به نام هستی بخش مهربان

هوش مصنوعی – پاییز ۹۸

پروژه شماره ۴

علی شهیدی - ۸۱۰۱۹۴۳۴۱

۱) در ابتدا ۳۰(یا ۲۰) درصد داده ها را به عنوان تست برمیدارم و سپس تابع زیر را اجرا میکنم و پس از تخمین زدن مقدار target هر داده ورودی دقت را بدست می آورم.

```
def classification(train_data, test_data):
    train_label = train_data[TARGET]
    train_data = train_data.drop('target', axis=1)

    test_label = test_data[TARGET]
    test_data = test_data.drop('target', axis=1)

    dt = DecisionTreeClassifier()
    dt.fit(train_data, train_label)


    pred = dt.predict(test_data)
    test_accuracy = accuracy_score(test_label, pred)
```

۲)

۱. داده هایی که برای train در بخش قبل استفاده میشد را به تابع زیر دادم و به من ۵ تا زیرمجموعه ۱۵۰تایی برمیگرداند.

```
def subset_gen(data):
    df_trimmed = []
    for i in range(NUM_OF_SUBSET):
        chosen_idx = np.random.choice(len(data),
replace=False, size=150)
        df_trimmed.append(data.iloc[chosen_idx])
    return df_trimmed
```

۲. برای این روش هر بار تابع `classification` را جدا برای هر یک از این زیرمجموعه ها که دربخش یک بدست آوردم ران کردم. سپس تخمین ها را در لیستی نگه داشتم (در اینجا ۵*۱۵۰ تایی) و بعد از آن بیشترین رای را انتخاب کردم. (مثلا ۳ نفر به ۱ رای داده بودند target آن خانه را یک گرفتم)

۳. همانطور که در فروم گفته بودند این بخش را فقط برای درخت تصمیم زدم. به این صورت که در ابتدا دقت را با همه ویژگی ها حساب کرده و سپس هر بار یک ویژگی را حذف کردم و دقت را محاسبه کردم. (همه دقت ها در جواب آمده اند) و در آخر ویژگی ای که با حذفش دقت افت کمتری دارد را چاپ کردم. باید دقت داشت که هر بار این ویژگی متفاوت است و بستگی به داده ورودی انتخاب شده و عوامل دیگر دارد.

۴. به صورت تصادفی انتخاب کرده و محاسبه کردم:

```
data = train_data.drop('target', axis=1)
data = data.sample(NUM_OF_ATT, axis=1)
data.insert(len(data.columns), "target",
train_data["target"])
```

۵. این بخش را به دو صورت پیاده سازی کردم: در اول کار برای همه ۵ درخت ویژگی هایی که از مرحله قبل داشتم را استفاده کردم. برای بخش دوم هر درخت ۵ ویژگی جدا دارد. که دقت ها نیز با هم متفاوت شد.

سوال‌ها:

۱-

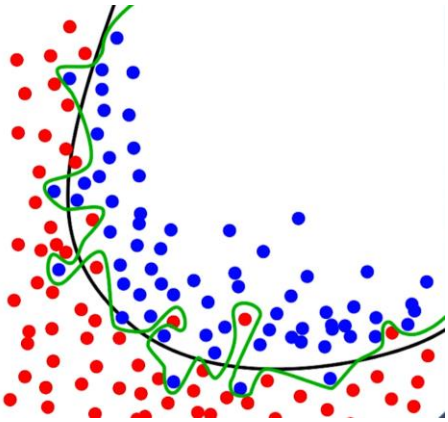
Bootstrapping یک روش نمونه گیری میباشد. از میان n نمونه ی موجود ، k نمونه با جایگزینی انتخاب می شوند. سپس الگوریتم یادگیری خود را روی هر یک از این نمونه ها اجرا می کنیم. نکته نمونه برداری از جایگزین ها، نمونه برداری مجدد از نمونه برداری های تصادفی است . اگر بدون تعویض انجام شود، نمونه های کشیده شده به نمونه های قبلی بستگی دارند و بنابراین تصادفی نیستند.

مزیت: این روش طوری عمل میکند که انگار در خدمت موجودیت بزرگتری هستند (چند تخمینگر کوچک در کنار هم یک تخمینگر بزرگ و قدرتمند را تشکیل میدهند با درنظر گرفتن حداکثر رای ها.) در نتیجه این تخمینگر قویتر به کاهش واریانس کمک میکند و از overfitting جلوگیری میکند.

توضیح: از آنجا که جنگل تصادفی بر روی نمونه های داده ها ساخته شده است ، خروجی های درختان جداگانه می تواند به عنوان متغیرهای تصادفی توزیع شده یکسان مشاهده شود. بنابرین با میانگین گرفتن از این داده ها برای k درخت, واریانس تخمین نهایی برابر با مقدار $p*sigma^2 + (1 - p)sigma^2 / k$ خواهد بود. در اینجا p همبستگی زوجی بین درختان است. اگر k خیلی بزرگ باشد عبارت سمت راستی حذف میشود و فقط $p*sigma^2$ باقی میماند.

-۲

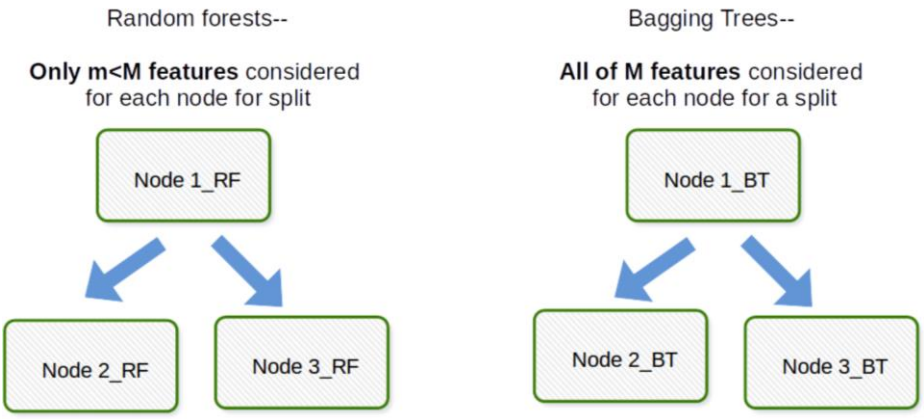
Overfitting وقتی رخ میدهد که پیش بینی ما خیلی وابسته به داده ی train شده باشد و در این صورت اگر داده ای با مشخصات متفاوت از داده ی train شده بیاید نتواند آن را درست تخمین بزند. مثلا عکس زیر دو نوع پیش بینی را نمایش میدهد. که همانطور که مشاهده میکنیم خط سبز خیلی دقیق تر داده ی train را تخمین زده است ولی خط سیاه در مواجه با داده های جدید به طور کلی بهتر عمل میکند. به عبارت دیگر Overfittingخطای مدل سازی است که وقتی یک تابع خیلی نزدیک به مجموعه محدودی از نقاط داده قرار میگیرد رخ می دهد.



درخت تصمیم واحد نسبت به تغییرات داده بسیار حساس است و به راحتی می تواند در به سمت نویر overfitشد. وقتی از الگوریتم bagging استفاده میکنیم به دلیل انتخاب تصادفی ویژگی ها و داده ها overfitting کاهش میابد. با این حال ، خطای تخمین آن به صفر نمی رسد.

-۳

شبیه هم هستند با این تفاوت که در جنگل تصادفی فقط زیرمجموعه ای از ویژگی ها انتخاب میشود و بهترین ویژگی برای تقسیم درخت استفاده میشود ولی در bagging همه ویژگی ها استفاده میشوند و از همه ویژگی ها برای جدا کردن نودها استفاده میشود.



از جنگل تصادفی بنابر دلایلی مختلفی استفاده میشود:

۱) درخت تصمیم نسبت به overfitting حساس است و جنگل تصادفی آن را کاهش میدهد.
۲) ساختن درخت تصمیم به این بستگی دارد که یک الگوریتمی داشته باشیم که بهترین انتخاب را در هر نود تخمین بزند (هزینه دارد) میتواند از الگوریتم هایی استفاده کرد که به صورت محلی کار کنند یا از درخت تصمیم استفاده کرد.

۴-

قسمت ۱ درخت تصمیم است.

قسمت ۲,۲ الگوریتم Bagging.

قسمت آخر الگوریتم جنگل تصادفی.

الگوریتم قسمت ۲,۲ از همه بهتر جواب میدهد چون از همه ویژگی ها استفاده کرده و جنگل تصادفی را پیاده‌سازی کرده است. بین قسمت ۱و ۵ دقت کاملا به ویژگی هایی که انتخاب کرده ایم بستگی دارد و نمیتوان مقایسه قاطعانه ای بین آن دو انجام داد.

جواب ها برای هر بخش با ۷۰ درصد داده‌ی **train** شده:

```
"F:\term 9\AI\Project 4\CA4\venv\Scripts\python.exe" "F:/term 9/A
********************
 Train Data
target
0     98
1    114
Name: age, dtype: int64
Total  212
********************
 Test Data
target
0    40
1    51
Name: age, dtype: int64
Total  91
********************
 Classification
Decision Tree accuracy score
    test: 0.8241758241758241
********************
 Classification With Removing One Attribute
test accuracy without removing any attribute 0.7802197802197802
test accuracy with removing age  =  0.7802197802197802
test accuracy with removing sex  =  0.7912087912087912
test accuracy with removing cp  =  0.8021978021978022
test accuracy with removing trestbps  =  0.8131868131868132
test accuracy with removing chol  =  0.8241758241758241
test accuracy with removing fbs  =  0.7802197802197802
test accuracy with removing restecg  =  0.8021978021978022
test accuracy with removing thalach  =  0.7692307692307693
test accuracy with removing exang  =  0.8021978021978022
test accuracy with removing oldpeak  =  0.7912087912087912
test accuracy with removing slope  =  0.7802197802197802
test accuracy with removing ca  =  0.7032967032967034
test accuracy with removing thal  =  0.8021978021978022

min accuracy is resulted with removing  age  =  0.78021978021978
********************
```

```
test accuracy with removing slope  =  0.7802197802197802
test accuracy with removing ca  =  0.7032967032967034
test accuracy with removing thal  =  0.8021978021978022

min accuracy is resulted with removing  age  =  0.7802197802197802
********************
 Bagging
Random Forest accuracy score
    test: 0.8351648351648352
********************
 Calculate accuracy of the decision tree with 5 random attributes
Columns =  ['age', 'restecg', 'thal', 'cp', 'ca', 'target']
Decision Tree accuracy score
    test: 0.7692307692307693
********************
 Calculate accuracy of the random forest with 5 random attributes
Columns =  ['age', 'restecg', 'thal', 'cp', 'ca', 'target']
Random Forest accuracy score
    test: 0.8241758241758241
********************
 Calculate accuracy of the random forest with 5 random attributes for each tree
Columns =  ['age', 'ca', 'cp', 'thal', 'restecg', 'target']
Columns =  ['age', 'ca', 'restecg', 'cp', 'thal', 'target']
Columns =  ['thal', 'restecg', 'cp', 'ca', 'age', 'target']
Columns =  ['cp', 'age', 'ca', 'thal', 'restecg', 'target']
Columns =  ['age', 'restecg', 'cp', 'thal', 'ca', 'target']
Random Forest accuracy score
    test: 0.7692307692307693
```

بار دوم ران شدن:

```
"F:\term 9\AI\Project 4\CA4\venv\Scripts\python.exe" "F:/term 9/AI/Proj
********************
 Train Data
target
0     92
1    114
Name: age, dtype: int64
Total  206
********************
 Test Data
target
0    46
1    51
Name: age, dtype: int64
Total  97
********************
 Classification
Decision Tree accuracy score
    test: 0.8762886597938144
********************
 Classification With Removing One Attribute
test accuracy without removing any attribute 0.865979381443299
test accuracy with removing age  =  0.7938144329896907
test accuracy with removing sex  =  0.8144329896907216
test accuracy with removing cp  =  0.7628865979381443
test accuracy with removing trestbps  =  0.8041237113402062
test accuracy with removing chol  =  0.8041237113402062
test accuracy with removing fbs  =  0.8247422680412371
test accuracy with removing restecg  =  0.845360824742268
test accuracy with removing thalach  =  0.845360824742268
test accuracy with removing exang  =  0.8762886597938144
test accuracy with removing oldpeak  =  0.845360824742268
test accuracy with removing slope  =  0.865979381443299
test accuracy with removing ca  =  0.845360824742268
test accuracy with removing thal  =  0.7422680412371134

min accuracy is resulted with removing  slope  =  0.865979381443299
********************
```

```
test accuracy with removing slope  =  0.865979381443299
test accuracy with removing ca  =  0.845360824742268
test accuracy with removing thal  =  0.7422680412371134

min accuracy is resulted with removing  slope  =  0.865979381443299
********************
 Bagging
Random Forest accuracy score
    test: 0.8350515463917526
********************
 Calculate accuracy of the decision tree with 5 random attributes
Columns =  ['restecg', 'trestbps', 'thalach', 'sex', 'thal', 'target']
Decision Tree accuracy score
    test: 0.6804123711340206
********************
 Calculate accuracy of the random forest with 5 random attributes
Columns =  ['restecg', 'trestbps', 'thalach', 'sex', 'thal', 'target']
Random Forest accuracy score
    test: 0.6907216494845361
********************
 Calculate accuracy of the random forest with 5 random attributes for each tree
Columns =  ['restecg', 'thalach', 'sex', 'thal', 'trestbps', 'target']
Columns =  ['restecg', 'thal', 'sex', 'trestbps', 'thalach', 'target']
Columns =  ['thalach', 'sex', 'thal', 'restecg', 'trestbps', 'target']
Columns =  ['restecg', 'sex', 'trestbps', 'thal', 'thalach', 'target']
Columns =  ['restecg', 'trestbps', 'sex', 'thalach', 'thal', 'target']
Random Forest accuracy score
    test: 0.6804123711340206
```

با ۲۰ درصد داده‌ی تست:

```
********************
 Train Data
target
0    117
1    129
Name: age, dtype: int64
Total  246
********************
 Test Data
target
0    21
1    36
Name: age, dtype: int64
Total  57
********************
 Classification
Decision Tree accuracy score
    test: 0.8070175438596491
********************
 Classification With Removing One Attribute
test accuracy without removing any attribute 0.7543859649122807
test accuracy with removing age  =  0.8421052631578947
test accuracy with removing sex  =  0.7017543859649122
test accuracy with removing cp  =  0.7543859649122807
test accuracy with removing trestbps  =  0.6842105263157895
test accuracy with removing chol  =  0.7368421052631579
test accuracy with removing fbs  =  0.7543859649122807
test accuracy with removing restecg  =  0.7894736842105263
test accuracy with removing thalach  =  0.7368421052631579
test accuracy with removing exang  =  0.8070175438596491
test accuracy with removing oldpeak  =  0.7368421052631579
test accuracy with removing slope  =  0.7368421052631579
test accuracy with removing ca  =  0.6842105263157895
test accuracy with removing thal  =  0.7719298245614035
```

```
test accuracy with removing ca  =  0.6842105263157895
test accuracy with removing thal  =  0.7719298245614035

min accuracy is resulted with removing  cp  =  0.7543859649122807
********************
 Bagging
Random Forest accuracy score
    test: 0.7719298245614035
********************
 Calculate accuracy of the decision tree with 5 random attributes
Columns =  ['trestbps', 'ca', 'restecg', 'thal', 'cp', 'target']
Decision Tree accuracy score
    test: 0.7719298245614035
********************
 Calculate accuracy of the random forest with 5 random attributes
Columns =  ['trestbps', 'ca', 'restecg', 'thal', 'cp', 'target']
Random Forest accuracy score
    test: 0.8070175438596491
********************
 Calculate accuracy of the random forest with 5 random attributes for each tree
Columns =  ['ca', 'restecg', 'cp', 'trestbps', 'thal', 'target']
Columns =  ['ca', 'cp', 'thal', 'restecg', 'trestbps', 'target']
Columns =  ['thal', 'ca', 'restecg', 'trestbps', 'cp', 'target']
Columns =  ['cp', 'thal', 'ca', 'trestbps', 'restecg', 'target']
Columns =  ['cp', 'restecg', 'thal', 'ca', 'trestbps', 'target']
Random Forest accuracy score
    test: 0.7543859649122807
```