导入类库

```
import pandas as pd
from sklearn.model_selection import learning_curve, train_test_split,GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from matplotlib import pyplot as plt
import seaborn as sns
```

数据加载

```
data = pd.read_csv('./UCI_Credit_Card.csv')
```

Name	Type	Size	Value
data	DataFrame	(30000, 25)	Column names: ID, LIMIT_BAL, SEX, EDUCATION, MARRIAGE, AGE, PAY_0, PAY ...

数据探索

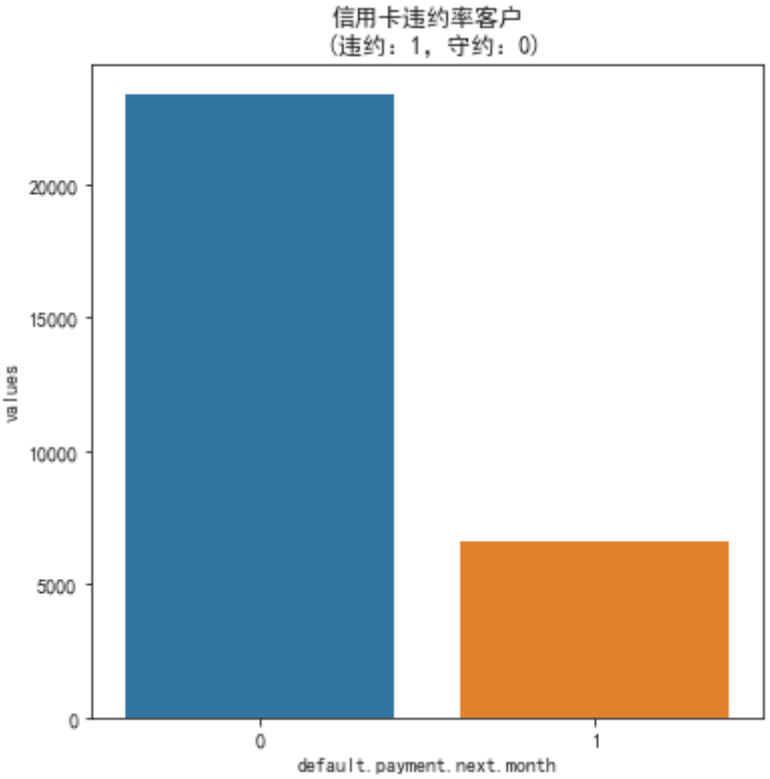
```
explore = data.describe(percentiles=[], include='all').T
# describe()函数自动计算非空值数，需要手动计算空值数
explore['null'] = len(data) - explore['count']

explore = explore[['null', 'max', 'min']]
explore.columns = [u'空值数', u'最大值', u'最小值']
```

Index	空值数	最大值	最小值
ID	0	30000	1
LIMIT_BAL	0	1e+06	10000
SEX	0	2	1
EDUCATION	0	6	0
MARRIAGE	0	3	0
AGE	0	79	21
PAY_0	0	8	-2
PAY_2	0	8	-2
PAY_3	0	8	-2
PAY_4	0	8	-2
PAY_5	0	8	-2
PAY_6	0	8	-2
BILL_AMT1	0	964511	-165580
BILL_AMT2	0	983931	-69777

通过条形图表探索数据

查看下一个月违约率的情况

```
next_month = data['default.payment.next.month'].value_counts()
print(next_month)
df = pd.DataFrame({'default.payment.next.month': next_month.index,'values':
next_month.values})
# 用来正常显示中文标签
plt.rcParams['font.sans-serif']=['SimHei']
plt.figure(figsize = (6,6))
plt.title('信用卡违约率客户\n (违约: 1, 守约: 0)')
sns.set_color_codes("pastel")
sns.barplot(x = 'default.payment.next.month', y="values", data=df)
locs, labels = plt.xticks()
plt.show()
```



特征选择

去掉ID字段、最后一个结果字段即可

```
#ID这个字段没有用,ID在列，所以删除的时候axis=1
data.drop(['ID'], inplace=True, axis =1)
#取出目标值
target = data['default.payment.next.month'].values
#取出列名
columns = data.columns.tolist()
#移除列名
columns.remove('default.payment.next.month')
#筛选出训练特征
features = data[columns].values
```

测试数据

target	int64	(30000,)	[1 1 0 ... 1 1 1]
features	float64	(30000, 23)	[[2.0000e+04 2.000 0.0000e+0 ...

拆分数据

```
# 30%作为测试集，其余作为训练集,stratify = target,y值得分布在训练和测试是一样的
train_x, test_x, train_y, test_y = train_test_split \
(features, target, test_size=0.30, stratify = target, random_state = 1)
```

测试数据拆分

test_x	float64	(9000, 23)
test_y	int64	(9000,)
train_x	float64	(21000, 23)
train_y	int64	(21000,)

分类器初始化

```
# 构造各种分类器
classifiers = [
    SVC(random_state = 1, kernel = 'rbf'),
    DecisionTreeClassifier(random_state = 1, criterion = 'gini'),
    RandomForestClassifier(random_state = 1, criterion = 'gini'),
    KNeighborsClassifier(metric = 'minkowski'),
]
# 分类器名称
classifier_names = [
            'svc',
            'decisiontreeclassifier',
            'randomforestclassifier',
            'kneighborsclassifier',
]
# 分类器参数
classifier_param_grid = [
            {'svc__C':[1], 'svc__gamma':[0.01]},
            {'decisiontreeclassifier__max_depth':[6,9,11]},
            {'randomforestclassifier__n_estimators':[3,5,6]} ,
            {'kneighborsclassifier__n_neighbors':[4,6,8]},
]
```

调试参数设计，决策树3种最大深度[3,6,9]

随机森林设置3个决策树个数的取值[3,5,6]

knn取3个n值调试[4,6,8]

训练分类器及测试

```
# 对具体的分类器进行GridSearchCV参数调优，返回response作为最后结果
def GridSearchCV_work(pipeline, train_x, train_y, test_x, test_y, param_grid, score =
'accuracy'):
```

```
    response = {}
    gridsearch = GridSearchCV(estimator = pipeline, param_grid = param_grid, scoring =
score)
    # 寻找最优的参数 和最优的准确率分数
    search = gridsearch.fit(train_x, train_y)
    print("GridSearch最优参数: ", search.best_params_)
    print("GridSearch最优分数:  %0.4lf" %search.best_score_)
    predict_y = gridsearch.predict(test_x)
    print("准确率 %0.4lf" %accuracy_score(test_y, predict_y))
    response['predict_y'] = predict_y
    response['accuracy_score'] = accuracy_score(test_y,predict_y)
    return response

for model, model_name, model_param_grid in zip(classifiers, classifier_names,
classifier_param_grid):
    pipeline = Pipeline([
            ('scaler', StandardScaler()),
            (model_name, model)
    ])
    result = GridSearchCV_work(pipeline, train_x, train_y, test_x, test_y,
model_param_grid , score = 'accuracy')
```

GridSearchCV是 Python 的参数自动搜索模块。 我们只要告诉它想要调优的参数有哪些以及参数的取值范围，它就会把所有的情况都跑一遍，然 后告诉我们哪个参数是最优的，结果如何。

---	svm	dtclassfier	randomtree	knn
最优参数	{'svc__C': 1, 'svc__gamma': 0.01}	{'decisiontreeclassifier__max_depth': 6}	{'randomforestclassifier__n_estimators': 6}	{'kneighborsclassifier__n_neighbors': 8}
最优分数	0.8174	0.8186	0.8000	0.8040
准确率	0.8172	0.8113	0.7997	0.8036