广东金融学院实验报告

课程名称：数据挖掘

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验编号  及实验名称 | 实验三 数据挖掘案例实践 | | | 系 别 | 互联网学院 |
| 姓 名 | 林旋华 | 学号（完整学号） | 181543306 | 班 级 | 18计科(3)班 |
| 实验地点 | 实验楼102 | 实验日期 | 2021.06.17 | 实验时数 | 2 |
| 指导老师 | 侯昉 | 同组其他成员 | 无 | 成 绩 |  |
| 一、实验目的及要求  1、进一步熟悉基本的数据获取、整理和分析流程及其操作。  2、掌握基本关联规则挖掘、分类、聚类和相关回归分析的算法。 | | | | | |
| 二、实验环境及相关情况（包含使用软件、实验设备、主要仪器及材料等）  1．使用软件：Python开发工具  2．实验设备：个人计算机 | | | | | |
| 三、实验内容及步骤（包含简要的实验步骤流程）  1.自行搜集数据，使用本学期学习的某个算法对其进行数据整理、分析和挖掘。并对得到的结果进行分析总结。  部分数据下载网站：  <http://archive.ics.uci.edu/ml/datasets.php>  <https://www.kesci.com/home/dataset>  <https://www.zhihu.com/question/342295029>  案例一：自动识别窃漏电用户-贝叶斯  案例二：自动识别窃漏电用户--逻辑回归  案例三：CART--决策树  案例四：APP（Apple IOS app store）--数据分析 | | | | | |
| 四、实验结果（包括程序或图表、结论陈述、数据记录及分析等，可附页） **案例一：自动识别窃漏电用户-贝叶斯****实验步骤：** **①导入库**  from pandas import read\_csv  from sklearn.naive\_bayes import GaussianNB  **②导入数据**  data\_bayes = read\_csv('nbc\_model.csv')  data\_bayes.head() #显示前5行数据  #data\_bayes.tail() #显示后5行  #data\_bayes.columns #查看列名  #data\_bayes.info() #查看各字段的信息  #data\_bayes.shape #查看数据集行列分布，几行几列  #data\_bayes.describe() #查看数据的大体情况  #data\_bayes.dtypes #查看数据集列变量类型    data\_bayes = read\_csv('nbc\_model.csv')  data\_bayes.head()    **③数据提取**  **#导入numpy库**  import numpy as np  **# 将数据分为输入数据和输出结果**  array = data\_bayes.values  X = array[:,2:5]  Y = array[:,5]  X=X.astype(np.int16)  Y=Y.astype(np.int16)  **④模型搭建**  model\_bayes = GaussianNB() #GaussianNB没有参数，不需要调参  fit1 = model\_bayes.fit(X, Y)  #预测  Y\_predicted= fit1.predict(X) #这里应该用测试集  **⑤模型评估**  #导入模型评估的库  from sklearn.metrics import classification\_report  **#模型评估**  scores\_bayes=fit1.score(X, Y) #这里应该用测试集  print("准确率:", scores\_bayes)  print(classification\_report(Y, Y\_predicted))    **#绘制ROC曲线**  import matplotlib.pyplot as plt  from sklearn.metrics import roc\_curve  #roc\_curve输出为tpr、fpr假正和真正概率，且第二个参数一定要是概率估计或者置信度  fpr,tpr,thresholds = roc\_curve(Y,fit1.predict\_proba(X)[:,1],pos\_label=1)  #pos\_labels设置的为感兴趣方的标签,predict\_probs前面输出的是0的概率，后面输出的是1的概率  **#查看结果与概率的对应情况**  plt.plot(fpr,tpr,linewidth=2,label="ROC")  plt.xlabel("false presitive rate")  plt.ylabel("true presitive rate")  plt.ylim(0,1.05)  plt.xlim(0,1)  plt.legend(loc=4)#图例的位置  plt.show()   **案例二：自动识别窃漏电用户--逻辑回归** **①导入所需的库**  #-\*- coding:utf-8 -\*-  import pandas as pd  import numpy as np  from sklearn.model\_selection import KFold  from sklearn.model\_selection import cross\_val\_score  from sklearn.linear\_model import LogisticRegression  **②导入并查看数据**  data\_lr = pd.read\_csv('logist\_model.csv',sep=',',encoding='utf-8')  print(data\_lr.head(10))    **③将数据分为训练数据和测试数据**  array = data\_lr.values  X\_train =array[0:200,2:5]  Y\_train = array[0:200,5]  X\_test =array[200:291,2:5]  Y\_test = array[200:291,5]  print(array.shape)  print(array[1:5])    **④逻辑回归**  #num\_folds = 10  #seed = 7  #kfold = KFold(n\_splits=num\_folds, random\_state=seed)  model = LogisticRegression()  model.fit(X\_train.astype('int'), Y\_train.astype('int'))    **⑤效果评估**  **#准确率**  scores = cross\_val\_score(model,X\_train.astype('int'), Y\_train.astype('int'), cv=10)  print("准确率", np.mean(scores), scores)    **#混淆矩阵**  from sklearn.metrics import confusion\_matrix  predicted = model.predict(X\_test.astype('int'))  matrix = confusion\_matrix(Y\_test.astype('int'), predicted)  classes = ['0', '1']  dataframe = pd.DataFrame(data=matrix,  index=classes,  columns=classes)  print(dataframe)    **#AUC**  from sklearn.metrics import roc\_curve, auc  predictions = model.predict\_proba(X\_test.astype('int'))  fpr, tpr, thresholds = roc\_curve(Y\_test.astype('int'), predictions[:,1])  roc\_auc = auc(fpr, tpr)  import matplotlib.pyplot as plt  plt.plot(fpr, tpr,'b', label='auc=%0.2f' % roc\_auc)  plt.legend(loc ='lower right')  plt.plot([0, 1],[0,1],'r--')  plt.xlim([0.0, 1.0])  plt.ylim([0.0, 1.0])  plt.xlabel("fpr")  plt.ylabel("tpr")  plt.show()   **案例三：CART--决策树** **①加载所需库**  import pandas as pd  import numpy as np  from sklearn.model\_selection import train\_test\_split  from sklearn.tree import DecisionTreeClassifier  **②导入数据**  # csv文件是通过pandas模块中的read\_csv函数进行读取。  # sep：指定分隔符，encoding：指定文件编码等，对于文本文件含有中文的，其文件编码通常为uft-8。  data\_cart = pd.read\_csv('tree.csv',sep = ',',encoding = 'utf-8')  print(data\_cart.shape)  data\_cart.columns    data\_cart.head()    **③拆分训练集与测试集**  array = data\_cart.values  X =array[:,0:7]  Y = array[:,7]  test\_size = 0.30  seed = 4  X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=test\_size, random\_state=seed)  #X\_train：所要划分的样本特征集  #train\_target：所要划分的样本结果  #test\_size：样本占比，如果是整数的话就是样本的数量  #random\_state：是随机数的种子。在需要重复试验的时候，保证得到一组一样的随机数。  #采用sklearn模块构建cart决策树  cart\_tree = DecisionTreeClassifier()  **④模型训练、预测、评估**  cart\_tree.fit(X\_train, Y\_train)  data\_cart\_pre = cart\_tree.predict(X\_test)  sum(data\_cart\_pre == Y\_test)/float(len(Y\_test))  结果：  cart\_tree    **#评估具体指标**  from sklearn import metrics  print("tes\_score:", cart\_tree.score(X\_test, Y\_test))  y\_pred = cart\_tree.predict(X\_test)  print("查准率:",metrics.precision\_score(Y\_test, data\_cart\_pre))  print("召回率:",metrics.recall\_score(Y\_test, data\_cart\_pre))  print("F1\_score:",metrics.f1\_score(Y\_test, data\_cart\_pre))    **⑤绘制决策树图（需要安装graphviz工具和及配置）**  import graphviz  import sklearn.tree as tree  dot\_data = tree.export\_graphviz(cart\_tree, out\_file=None)  graph = graphviz.Source(dot\_data)  graph.render("cart\_tree\_1")    **⑥剪枝-设置待选的参数**  **#导入所需库**  #from sklearn.grid\_search import GridSearchCV  from sklearn.model\_selection import GridSearchCV  from sklearn.model\_selection import StratifiedKFold  decision\_tree\_classifier = DecisionTreeClassifier()  parameter\_grid = {'max\_depth':[1,2,3,4,5],  'max\_features':[1,2,3]}  **#将不同参数带入**  gridsearch = GridSearchCV(decision\_tree\_classifier,  param\_grid = parameter\_grid,  cv = 10)  gridsearch.fit(X\_train,Y\_train)  #得分最高的参数值，并构建最佳的决策树  best\_param = gridsearch.best\_params\_  best\_decision\_tree\_classifier = DecisionTreeClassifier(max\_depth=best\_param['max\_depth'], max\_features=best\_param['max\_features'])  #best\_decision\_tree\_classifier  best\_decision\_tree\_classifier.fit(X\_train,Y\_train)    best\_decision\_tree\_classifier\_pre = best\_decision\_tree\_classifier.predict(X\_test)  sum(best\_decision\_tree\_classifier\_pre == Y\_test)/float(len(Y\_test))    **#绘制决策树图**  dot\_data = tree.export\_graphviz(best\_decision\_tree\_classifier, out\_file=None)  graph = graphviz.Source(dot\_data)  graph.render("clf\_cart3")    **案例四：APP（Apple IOS app store）--数据分析（Pyecharts)**  不断变化的移动环境是一个充满挑战的导航空间。移动设备在台式机上的百分比仅在增加。Android占据智能手机市场约53.2％，而iOS则占43％。为了吸引更多人下载您的应用，您需要确保他们可以轻松找到您的应用。移动应用程序分析是了解现有策略以促进增长和保留未来用户的一种好方法。  如今，随着数以百万计的应用程序的使用，以下数据集已成为在iOS应用程序商店中获取最流行应用程序的关键。此数据集包含7000多个Apple iOS移动应用程序详细信息。数据是从Apple Inc.网站上的iTunes Search API中提取的，为了分析App特性，从Kaggle中找到数据集。  1.数据集来源([https://www.kaggle.com/ramamet4/app-store-apple-data-set-10k-apps](https://www.kaggle.com/ramamet4/app-store-apple-data-set-10k-apps" \t "http://localhost:8889/notebooks/pyechart%E5%8F%AF%E8%A7%86%E5%8C%96/_blank))   |  |  |  | | --- | --- | --- | | **编号** | **变量** | **中文解释** | | **2** | “track\_name” | 应用名称 | | **3** | “size\_bytes” | 大小（以字节为单位） | | **4** | “currency” | “货币”：货币类型 | | **5** | “price” | “价格”：价格金额 | | **6** | “rating\_count\_tot” | 用户评分计数（所有版本） | | **7** | “rating\_count\_ver” | 用户评分计数（当前版本） | | **8** | “user\_rating” | 平均用户评级值（对于所有版本） | | **9** | “user\_rating\_ver” | 平均用户评分值（当前版本） | | **10** | “ver” | 最新版本代码 | | **11** | “cont\_rating” | 内容分级 | | **12** | “prime\_genre” | 主要类型 | | **13** | “sup\_devices.num” | 支持设备的数量 | | **14** | “ipadSc\_urls.num” | 显示的屏幕截图数量 | | **15** | “lang.num” | 支持的语言数量 | | **16** | “vpp\_lic” | 启用基于Vpp设备的许可 |   import numpy as np  import pandas as pd  from pyecharts.globals import ThemeType  from pyecharts.faker import Faker  from pyecharts import options as opts  from pyecharts.charts import Bar  from pyecharts.charts import Line  from pyecharts.charts import Bar, Grid, Line,Scatter,Tab  # -\*- coding: utf-8 -\*-  #数据文件的读取  r1=pd.read\_csv('AppleStore.csv',sep = ',',encoding = 'utf-8')  r1.head()    f = open("AppleStore.csv","rb")#二进制格式读文件  while True:  line = f.readline()  if not line:  break  else:  try:  #print(line.decode('utf8'))  line.decode('utf8')  #为了暴露出错误，最好此处不print  except:  print(str(line))  2.分析思路  以所有版本评分均分（user\_rating）为指标，筛选出“优质游戏APP”，以“一般游戏APP”为对照，统计描述二者不同变量下的特性。   1. 描述性分析   **图1：总体APP分析**  #按类别分组groupby  p0=r1.groupby('prime\_genre').size().sort\_values(ascending=False)  from pyecharts.charts import Page, Pie  from pyecharts import options as opts  #设置风格（总结：ThemeType.LIGHTLIGHT、DARK、CHALK、ESSOS、INFOGRAPHIC、MACARONS、  #PURPLE\_PASSION、ROMA、ROMANTIC、SHINE、VINTAGE、WALDEN、WESTEROS、WONDERLAND）  f2=Pie(init\_opts=opts.InitOpts(theme=ThemeType.DARK)).add("", [list(z) for z in zip(p0.index.tolist(), p0.tolist())])  f2.set\_series\_opts(label\_opts=opts.LabelOpts(is\_show=False))  f2.set\_global\_opts(  title\_opts=opts.TitleOpts(title="总体APP分析"),  legend\_opts=opts.LegendOpts(  orient="vertical", pos\_top="15%", pos\_left="2%"  ),  )  f2.render\_notebook()    图 1 总体APP分析 **图2 ：箱型图 APP分类箱线图（异常值分析）** from pyecharts.charts import Boxplot  def c1(x):  return x.price.values  t3=r1.groupby("prime\_genre").apply(c1)  x=["Book","Business","Catalogs","Education",'Entertainment','Finance','Food & Drink','Games','Health & Fitness','Lifestyle','Medical',  'Medical','Music','Navigation','News','Photo & Video','Productivity','Reference','Shopping','Social Networking','Social Networking','Sports'  ,'Sports','Travel','Utilities','Weather']  y=[t3[x[0]],t3[x[1]],t3[x[2]],t3[x[3]],t3[x[4]],t3[x[5]],t3[x[6]],t3[x[7]],t3[x[8]],t3[x[9]],t3[x[10]],t3[x[11]],t3[x[12]],t3[x[13]],t3[x[14]],t3[x[15]],t3[x[16]],t3[x[17]],  t3[x[18]],t3[x[19]],t3[x[20]],t3[x[21]],t3[x[22]],t3[x[23]],t3[x[24]],t3[x[25]]]  c = Boxplot(init\_opts=opts.InitOpts(theme=ThemeType.WESTEROS))  c.add\_xaxis(x).add\_yaxis("price", c.prepare\_data(y))  c.set\_global\_opts(title\_opts=opts.TitleOpts(title="price"),  datazoom\_opts=opts.DataZoomOpts(is\_show=True))  c.render\_notebook()  在保留的7197条记录中，按APP类型绘制条形图，共有23种不同类型APP，其中games类的APP一枝独秀，约为总数的53%。对总体按不同类型APP进行比较，绘制箱线图如下。    图 2箱型图 APP分类箱线图（异常值分析）  由图可知，游戏APP的大小分布较广；绝大多数APP的价格低于等于50，仅0.1%的APP价格大于50，其中分类为Education的两款APP价格最高（249.99，299.99）.  **图 3游戏APP总版本评分计数柱形图**  p1=r1[r1.prime\_genre=='Games'].groupby('user\_rating').size().sort\_index(ascending=False)  import json  import os  from pyecharts import options as opts  from pyecharts.charts import Page, PictorialBar  from pyecharts.globals import SymbolType  def pictorialbar\_base() -> PictorialBar:  c= (PictorialBar(init\_opts=opts.InitOpts(theme=ThemeType.VINTAGE)).add\_xaxis(p1.index.tolist()).add\_yaxis("游戏APP总版本评分计数柱形图",p1.tolist()).set\_series\_opts(  label\_opts=opts.LabelOpts(is\_show=False),  markpoint\_opts=opts.MarkPointOpts(  data=[  opts.MarkPointItem(type\_="max", name="最大值"),  opts.MarkPointItem(type\_="min", name="最小值"),  opts.MarkPointItem(type\_="average", name="平均值"),  ]  )))  return c  pictorialbar\_base().render\_notebook()  在3862条游戏APP的记录中，绘制ratings计数柱形图，其中评分4.5的APP最多，占48.06%。由图可知，App Store里能生存的游戏几乎都是精品，用户们评分都挺高。    图 3游戏APP总版本评分计数柱形图 **图4优质游戏APP-price计数柱形图** r1.rating\_count\_tot.mean() #所有版本的评价人数比较多    r1.rating\_count\_ver.mean() #先行版的评价人数    p2=r1[(r1.rating\_count\_tot>12892.907183548701)&(r1.user\_rating.astype('float').astype('int')>4)].price.value\_counts() #采用所有版数据  r1[(r1.rating\_count\_tot>12892.907183548701)&(r1.user\_rating>4)].price.mean()  from pyecharts.charts import Bar  def bar\_base() -> Bar:  c= (Bar(init\_opts=opts.InitOpts(theme=ThemeType.ESSOS)).add\_xaxis(p2.index.tolist()).add\_yaxis("",p2.tolist()).set\_series\_opts(  label\_opts=opts.LabelOpts(is\_show=False),  markpoint\_opts=opts.MarkPointOpts(  data=[  opts.MarkPointItem(type\_="max", name="最大值"),  opts.MarkPointItem(type\_="min", name="最小值"),  opts.MarkPointItem(type\_="average", name="平均值"),  ]  )))  return c  bar\_base().render\_notebook()  游戏APP的评论人数分为所有版本（rating\_count\_tot）和现行版本（rating\_count\_ver），均值分别为12892.907183548701和460.3739057940809。所有版本评分统计量更大，数据相对更可靠。因此，将所有版本评分人数超过均值以及ratings大于等于4.0的记录视为优质游戏APP（假定其为受欢迎APP），则相对的为一般的游戏APP（假定其为相对不受欢迎APP）。    图 4 优质游戏APP-price计数柱形图 **图5 一般游戏APP-price计数柱形图** r1[(r1.rating\_count\_tot<12892.907183548701)&(r1.user\_rating<4)].pri  r1.price=r1.price\*6.99 #中美汇率 转化称人民币价格  p3=r1[(r1.rating\_count\_tot<12892.907183548701)&(r1.user\_rating<4)].price.value\_counts()  from pyecharts.charts import Bar  def bar\_base() -> Bar:  c = (Bar(init\_opts=opts.InitOpts(theme=ThemeType.ESSOS)).add\_xaxis(p3.index.tolist()).add\_yaxis("一般游戏APP-price计数柱形图",p3.tolist()).set\_series\_opts(  label\_opts=opts.LabelOpts(is\_show=False),  markpoint\_opts=opts.MarkPointOpts(  data=[  opts.MarkPointItem(type\_="max", name="最大值"),  opts.MarkPointItem(type\_="min", name="最小值"),  opts.MarkPointItem(type\_="average", name="平均值"),  ]  )))  return c  bar\_base().render\_notebook()  相对一般的2296个游戏APP，price的均值为0.4924876441515657，其中免费的APP有1386个（60.365%），计数柱形图如下。    图5一般游戏APP-price计数柱形图 **图6 优质和相对一般的游戏APP的支持设备计数柱形对比图** r1[(r1.rating\_count\_tot>12892.907183548701)&(r1.user\_rating>4)].price.mean()  p4=r1[(r1.rating\_count\_tot>12892.907183548701)&(r1.user\_rating>4)]['sup\_devices.num'].value\_counts()  p5=r1[(r1.rating\_count\_tot<12892.907183548701)&(r1.user\_rating<4)]['sup\_devices.num'].value\_counts()  p0=pd.concat([p4,p5],axis=1,sort=False,keys=['0','1']).fillna(0)  bar = Bar(init\_opts=opts.InitOpts(theme=ThemeType.ESSOS))  bar.add\_xaxis(p0['0'].index.tolist())  bar.add\_yaxis("相对一般的游戏APP- sup\_devices.num计数柱形图", p0['1'].tolist())  bar.set\_global\_opts(title\_opts=opts.TitleOpts(title='款式')  )  bar1 = Bar(init\_opts=opts.InitOpts(theme=ThemeType.ESSOS))  bar1.add\_xaxis(p0['0'].index.tolist())  bar1.add\_yaxis("优质游戏APP- sup\_devices.num计数柱形图", p0['0'].tolist())  bar1.set\_global\_opts(title\_opts=opts.TitleOpts(title='款式'))  g1 =Tab()  g1.add(bar, "相对一般的游戏APP- sup\_devices.num计数柱形图")  g1.add(bar1,"优质游戏APP- sup\_devices.num计数柱形图")  g1.render\_notebook()  分析：优质APP中，免费的所占比例更高，而且最高收费6.99比相对一般的APP（29.99）更低，符合常识，免费的东西用户更愿意尝试，当然更受欢迎啦。    图 6优质和相对一般的游戏APP的支持设备计数柱形对比图 | | | | | |
| 五、实验总结（包括心得体会、问题回答及实验改进意见，可附页）  通过本次实验，我使用了朴素贝叶斯分类、决策树、逻辑回归等算法分别自动识别窃漏电用户、cart数据集对其进行数据整理、分析和挖掘。对数据挖掘常见的算法有了进一步的了解。  Ⅰ.朴素贝叶斯分类是利用已经标记好类别的样本，通过分类识别函数，使分类结果达到所要求性能的过程  Ⅱ.CART 使用 Gini 指数作为选择特征的准则；  ID3 使用信息增益作为选择特征的准则；  C4.5 使用信息增益比作为选择特征的准则；  C5.0 C4.5 升级版，适用于大数据量；  CHAID 根据统计检验来确定自变量和分割点的选择；  Ⅲ.逻辑回归分两步:  第一步是建立预测函数，预测函数是用自变量预测因变量结果概率的一个函数；  第二步是构建损失函数（目标函数），损失函数表示的是预测值与真实值的平均差异。  Ⅳ.AppleStore数据集的数据分析及可视化  学会了用pyechart这个国产库绘制可视化图表进行数据分析。 | | | | | |
| 六、教师评语  1、完成所有规定的实验内容，实验步骤正确，结果正确；  2、完成绝大部分规定的实验内容，实验步骤正确，结果正确；  3、完成大部分规定的实验内容，实验步骤正确，结果正确；  4、基本完成规定的实验内容，实验步骤基本正确，所完成的结果基本正确；  5、未能很好地完成规定的实验内容或实验步骤不正确或结果不正确。  评定等级：  签名： | | | | | |