客户流失判断-最终报告

1. 数据预处理

数据属性共15个，分别为：

- ID：编号
- Contract：是否有合同
- Dependents：是否有家属
- DeviceProtection：是否有设备保护
- IntenrnetService：是否有互联网服务
- MonthlyCharges：月度费用
- MultipleLines：是否有多条线路
- Partner：是否有配偶
- PaymentsMethod：付款方式
- PhoneService：是否有电话服务
- SeniorCitizen：是否为老年人
- TVProgram：是否有电视节目
- TotalCharges：总费用
- gender：性别
- tenure：任期年数

在训练集中，增加了标签属性：

- Label：用户是否流失

1.1 数据摘要

训练集和测试集数据样例个数分别为5227和1307，在训练集中客户流失比例约占37%。

各属性的数据类型如下：

```
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ID                5227 non-null   int64
 1   Contract          5227 non-null   object
 2   Dependents        5227 non-null   object
 3   DeviceProtection  5227 non-null   object
 4   InternetService   5227 non-null   object
 5   MonthlyCharges    5227 non-null   float64
 6   MultipleLines     5227 non-null   object
 7   Partner           5227 non-null   object
 8   PaymentMethod     5227 non-null   object
 9   PhoneService      5227 non-null   object
 10  SeniorCitizen     5227 non-null   int64
 11  TVProgram         5227 non-null   object
 12  TotalCharges      5227 non-null   float64
 13  gender            5227 non-null   object
 14  tenure            5227 non-null   int64
 15  Label             5227 non-null   object
dtypes: float64(2), int64(3), object(11)
```

查看各属性取值发现如下:

```
(ID                5227
 Contract             3
 Dependents           2
 DeviceProtection     3
 InternetService      3
 MonthlyCharges    2620
 MultipleLines        3
 Partner              2
 PaymentMethod        4
 PhoneService         2
 SeniorCitizen        2
 TVProgram            3
 TotalCharges      5016
 gender               2
 tenure              73
 Label                2
 dtype: int64,
 ID                1307
 Contract             3
 Dependents           2
 DeviceProtection     3
 InternetService      3
 MonthlyCharges     900
 MultipleLines        3
 Partner              2
 PaymentMethod        4
 PhoneService         2
 SeniorCitizen        2
 TVProgram            3
 TotalCharges      1289
 gender               2
 tenure              72
 dtype: int64)
```

经观察后初步发现：

- PhoneService为No的MultipleLines为No phone service，二者存在关联
- InternetService为No的DeviceProtection，TVProgram为No internet service，存在关联
- tenure为0的8行数据对应的TotalCharfes为2283.300441，在TotalCharges中为最大值，考虑可能是为tenure为0的用户总费用设为最大值。
- 大部分属性的取值个数为2-4个，且数值类型为object，需要进行处理。

1.2 数据清洗

主要任务包括：

- 二值属性转为0/1，其中yes为1,No为0
- 多值属性用one-hot，去掉第一个，将处理后数据附加在数据后
- 包含No xx service的多值属性，one-hot值去掉该列
- 删除处理前的属性列

将数据类型为object，取值为yes/no的属性Dependents','Partner','PhoneService', 'Label'替代为1/0

```
collist = ['Dependents','Partner','PhoneService','Label']
# map函数
def binary_map(x):
    return x.map({'Yes':1,"No":0})
custmer_train[collist] = custmer_train[collist].apply(binary_map)
col = ['Dependents','Partner','PhoneService']
custmer_test[col] = custmer_test[col].apply(binary_map)
```

多值属性如'Contract','InternetService','PaymentMethod','gender'修改为One-hot属性并去掉第一个新属性。

```
# train one-hot
# 3个属性，新增2*2+3*1+1*1
dummy1 =
pd.get_dummies(custmer_train[['Contract','InternetService','PaymentMethod'
,'gender']],
                        drop_first=True)

# 拼接到源数据
custmer_train = pd.concat([custmer_train,dummy1],axis=1)

# test
dummy2 =
pd.get_dummies(custmer_test[['Contract','InternetService','PaymentMethod',
'gender']],
                        drop_first=True)
custmer_test = pd.concat([custmer_test,dummy2],axis=1)
```

针对一些具有no xx service属性值的数据如‘DeviceProtection’， ‘MultipleLines’, ‘TVProgram’， 考虑到该类取值较少且对结果影响较小，one-hot处理后去掉该属性。

将与上述操作相关的源数据删除后的数据类型如下：

```
RangeIndex: 5227 entries, 0 to 5226
Data columns (total 23 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   ID                                      5227 non-null   int64
 1   Dependents                              5227 non-null   int64
 2   MonthlyCharges                          5227 non-null   float64
 3   Partner                                 5227 non-null   int64
 4   PhoneService                            5227 non-null   int64
 5   SeniorCitizen                           5227 non-null   int64
 6   TotalCharges                            5227 non-null   float64
 7   tenure                                  5227 non-null   int64
 8   Label                                   5227 non-null   int64
 9   Contract_One year                       5227 non-null   uint8
 10  Contract_Two year                       5227 non-null   uint8
 11  InternetService_Fiber optic             5227 non-null   uint8
 12  InternetService_No                      5227 non-null   uint8
 13  PaymentMethod_Credit card (automatic)   5227 non-null   uint8
 14  PaymentMethod_Electronic check          5227 non-null   uint8
 15  PaymentMethod_Mailed check              5227 non-null   uint8
 16  gender_Male                             5227 non-null   uint8
 17  DeviceProtection_No                     5227 non-null   uint8
 18  DeviceProtection_Yes                    5227 non-null   uint8
 19  MultipleLines_No                        5227 non-null   uint8
 20  MultipleLines_Yes                       5227 non-null   uint8
 21  TVProgram_No                            5227 non-null   uint8
 22  TVProgram_Yes                           5227 non-null   uint8
```

查看多值属性的相关数值统计信息如下：

	tenure	MonthlyCharges	SeniorCitizen	TotalCharges
count	5227.000000	5227.000000	5227.000000	5227.000000
mean	28.775971	66.823765	0.118615	2084.477153
std	24.293077	28.862749	0.323366	2183.825066
min	0.000000	18.250000	0.000000	18.800000
25%	5.000000	45.000000	0.000000	292.979609
50%	23.000000	74.200000	0.000000	1218.650000
75%	51.000000	89.900000	0.000000	3373.825000
90%	67.000000	101.395283	1.000000	5683.670000
95%	71.000000	106.196180	1.000000	6632.480000
99%	72.000000	114.037000	1.000000	7937.495000
max	72.000000	118.600000	1.000000	8564.750000

以上四种数据分布范围较广，需要对其进行特征缩放处理。

1.3 特征缩放

由于tensure，MonthlyCharges，TotalCharges数值分布范围广，需要进行特征缩放。

将数据集按照7：3划分为训练集和测试集，使用StandardScaler()将三个属性缩放后的结果如下：

	Dependents	MonthlyCharges	Partner	PhoneService	SeniorCitizen	TotalCharges	tenure
834	1	-0.382688	0	1	0	0.269931	0.747357
800	0	-1.597290	0	1	0	-0.942472	-1.144483
273	1	-1.049766	0	0	0	-0.766467	-0.733213
4684	0	0.726221	0	1	1	-0.198104	-0.404198
5051	0	0.660379	1	1	0	-0.767157	-0.979975

1.4 特征选择

使用RFE对进行特征选择，得出17种特征如下：

```
[('Dependents', True, 1),
 ('MonthlyCharges', True, 1),
 ('Partner', True, 1),
 ('PhoneService', False, 4),
 ('SeniorCitizen', True, 1),
 ('TotalCharges', True, 1),
 ('tenure', True, 1),
 ('Contract_One year', True, 1),
 ('Contract_Two year', True, 1),
 ('InternetService_Fiber optic', True, 1),
 ('InternetService_No', True, 1),
 ('PaymentMethod_Credit card (automatic)', False, 5),
 ('PaymentMethod_Electronic check', True, 1),
 ('PaymentMethod_Mailed check', True, 1),
 ('gender_Male', False, 3),
 ('DeviceProtection_No', True, 1),
 ('DeviceProtection_Yes', True, 1),
 ('MultipleLines_No', True, 1),
 ('MultipleLines_Yes', True, 1),
 ('TVProgram_No', False, 2),
 ('TVProgram_Yes', True, 1)]
```

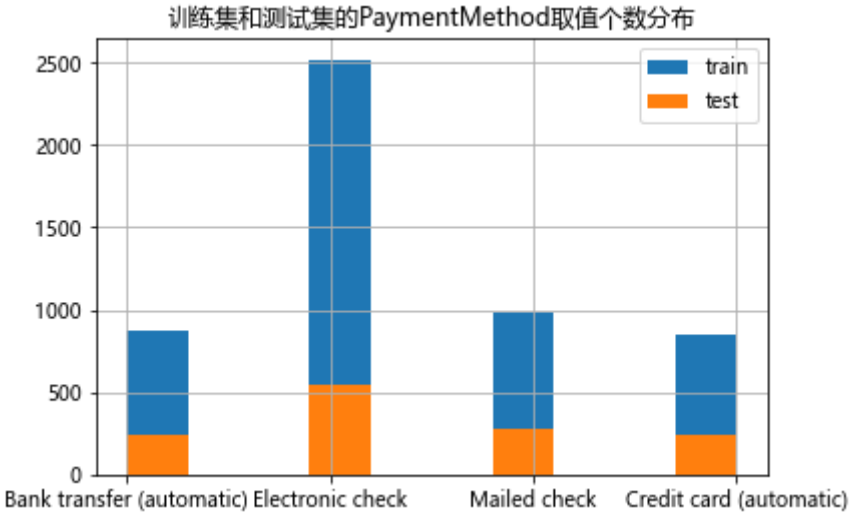
后续特征选择个数将作为超参数进行调整，选出模型表现最优的特征作为模型主要特征。

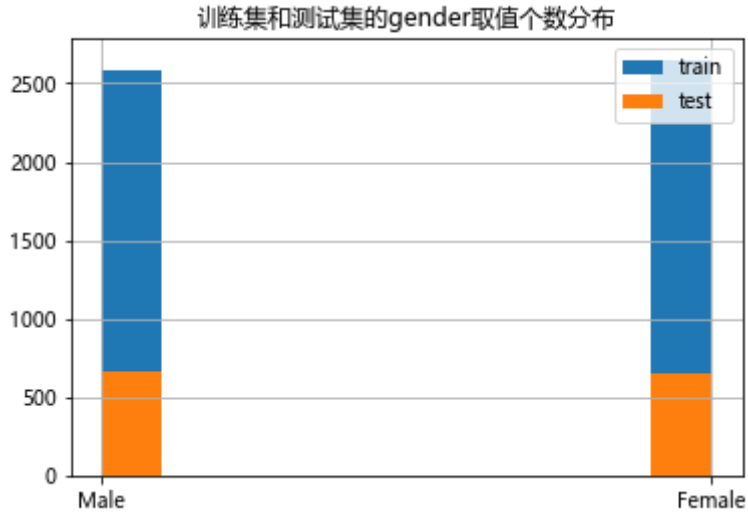
2. 数据可视化

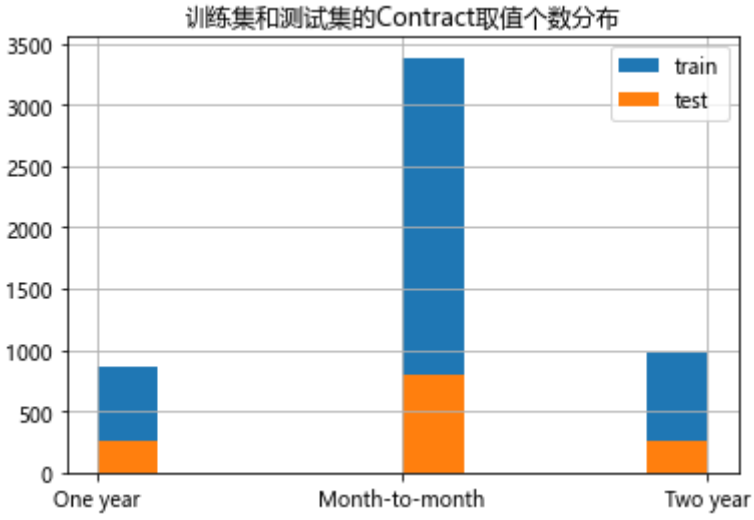
通过可视化技术对源数据进行探索性分析如下。

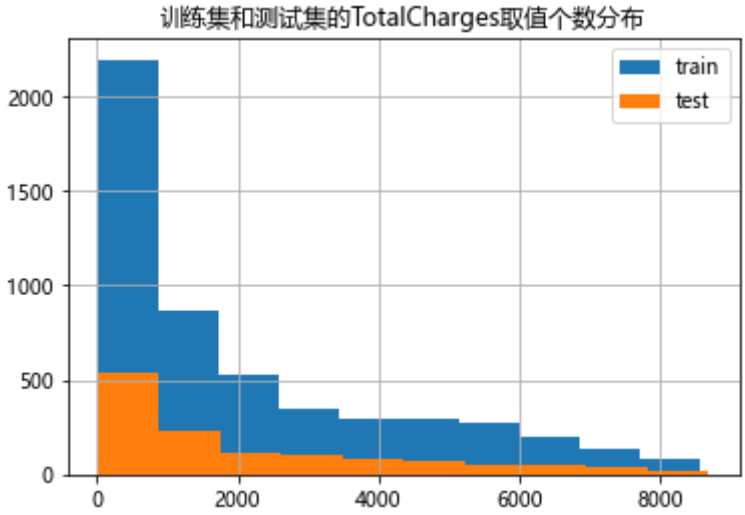
2.1 数据分布可视化

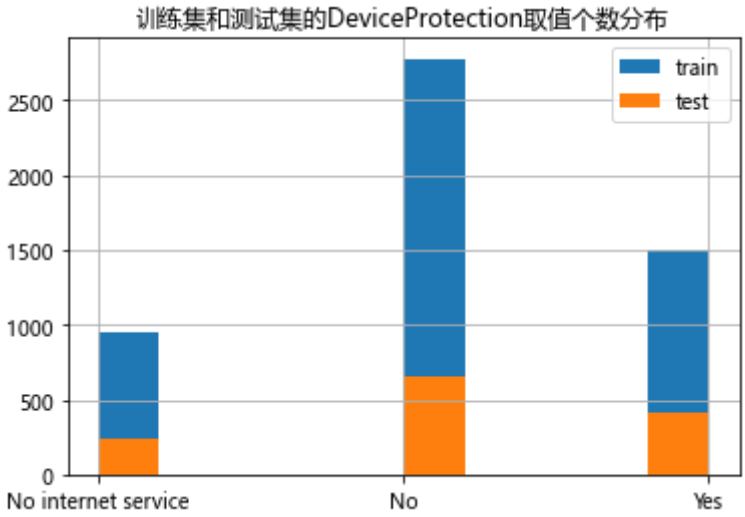
查看各属性在训练集和测试集的取值分布

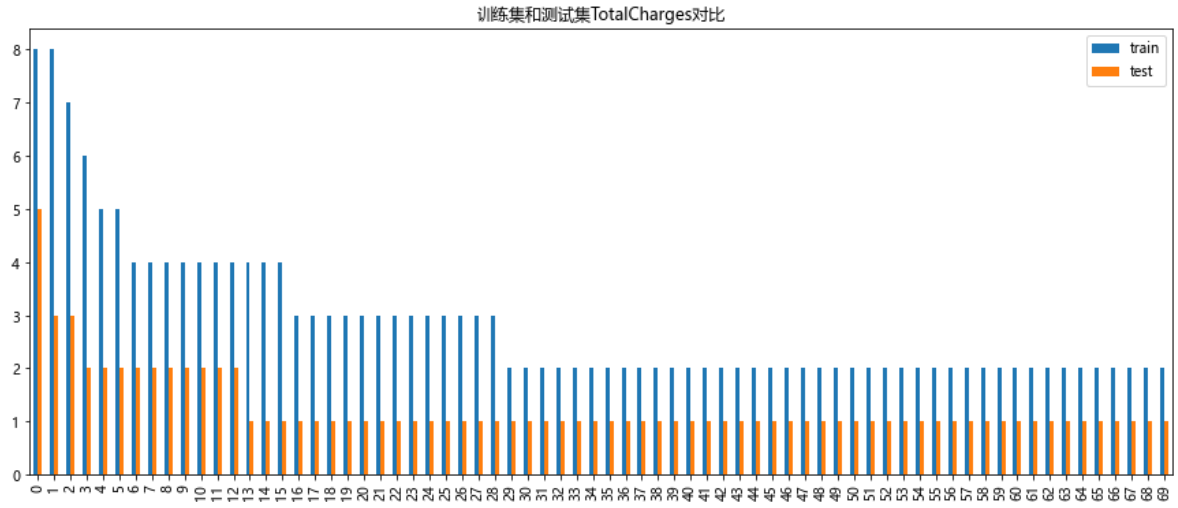

训练集和测试集的PaymentMethod取值个数分布
train
test
2500
2000
1500
1000
500
0
Bank transfer (automatic)
Electronic check
Mailed check
Credit card (automatic)


训练集和测试集的gender取值个数分布
train
test
2500
2000
1500
1000
500
0
Male
Female


训练集和测试集的Contract取值个数分布
train
test
3500
3000
2500
2000
1500
1000
500
0
One year
Month-to-month
Two year

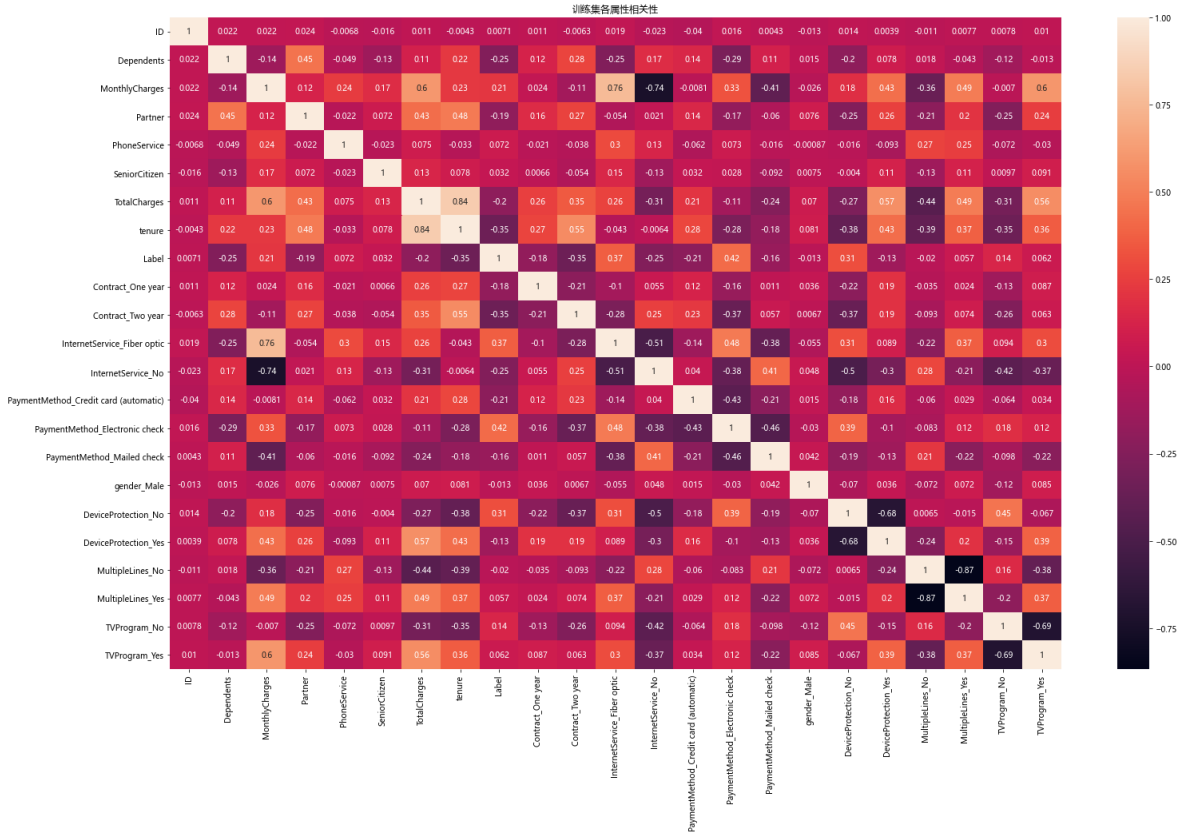






经上述分析，可以看出训练集和测试集在数据分布比例上基本一致。

2.2 相关性可视化

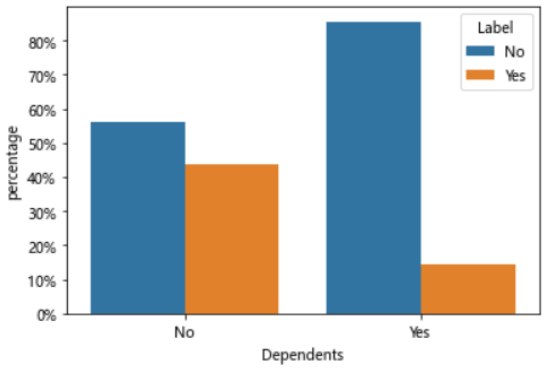


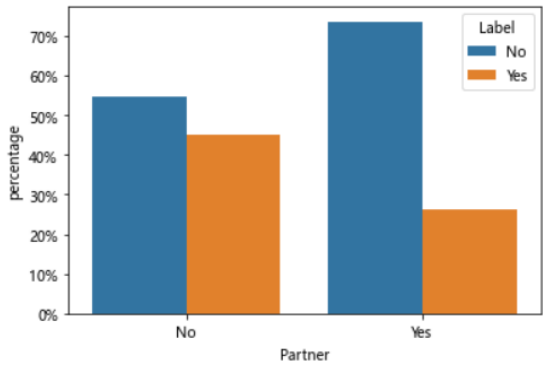
2.3 客户属性维度分析

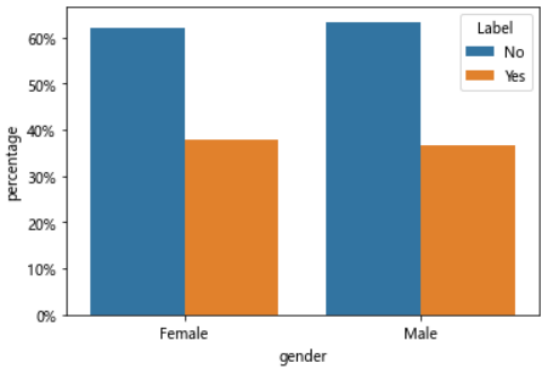
主要任务包括：

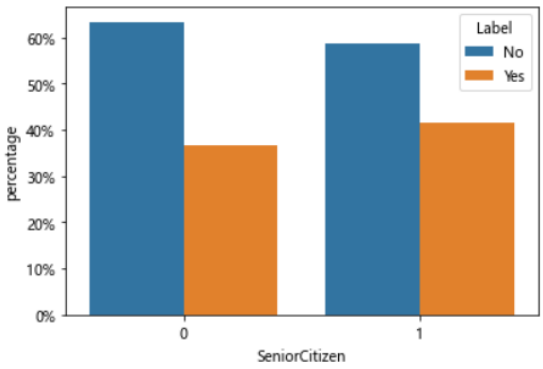
- 针对二值属性统计不同属性值的客户流失情况。
- 针对三值属性分析不同属性值的客户流失情况，衡量预处理的合理性。
- 针对合同签约期限分析不同期限的客户流失情况。
- 针对数值属性采用核函数绘图分析客户流失情况。

二值属性包括'Dependents','Partner','gender','SeniorCitizen'，分析四个属性与客户流失的关系如下：





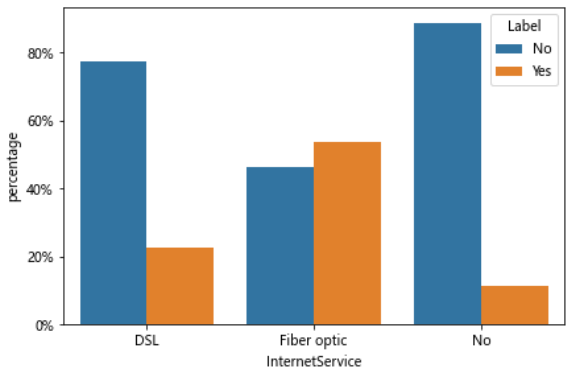


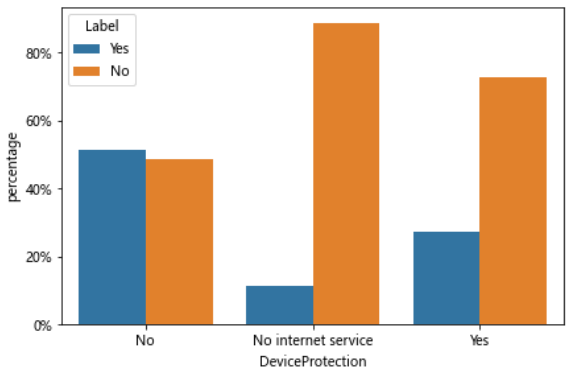


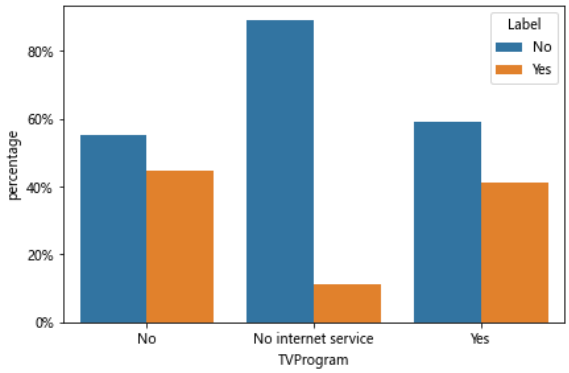
根据上述图表结果可得出以下结论：

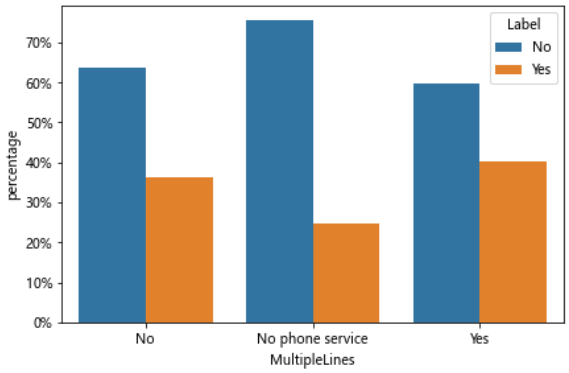
- 有家属和无家属的客户流失量相差最大，其中有家属客户流失量更少。
- 单身用户的客户流失量大于非单身用户，差距较为明显。
- 性别及年老与否对客户流失量几乎无影响。其中，老年人比非老年人的客户流失量高，但程度不大。

针对'InternetService','DeviceProtection','TVProgram','MultipleLines'属性，绘出关系图如下：









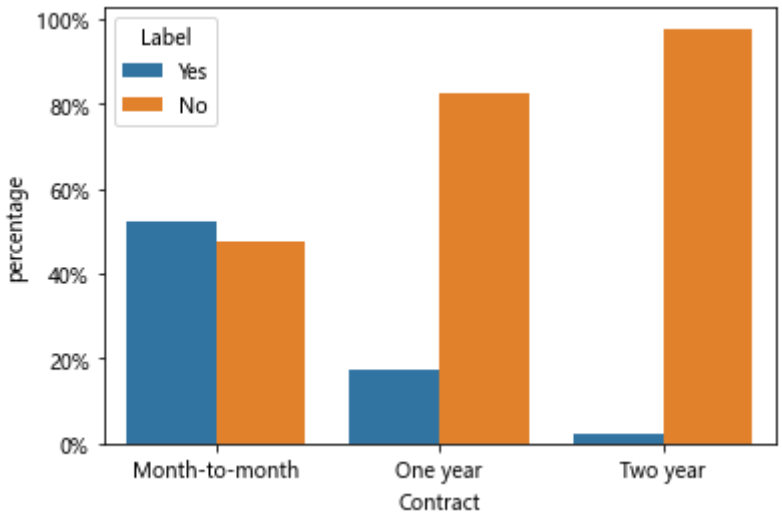
部分代码如图所示：

```
# 客户三值数据分析
def multi_barplot_percentages(df,features):
    plt.figure(figsize=(15,15))
    fig=plt.subplot()
    cnt=0
    for feature in features:
        g=df.groupby(by=feature)['Label'].value_counts(1).to_frame()
        g.rename(columns={'Label':'percentage'},inplace=True)
        g.reset_index(inplace=True)

        plt.subplot(321+cnt)
        ax=sns.barplot(x=feature,y='percentage', hue='Label', data=g)
        ax.set_yticklabels(['{:,.0%}'.format(y) for y in ax.get_yticks()])
        cnt+=1
    plt.show()
```

对客户使用情况进行分析后得出：

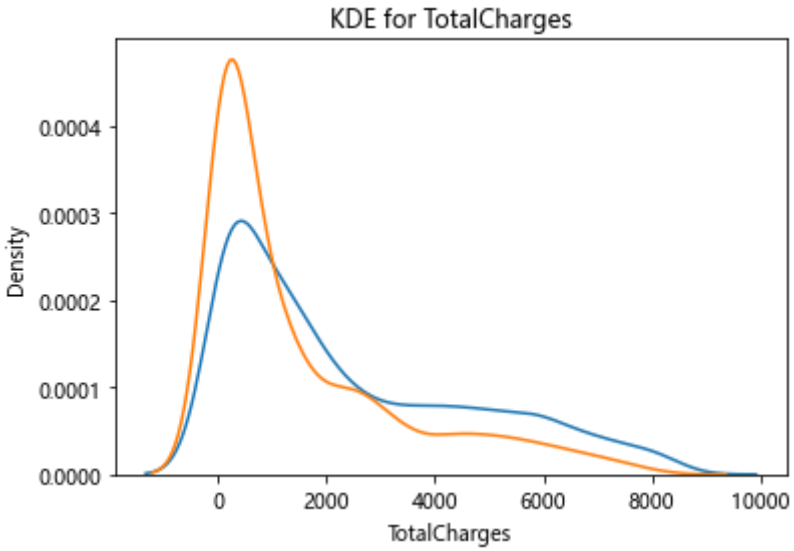
- 具有光纤服务，无设备保护的用户较其他用户易于流失。
- No Internet service在TVProgram以及DeviceProtection中客户流失较少，将两个属性作为one-hot时的删除项影响较小。

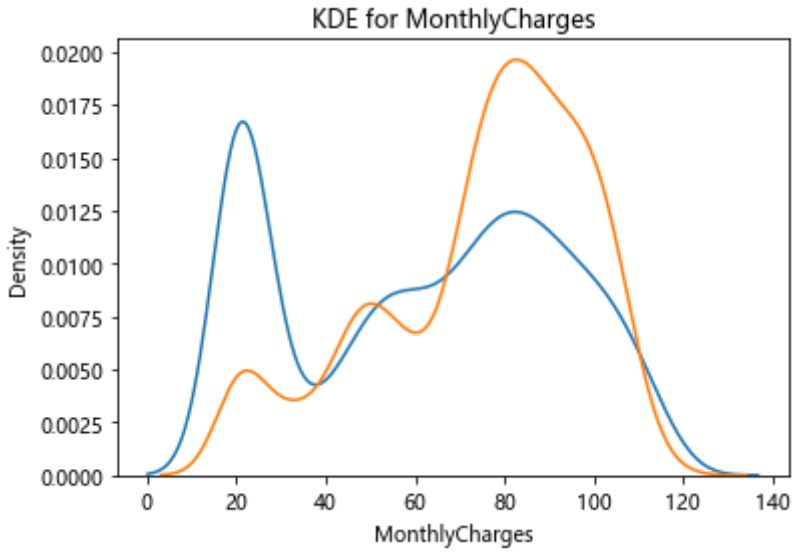
客户合同签约时间与流失情况关系如下：



可以看出，合同签约期限越长，客户流失越严重，说明该业务可能在长期内很难为用户提供满意的服务，从而造成客户流失，今后可能需要优化业务使其满足用户长期需要。

通过核函数分析数值属性的用户流失率如下：





说明在支付费用方面，月支付较高的用户更容易流失，可能由于价格过高使消费者望而却步；总费用较低的用户更容易流失，可能该业务仍需要优化。

2.4 总结

单身用户及无家属的独居人士更容易流失，可以继续深入调查这些用户的需求并改进业务，以达到扩展客户来源的效果。合同期限越长的用户越容易流失，说明产品不仅要考虑到短期效果好的目标，还要优化产品满足用户长期使用的需求，从而巩固老用户。针对支付费用方面，为了减少因月支付较高引起的客户流失，可以适当调整月费用。

3. 模型选取

3.1 GBDT

GBDT通过多轮迭代,每轮迭代产生一个弱分类器，每个分类器在上一轮分类器的残差基础上进行训练。由于训练的过程是通过降低偏差来不断提高最终分类器的精度，对弱分类器的要求一般是足够简单，并且是低方差和高偏差的。弱分类器一般会选择为CART TREE（也就是分类回归树），每个分类回归树的深度不会很深,最终的总分类器是将每轮训练得到的弱分类器加权求和得到的。

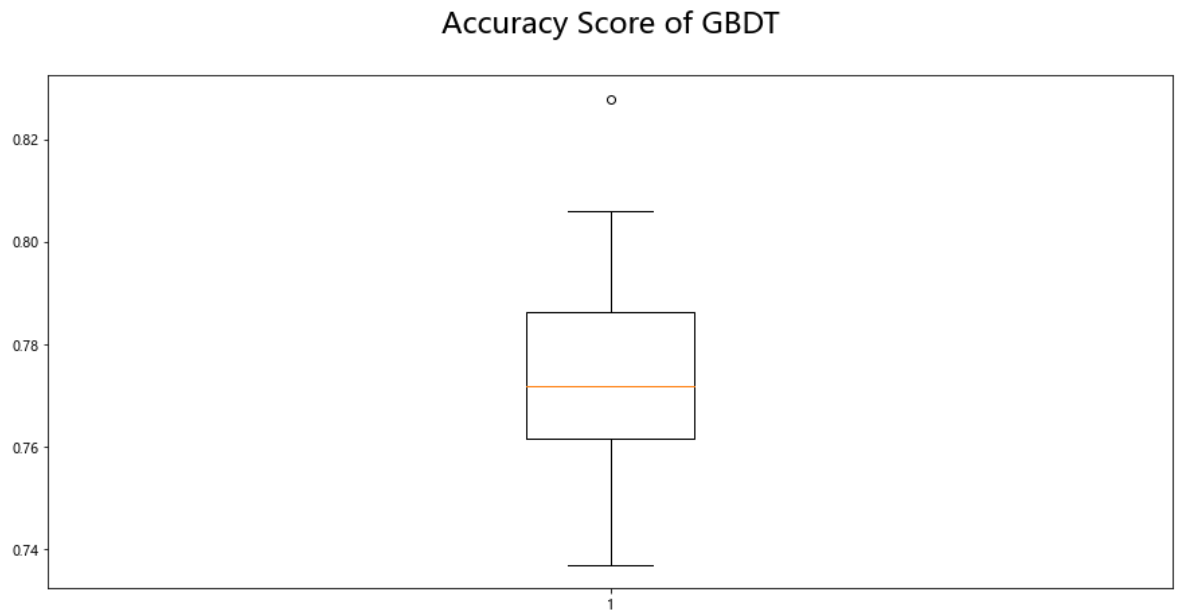
将训练集按照7：3划分后，使用特征选择后的17个主要特征训练GBDT分类器，通过10折交叉验证得出模型准确率，结果表明当选择max_depth=2,n_estimators=60时各评价指标达到当前最优。

模型效果评估如下：

GBDT预测效果

acc	pri	rec	auc
0.769	0.686	0.740	0.827

10次交叉验证的准确率盒图如下：



部分代码如下：

```
clf = Pipeline(steps=[("classifier",XGBClassifier(max_depth=2,n_estimators=60,n_jods=3

# 训练
clf.fit(X_train[col], y_train)
pickle.dump(clf, open("churn_xgb.model", "wb"))

# 预测
model = pickle.load(open("churn_xgb.model", "rb"))
predictions = model.predict(X_test[col])
predict_proba = model.predict_proba(X_test[col])[:, 1]

# 模型评价
print("acc:", accuracy_score(y_test, predictions))
print("pri:", precision_score(y_test, predictions))
print("rec:", recall_score(y_test, predictions))
print("auc:", roc_auc_score(y_test, predict_proba))
```

3.2 逻辑回归

逻辑回归（Logistic Regression）虽然被称为回归，但其实际上是分类模型，并且常常被用于二分类，而我们的问题预测客户流失也可以建模为一个二分类问题，流失为0，存在为1。

对于所有数据集假设一条直线可以将数据完成线性可分，决策边界可以用一个线性函数表示，逻辑回归需要在线性回归的基础上再加一层，它要找到分类概率与输出向量的直接关系，一般选用sigmoid函数，在求解过程中，我们一般选用最大化似然函数的方法进行求解。

实验首先对数据集的属性进行分析，其中ID为连续无意义属性，将其剔除，其余属性进行预处理后全部进行训练，对于训练和测试数据集的划分，我们随机取出100条数据集作为测试数据集，其余作为训练数据，最终验证数据集的f1值为0.82。**在最终比赛提交的测试集中，得分0.7758。**

3.3 神经网络

神经网络（neural network）是一种模仿生物神经网络的结构和功能的数学模型或计算模型。神经网络由大量的人工神经元联结进行计算。神经网络是一种运算模型，由大量的神经元和之间相互的联接构成。每个神经元代表一种特定的输出函数，称为激活函数。每两个节点间的连接都代表一个对于通过改连接信号的加权值，称之为权重。

对于本项目所面临的问题，利用神经网络去解决，输入部分是21个神经元，刨除了无效字段ID，输出层为21*32，添加了两个全联接层，两个连接层分别为32*32、32*16，最后输出层为16*2。激活函数全部使用tanh函数，在输出部分，加入一个softmax函数进行二分类预测，两个值分别为成为0、1的概率，取概率最大的id，作为最后的预测结果。

优化器选用Adam优化器，Adam优化器结合了AdaGrad和RMSPRop两种算法的优点，AdaGrad能够对每个不同的参数调整不同的学习率，对频繁变化的参数以更小的步长进行更新，而稀疏的参数以更大的步长进行更新，公式如下：

这种方法在数据分布稀疏的场景下，能更好的利用稀疏梯度的信息，比标准的SGD算法更有效的收敛，但是它也有一定的缺陷，分母项对梯度平方不断累积，随着时间步的增加，分母项越来越大，最终导致学习率收缩到太小无法进行有效更新。

RMSProp结合梯度平方的指数移动平均数来调节学习率的变化，能够在不稳定的目标函数情况下进行很好的收敛。它能够克服AdaGRadt梯度急剧减小的问题，在很多应用中都展示出优秀的学习率自适应能力，尤其在不稳定的目标函数下，表现更好。

Adam对梯度的一阶矩估计和二阶矩估计进行综合考虑，计算出更新步长。它的实现简单，计算高效，对内存需求少，参数的更新不受梯度的伸缩性变换影响，能够适应于不稳定的目标函数。公式如下：

经过大量实验，**发现在batch_size为16，epoch为10，lr为0.0001时效果最好，测试集得分为0.7785。**

3.4 SVM

支持向量机（Support Vector Machines）是一种二分类分类模型，其基本模型为定义在特征空间上间隔最大的线性分类器，此外，支持向量机还支持核技巧。

支持向量机的学习策略是求解能够正确划分训练数据集并且几何间隔最大的分离超平面，其本质算法为求解凸二次规划的最优化算法。

根据学习方法，支持向量机模型包含线性可分支持向量机、线性支持向量机、非线性支持向量机三大类。当训练数据线性可分时，通过硬间隔最大化学习线性分类器；训练数据近似线性可分时，采用软间隔最大化学习线性分类器（引入松弛变量等）；训练数据线性不可分时，通过核技巧及软间隔最大化学习非线性支持向量机。

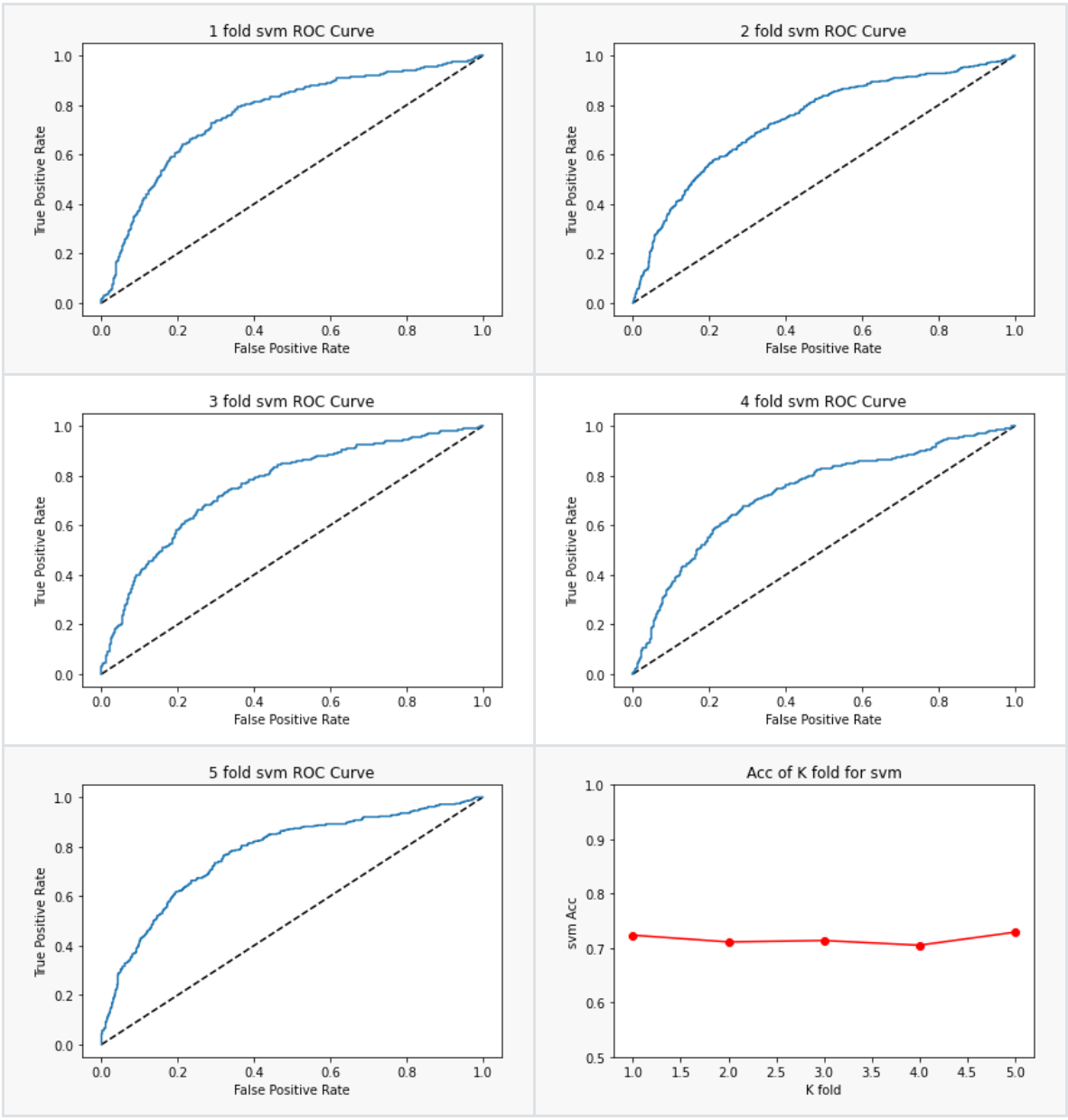
下面采用SVM方法实现客户流失数据预测。部分关键代码如下：

```
# ========== SVM分类器训练及预测 ===========
def svm_pred(k, x_t, y_t, x_v, y_v, test_data):
    # ====== 训练SVM ========
    # 当核函数为rbf（高斯核函数）时，通过调整参数C和gamma，提高分类器性能
    # rbf，C=0.8，gamma=0.001时达到rbf最佳（测试集上75.82%）
    model = svm.SVC(C=0.8, gamma=0.001, kernel='rbf')
#     # 核函数为linear时，C=0.8时，通过调整参数C，达到linear最佳（测试集上77.81%）
#     model = svm.SVC(C=0.8, gamma=0.001, kernel='linear')

    model.fit(x_t, y_t)
    print(str(k+1) + " fold model score:" + str(model.score(x_t, y_t)))

    # ====== 验证模型精度 ========
    y_v_pred = model.predict(x_v)

    # ====== AUC ROC 绘制 =======
    y_dist = model.decision_function(x_v)
    AucRoc(y_v, y_dist, 'svm', k)

    # ====== 预测客户流失 ========
    x_test = test_data[:, 1:]
    y_pred = model.predict(x_test)
    y_pred.astype(np.int16)
    y_ret = ['Yes' if y_pred[i] == 1 else 'No' for i in range(len(y_pred))]

    np.savetxt("svm_pred.csv", y_ret, fmt='%s')
    return accuracy_score(y_v_pred, y_v)
```

采用分层五折交叉验证法将数据集进行划分，并绘出ACC变化以及ROC,AUC曲线。

SVM五折交叉验证

	1-fold	2-fold	3-fold	4-fold	5-fold
acc	0.724	0.711	0.714	0.705	0.729
auc	0.762	0.738	0.759	0.732	0.772



3.5 K近邻

k近邻法（K-Nearest Neighbor）是一种基本分类回归方法，k近邻法假设给定一个训练数据集，其中实例类别已定，分类时，对新的实例，根据其k个最近邻的训练实例类别，通过多数表决方式进行预测。k近邻本质是利用训练数据集对特征向量空间进行划分，并作为其分类的“模型”。k值的选择、距离度量、分类决策规则是k近邻法的三个基本要素。

下面采用KNN方法实现客户流失数据预测。部分关键代码如下：

```python
# ========== KNN分类器训练及预测 ==========
def knn_pred(k, x_t, y_t, x_v, y_v, test_data):
    # 训练KNN模型，通过调整参数k（40为最佳结果，测试集上达到75.21%）
    clf = neighbors.KNeighborsClassifier(40)
    clf.fit(x_t, y_t)
    print(str(k+1) + " fold model score:" + str(clf.score(x_t, y_t)))

    # ====== 验证模型精度 ========
    y_v_pred = clf.predict(x_v)

    # ====== AUC ROC 绘制 =======
    y_dist = clf.predict_proba(x_v)[:, 1]
    AucRoc(y_v, y_dist, 'knn', k)

    # ====== 预测客户流失 ========
    x_test = test_data[:, 1:]
    y_pred = clf.predict(x_test)
    y_pred.astype(np.int16)
    y_ret = ['Yes' if y_pred[i] == 1 else 'No' for i in range(len(y_pred))]

    np.savetxt("knn_pred.csv", y_ret, fmt='%s')
    return accuracy_score(y_v_pred, y_v)
```

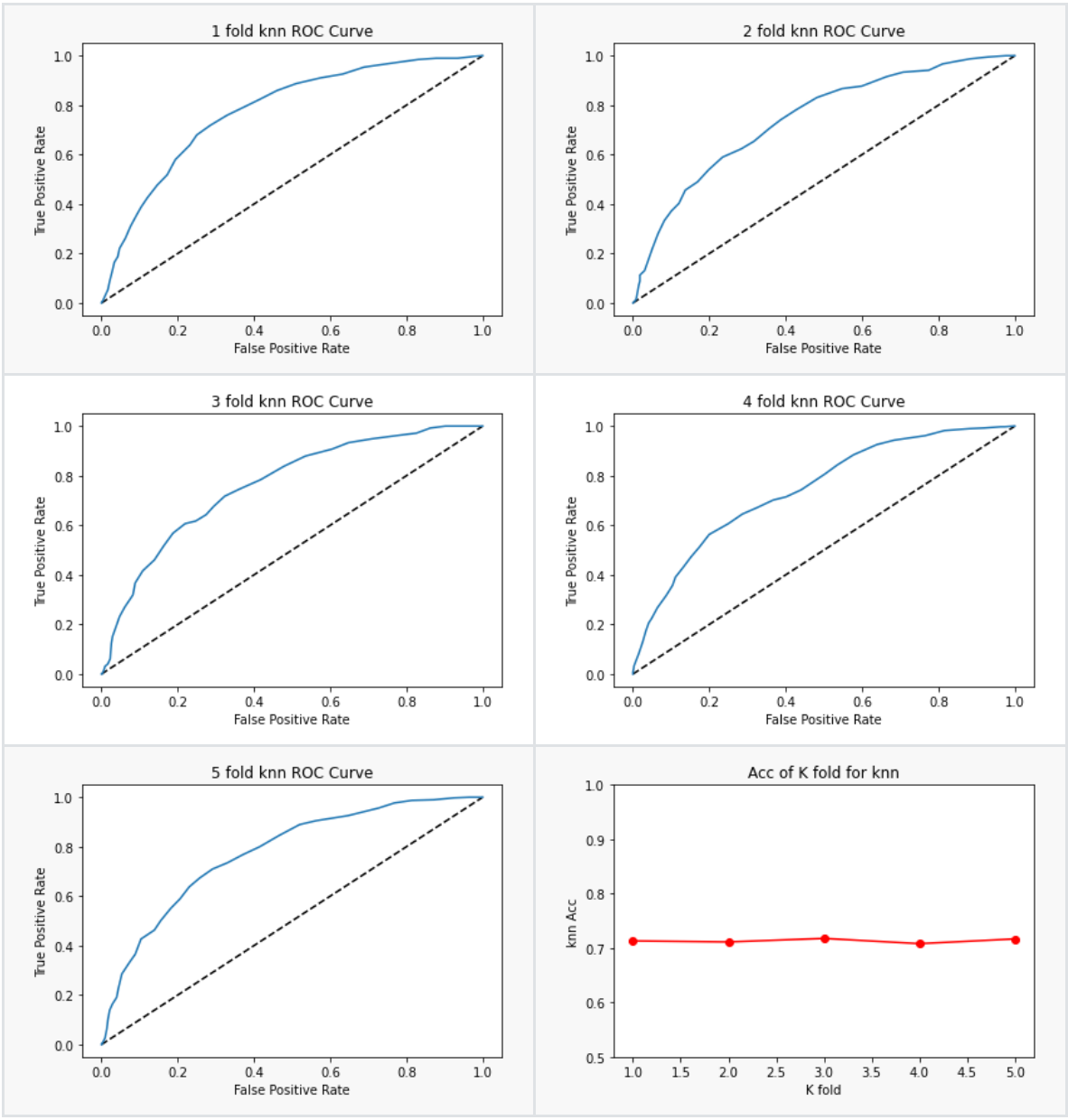
```python
# 采用分层五折交叉验证法将数据集进行划分
def kfold_cross(train_data, test_data, method):
    X = train_data[:, 1:-1]
    y = train_data[:, -1]
    skf = StratifiedKFold(n_splits=5).split(X, y)
    val_acc = []

    for k, (train_index, valid_index) in enumerate(skf):
        # print("TRAIN:", train_index, "VALID:", valid_index)
        X_train, X_valid = X[train_index], X[valid_index]
        y_train, y_valid = y[train_index], y[valid_index]

        if method == 'svm':
            acc = svm_pred(k, X_train, y_train, X_valid, y_valid, test_data)
        elif method == 'knn':
            acc = knn_pred(k, X_train, y_train, X_valid, y_valid, test_data)
        val_acc.append(acc)
        print(method + " " + str(k+1) + " fold acc: " + str(acc) + "\n")

    print(method + " mean acc: " + str(np.mean(val_acc)) + "\n")
    draw_chart(val_acc, method)
```

```python
# ====== Acc画图函数 =========
def draw_chart(y_label, tit):
    x = [1, 2, 3, 4, 5]
    plt.plot(x, y_label, 'ro-')
    plt.xlabel("K fold")
    plt.ylabel(tit + " Acc")
    plt.ylim(ymax=1.0, ymin=0.5)
    plt.title("Acc of K fold for " + tit)
    name = tit + ".jpg"
    plt.savefig(name)
    plt.show()

# ======== ROC曲线绘制 =========
def AucRoc(y_hat, y_pred, tit, k):
    # ====== 计算ROC =======
    fpr, tpr, threshold = roc_curve(y_hat, y_pred)

    # ====== 计算AUC =======
    ras = roc_auc_score(y_hat, y_pred)
    print(str(k+1) + 'fold AUC: ' + str(ras))

    # ======= 绘图 =========
    plt.plot([0, 1], [0, 1], 'k--')
    plt.plot(fpr, tpr)
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    roc_tit = str(k+1) + ' fold ' + tit + ' ROC Curve'
    plt.title(roc_tit)
    name = roc_tit + ".jpg"
    plt.savefig(name)
    plt.show()
```

采用分层五折交叉验证法将数据集进行划分，并绘出ACC变化以及ROC,AUC曲线。

KNN五折交叉验证

	1-fold	2-fold	3-fold	4-fold	5-fold
acc	0.713	0.711	0.718	0.708	0.717
auc	0.762	0.744	0.763	0.745	0.775



3.6 决策树

决策树算法起源于E.B.Hunt等人于1966年发表的论文“experiments in Induction”，但真正让决策树成为机器学习主流算法的还是Quinlan（罗斯.昆兰）（2011年获得了数据挖掘领域最高奖KDD创新奖），昆兰在1979年提出了ID3算法，掀起了决策树研究的高潮。现在最常用的决策树算法是C4.5是昆兰在1993年提出的。

顾名思义，决策树就是一棵树，一颗决策树包含一个根节点、若干个内部结点和若干个叶结点；叶结点对应于决策结果，其他每个结点则对应于一个属性测试；每个结点包含的样本集合根据属性测试的结果被划分到子结点中；根结点包含样本全集，从根结点到每个叶子结点的路径对应了一个判定测试序列。下面直接上个图，让大家看下决策树是怎样决策的（以二元分类为例），图中红线表示给定一个样例（表中数据）决策树的决策过程：

 1

主要是对训练集进行预处理 去掉Label列 之后对训练集进行随机分配2:8比例 测试方法准确度。

部分代码如下：

```
from sklearn import tree
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)
pred_tree = clf.predict(X_test)
print(accuracy_score(y_test,pred_tree))
```

```
y_tree=clf.predict(test)
```

```
y_tree
```

对结果进行输出

```
array([0, 1, 0, ..., 1, 1, 1], dtype=int64)
```

```
sub1 = pd.read_csv('C:/Users/15192/Desktop/sub_test.csv')
sub1
```

	ID	Label
0	5227	NaN
1	5228	NaN
2	5229	NaN
3	5230	NaN
4	5231	NaN
...	...	...
1302	6529	NaN
1303	6530	NaN
1304	6531	NaN
1305	6532	NaN
1306	6533	NaN

1307 rows × 2 columns

将结果保存在CSV中

```
sub1['Label']= y_tree
sub1 = sub1.set_index('ID')
sub1.to_csv('C:/Users/15192/Desktop/tree_result.csv')
```

```
comptree = pd.read_csv('C:/Users/15192/Desktop/数据挖掘/大作业/tree_result.csv')
```

```
collist = ['Label']
def binary_map(x):
    return x.map({1:'Yes',0:"No"})
comptree[collist] = comptree[collist].apply(binary_map)
```

```
comptree
```

	ID	Label
0	5227	No
1	5228	Yes
2	5229	No
3	5230	Yes
4	5231	No
...	...	...
1302	6529	Yes
1303	6530	No
1304	6531	Yes
1305	6532	Yes
1306	6533	Yes

1307 rows × 2 columns

```
comptree.to_csv('C:/Users/15192/Desktop/数据挖掘/大作业/comptree.csv')
```

3.7 梯度提升（GBM）

因为朴素贝叶斯方法精确度低 不适用 因此改用gbm方法。

梯度提升是一种用于回归和分类问题的机器学习技术，该技术以弱预测模型(通常为决策树)的集合的形式产生预测模型。

任何监督学习算法的目标是定义一个损失函数并将其最小化。让我们看看梯度提升算法的数学运算。假设我们将均方误差(MSE)定义为：

我们希望我们的预测，使我们的损失函数(MSE)最小。通过使用**梯度下降**并根据学习速率更新我们的预测，我们可以找到MSE最小的值。

因此，我们更新预测使得我们的残差总和接近于0(或最小值)，并且预测值足够接近实际值。

梯度提升的直观理解

梯度提升背后的逻辑很简单，(可以直观地理解，不使用数学符号)。我希望阅读这篇文章的人可能会很熟悉`simple linear regression`模型。

线性回归的一个基本假设是其残差之和为0，即残差应该在零附近随机扩散。

现在将这些残差视为我们的预测模型犯下的错误。虽然，树模型*(考虑决策树作为我们梯度提升的基础模型)*不是基于这样的假设，但如果我们从逻辑上(而不是统计上)考虑这个假设

所以，背后的直觉`梯度提升（gradient boosting）`算法是重复利用残差中的模式，并加强一个弱预测模型并使其更好。一旦我们达到残差没有任何可模拟模式的阶段，我们可以停止建模残差(否则可能导致过拟合)。在算法上，持续最小化损失函数，使得测试损失达到最小值。

> 综上所述，
> •我们首先用简单的模型对数据进行建模并分析数据中的错误。
> •这些错误表示难以用简单模型拟合的数据点。
> •然后对于以后的模型，我们特别关注那些难以拟合的数据点，以使他们正确。
> •最后，我们通过给每个预测变量赋予一些权重来组合所有预测变量。

部分代码如下：

```
gbm = LGBMClassifier()
gbm.fit(X_train,y_train)
pred_gbm = gbm.predict(X_test)
print(accuracy_score(y_test,pred_gbm))//对比精确度
```

对测试集进行预测后将结果输出为csv，并提交至比赛网站。

4. 实验结果

在进行特征选取后使用模型进行客户流失判断，使用GBDT并进行微调后的预测准确率在76.7%，进一步的优化，所有模型的结果及分析将在最终报告中展示。

在测试集上运行模型，并将结果按照比赛要求的格式储存并提交，得到实验结果。在多个模型下的准确率如下表所示：

	训练集	测试集
GBDT	0.769	0.773
逻辑回归	0.821	0.776
神经网络	0.781	0.779
SVM	0.717	0.778
K近邻	0.713	0.752
决策树	0.691	0.708
GBM	0.782	0.793

其中，逻辑回归在训练集上有最高准确率0.821，GBM在测试集上具有0.793的准确率，在所有模型中效果最好，最终选用GBM进行模型预测。

5. 存在的问题

问题1：在对DeviceProtection可视化时发现这部分的数据No Internet service中标记了大量的客户流失，然而在进行预处理时对数据进行了删除，削弱了该属性对模型预测效果的影响，考虑今后可针对该属性进行改进。

6. 总结

通过完成本次项目，我们在实践中了解了各种模型的基本原理，掌握了数据挖掘的基本流程，并对实验中遇到的问题进行改进，针对模型准确率提升采用了交叉验证等方法，结果表现良好。在今后的项目中我们会更加注重数据预处理以及模型效果提升的方法。