Python语言程序设计 实验报告

学号：2012682　　姓名：韩佳迅　　专业：计算机科学与技术

一、摘要

给定数据集是信息的标题、出处、相关链接以及相关评论，要求尝试判别信息真伪。本实验使用机器学习中的 **随机森林** 方法对文本内容进行 **二分类真假预测。**

首先，**获取数据集 + 数据处理**，加载train、test数据集，构建停用词列表，对 Title、Report Content 的文本特征进行分词、去停用词，并与 Ofiicial Account Name 的文本特征（该特征不分词）拼接；其次，进行 **特征工程**，将处理后的文本特征进行TF-IDF向量化、标准化，以及文本情感分析共同构成分类特征；然后，进行 **机器学习算法训练**，调用 sklearn 中的函数进行 RandomForest 训练，并调整参数至较优，最后，进行 **模型评估**，分析训练后的 **Accuracy**、**Precision**、**Recall**、**F1**、**ROC**、**AUC**。

二、具体分析

（一）获取数据集 + 数据处理

分析所给数据集，得到可用于构建特征的是 Ofiicial Account Name、Title、Report Content，进一步分析，发现需要对 Title、Report Content 进行jieba分词、去停用词，而由于Ofiicial Account Name本身就是专有特征词，不需要再分词，所以先对 Title、Report Content 处理好后直接与 Ofiicial Account Name拼接即可。

```
# 储存停用词列表
def stopwordlist():
    stopwords = [line.strip() for line in open("Chinesestopwords.txt",
encoding="UTF-8").readlines()]
    return stopwords

# 读取数据 + 数据处理
def product(path):
    data = pd.read_csv(path)
    label = data["label"]
    title = data["Title"]
    name = data["Ofiicial Account Name"]
    content = data["Report Content"]

    # 将每行的name值暂存进列表，以便下面与分词后的title和content拼接
    new_name = []
    for tem in name:
        res = ""
        res += str(tem)
        new_name.append(res)

    # 将title与content拼接，去停用词、jieba分词 + 与name拼接
```

```
    temp_data = title + content
    new_data = [] # new_data存放处理好的数据
    stopwords = stopwordlist()
    i = 0
    for sentense in temp_data:
        tmp = jieba.cut(str(sentense).strip())  # jieba分词
        res = ""
        for word in tmp:
            if word not in stopwords:
                if word != '\t':
                    res += word
                    res += "  "
        res += new_name[i]
        i += 1
        new_data.append(res)

    # print(new_data)  # 打印分割后的数据，看是否正确
    return new_data, label
```

打印数据（new_data）如下：

```
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\HJX\AppData\Local\Temp\jieba.cache
Loading model cost 0.532 seconds.
Prefix dict has been built successfully.

Out[4]: ['中国  反腐  风刮到  阿根廷  美到  瘫痪  女  总统  本子  摊上  大事  内容  不符  环球人物',
 '腾讯  如懿传  道歉  这部  亿大剧  上映  第一天  遭  网友  狂吐槽  愣  拍成  村头  恋曲  满口胡  言  西湖之声',
 '顺风  车  司机  奸杀  20  岁  女  乘客  落网  视频  曝光  滴滴  道歉  …  厦门晚报',
 '偶遇  鹿晗  关晓彤  旅行  七夕  情侣  真滴  甜  领个  屁证  妹  七夕  几天  前  图  今天  拿来  博  眼球  腾讯娱乐',
 '赵丽颖  冯绍峰  即将  公布  恋情  网友  曝  曝  没  区别  事件  实  腾讯娱乐',
 '六年  前  强制  堕胎  六年  强制  交配  消息  不实  ##  强制  生育  那段  太扯  投资者报',
 '不忍  直视  外面  下雨  杭州  公寓  成  水帘洞  墙角  缝隙  长出  蘑菇  …  无证据  说明  投诉  人数  众多  诱导  读者  行为  FM93交通之声',
 '五部委  喊话  年底  前  投案  从轻  处罚  部委  国务院  下属  机构  称谓  最高  法  最高检  脱离  政府  国家  组成  机构  这是常识性  错误  央视新闻',
 '上海  迪士尼  气球  遭  哄抢  网友  实在  丢人  东方网  记者  月  24  日  报道  昨天  月  23  日  上海  迪士尼  乐园  内有  游客  公然  哄抢  售卖  气球  东方网  记者  现场  知情  人士处  获悉  此事  源于  一场  误会  弄清  原委  多数  游客  很快  气球  归还  央视新闻',
 '自如  屡  曝  甲醛  超标  房客  生病  退  租金  需  封口  内容  实  央视新闻',
 '魏  璎珞  央视  电影频道  点名  批评  希望  这是  第一次  .  最后  一次  !  不想  说  赤裸裸  嫉妒  环球网',
```

（二）特征工程

- **Tf-idf 文本特征化 + StandardScaler标准化**

（1）对训练集、测试集数据以相同的 Tf-idf 方法进行文本特征转化。

TF-IDF 原理：

将 TF（词频——某个关键词在整篇文章中出现的频率）与 IDF（逆文档频率——表示关键词的普遍程度）相乘来计算 Tf-idf，并将所得文档标准化为单位长度。

代码实现方法：

使用 Sklearn 的 TfidfVectorizer，将分词后的文档通过 Tf-idf 值来进行表示，即用一个 Tf-idf 值的矩阵来表示文档。

关于 fit_transform 和 transform:

1）fit_transform作用于train：

fit_transform是fit和transform的组合，既包括了训练又包含了转换。

对训练集数据train先拟合fit，找到部分的整体指标，如均值、方差、最大值最小值等等（根据具体转换的目的）

然后对该train进行转换transform，从而实现数据的标准化、归一化等等。

2）transform作用于test:

意味着将train中算出的各项参数直接应用于test而无需计算参数了。

（2）对训练集、测试集数据以相同的 StandardScaler 方法进行标准化。 StandardScaler 原理：

对数据的每一个特征维度进行去均值和方差归一化。

标准差标准化（standardScale）使得经过处理的数据符合标准正态分布，即均值为0，标准差为1，其转化函数为：$x^* = \frac{x - \mu}{\sigma}$

其中μ为数据的均值，σ为数据的标准差。

代码实现方法：

使用 Sklearn 的 StandardScaler，实例化它的一个转换器类，用于保存训练集中的均值、方差参数，然后直接用于转换测试集数据。

关于 fit_transform 和 transform（同TfidfVectorizer）：

1）fit_transform作用于train：

fit_transform是fit和transform的组合，既包括了训练又包含了转换。

对训练集数据train先拟合fit，找到部分的整体指标，如均值、方差、最大值最小值等等（根据具体转换的目的）

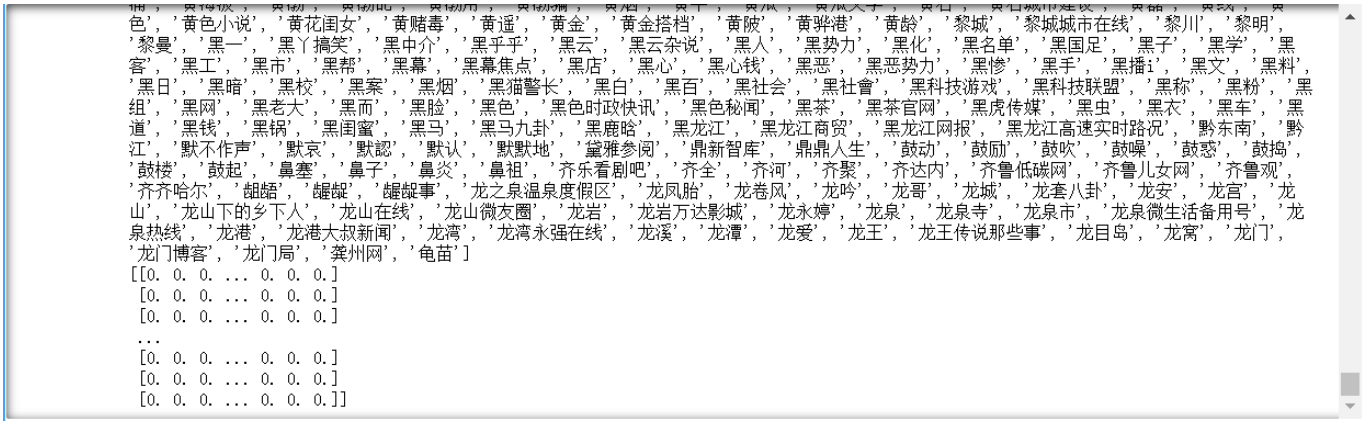
然后对该train进行转换transform，从而实现数据的标准化、归一化等等。

2）transform作用于test:

意味着将train中算出的各项参数直接应用于test而无需计算参数了

```
def handle(train, test):

    # Tf-idf向量化
    transfer_data = TfidfVectorizer()
    train = transfer_data.fit_transform(train)
    test = transfer_data.transform(test)
    # print(transfer_data.get_feature_names()) # 打印tf-idf后的特征词名字

    # StandardScaler标准化
    transfer_stand = StandardScaler(with_mean=False)
    train_stand = transfer_stand.fit_transform(train)
    test_stand = transfer_stand.transform(test)
    # print(train_stand.toarray()) # 打印标准化后的矩阵

    return  train_stand.toarray(),test_stand.toarray()
```

打印数据（tf-idf后的特征词名字 、 标准化后的矩阵）如下：



- **文本情感特征分析**

分析所给数据集，发现 title、content 对应的一些文本有较强的情感倾向，因此决定，分析它们的情感特点，并给出具体的情感分析得分，使其构成一个特征值。

原理：

一句话由若干词语组成，遍历这些词语，找到其中的积极、消极词汇，进行加减分，并且根据这些情感词前面的程度词、否定词决定加分 / 减分 以及加减分的大小程度，最后计算出这句话的情感分析得分。

代码实现方法：

1）先将 title 和 content 拼接；

2）使用 jieba 分词，把一句话拆分成单独的词语；

3）加载积极词语、消极词语、否定词、程度词等，并为每种词分别赋予它们对应的分数；

4）遍历计算每句话的情感得分，构成特征值。

5）标准化所得数据。

```
# 加载词库，并为每种词分别赋予它们对应的分数；
def loadDict(fileName, score):
    wordDict = {}
    with open(fileName,encoding="UTF-8") as fin:
        for line in fin:
            word = line.strip()
            wordDict[word] = score
    return wordDict

def loadExtentDict(fileName, level):
    extentDict = {}
    for i in range(level):
        with open(fileName + str(i + 1) + ".txt", encoding="UTF-8") as fin:
            for line in fin:
                word = line.strip()
                extentDict[word] = i + 1
    return extentDict

# 计算情感得分
```

```
def getScore(content):
    postDict = loadDict("正面情感词语.txt", 1)  # 积极情感词典
    negDict = loadDict("负面情感词语.txt", -1)  # 消极情感词典
    inverseDict = loadDict("否定词.txt", -1)  # 否定词词典
    extentDict = loadExtentDict("程度级别词语", 6)
    punc = loadDict("标点符号.txt", 1)
    exclamation = {"!": 2, "！": 2}

    words = jieba.cut(content)
    wordList = list(words)

    totalScore = 0  # 记录最终情感得分
    lastWordPos = 0  # 记录情感词的位置
    lastPuncPos = 0  # 记录标点符号的位置
    i = 0  # 记录扫描到的词的位置

    for word in wordList:
        if word in punc:
            lastPuncPos = i

        if word in postDict:
            if lastWordPos > lastPuncPos:
                start = lastWordPos
            else:
                start = lastPuncPos

            score = 1
            for word_before in wordList[start:i]:
                if word_before in extentDict:
                    score = score * extentDict[word_before]
                if word_before in inverseDict:
                    score = score * -1
            for word_after in wordList[i + 1:]:
                if word_after in punc:
                    if word_after in exclamation:
                        score = score + 2
                    else:
                        break
            lastWordPos = i
            totalScore += score
        elif word in negDict:
            if lastWordPos > lastPuncPos:
                start = lastWordPos
            else:
                start = lastPuncPos
            score = -1
            for word_before in wordList[start:i]:
                if word_before in extentDict:
                    score = score * extentDict[word_before]
                if word_before in inverseDict:
                    score = score * -1
            for word_after in wordList[i + 1:]:
                if word_after in punc:
                    if word_after in exclamation:
```

```
                        score = score - 2
                    else:
                        break
            lastWordPos = i
            totalScore += score
        i = i + 1

    return totalScore

# 处理文件中的情感特征
def haddle_emotion(path):
    data = pd.read_csv(path)
    title = data["Title"]
    content = data["Report Content"]
    combine = title + content
    emolist=[]
    for sentence in combine:
        sc=[]
        sc.append(getScore(sentence))
        emolist.append(sc)

    transfer_stand = StandardScaler(with_mean=False)
    emolist = transfer_stand.fit_transform(emolist)
    print(emolist)
    return emolist
```

打印数据（特征值 emolist）如下：

```
emolist:  [[0.        ]
 [0.        ]
 [0.        ]
 ...
 [0.72707618]
 [2.90830474]
 [6.90722375]]
```

（三）机器学习算法训练

本实验选择使用随机森林算法对数据进行分类。

代码实现方法：

1）实例化随机森林转换器类RandomForestClassifier()；

2）在调设置好的分类器上训练；

3）计算准确率得分和预测值。

```
def random_forest(train_data, test_data, train_label, test_label):
    rfc = RandomForestClassifier(n_estimators=161, random_state=90)
```

```
    rfc = rfc.fit(train_data, train_label)
    # 准确率
    result = rfc.score(test_data, test_label)
    print("score准确率：", result)
    # 预测值
    y_predict = rfc.predict(test_data) # 预测值 y_predict 真实值 test_label
    return y_predict
```

打印数据（训练结果）如下：

```
score准确率：  0.9116457943003649
```

（四）模型评估

评测指标原理

准确率：

准确率（Accuracy）是指预测正确的比例。

$$准确率 = Accuracy = A = \frac{预测正确的样本数}{样本总数}$$

➢ 感觉**正确率**更好

		Actual		
		Positive	Negative	
Predicted	Positive	TP	FP	
	Negative	FN	TN	
				$TP+FP+FN+TN$

$$准确率 = Accuracy = A = \frac{预测正确的正类数 + 预测正确的负类数}{样本总数} = \frac{TP+TN}{TP+FP+FN+TN}$$

精确率（Precision）：

精确率是指模型**正确预测正类**的频率。

		Actual		
		Positive	Negative	
Predicted	Positive	TP	FP	TP + FP
	Negative	FN	TN	
		TP + FN		

$$精确率 = Precise = P = \frac{\text{TP}}{Predicted\ Positive} = \frac{TP}{TP+FP}$$

召回率（Recall）：

在所有**实际的正类标签**中，模型正确地识别出了多少个？

		Actual		
		Positive	Negative	
Predicted	Positive	TP	FP	TP + FP
	Negative	FN	TN	
		TP + FN		

$$召回率 = Recall = R = \frac{\text{TP}}{Actual\ Positive} = \frac{TP}{TP+FN}$$

F1：

精确率和召回率的调和平均值。 $\frac{1}{F1} = \frac{1}{2}\left(\frac{1}{R}+\frac{1}{P}\right)$

		Actual		
		Positive	Negative	
Predicted	Positive	TP	FP	TP + FP
	Negative	FN	TN	
		TP + FN		

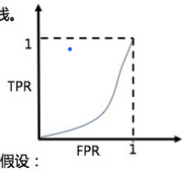
$$F1 = \frac{2}{\frac{1}{R}+\frac{1}{P}} = \frac{2*P*R}{P+R} = \frac{2TP}{2TP+FP+FN}$$

2. ROC

ROC（Receiver Operating Characteristic）Curve，受试者工作特征曲线。

不同**分类阈值**下的正类率（TPR）和假正类率（FPR）构成的曲线。

其中，TPR是**纵轴**，FPR**是横轴**。



绘制ROC曲线

模型预测的得分，将正类与负类区分开。以逻辑回归模型为例，假设：

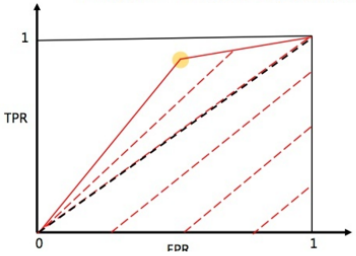
- 逻辑回归值高于 0.8 的，则归正类；
- 逻辑回归值低于 0.8 的，则归反类。

➢ 每个分类阈值，对应一个点(FPR，TPR)，描绘出每一个点即是ROC曲线。

3. AUC

AUC（**A**rea **U**nder the ROC **C**urve），ROC曲线下的面积。

AUC的原理： ROC曲线下面积越大，则分类模型效果越优。



AUC的几何意义： ROC曲线下的面积

➢ **完美分类**：AUC = 1

➢ **随机分类**：AUC = 0.5

实现

根据真实值、预测值，计算TP、FP、FN、TN，代入公式计算；

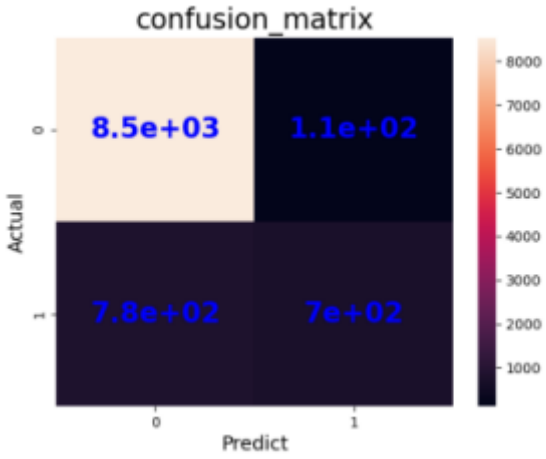
绘图实现可视化。

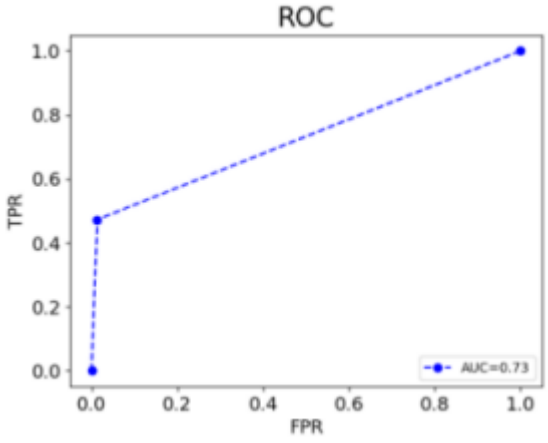
```
def binary_confusion_matrix(acts, pres):
    TP, FP, TN, FN = 0, 0, 0, 0
    for i in range(len(acts)):
        if acts[i] == 1 and pres[i] == 1:
            TP += 1
        if acts[i] == 0 and pres[i] == 1:
            FP += 1
        if acts[i] == 1 and pres[i] == 0:
            FN += 1
        if acts[i] == 0 and pres[i] == 0:
            TN += 1

    # 混淆矩阵可视化
    labels = [0, 1]
    cm = confusion_matrix(acts, pres, labels)
    print(cm)
    sns.heatmap(cm, annot=True, annot_kws={'size': 20, 'weight': 'bold', 'color':
'blue'})
    plt.rc('font', family='Arial Unicode MS', size=14)
    plt.title('confusion_matrix', fontsize=20)
    plt.xlabel('Predict', fontsize=14)
    plt.ylabel('Actual', fontsize=14)
    plt.show()

    # 精确率Precision
    P = TP / (TP + FP)
    print("精确率Precision:", P)

    # 召回率Recall
    R = TP / (TP + FN)
    print("召回率Recall:", R)

    # F1
    F1 = 2 / (1 / P + 1 / R)
    print('F1:', F1)

    # 准确率Accuracy
    A = (TP + TN) / (TP + FP + FN + TN)
    print("准确率Accuracy:", A)

    # ROC AUC
    act = np.array(acts)
    pre = np.array(pres)
    FPR, TPR, thresholds = metrics.roc_curve(act, pre)
    AUC = auc(FPR, TPR)
    print('AUC:', AUC)
    plt.rc('font', family='Arial Unicode MS', size=14)
    plt.plot(FPR, TPR, label="AUC={:.2f}".format(AUC), marker='o', color='b',
```

```
linestyle='--')
    plt.legend(loc=4, fontsize=10)
    plt.title('ROC', fontsize=20)
    plt.xlabel('FPR', fontsize=14)
    plt.ylabel('TPR', fontsize=14)
    plt.show()

    return TP, FP, TN, FN
```

打印数据（评估指标）如下：

```
score准确率： 0.9116457943003649
[[8545  114]
 [ 782  700]]
精确率Precision: 0.85995085995086
召回率Recall: 0.47233468286099867
F1: 0.6097560975609756
准确率Accuracy: 0.9116457943003649
AUC: 0.7295845951549479
```





综合分析各项指标以及测试集的正反例分布，可知：

1. 准确率较高，达到91+，说明模型预测正确的比例较好；
2. 精确率较高，说明在所有预测为正确的例子中，实际也为正确的比例较高，模型正确预测正类的比例较好；
3. 召回率较低，说明在所有实际的正例中模型正确预测的比例较低，不能更好地分类出所有的正例；
4. F1综合了准确率和召回率，表现为0.6
5. ROC、AUC表明模型的TPR较差，FPR较好，整体分类表现良好。

三、总结

本实验通过：

```
1. 获取数据集 + 数据处理（分词、去停用词）
2. 进行特征工程（Tf-idf、标准化、情感分析）
3. 机器学习算法训练（随机森林）
4. 模型评估(Accuracy、Precision、Recall、F1、ROC、AUC)
```

的方法对所给样本集进行分类，最后达到 91+ 的准确率和其余较好的分类指标，最终完成了分类功能。