_train, x_test, y_train, y_test = split_data(data)
y_pred = predict(x_train, x_test, y_train)
print_result(y_test, y_pred)
```

```
model precision:0.60 recall:0.49
```

结果显示，精准度为0.60，召回率为0.49

> 在运行代码时，发现传入清洗的数据为去除掉标签的数据，所以woe不能做了，故按照原本代码进行，得出的结果如下：
>
> ```
> data = pd.read_csv('bank-additional-full.csv', sep=';')
> x_train, x_test, y_train, y_test = split_data(data)
> y_pred = predict(x_train, x_test, y_train)
> print_result(y_test, y_pred)
> ```
>
> ```
> model precision:0.60 recall:0.49
> ```
>
> 即精准度与召回率相同，重复多次结果不变。

SVM

```
data = pd.read_csv('bank-additional-full.csv', sep=';')
x_train, x_test, y_train, y_test = split_data(data)
y_pred = predict(x_train, x_test, y_train)
print_result(y_test, y_pred)
```

```
model precision:0.66 recall:0.40
```

结果显示，精准度为0.66，召回率为0.40（该指标为多次调参，包括使用'poly','sigmoid','rbf'核函数以及调整惩罚项得到的较优结果）。该模型训练速度很慢。

> 如果使用WOE编码，则效果如下：
>
> ```
> data = pd.read_csv('bank-additional-full.csv', sep=';')
> x_train, x_test, y_train, y_test = split_data(data)
> y_pred = predict(x_train, x_test, y_train)
> print_result(y_test, y_pred)
> ```
>
> ```
> model precision:0.65 recall:0.42
> ```
>
> 精准度降低0.01，召回率提高0.02.

LR

```
data = pd.read_csv('bank-additional-full.csv', sep=';')
x_train, x_test, y_train, y_test = split_data(data)
y_pred = predict(x_train, x_test, y_train)
print_result(y_test, y_pred)
```

```
model precision:0.64 recall:0.42
```

结果显示，精准度为0.64，召回率为0.42。 该模型训练速度比较快。

> 如果使用自定义WOE编码，则效果如下：
>
> ```
> data = pd.read_csv('bank-additional-full.csv', sep=';')
> x_train, x_test, y_train, y_test = split_data(data)
> y_pred = predict(x_train, x_test, y_train)
> print_result(y_test, y_pred)
> ```
>
> ```
> model precision:0.66 recall:0.39
> ```

> 精准度提高了0.02，召回率降低了0.01.

DecisionTree

```
data = pd.read_csv('bank-additional-full.csv', sep=';')
x_train, x_test, y_train, y_test = split_data(data)
y_pred = predict(x_train, x_test, y_train)
print_result(y_test, y_pred)
```

```
model precision:0.51 recall:0.50
```

结果显示，精准度为0.51，召回率为0.50. 该模型训练速度非常快。

> 如果使用自定义MOE编码，则效果如下：
>
> ```
> data = pd.read_csv('bank-additional-full.csv', sep=';')
> x_train, x_test, y_train, y_test = split_data(data)
> y_pred = predict(x_train, x_test, y_train)
> print_result(y_test, y_pred)
> ```
>
> ```
> model precision:0.52 recall:0.51
> ```
>
> 精准度和召回率都提升了0.01.

MLPClassifier

```
data = pd.read_csv('bank-additional-full.csv', sep=';')
x_train, x_test, y_train, y_test = split_data(data)
y_pred = predict(x_train, x_test, y_train)
print_result(y_test, y_pred)
```

```
model precision:0.60 recall:0.53
```

结果显示，精准度为0.60，召回率为0.53. 该模型训练速度在迭代次数小于1000次时比较快。但是在100-1000次迭代中产生的结果没有差异。该模型训练速度中等。

> 如果使用自定义MOE编码，则效果如下：
>
> ```
> data = pd.read_csv('bank-additional-full.csv', sep=';')
> x_train, x_test, y_train, y_test = split_data(data)
> y_pred = predict(x_train, x_test, y_train)
> print_result(y_test, y_pred)
> ```
>
> ```
> model precision:0.36 recall:0.94
> ```
>
> 精准度降低0.15，召回率提高0.44.

RandomForest

```
data = pd.read_csv('bank-additional-full.csv', sep=';')
x_train, x_test, y_train, y_test = split_data(data)
y_pred = predict(x_train, x_test, y_train)
print_result(y_test, y_pred)
```

```
model precision:0.75 recall:0.14
```

结果显示，精准度为0.75，召回率为0.14. 该模型设置最大树深度为1,2时，无法进行预测，以上结果为最大深度为3. 继续加大最大深度界，精准度约为0.73，召回率约到0.17，与上述结果无大差异。该模型训练速度中等。

> 如果使用自定义WOE编码，则效果如下：
>
> ```
> data = pd.read_csv('bank-additional-full.csv', sep=';')
> x_train, x_test, y_train, y_test = split_data(data)
> y_pred = predict(x_train, x_test, y_train)
> print_result(y_test, y_pred)
> ```
>
> ```
> model precision:0.79 recall:0.13
> ```
>
> 精准度有了0.04的提升。

AdaBoost

```
data = pd.read_csv('bank-additional-full.csv', sep=';')
x_train, x_test, y_train, y_test = split_data(data)
y_pred = predict(x_train, x_test, y_train)
print_result(y_test, y_pred)
```

```
model precision:0.67 recall:0.42
```

结果显示，精准度为0.67，召回率为0.42. 该模型训练速度较慢。

> 如果使用自定义WOE编码，则效果如下：
>
> ```
> data = pd.read_csv('bank-additional-full.csv', sep=';')
> x_train, x_test, y_train, y_test = split_data(data)
> y_pred = predict(x_train, x_test, y_train)
> print_result(y_test, y_pred)
> ```
>
> ```
> model precision:0.67 recall:0.42
> ```
>
> 效果不变

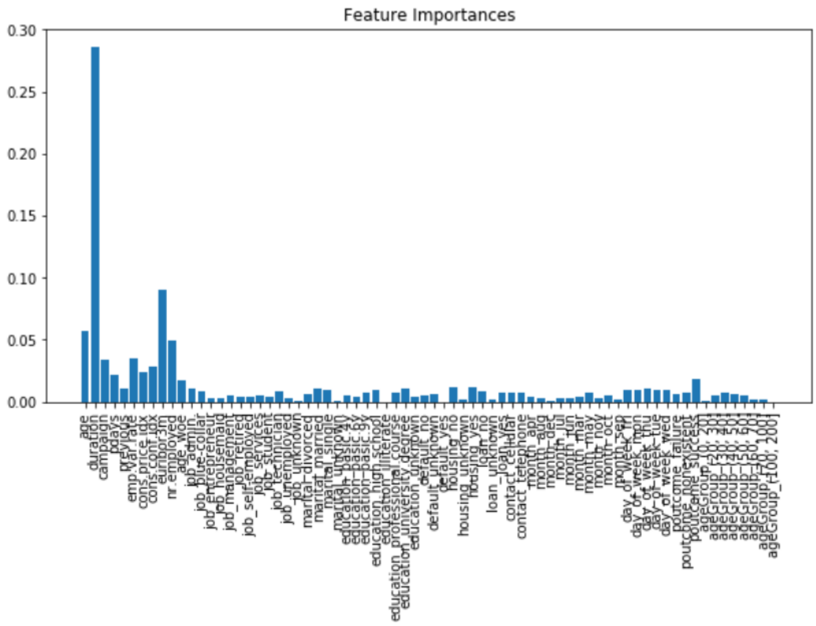
自定义LR与perceptron

由于数据类型的问题无法使用。

可视化分析

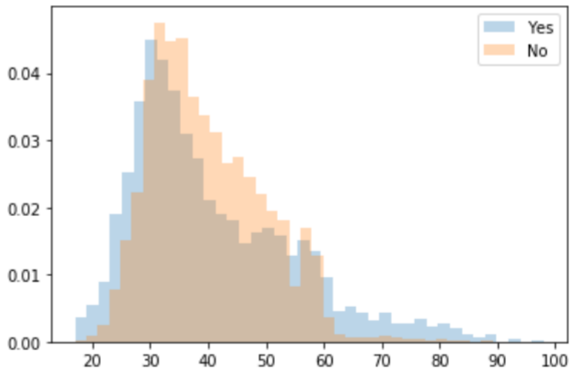
特征重要性分析

利用随机森林，绘制特征重要性图像：



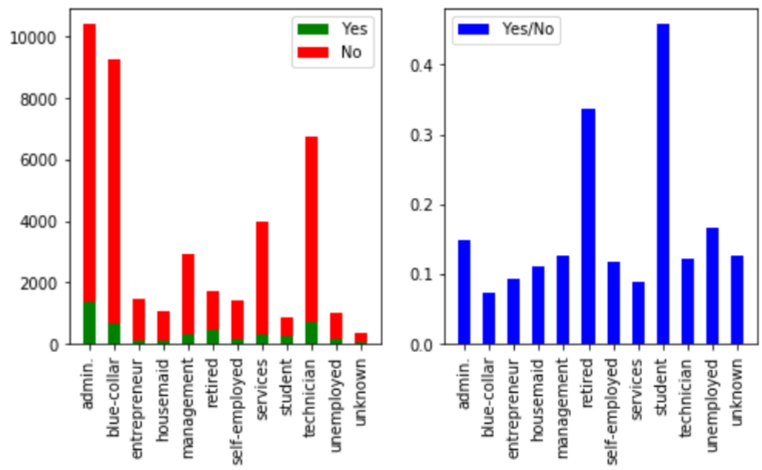
年龄结构

```
kwargs = dict(histtype='stepfilled', alpha=0.3, normed=True, bins=40)
plt.hist(data[data['y']=='yes']['age'],label = "Yes", **kwargs)
plt.hist(data[data['y']=='no']['age'],label = "No", **kwargs)
plt.legend()
plt.show()
```



职业分析

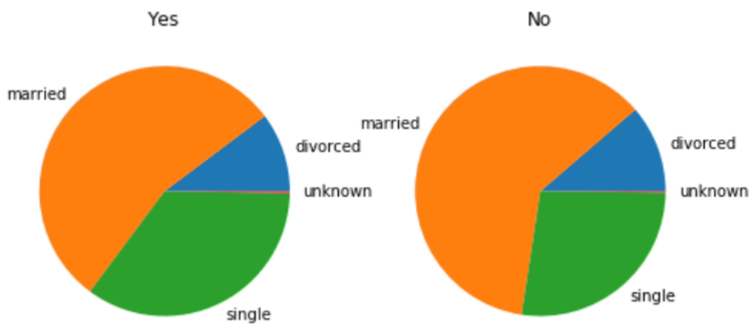
```
job = data[data['y']=='yes'].groupby('job').count()['y'].index
job_count_yes = data[data['y']=='yes'].groupby('job').count()['y']
job_count_no = data[data['y']=='no'].groupby('job').count()['y']
width = 0.5
plt.figure(figsize=(8, 4))
plt.subplot(121)
plt.bar(job,job_count_yes, width, color='green', label='Yes')
plt.bar(job,job_count_no, width, bottom = job_count_yes, color='red',
label='No')
plt.xticks(rotation=90)
plt.legend()
plt.subplot(122)
plt.bar(job,job_count_yes/job_count_no, width, color='blue', label='Yes/No')
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



可见，学生存款比例最高，退休工人次之。

婚姻状况

```python
marital = data[data['y']=='yes'].groupby('marital').count()['y'].index
marital_count_yes = data[data['y']=='yes'].groupby('marital').count()['y']
marital_count_no = data[data['y']=='no'].groupby('marital').count()['y']
plt.figure(figsize=(8, 4))
plt.subplot(121)
plt.title('Yes')
plt.pie(marital_count_yes.values,labels = marital)
plt.subplot(122)
plt.title('No')
plt.pie(marital_count_no.values,labels = marital)
plt.show()
```



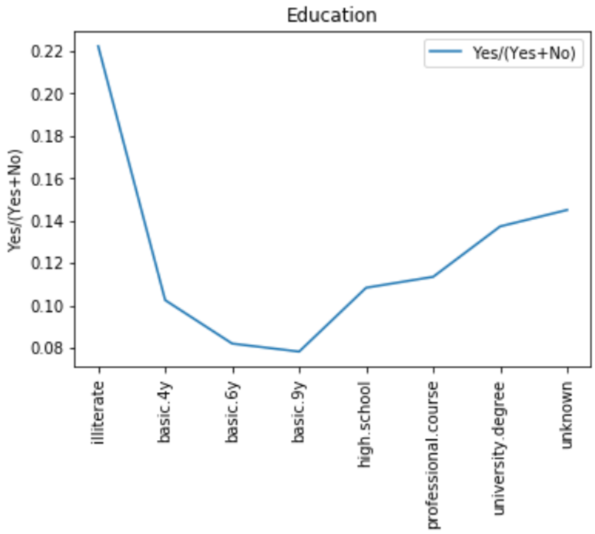
可见，在存款人中（左图），已婚人士与单身人士占比较大。在非存款人中（右图），已婚人士占比较大。

学历影响

```python
edu = ["illiterate", "basic.4y", "basic.6y", "basic.9y", "high.school", "professional.course", "university.degree","unknown"]

education_count_yes = data[data['y']=='yes'].groupby('education').count()['y']
education_count_no = data[data['y']=='no'].groupby('education').count()['y']
#按照学历对数据行重新排序
education_count_yes = education_count_yes.reindex(index=edu)
education_count_no = education_count_no.reindex(index=edu)

index = education_count_yes.index
fig = plt.figure(figsize=(6, 4))
axes=fig.add_subplot(1,1,1)
axes.plot((education_count_yes/(education_count_yes+education_count_no)).values,label = 'Yes/(Yes+No)')
axes.set_xticks(np.arange(len(edu)))
axes.set_xticklabels(edu)
axes.set_title("Education")
axes.set_ylabel('Yes/(Yes+No)')
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



可见，受中等教育的人存款的倾向比较低，而受教育贫乏的人群存款的倾向强烈。

结论

在本次实验中，对银行潜客信息进行了挖掘并处理分析，了解了金融营销的应用场景，学会了面对大数据的处理方式与预测方式，在多个预测函数中也通过不断调整参数来优化模型，同时通过参考资料，学会了分析数据于plt可视化，对于不熟悉的可视化方面也有了了解。

> 参考：https://codechina.csdn.net/mirrors/leungBH/BankMarketing/-/tree/master/code