Web 信息处理与应用 Lab2-1:
西班牙银行机构电话营销预测

张劲暾 PB16111485

2019 **年** 1 **月** 18 **日**

目录

1 数据集描述 **3**

2 数据预处理 **3**

2.1 age: 用户年龄 . . . 3
2.2 job: 职业 . . . 4
2.3 marital: 婚姻状况 . . . 6
2.4 education: 受教育程度 . . . 6
2.5 default: 是否默认信用 . . . 6
2.6 housing: 是否有房贷 . . . 6
2.7 loan: 是否有个人贷款 . . . 6
2.8 contact: 联系方式类型 . . . 6
2.9 month: 本年度上一次联系的月份 . . . 7
2.10 day_of_week: 本周上一次访问的天 . . . 7
2.11 duration: 上次联系到目前的时间 . . . 7
2.12 campaign: 这次商业活动中联系了多少次 . . . 7
2.13 pdays: 上次商务活动中最后一次联系到现在的天数 . . . 7
2.14 previous: 本次商业活动前与此用户联系的次数 . . . 7
2.15 poutcome: 上次商业活动结果 . . . 7
2.16 emp.var.rate: 就业率 . . . 7
2.17 cons.price.idx: 消费者价格指数 . . . 7
2.18 cons.conf.idx: 消费者信心指数 . . . 8
2.19 euribor3m: 欧元区同业拆借利率 . . . 8
2.20 nr.employed: 就业人数 . . . 8
2.21 y: 用户是否定期存款 . . . 8

3 KNN 算法 **8**

4 LogisticRegression 算法 **9**

5 测试结果与分析 **10**

5.1 KNN 算法运行结果与解读 . . . 10
5.2 LogisticRegression 算法运行结果与解读 . . . 10

1 数据集描述

UCI 数据集：Bank Marketing Data Set

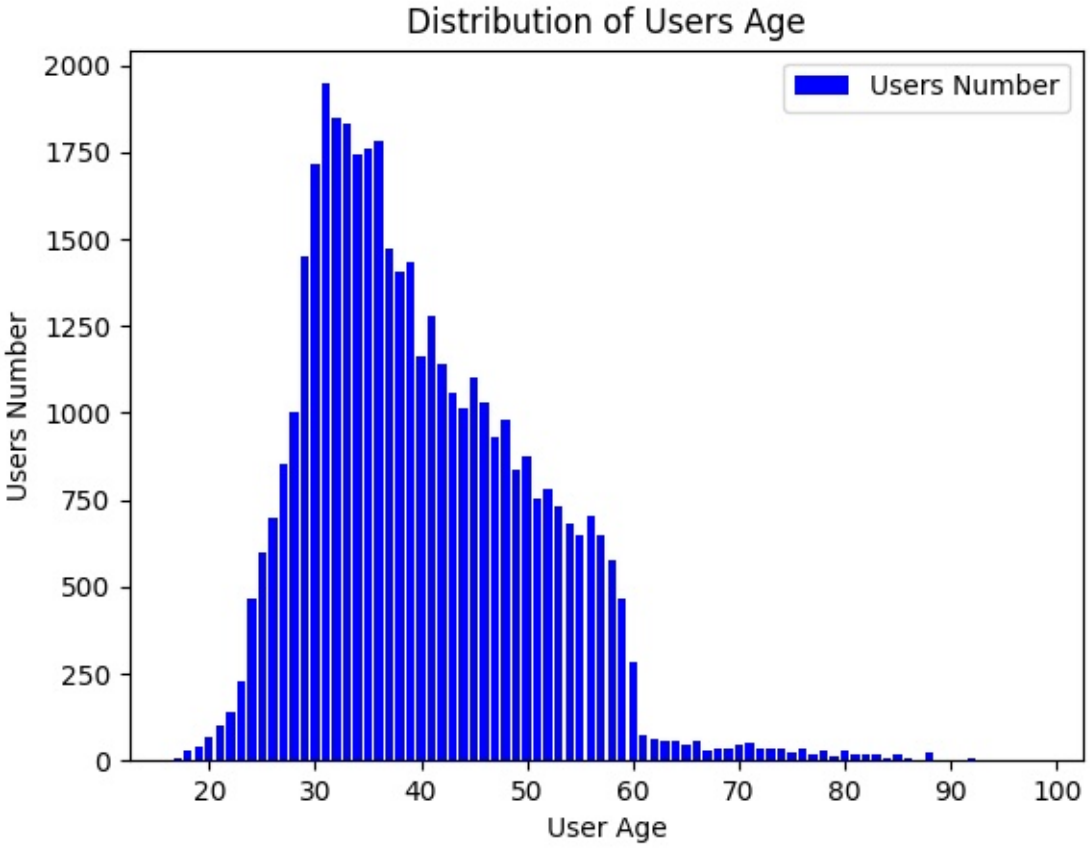
数据集一共 41188 条银行向用户推荐定期存款的记录，每条记录包括如下属性，其中 y 属性的含义是用户最终是否订购了定期存款，是需要预测的标签：

属性名称	类型	含义
age	int	用户的年龄
job	str	用户的职业
marital	str	用户的婚姻状况
education	str	用户的受教育程度
default	str	用户是否默认信用
housing	str	用户是否有房贷
loan	str	用户是否有个人贷款
contact	str	用户的联系方式类型
month	str	用户本年度上一次联系的月份
day_of_week	str	用户本周上一次访问的天
duration	int	上次联系（订购定期存款）到目前的时间
campaign	int	这次商业活动中联系了多少次
pdays	int	上次商务活动中最后一次联系到现在的天数
previous	int	本次商业活动前与此用户联系的次数
poutcome	str	上次商业活动结果
emp.var.rate	float	就业率
cons.price.idx	float	消费者价格指数
cons.conf.idx	float	消费者信心指数
euribor3m	float	欧元区同业拆借利率
nr.employed	float	就业人数
y	str	用户是否定期存款

2 数据预处理

2.1 age: 用户年龄

用户的年龄分布如图所示：

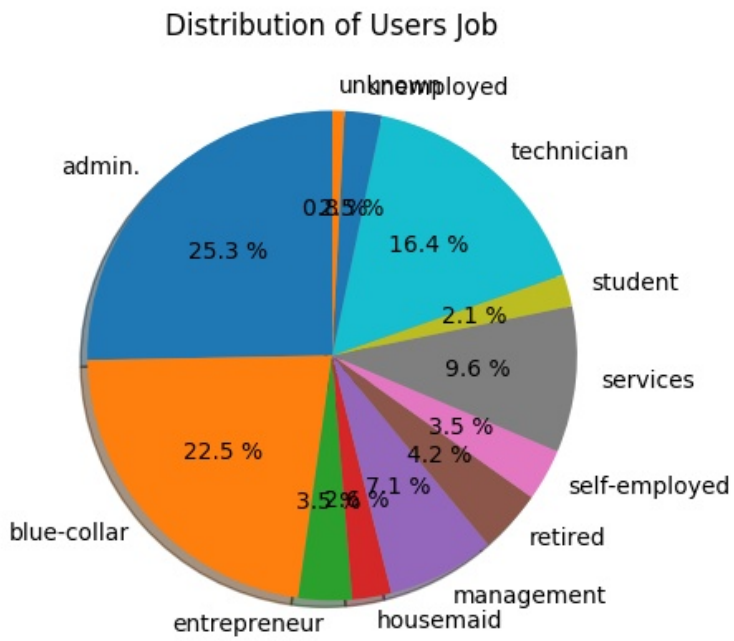


考虑到不同年龄段的用户需求以及人数的分布，我将年龄划分为 5 个区间并进行独热码编码，如表格所示：

年龄区间	独热码
[17,25]	[0,0,0,0,1]
[25,35]	[0,0,0,1,0]
[35,45]	[0,0,1,0,0]
[45,60]	[0,1,0,0,0]
[60,98]	[1,0,0,0,0]

2.2 job: 职业

用户的职业分布如图所示：



对职业进行独热码编码如表，将未知数据填充为众数：

职业	独热码
admin.	[0,0,0,0,0,0,0,0,0,0,1]
blue-collar	[0,0,0,0,0,0,0,0,0,1,0]
entrepreneur	[0,0,0,0,0,0,0,0,1,0,0]
housemaid	[0,0,0,0,0,0,0,1,0,0,0]
management	[0,0,0,0,0,0,1,0,0,0,0]
retired	[0,0,0,0,0,1,0,0,0,0,0]
self-employed	[0,0,0,0,1,0,0,0,0,0,0]
services	[0,0,0,1,0,0,0,0,0,0,0]
student	[0,0,1,0,0,0,0,0,0,0,0]
technician	[0,1,0,0,0,0,0,0,0,0,0]
unemployed	[1,0,0,0,0,0,0,0,0,0,0]
unknown	[0,0,0,0,0,0,0,0,0,0,1]

2.3 marital: 婚姻状况

对用户的婚姻状况进行独热码编码如表，将未知数据填充为众数：

婚姻状况	独热码
divorced	[0,0,1]
married	[0,1,0]
single	[1,0,0]
unknown	[0,1,0]

2.4 education: 受教育程度

对用户的受教育程度进行独热码编码如表，将未知数据填充为众数：

受教育程度	独热码
basic.4y	[0,0,0,0,0,0,1]
basic.6y	[0,0,0,0,0,1,0]
basic.9y	[0,0,0,0,1,0,0]
high.school	[0,0,0,1,0,0,0]
illiterate	[0,0,1,0,0,0,0]
professional.course	[0,1,0,0,0,0,0]
university.degree	[1,0,0,0,0,0,0]
unknown	[1,0,0,0,0,0,0]

2.5 default: 是否默认信用

这一项表示的是用户是否默认信用，绝大多数 (41185 条) 都是没有或未知，直接放弃

2.6 housing: 是否有房贷

用 0 表示没有房贷，1 表示有房贷，缺失数据默认填充为 1。

2.7 loan: 是否有个人贷款

用 0 表示没有个人贷款，1 表示有个人贷款，缺失数据默认填充为 0。

2.8 contact: 联系方式类型

用 0 表示固定电话联系方式，用 1 表示移动电话联系方式。

2.9 month: 本年度上一次联系的月份

这一项数据不便表示为"某种度量上的距离"，这对 KNN 算法和 LogisticRegression 算法是非常不友好的，所以我最终选择放弃这一项数据，试验结果表明这是有助于预测结果的准确率提高的。

2.10 day_of_week: 本周上一次访问的天

这一项数据不便表示为"某种度量上的距离"，这对 KNN 算法和 LogisticRegression 算法是非常不友好的，所以我最终选择放弃这一项数据，试验结果表明这是有助于预测结果的准确率提高的。

2.11 duration: 上次联系到目前的时间

这一项在数据集中说明有规则关系 duration = 0 ===> y = 'yes'，所以是不能使用的项。

2.12 campaign: 这次商业活动中联系了多少次

这一项不做离散化处理, 但观察到分布为长尾趋势, 所以特征值取为 log2(1 + n)。

2.13 pdays: 上次商务活动中最后一次联系到现在的天数

这一项绝大多数是无意义的 999, 直接放弃。

2.14 previous: 本次商业活动前与此用户联系的次数

这一项不做离散化处理, 但观察到分布为长尾趋势, 所以特征值取为 log2(1 + n)。

2.15 poutcome: 上次商业活动结果

用 -1 表示'failure'，用 0 表示'nonexistent'，用 1 表示'success'。

2.16 emp.var.rate: 就业率

实值直接作为特征值。

2.17 cons.price.idx: 消费者价格指数

实值正则化。

2.18 cons.conf.idx: 消费者信心指数

实值正则化。

2.19 euribor3m: 欧元区同业拆借利率

实值正则化。

2.20 nr.employed: 就业人数

实值正则化。

2.21 y: 用户是否定期存款

在 LogisticRegression 算法中，将'yes'转换为 1，'no'状换为 0，方便计算。

3 KNN 算法

有了预处理到得到的特征向量，在本数据集上运行 KNN 算法的基本步骤如下：

```
    def KnnClassifier(TrainX,TrainY,TargetX,K):
    '''
        knn分类器:(输入输出类型均为numpy.array)
        输入:
            TrainX:训练数据集特征向量集合
            TrainY:训练数据集标签集合,与特征向量集合顺序相同
            TargetX:预测数据集特征向量集合
            K:考虑最邻近的K个点
        输出:
            TargetY:预测数据集标签集合,与特征向量集合顺序相同
    '''
    if(TrainX.shape[0] != TrainY.shape[0]):
        raise TypeError("TrainX should have same size with TrainY.")
    TargetY = []
    for i in range(TargetX.shape[0]):
        DifferentMatrix = np.tile(TargetX[i],(TrainX.shape[0],1)) - TrainX
        DiffSquarMatrix = DifferentMatrix ** 2
        Distance = DiffSquarMatrix.sum(axis = 1) ** 0.5
```

```
            DistanceSortedOrder = Distance.argsort()
            NeighLabelCount = {}
            for i in range(1,K + 1):
                NeighLabel = TrainY[DistanceSortedOrder[i]]
                NeighLabelCount[NeighLabel] =
                NeighLabelCount.get(NeighLabel,0) +
                ClassWeight[NeighLabel] / ( 1 + Distance[DistanceSortedOrder[i]] )
            SortedClassCount =
            sorted(NeighLabelCount.items(),key = lambda x:x[1],reverse = True)
            TargetY.append(SortedClassCount[0][0])
        return TargetY
```

4 LogisticRegression 算法

有了预处理到得到的特征向量，在本数据集上运行 LogisticRegression 算法的主要学习步骤如下：

```
    def fit(self,X,Y):
        '''
            拟合训练数据X,Y
        '''
        if(X.shape[0] != Y.shape[0]):
            raise TypeError("X should have same size with Y.")
        self.FeatureNum = X.shape[1]
        self.InstanceNum = X.shape[0]
        self.PredictProba = np.zeros(self.InstanceNum)
        self.FeatureWeight = np.zeros(self.FeatureNum)
        for i in range(self.MaxIterTimes):
            self.PredictProba = self.get_proba(X)
            WeightAveDiff =
            (1.0 / self.InstanceNum) * np.dot(X.T, (self.PredictProba - Y))
            WeightAveDiff +=
            ( (self.RegulizationCoef * self.FeatureWeight) / self.InstanceNum )
            self.FeatureWeight -= self.LearningtRate * WeightAveDiff
            loss = np.linalg.norm(WeightAveDiff)
            if loss < self.StopThreshold:
                break
            if i % 2500 == 0:
                print("Iteration " + str(i) + " , loss = " + str(loss))
```

5 测试结果与分析

5.1 KNN 算法运行结果与解读

对于不同的 K 值，运行结果如下：

K	Precision	Recall	F1-score	Accuracy
1	0.32748538011695905	0.30601092896174864	0.3163841807909605	0.8483709273182958
2	0.2857142857142857	0.45901639344262296	0.35220125786163525	0.806390977443609
3	0.24743589743589745	0.5273224043715847	0.33682373472949395	0.7619047619047619
4	0.2372340425531915	0.6092896174863388	0.34150076569678406	0.7305764411027569
5	0.22482893450635386	0.6284153005464481	0.3311735061195104	0.7089598997493735
6	0.25769230769230766	0.5491803278688525	0.3507853403141361	0.7669172932330827
7	0.30976430976430974	0.5027322404371585	0.3833333333333333	0.8145363408521303
8	0.31789137380191695	0.5437158469945356	0.40120967741935487	0.8139097744360902
9	0.31101190476190477	0.5710382513661202	0.4026974951830443	0.8057644110275689
10	0.30939226519337015	0.6120218579234973	0.4110091743119266	0.7988721804511278
11	0.31129476584022037	0.6174863387978142	0.41391941391941395	0.7994987468671679
12	0.32248062015503876	0.5683060109289617	0.4114737883283877	0.8135964912280702
13	0.3441780821917808	0.5491803278688525	0.4231578947368421	0.8283208020050126
14	0.34390651085141904	0.5628415300546448	0.42694300518134715	0.8267543859649122

可以看到在 K 大于 10 以后，基本上满足 **Precision > 0.3, Recall > 0.5, F1-score** 值基本维持在 0.4 左右，**Accuracy** 值基本维持在 0.8 以上，而 y 的比例大概是 7: 1，也就是说，KNN 分类的结果大概把选择的用户订购的概率提高了三倍，涵盖了一半以上潜在的会订购定期存款的用户，还是很有意义的。

5.2 LogisticRegression 算法运行结果与解读

经过大量调参之后，获得的最佳结果与相应的参数如下所示：

LogisticRegressionClassifier(LearningtRate=0.05,JudgeThreshold=0.25,MaxIterTimes=20000)

Precision = 0.46647230320699706, Recall = 0.4371584699453552,

F1-score = 0.45133991537376583, Accuracy = 0.8781328320802005

可以看到对于用户订购意愿预测的准确性比 KNN 低，但召回率却比较低，与 Random Result 相比大概把选择的用户订购的概率提高了四倍，涵盖了百分之四十以上潜在的会订购定期存款的用户，还是很有意义的。