Problem Set 1

孟昶-2021214431-Problem Set 1

September 16, 2021

Question1: Try to choose one Classification Model from scikit-learn Models and explain its theory and mean-ing of parameters. Loading the breast cancer dataset and apply the model you choose on the dataset,show your code and your precision on the task.

Answer1:

我选择 DecisionTreeClassifier

理论：决策树学习是一种采用树状结构的有监督机器学习方法。决策树是一个预测模型，表示对象特征和对象值之间的一种映射。其原理通俗的说，即给定一个输入值，从树节点不断往下走，直至走到叶节点，这个叶节点就是对输入值的一个预测或者分类。Sklearn 中实现了其三种算法 ID3、C4.5、CART，本人选择了 CART 进行试验，CART 算法其核心公式就是基尼指数的计算，基尼指数越大不确定越大，基尼指数的计算公式为：

$$Gini\,(D) = 1 - \sum p_i^2$$

其中 p_i 是 D 中元组 C_i 类的概率。

参数：

criterion:"gini", "entropy",default="gini": 特征选择标准。

splitter:"best","random",default="best": 特征划分标准，best 在特征的所有划分点中找出最优的划分点，random 随机的在部分划分点中找局部最优的划分点。

max_depth:int,default=None: 决策树最大深度。

min_samples_split:int or float, default=2: 内部节点再划分所需最小样本数。

min_samples_leaf:int or float, default=1: 叶子节点（即分类）最少样本数。

min_weight_fraction_leaf:float,default=0.0: 叶子节点（即分类）最小的样本权重和。

max_features：int, float or "auto", "sqrt", "log2", default=None：在划分数据集时考虑的最多的特征值数量。

random_state：int, RandomState instance or None, default=None：控制估计器的随机性。

max_leaf_nodes：int, default=None：最大叶子节点数。通过设置最大叶子节点数，可以防止过拟合。默认值 None，默认情况下不设置最大叶子节点数。如果加了限制，算法会建立在最大叶子节点数内最优的决策树。如果特征不多，可以不考虑这个值，但是如果特征多，可以加限制，具体的值可以通过交叉验证得到。

min_impurity_decrease：float, default=0.0：如果此分割导致杂质减少，则节点将被分割。

class_weight：dict, list of dict or "balanced", default=None：类别的权重。

ccp_alpha：non-negative float, default=0.0：复杂度参数。用于最小成本复杂度修剪。

代码：

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
# 载入brest_cancer数据集，并划分为训练集和测试集
X,y = load_breast_cancer(return_X_y=True)
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=0)

# 决策树分类
clf = DecisionTreeClassifier(random_state=0)
# 返回剪枝的过程中每一个ccp_alphas和impurities的值
path = clf.cost_complexity_pruning_path(X_train,y_train)
ccp_alphas, impurities = path.ccp_alphas, path.impurities

clfs = []
# 遍历每个ccp_alpha并作图表
for ccp_alpha in ccp_alphas:
    clf = DecisionTreeClassifier(random_state=0, ccp_alpha=ccp_alpha)
    clf.fit(X_train, y_train)
    clfs.append(clf)
clfs = clfs[:-1]
ccp_alphas = ccp_alphas[:-1]

# 尝试每一个ccp_alpha在训练和测试中的效果
train_scores = [clf.score(X_train, y_train) for clf in clfs]
test_scores = [clf.score(X_test, y_test) for clf in clfs]

fig, ax = plt.subplots()
ax.set_xlabel("alpha")
ax.set_ylabel("accuracy")
ax.set_title("Accuracy vs alpha for training and testing sets")
ax.plot(ccp_alphas, train_scores, marker='o', label="train",
        drawstyle="steps-post")
ax.plot(ccp_alphas, test_scores, marker='o', label="test",
        drawstyle="steps-post")
temp = DecisionTreeClassifier(random_state=0, ccp_alpha=0)
temp.fit(X_train,y_train)
print('if ccp_alphas = 0:')
print('The precision of train dataset is:',temp.score(X_train,y_train))
print('The precision of test dataset is:',temp.score(X_test,y_test))
temp1 = []
alpha = []
for ccp_alpha in ccp_alphas:
    temp1 = DecisionTreeClassifier(random_state=0, ccp_alpha=ccp_alpha)
    temp1.fit(X_train, y_train)
    if max(test_scores)==temp1.score(X_test,y_test):
        alpha.append(ccp_alpha)
        num = temp1.score(X_train,y_train)
print('The max precision of test dataset is:',max(test_scores),'and at this time the ccp_alphas is
    :',min(alpha))
print('and the precision of train dataset is:',num)

ax.legend()
plt.show()
```

结果：

```
if ccp_alphas = 0:
The precision of train dataset is: 1.0
The precision of test dataset is: 0.8811188811188811
The max precision of test dataset is: 0.9370629370629371
 and at this time the ccp_alphas is: 0.011443661971830986
and the precision of train dataset is: 0.9647887323943662
```

Figure 1: 结果图

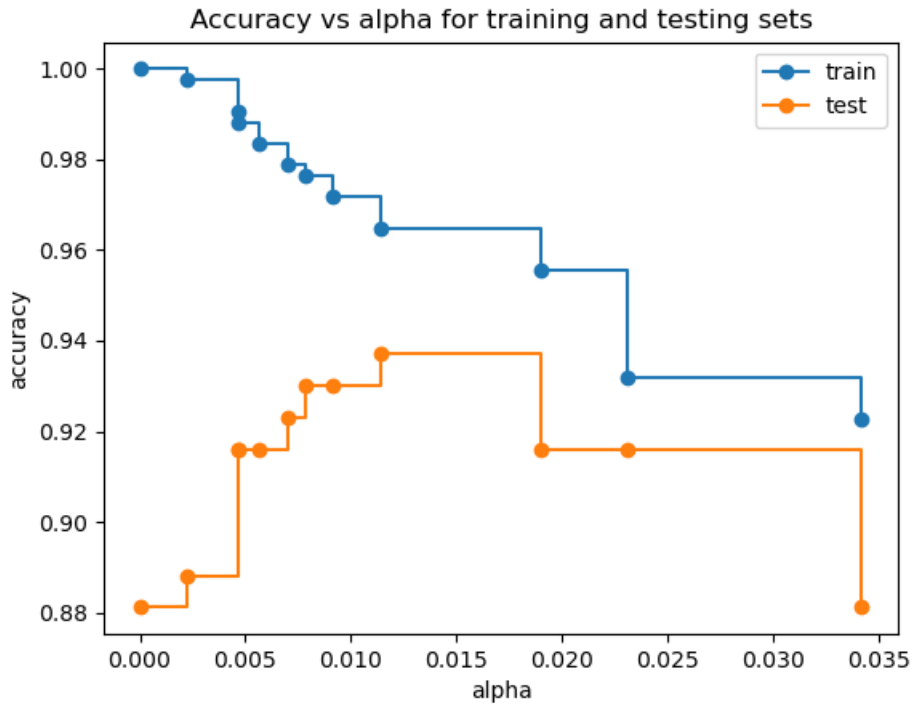

Figure 2: 精度随着剪枝参数变化趋势图

Question2: Classfication with Nearest Neighbours. In this question, you will use the scikit-learn' s KNN classifer to classify real vs. fake news headlines. The aim of this question is for you to read the scikit-learn API and get comfortable with training/validation splits. We will use a dataset of 1298 "fake news" headlines (which mostly include headlines of articles classfied as biased, etc.) and 1968 "real" news headlines, where the "fake news"headlines are from https://www.kaggle.com/mrisdal/fake-news/data and "real news"headlines are from https://www.kaggle.com/therohk/million-headlines.Write a function load data which loads the data, preprocesses it using a CountVectorizer (http://scikit-learn.org/stable/modules/classes.htmlmodule-sklearn.feature extraction.text), and splits the entire dataset randomly into 80% training, 10% validation, and 10% test examples.

Answer2:

代码:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import csv
import os
import random
import itertools
# path = r'D:\Users\53263\courses\tongji'
corpos = pd.DataFrame(columns=['text','kind'])
def load_data():
    # csv.field_size_limit(500 * 1024 * 1024)
    temp_fake = []
    temp_true = []
    with open('clean_fake.txt','r',encoding='utf-8') as f_fake:

        temp = 0
        for i in f_fake.readlines():
            # temp_fake.append(i[4])
            temp += 1
            corpos.loc[len(corpos)] = [i,'fake']
            if temp == 1298: break
    with open('clean_real.txt','r',encoding='utf-8') as f_true:

        temp = 0
        for i in f_true.readlines():
            temp += 1
            # temp_true.append(i[1])
            corpos.loc[len(corpos)] = [i,'true']
            if temp == 1968: break
    # with open('fake.txt','w',encoding='utf-8') as r_fake:
    #     for i in temp_fake:
    #         r_fake.write(i+'\n')
    # with open('true.txt','w',encoding='utf-8') as r_true:
    #     for i in temp_true:
    #         r_true.write(i+'\n')
```

```
40
41
42  def main():
43      load_data()
44      # print(corpos)
45      cv = CountVectorizer()
46      countvector = cv.fit_transform(corpos.iloc[:,0]).toarray()
47
48      # 转换数字
49      kind = np.unique(corpos['kind'].values)
50      nkind = np.zeros(len(countvector))
51      for i in range(len(kind)):
52          index = corpos[corpos['kind']==kind[i]].index
53          nkind[index] = i+1
54      pca = PCA(n_components=2)
55      newvector = pca.fit_transform(countvector)
56      plt.figure()
57      for i,c,m in zip(range(len(kind)),['r','b'],['o','^']):
58          index = corpos[corpos['kind']==kind[i]].index
59          x = newvector[index,0]
60          y = newvector[index,1]
61          plt.scatter(x,y,c=c,marker=m,label=kind[i])
62      plt.legend()
63      plt.xlim(-1,3)
64      plt.ylim(-2.5,5)
65      plt.xlabel('X Label')
66      plt.ylabel('Y Label')
67      plt.show()
68      # 划分数据集
69      index = np.arange(0,len(countvector),1)
70      random.shuffle(index)
71      index_train,index_val,index_test = index[:int(len(countvector)*0.8)],index[int(len(countvector
            )*0.8):int(len(countvector)*0.9)],\
72          index[int(len(countvector)*0.9):]
73      X_train = countvector[index_train]
74      y_train = corpos.iloc[index_train,1]
75
76      X_val = countvector[index_val]
77      y_val = corpos.iloc[index_val,1]
78
79      X_test = countvector[index_test]
80      y_test = corpos.iloc[index_test,1]
81
82      # knn分类
83      knn = KNeighborsClassifier(n_neighbors = 5)
84      knn.fit(X_train,y_train)
85      # 精度计算
86      print('The accuracy of val is: ',knn.score(X_val,y_val))
87      y_pred = knn.predict(X_test)
88      print('The accuracy of test is: ',knn.score(X_test,y_test))
89      acc = np.mean(y_pred == y_test)
90
91      # 计算混淆矩阵
92      knn_confusion = confusion_matrix(y_test,y_pred)
93      plt.imshow(knn_confusion,interpolation='nearest',cmap=plt.cm.Oranges)
94      plt.xlabel('y_pred')
95      plt.ylabel('y_True')
96      tick_marks = np.arange(len(kind))
```

```
    plt.xticks(tick_marks,kind,rotation=90)
    plt.yticks(tick_marks,kind)
    plt.colorbar()
    plt.title('confustion_matrix')
    for i,j in itertools.product(range(len(knn_confusion)),range(len(knn_confusion))):
        plt.text(i,j,knn_confusion[j,i],
                 horizontalalignment="center")
    plt.show()


if __name__ == '__main__':
    main()
```

结果：

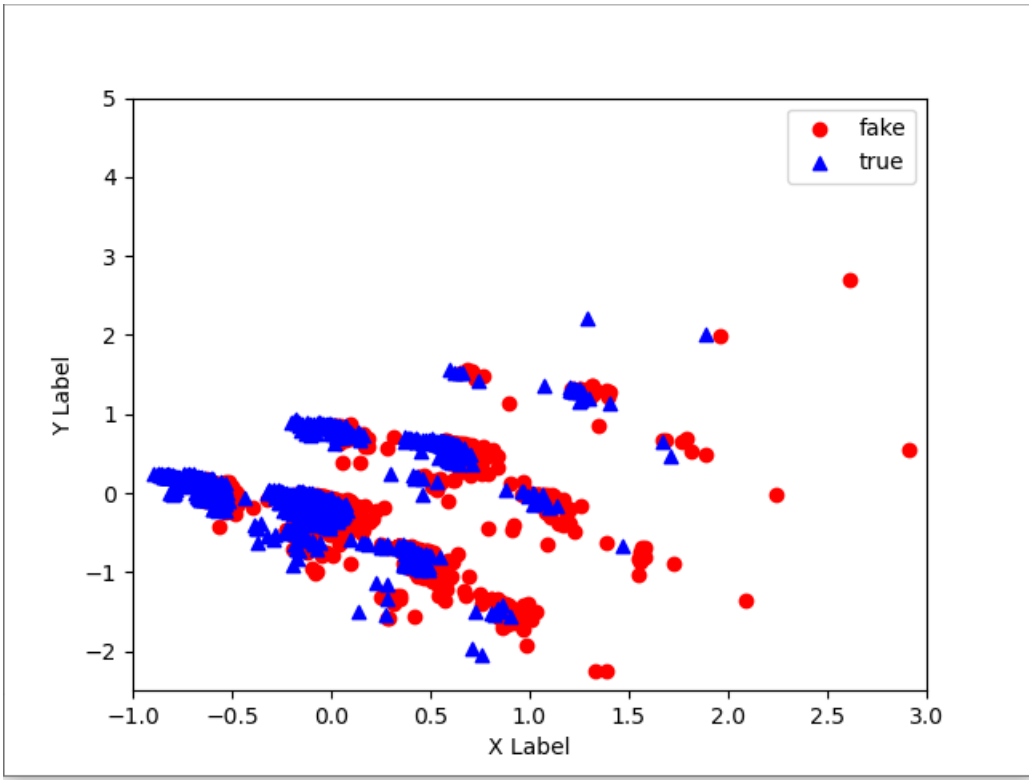

Figure 3: 数据分析散点图

```
The accuracy of val is:  0.70030581039755351
The accuracy of test is:  0.68195718654434251
```

Figure 4: 在评估集和测试集上的精度

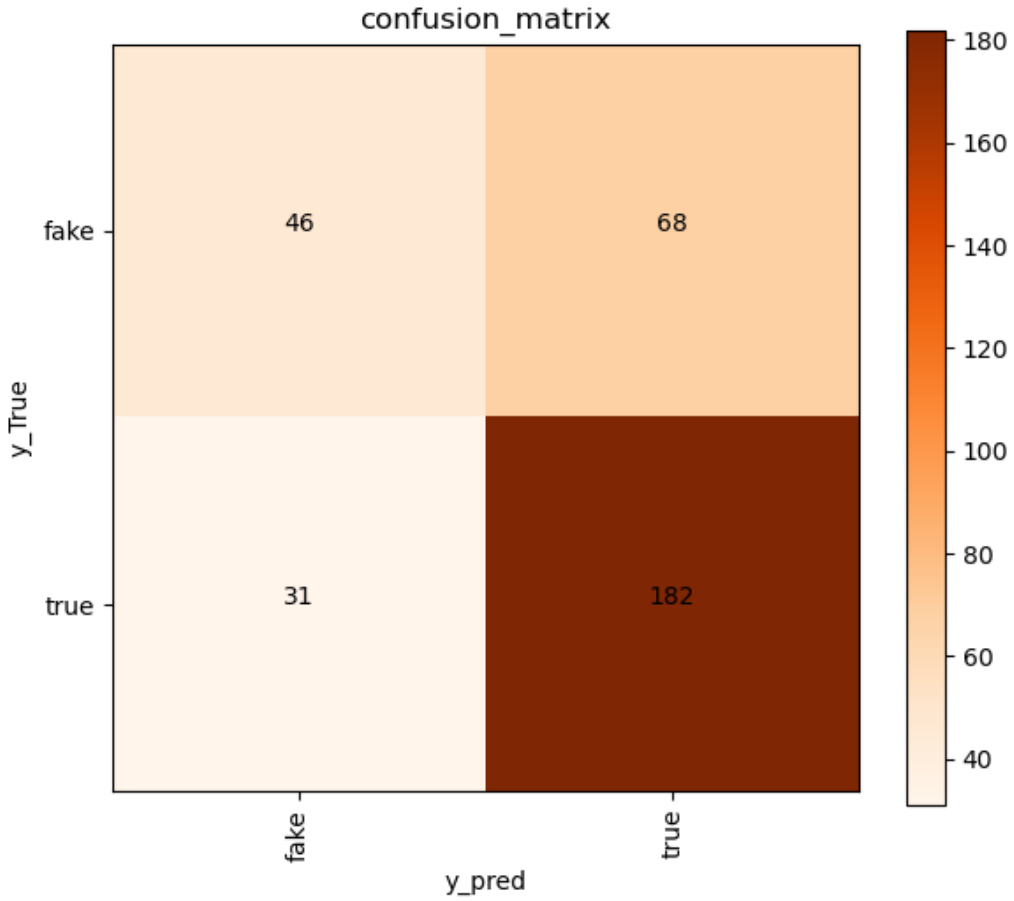

Figure 5: 混淆矩阵

Question3: Read the paper "Efficient Person Search: An Anchor-Free Approach" and give your review. (Hint: you can refer to " Tips and advice when you review a scientific paper" and browser the OpenReview website)

Answer3:

这篇文章主要解决了当前 SOTA 模型在执行对人员搜索的策略时会引入较高计算开销的问题，文章作者认为，在行人搜索中应用 anchor_free 的方法会出现尺度失准、区域失准和任务失准三个问题，并提出了一种更简单有效的 anchor_free 模型，即 AlignPS。在 AlignPS 基础上增加 ROI-Align 的 head，从而最终得到 ROIAlignPS，该模型同时保证了模型的鲁棒性和效率。在最后，文章作者在 CUHK-SYSU 和 PRW 数据集上与 SOTA 模型对比，充分展示了其模型的优越性。虽然这篇文章非常好，但是也存在了一些小问题，比如未设置足够多的消融实验来纵向比较算法的改进体现在哪；还有可以考虑在更多的 GPU 上运行，探索其速度的优越性。

Question4: Write your Project proposal and your plan.(Please choose one competition from "AI innovation and application competition" , "QQ Browser 2021 AI algorithm competition" or "ML Reproducibility Chal- lenge 2021")

Answer4:

我们选择的是中关村与旷视合办的"中关村数字科技联合创新大赛"的"旷视 AI 智慧交通开源赛道", 这项比赛目前正在进行中。比赛的具体任务是在复杂道路场景下对十字路口的交通标志进行检测。由于数据集中的交通标志图片都采集于真实场景, 它们将不可避免地受到多种图像退化影响, 例如, 光照、模糊、过曝光、遮挡。此外, 交通标志的小目标性也给检测工作带来了困难。因此, 这是一个集数据预处理、小目标检测等综合技术于一身的比赛任务。目前报名已截止, 共有 177 支队伍参加。本小组共四人, 组长为张恒煜, 成员为孟昶、李曦辉、何春明。我们的原定计划是由一人负责数据预处理算法, 一人负责小目标检测算法, 两人负责组合起来的检测系统的性能验证、分析与优化。实际操作中, 大家各有侧重却也紧密合作, 大部分决策都由四人共同商讨而做出决定。小组成员目前正在积极合作, 每天都会有针对算法提升的点子经过探讨而迸发。