Data Mining —Adult

陶睿
201714738

数据集简介

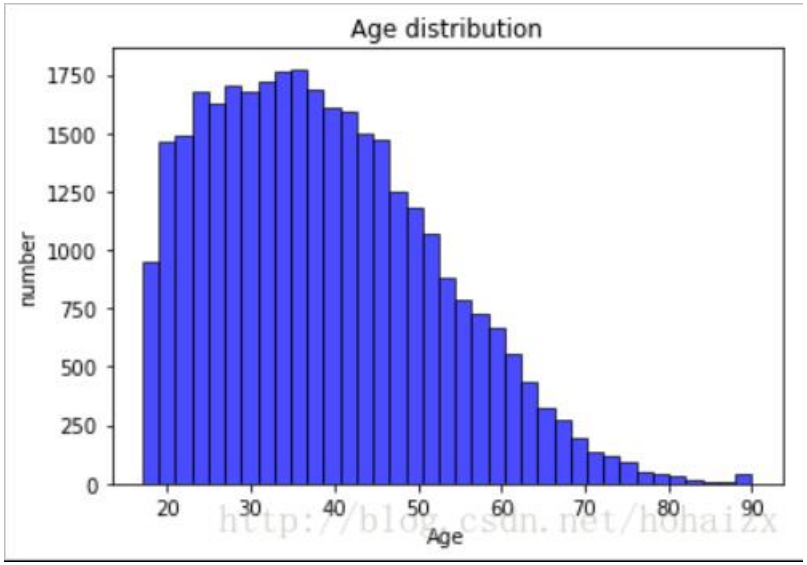
该数据从美国1994年人口普查数据库中抽取而来，因此也称作“人口普查收入”数据集，训练数据32561条和测试数据16281条。该数据集类变量为年收入是否超过50k$，是一个分类数据集，用来预测年收入是否超过50k$。

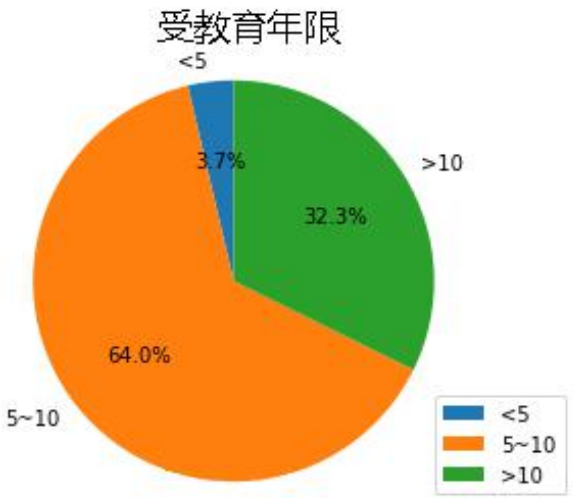
用训练数据训练的分类器，对测试数据进行测试。计算准确率（Precision），精度（Accuracy），召回率（Recall）

属性名	类型	含义
age	continuous	年龄
workclass	discrete	工作类别
fnlwgt	continuous	序号
education	discrete	受教育程度
education-num	continuous	受教育时间
marital-status	discrete	婚姻状况
occupation	discrete	职业
relationship	discrete	社会角色
race	discrete	种族
sex	discrete	性别
capital-gain	continuous	资本收益
capital-loss	continuous	资本支出
hours-per-week	continuous	每周工作时间
native-country	discrete	国籍

简单分析

学历	年收入>50K比例
博士	75%
硕士	57%
学士	42%





实验环境

系统和语言

- Windows 10
- Python 2.7.15 & Python3.6.3

Python包

- Numpy 1.14.5
- Pandas 0.23.1
- Scikit-learn 0.19.1
- Matplotlib 2.2.2
- Tensorflow 1.8.0

数据预处理

- data —> csv
- 剔除缺省值
- 离散型—>数值型 One hot encode
- feature scaling & PCA

分类器模型

- 支持向量机
- 贝叶斯分类器
- 随机梯度下降
- SVM/BNB/SGD组合分类器
- DNNLinear组合分类器

SVM

- SVM之所以受欢迎度这么高，另一个重要的原因是它很容易核化(kernelized)，能够解决非线性分类问题。
- 核方法的idea是为了解决线性不可分数据，在原来特征基础上创造出非线性的组合，然后利用映射函数将现有特征维度映射到更高维的特征空间，并且这个高维度特征空间能够使得原来线性不可分数据变成了线性可分的。
- 我们在这里选用核’ rbf’ ，gamma = 1.代码如下：

```
print ('Training SVM')
clf_svm=svm.SVC(kernel='rbf',gamma=1.0,C=3.0)
clf_svm.fit(train_x_pca,train_y)
print ('Testing SVM')
predict_svm=clf_svm.predict(test_x_pca)
```

贝叶斯分类器

- 高斯贝叶斯分类器进行训练，参数使用默认，代码如下

```
print ('Training BNB')
clf_bnb=GaussianNB()
clf_bnb.fit(train_x_pca,train_y)
print ('Testing BNB')
predict_bnb=clf_bnb.predict(test_x_pca)
```

随机梯度下降法

- 随机梯度下降法(stochastic gradient descent)。有时也被称为迭代(iteration)/在线(on-line)梯度下降。随机梯度下降法每次只用一个样本对权重进行更新
- 虽然随机梯度下降被当作是梯度下降的近似算法，但实际上她往往比梯度下降收敛更快，因为相同时间内她对权重更新的更频繁。由于单个样本得到的损失函数相对于用整个训练集得到的损失函数具有随机性，反而会有助于随机梯度下降算法避免陷入局部最小点

```
print ('Training SGD')
clf_SGD=SGDClassifier(penalty='l1')
clf_SGD.fit(train_x_pca,train_y)
print ('Testing SGD')
predict_SGD=clf_SGD.predict(test_x_pca)
```

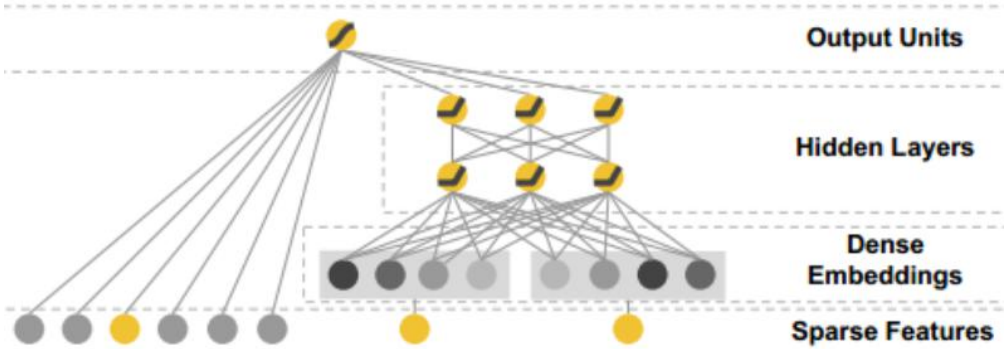
组合分类器

- 为了提高分类结果，我们将上面三个分类器进行组合，然后投票

```
class mv():
    def majority_voting(self,data):
        row=len(data.axes[0])
        col=len(data.axes[1])
        new_class=[]
        for i in range(0,row):
                count_zero=0
                count_one=0
                count_two=0
                count_three=0
                for j in range(0,col):
                     if   (data.ix[i,j] == 0):
                              count_zero = count_zero + 1
                     elif (data.ix[i,j]==1):
                              count_one = count_one + 1
                     elif (data.ix[i,j]==2):
                              count_two = count_two + 1
                     elif (data.ix[i,j]==3):
                              count_three = count_three +1

                bigger={"count_zero":count_zero, "count_one":count_one, "count_two":count_two,"count_three":count_three}
                help_max=max(bigger.iterkeys(), key=lambda k: bigger[k])
                if (help_max == "count_zero"):
                    new_class.append(0)
                elif (help_max == "count_one"):
                    new_class.append(1)
                elif (help_max == "count_two"):
                    new_class.append(2)
                elif (help_max == "count_three"):
                    new_class.append(3)
        return new_class
```

DNNLinear组合分类器

- Linear分类器和Deep分类器结合。




```
def build_estimator(model_dir, model_type):
  """Build an estimator."""
  if model_type == "wide":
    m = tf.estimator.LinearClassifier(
        model_dir=model_dir, feature_columns=base_columns + crossed_columns)
  elif model_type == "deep":
    m = tf.estimator.DNNClassifier(
        model_dir=model_dir,
        feature_columns=deep_columns,
        hidden_units=[100, 50])
  else:
    m = tf.estimator.DNNLinearCombinedClassifier(
        model_dir=model_dir,
        linear_feature_columns=crossed_columns,
        dnn_feature_columns=deep_columns,
        dnn_hidden_units=[100, 50])
  return m
```

结果&分析

- 每种参数都分别实验3-4次，取平均值
- 测试了PCA 维度，训练数据个数的影响
- 讨论结果

序号	Train num	Test num	PCA	CPU TIME	SVM	Bayes	SGD	Ensemble	average
1	100	1000	4	0.52	78.30	78.17	77.67	78.90	78.26
2	1000	1000	4	0.60	80.60	80.40	68.27	80.27	77.38
3	10000	1000	4	2.50	82.80	80.63	78.07	81.83	80.83
4	20000	1000	4	7.89	83.77	82.00	77.33	82.83	81.48
5	20000	1000	8	19.68	82.00	80.13	81.10	82.15	81.34
6	20000	1000	2	15.25	82.25	81.23	80.20	81.30	81.24

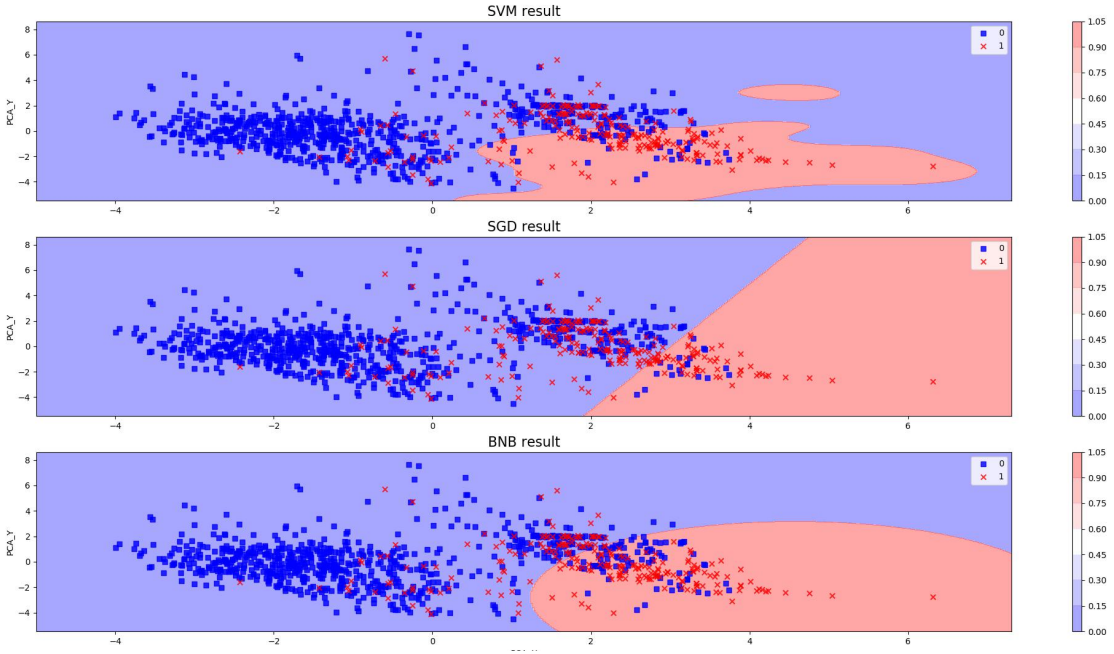
1.PCA维度的影响

- PCA降维后可以将14维数据降为dim维数据
- 分别选取dim为2 4 8
- 训练集数目20000，测试集数目1000
- dim为4时，所用时间最少。dim>4时，用时随dim增加而增加
- accuracy上没有明显差异。

序号	Train num	Test num	PCA	CPU TIME	SVM	Bayes	SGD	Ensemble	average
1	100	1000	4	0.52	78.30	78.17	77.67	78.90	78.26
2	1000	1000	4	0.60	80.60	80.40	68.27	80.27	77.38
3	10000	1000	4	2.50	82.80	80.63	78.07	81.83	80.83
4	20000	1000	4	7.89	83.77	82.00	77.33	82.83	81.48
5	20000	1000	8	19.68	82.00	80.13	81.10	82.15	81.34
6	20000	1000	2	15.25	82.25	81.23	80.20	81.30	81.24

2.训练集数目的影响

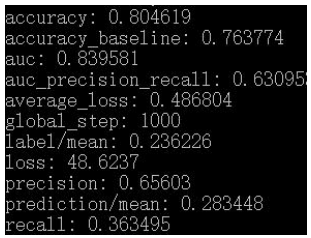
- PCA降维为4维数据
- 训练集数目100/10^3/10^4/2*10^4，测试集数目10^3
- 随着训练所用数据量增加，组合分类器accuracy从78.9提高到82.8。
- 其中SVM Bayes Ensemble提高比较明显，SGD波动比较大。

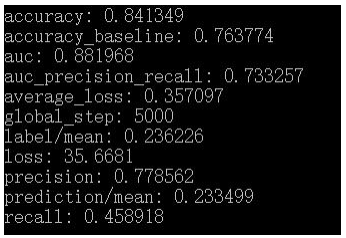


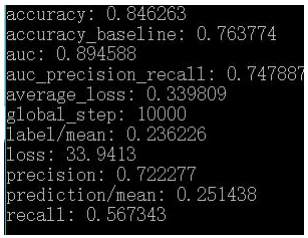

```
accuracy_svm:83.000000
accuracy_binomial naivebayes:81.100000
accuracy_SGD:80.400000
accuracy_ensemble:83.000000
svm classification
              precision    recall  f1-score   support

           0       0.86      0.93      0.89       762
           1       0.70      0.50      0.58       238

avg / total       0.82      0.83      0.82      1000

bnb classification
              precision    recall  f1-score   support

           0       0.86      0.89      0.88       762
           1       0.61      0.55      0.58       238

avg / total       0.81      0.81      0.81      1000

SGD classification
              precision    recall  f1-score   support

           0       0.81      0.98      0.88       762
           1       0.76      0.26      0.38       238

avg / total       0.80      0.80      0.76      1000

confusion matrix
[[712  50]
 [120 118]]
[[679  83]
 [106 132]]
[[743  19]
 [177  61]]
pca n = 2

Method all: 14.749609 CPU seconds
```

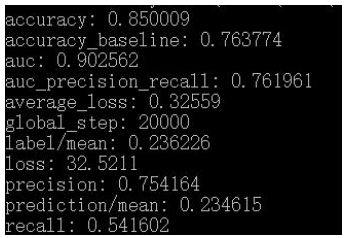
3.SVM、Bayes、SGD及其组合

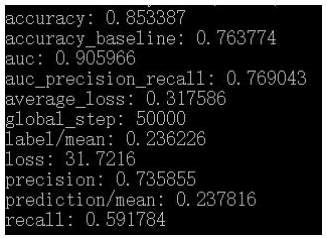
- train = 20000 test = 1000
- 四种方法准确率Accuracy分别为=83%、81%、80%、83%
- 对工资少于50K的分类结果普遍较好，对大于50K的分类很差。原因可能是工资大于50K的样本数少且分布混在在工资小于50K的样本中。











experiment	1	2	3	4	5
train_steps	1000	5000	10000	20000	50000
Accuracy	0.805	0.841	0.846	0.85	0.853

4.DNNLinear组合分类器结果

- 迭代次数从1000增加到50000，准确率Accuracy从0.805增长到0.853
- 迭代次数50000运行时间过久，放弃

谢谢观看

THANK YOU