安徽工业大学
ANHUI UNIVERSITY OF TECHNOLOGY

2022 - 2023 学年第 1 学期

机器学习课程设计报告

题　目：	基于 SVM 实现情感分析（英文文本分析）
姓　名：	张建才
学　院：	计算机科学与技术学院
专　业：	人工智能
学　号：	209074458
班　级：	智能 202
日　期：	2022 年 12 月 25 日

评　分：

声　明

本人郑重声明，所提交的课程设计由本人独立完成，保证不存在剽窃、抄袭他人成果的现象，文中所引用他人观点、材料、数据，图表等文献资料均以注释说明来源。

课程设计作者（签名）：张建才

2022 年 12 月 25 日

目　录

摘　要

英文文字的情感分析；SVM；数据清洗；多种评价指标；使用 SVM 预测英文文字的情感。文本特征分析。

一、课设目的与思路

目的：利用从推特抓取的评论，将其划分为积极、消极、中立，并以此为数据集训练模型，预测未知语句的情感。

思路：抓取数据；处理数据；准备好特征值、目标值，文本特征处理；划分数据集；预训练：使用多种算法训练，选取效果最好的。这里选取 SVM 作为模型训练。

二、模型算法

1. 模型简介

SVM 是找到集合边缘上的若干数据（称为支持向量），用这些点找出一个平面（称为决策面），使得支持向量到该平面的距离最大。

线性支持向量机学习算法如下：

选择惩罚参数,构造并求解凸二次规划问题；

求分离超平面，分类决策函数。

2. 评价指标

准确率(样本均衡时比较有意义)：所有预测正确的样本于总样本的比例，也就是常用的 scores，通常来说越接近 1 越好。

精确度(precesion)：也叫做查准率，表示少数类的预测准确率，也就是被预测为少数类的样本中，真正的少数类的占比。

召回率：也叫做查全率，敏感度等，表示的是少数类被预测正确的样本在少数类的占比，召回率越高表明捕获到的少数类越多，召回率是越高越好。

F1-measure：为了同时兼顾召回率和精确度，用召回率和精确度的调和平均来衡量两者之间的平衡 F1-measure 是在 0-1 之间的数，且越靠近 1 越好， 较高的 F1 保证了召回率和精确度都较高。

三、实验结果与分析

1.实验环境

硬件环境：CPU
软件环境：python3.11;sklearn1.2;（包括硬件环境和软件环境）

2.数据收集

https://github.com/saurabhv158/Sentiment-Analysis-using-Python

3.数据处理

1.使语句文本小写；
2.清理并删除文本中的上述停用词列表；
3.清除和删除标点符号；
4.清除和删除重复字符；
5.清理和删除 URL；
6.清除和删除数字；
7.删除短词，删除没有用的词；
8.把数据切分为特征 X 和标签 y；
9.切分数据集；
10.切分后训练集和测试集中的数据类型的比例跟切分前 y 中的比例一致。

4.模型训练

确定训练次数，添加惩罚项，每次数据输入时随机。

5.模型优化

因为特征比较多，维度较高，所以一般以训练次数为定值。

6. 模型测试与评估

```
from sklearn import svm
model = svm.SVC(kernel='linear')
model.fit(X_train, y_train)
model.score(X_test,y_test)
```

✓ 37m 58.5s

```
0.8653675819309123
```

结果的正确率还是蛮高的。就算是只训练了 1000 次，但是仍然可以达到一个很高的正确率。

```
from sklearn import svm
model = svm.SVC(kernel='linear',max_iter=1000)
model.fit(X_train, y_train)
model.score(X_test,y_test)
```

✓ 23m 49.3s

```
0.8662533215234721
```

7. 模型 UI 设计（可选）

无

四、结语

整合课程设计使用的比较简单，同时也是最基础的支持向量机，对数据集进行了较多的清理，但是对训练集没有进行较多的处理，如:K 折交叉验证；如降维：只取一部分的特征，或是合并几个特征，或是减低数据的类别；如升维：将几个特征合并成一个新的特征；寻找他们之间的现实意义的相关性。

训练也是极其简单：并未自造车轮，调用 sklearn 库，方便不少，从复杂的程序语言中解脱出来。

同时发现，这个线性分类器就可以实现，这一点与自身对数据的处理有关，这里我再赘述。总的来说，如果标签的值本身就是一种线性关系，那么，我们训练的自然也是线性。

不过，将语言转化为二维矩阵，使其能够被计算机理解。这是自然语言处理的一个很基本的思想。

参考文献

数据集：(2022, February 22). saurabhv158/Sentiment-Analysis-using-Python. Github. https://github.com/saurabhv158/Sentiment-Analysis-using-Python

思路：软件开发 | 如何使用机器学习来分析情感. Linux. https://linux.cn/article-15504-1.html

测试指标:机器学习 sklearn----支持向量机 SVC 模型评估指标_iostreamzl 的博客-CSDN 博客_sklearn svc. Blog. https://blog.csdn.net/weixin_43776305/article/details/121941230

模型简介：

支持向量机（SVM）——原理篇 - 知乎. Zhuanlan. https://zhuanlan.zhihu.com/p/31886934

SVM 多分类 - 止战 - 博客园. Cnblogs. https://www.cnblogs.com/zhizhan/p/4448668.html

附录：相关代码

```
import pandas as pd
import re
import string
import numpy as np
import random
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
#from wordcloud import WordCloud
from textblob import TextBlob
# Import the data
df=pd.read_json('Sample Data.txt', lines=True)

#convert the sample data to a csv file
df.to_csv('Sample Data.csv',index=None)
df.fillna('', inplace=True)
import nltk
from nltk.stem import WordNetLemmatizer
lemma = WordNetLemmatizer()
nltk.download('stopwords')
from nltk.corpus import stopwords
df['content']=df['content'].str.lower()
STOPWORDS = set(stopwords.words('english'))
def cleaning_stopwords(text):
    return " ".join([word for word in str(text).split() if word not in
STOPWORDS])
df['content']        =        df['content'].apply(lambda        text:
cleaning_stopwords(text))
import string
english_punctuations = string.punctuation
punctuations_list = english_punctuations
def cleaning_punctuations(text):
```

```
    translator = str.maketrans('', '', punctuations_list)
    return text.translate(translator)
df['content']= df['content'].apply(lambda x: cleaning_punctuations(x))
def cleaning_repeating_char(text):
    return re.sub(r'(.)1+', r'1', text)
df['content'] = df['content'].apply(lambda x:
cleaning_repeating_char(x))
def cleaning_URLs(data):
    return re.sub('((www.[^s]+)|(https?://[^s]+))',' ',data)
df['content'] = df['content'].apply(lambda x: cleaning_URLs(x))
def cleaning_numbers(data):
    return re.sub('[0-9]+', ' ', data)
df['content'] = df['content'].apply(lambda x: cleaning_numbers(x))
def transform_text(text):
    return ' '.join([word for word in text.split() if len(word) > 2])
df['content'] = df['content'].apply(lambda x: transform_text(x))
# Function which directly tokenize the tweet data
from nltk.tokenize import TweetTokenizer
tt = TweetTokenizer()
df['content']=df['content'].apply(tt.tokenize)
import nltk
st = nltk.PorterStemmer()
def stemming_on_text(data):
    text = [st.stem(word) for word in data]
    return data
df['content']= df['content'].apply(lambda x: stemming_on_text(x))
import nltk
nltk.download('wordnet')
lm = nltk.WordNetLemmatizer()
def lemmatizer_on_text(data):
    text = [lm.lemmatize(word) for word in data]
    return data
df['content'] = df['content'].apply(lambda x: lemmatizer_on_text(x))
#create a function to get the subjectivity
def getSubjectivity(text):
```

```
    return TextBlob(str(text)).sentiment.subjectivity
#create a function to get the polarity
def getpolarity(text):
    return TextBlob(str(text)).sentiment.polarity
#create two new columns
df['subjectivity']=df['content'].apply(getSubjectivity)
df['polarity']=df['content'].apply(getpolarity)
#create a function to compute the negative, neutral and positive
analysis
def getAnalysis(score):
    if score<0:
        return 'negative'
    elif score==0:
        return 'neutral'
    else:
        return 'positive'
df['analysis']=df['polarity'].apply(getAnalysis)
df.to_csv('text.csv')


import pandas as pd
import numpy as np
from nltk.stem.porter import PorterStemmer
import re
import string
import sys
sentiment_dataframe                                                    =
pd.read_csv("text.csv",usecols=['content','analysis'])
sentiment_dataframe['label']                                           =
pd.Categorical(sentiment_dataframe['analysis']).codes
y = sentiment_dataframe['label']
x=sentiment_dataframe['content']
from sklearn.feature_extraction.text import CountVectorizer
transfer = CountVectorizer()
feature_x = transfer.fit_transform(x)
```

```
text = feature_x.toarray()
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(text,y, test_size
= 0.20, random_state = 99)
import warnings
warnings.filterwarnings("ignore")
from sklearn import svm
model = svm.SVC(kernel='linear',max_iter=1000)
model.fit(X_train, y_train)
model.score(X_test,y_test)
X_test_result = model.predict(X_test)
from sklearn.metrics import f1_score
print(f1_score(y_test, X_test_result, average=None))
from sklearn.metrics import precision_score
print(precision_score(y_test, X_test_result, average=None))
from sklearn.metrics import recall_score
print(recall_score(X_test_result,y_test,average=None))
from joblib import dump
dump(model, 'SVM3.joblib')
print('Python: {}'.format(sys.version))
#print('sklearn: {}'.format(svm.__version__))
print('pandas: {}'.format(pd.__version__))
print('numpy: {}'.format(np.__version__))
#print('PorterStemmer: {}'.format(PorterStemmer.__version__))
```