به نام خدا

تمرین ٤

DecisionTree

KNN

Ensemble (AdaBoost)

درس مبانی داده کاوی

استاد : دکتر محسن غلامی

کسری صمدی<۹۹۳۶۲۳۰۳۰>

بهمن ۱٤۰۲

فهرست مطالب

فهرست تصاویر

مقدمه

در این تمرین، به معرفی و پیاده‌سازی سه الگوریتم درخت تصمیم[1]، k نزدیک‌ترین همسایه[2] و یادگیری ترکیبی (مدل آدابوست)[3] می‌پردازیم. کد هر کدام از الگوریتم‌ها همراه با دیتاست‌های آن‌ها در پوشه‌هایی با نام الگوریتم مربوطه قرار گرفته‌اند و همراه با این داکیومنت ارسال می‌شوند.

۱- الگوریتم درخت تصمیم

الگوریتم درخت تصمیم، یک ساختار درختی از داده‌های آموزشی ایجاد می‌کند که مجموعه‌ای از تصمیمات را نشان می‌دهد که منجر به پیش‌بینی متغیر هدف می‌شود. این الگوریتم با تقسیم بازگشتی داده‌ها به زیرمجموعه‌ها، بر اساس ویژگی‌هایی کار می‌کند که بیشترین اطلاعات را تا زمانی که پیشبینی یا تصمیم نهایی گرفته شود، عمل می‌کند.

۱-۱- مراحل اصلی الگوریتم درخت تصمیم

- با یک مجموعه داده شروع می‌شود و یک فیچر برای آزمایش انتخاب می‌شود.
- داده‌ها به زیر مجموعه‌هایی تقسیم می‌شوند که مقدار یکسانی برای فیچر انتخاب شده دارند.
- آنتروپی زیر مجموعه‌ها محاسبه می‌شود.
- با مقایسه آنتروپی قبل و بعد از پارتیشن‌بندی، بهره اطلاعاتی (Information gain) فیچر محاسبه می‌شود.
- فیچری با بالاترین بهره اطلاعاتی به عنوان گره بعدی درخت انتخاب می‌شود.
- مراحل ۲-۵ به صورت بازگشتی تکرار می‌شود تا زمانی که همه فیچرها استفاده شوند یا یک معیار توقف برآورده شود.
- یک برچسب کلاس به هر گره برگ درخت بر اساس اکثریت کلاس داده‌های آموزشی که در آن گره برگ قرار می‌گیرند، اختصاص داده می‌شود.

۱-۲- معرفی دیتاست

این دیتاست در مورد مجموعه‌ای از بیماران است که همه آنها از یک بیماری رنج می برند. در طول دوره درمان، هر بیمار به با یکی از ۵ دارو، داروی A، داروی B، داروی C، داروی X و Y درمان می‌شود و به عبارتی هر بیمار به یکی از این ۵ دارو پاسخ مثبت می‌دهد.

[1] DecisionTree
[2] KNN
[3] Ensemble_AdaBoost

در این بخش میخواهیم با این دیتاست، مدل درخت تصمیم بسازیم تا بفهمیم کدام دارو ممکن است برای یک بیمار با ویژگی‌های منحصر به فرد خود با همان بیماری در آینده مناسب باشد.

ویژگی‌های این مجموعه داده شامل سن، جنسیت، فشار خون ، کلسترول بیماران و نتیجه تست مقدار سدیم و پتاسیم خون است و متغیر هدف دارویی است که هر بیمار به آن پاسخ داده است.

ابتدا مدل درخت تصمیم را برا اساس بخشی از این دیتاست آموزش می‌دهیم و سپس از آن برای پیش‌بینی کلاس یک بیمار ناشناخته یا تجویز دارو برای بیمار جدید استفاده می‌کنیم.

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	drugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	drugY

عکس ۱ بخشی از دیتاست دارو همراه با ویژگی‌های آن

Kasra Samadi 993623030

HW4 DataMining

DecisionTree

Tasks :

1- Downloading data from IBM Object Storage

2- Obtaining information (#Rows, #Columns, Types of Columns, Missing values)

3- Pre-processing

4- Decision Tree

5- Evaluation and Confusion matrix

عکس ۲ فهرست بخش پیاده‌سازی الگوریتم درخت تصمیم

Task 1

Downloading data from IBM Object Storage

download link : https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-ML0101EN-SkillsNetwork/labs/Module%203/data/drug200.csv

Import libraries

```
from sklearn.tree import DecisionTreeClassifier
import numpy as np
import pandas as pd
import seaborn as sns
```

عکس ۳ دانلود دیتاست دارو و ایمپورت کتابخانه‌های مورد نیاز

Reading drug200.csv

```
df = pd.read_csv("drug200.csv", delimiter=",")
df[0:5]
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	drugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	drugY

عکس ۴ خواندن دیتاست drug200.csv

Task 2

Obtaining information

This dataFrame has 200 rows and 6 columns

```
[ ]: df.shape
```

```
[ ]: (200, 6)
```

عکس ۵ نمایش تعداد سطرها و ستون‌ها

Find columns names and their types

```
df.dtypes
```

```
Age              int64
Sex             object
BP              object
Cholesterol     object
Na_to_K        float64
Drug            object
dtype: object
```

عکس ۶ نوع‌های داده‌ای ستون‌ها

Check Missing values

```
df.isna().sum()
```

```
Age            0
Sex            0
BP             0
Cholesterol    0
Na_to_K        0
Drug           0
dtype: int64
```

عکس ۷ نمایش تعداد مقادیر از دسته رفته

در این دیتافریم مقادیر از دست رفته‌ای در ستون‌ها وجود ندارد و همچنین مقادیر ناسازگار نیز یافت نمی‌شود.

Task 3

Pre-processing

X as the Feature Matrix

y as the response vector (target)

```
X = df[['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K']].values
X[0:5]
```

```
array([[23, 'F', 'HIGH', 'HIGH', 25.355],
       [47, 'M', 'LOW', 'HIGH', 13.093],
       [47, 'M', 'LOW', 'HIGH', 10.114],
       [28, 'F', 'NORMAL', 'HIGH', 7.798],
       [61, 'F', 'LOW', 'HIGH', 18.043]], dtype=object)
```

عکس ۸ مرحله پیش‌پردازش : جداسازی متغیرهای مستقل

```
y = df["Drug"]
y[0:5]
```

```
0    drugY
1    drugC
2    drugC
3    drugX
4    drugY
Name: Drug, dtype: object
```

عکس ۹ مرحله پیش‌پردازش : جداسازی متغیر وابسته

Convert these categorical features to numerical values using pandas.get_dummies()

```
from sklearn import preprocessing
le_sex = preprocessing.LabelEncoder()
le_sex.fit(['F','M'])
X[:,1] = le_sex.transform(X[:,1])


le_BP = preprocessing.LabelEncoder()
le_BP.fit([ 'LOW', 'NORMAL', 'HIGH'])
X[:,2] = le_BP.transform(X[:,2])


le_Chol = preprocessing.LabelEncoder()
le_Chol.fit([ 'NORMAL', 'HIGH'])
X[:,3] = le_Chol.transform(X[:,3])

X[0:5]
```

```
array([[23, 0, 0, 0, 25.355],
       [47, 1, 1, 0, 13.093],
       [47, 1, 1, 0, 10.114],
       [28, 0, 2, 0, 7.798],
       [61, 0, 1, 0, 18.043]], dtype=object)
```

عکس ۱۰ مرحله پیش‌پردازش : تبدیل متغیرهای دسته‌ای به مقادیر عددی

Task 4

Decision Tree

Split data

```
from sklearn.model_selection import train_test_split
X_trainset, X_testset, y_trainset, y_testset = train_test_split(X, y, test_size=0.3, random_state=3)
```

عکس ۱۱ درخت تصمیم : جداسازی داده‌های تست و آموزش

Modeling

Create an instance of the DecisionTreeClassifier called drugTree with riterion="entropy"

```
drugTree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
drugTree # it shows the default parameters
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

عکس ۱۲ درخت تصمیم : ایجاد مدل

we will fit the data with the training feature matrix X_trainset and training response vector y_trainset

```
drugTree.fit(X_trainset,y_trainset)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

عکس ۱۳ درخت تصمیم : آموزش مدل

Prediction

```
predTree = drugTree.predict(X_testset)
```

```
print (predTree [0:5])
print (y_testset [0:5])
```

```
['drugY' 'drugX' 'drugX' 'drugX' 'drugX']
40     drugY
51     drugX
139    drugX
197    drugX
170    drugX
Name: Drug, dtype: object
```

عکس ۱۴ درخت تصمیم : پیش‌بینی مدل با داده‌های تست

Task 5

Evaluation and Confusion matrix

```
from sklearn.metrics import f1_score, recall_score, precision_score, accuracy_score, confusion_matrix
print("DecisionTrees's Accuracy:", accuracy_score(y_testset, predTree))
print("DecisionTrees's F1 score:", f1_score(y_testset, predTree, average='weighted'))
print("DecisionTrees's Recall score: ", recall_score(y_testset, predTree, average='weighted'))
print("DecisionTrees's Precision score: ", precision_score(y_testset, predTree, average='weighted'))
```

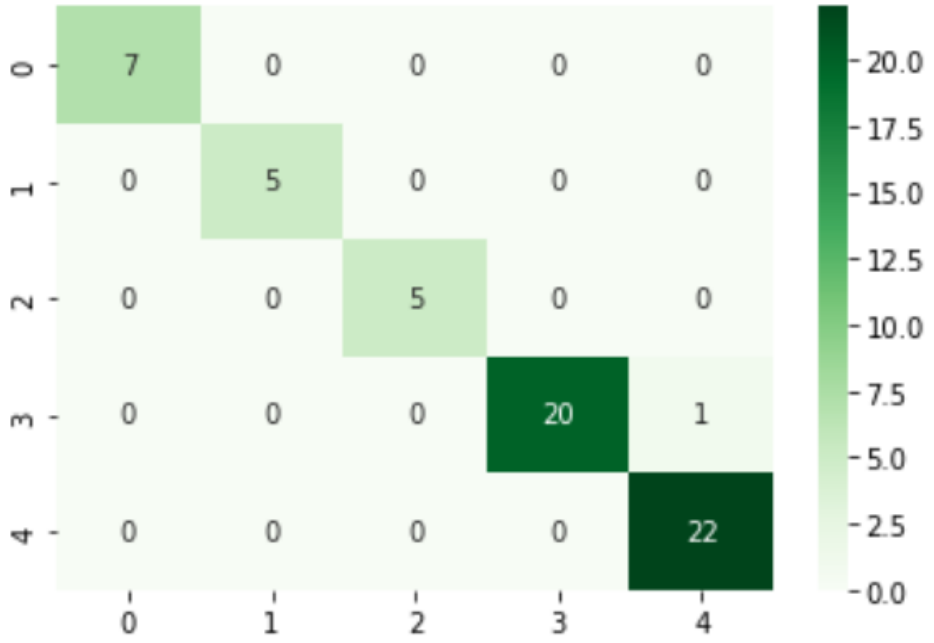
```
DecisionTrees's Accuracy: 0.9833333333333333
DecisionTrees's F1 score: 0.9833152664859981
DecisionTrees's Recall score:  0.9833333333333333
DecisionTrees's Precision score:  0.9840579710144927
```

عکس ۱۵ درخت تصمیم : ارزیابی مدل با معیارهای مختلف

Confusion matrix

```
con_matrix=confusion_matrix(y_testset, predTree)
sns.heatmap(con_matrix,annot=True, cmap="Greens")
```

```
<AxesSubplot:>
```



عکس ۱۶ درخت تصمیم : ارزیابی مدل با ماتریس درهم‌ریختگی

۲- الگوریتم K نزدیک‌ترین همسایه

الگوریتم K نزدیک‌ترین همسایه به‌عنوان یک مدل مبتنی بر نمونه (instance-based method) یا یک یادگیرنده‌ی تنبل (lazy learner) شناخته می‌شود؛ زیرا یک مدل داخلی ایجاد نمی‌کند و از داده‌های آموزش عملکرد متمایز را یاد نمی‌گیرد؛ فقط نمونه‌های آموزشی را حفظ می‌کند که به‌عنوان «دانش» برای مرحله‌ی پیش‌بینی استفاده می‌شود.

این الگوریتم برای مسائل طبقه‌بندی، k نزدیک ترین همسایه را پیدا کرده و با اکثریت آرا نزدیک‌ترین همسایگان کلاس را پیش‌بینی می‌کند.

برای مسائل رگرسیون، k نزدیک‌ترین همسایه را پیدا کرده و با محاسبه‌ی میانگین مقدار نزدیک‌ترین همسایه‌ها، مقدار مدنظر را پیش‌بینی می‌کند.

۲-۱- مراحل اصلی الگوریتم K نزدیک‌ترین همسایه

۱. داده‌ها را بارگذاری می‌کنیم.

۲. مقدار K را تعیین می‌کنیم که همان تعداد نزدیک‌ترین همسایه‌ها هستند.

۳. برای هر نمونه داده فاصله‌ی نمونه داده‌ی جدید را با نمونه داده‌های موجود محاسبه می‌کنیم.

٤. لیست را براساس فاصله‌ی نمونه داده‌ها، از کمترین به بیشترین فاصله، مرتب می‌کنیم.

٥. K تا از اولین نمونه‌های فهرست مرتب‌شده را به‌عنوان K نزدیک‌ترین همسایه انتخاب می‌کنیم.

٦. برچسب این K نمونه را بررسی می‌کنیم و بر اساس بیشترین برچسب موجود، مقدار و برچسب نمونه جدید بدست می‌آید.

۲-۲- معرفی دیتاست

دیتاست این الگوریتم، مجموعه داده‌های گل‌های ایریس می‌باشد. ویژگی‌های این مجموعه داده شامل طول کاسبرگ، عرض کاسبرگ، طول گلبرگ و عرض گلبرگ است و متغیر هدف نوع گونه‌ی گل ایریس می‌باشد.

ابتدا مدل درخت تصمیم را برا اساس بخشی از این دیتاست آموزش می‌دهیم و سپس از آن برای پیش‌بینی نوع یک گل ایریس ناشناخته استفاده می‌کنیم.

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	85	5.4	3.0	4.5	1.5	Iris-versicolor
1	123	7.7	2.8	6.7	2.0	Iris-virginica
2	29	5.2	3.4	1.4	0.2	Iris-setosa
3	25	4.8	3.4	1.9	0.2	Iris-setosa
4	76	6.6	3.0	4.4	1.4	Iris-versicolor

عکس ۱۷ بخشی از دیتاست گل‌های ایریس همراه با ویژگی‌های آن

۲-۳- کد پایتون و خروجی‌ها

HW4 DataMining

KNN

Tasks :

1- Downloading Iris Data Set

2- Obtaining information (#Rows, #Columns, Types of Columns, Missing values)

3- Pre-processing

4- KNN

5- Evaluation and Confusion matrix

عکس ۱۸ فهرست بخش پیاده‌سازی الگوریتم KNN

Task 1

Downloading Iris Train and Iris Test

Import libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn import preprocessing
```

عکس ۱۹ دانلود دیتاست گل‌های ایریس و ایمپورت کتابخانه‌های مورد نیاز

Reading Iris_train.csv and Iris_test.csv

```
df_train = pd.read_csv("Iris_train.csv")
df_test = pd.read_csv("Iris_test.csv")
```

```
df_train.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	85	5.4	3.0	4.5	1.5	Iris-versicolor
1	123	7.7	2.8	6.7	2.0	Iris-virginica
2	29	5.2	3.4	1.4	0.2	Iris-setosa
3	25	4.8	3.4	1.9	0.2	Iris-setosa
4	76	6.6	3.0	4.4	1.4	Iris-versicolor

عکس ۲۰ خواندن دیتاست Iris_train.csv و Iris_test.csv

مجموعه داده‌های تست و آموزش به صورت جداگانه در فایل های csv قرار گرفته‌اند و در نتیجه نیازی به جداسازی (split) مجدد داده‌ها نمی‌باشد.

Task 2

Obtaining information

```
df_train.shape
```

```
(120, 6)
```

```
df_test.shape
```

```
(30, 6)
```

عکس ۲۱ نمایش تعداد سطرها و ستون‌ها مجموعه داده‌های تست و آموزش

Find columns names and their types

```
df_train.dtypes
```

```
Id                int64
SepalLengthCm     float64
SepalWidthCm      float64
PetalLengthCm     float64
PetalWidthCm      float64
Species            object
dtype: object
```

عکس ۲۲ نوع‌های داده‌ای ستون‌ها

Check Missing values

```
df_train.isna().sum() , df_train.isna().sum()
```

```
(Id              0
 SepalLengthCm    0
 SepalWidthCm     0
 PetalLengthCm    0
 PetalWidthCm     0
 Species          0
 dtype: int64,
 Id              0
 SepalLengthCm    0
 SepalWidthCm     0
 PetalLengthCm    0
 PetalWidthCm     0
 Species          0
 dtype: int64)
```

عکس ۲۳ نمایش تعداد مقادیر از دسته رفته

در این دیتافریم مقادیر از دست رفته‌ای در ستون‌ها وجود ندارد و همچنین مقادیر ناسازگار نیز یافت نمی‌شود.

Task 3

Pre-processing

X as the Feature Matrix

y as the response vector (target)

```
X_train = df_train[['SepalLengthCm', 'SepalWidthCm','PetalLengthCm', 'PetalWidthCm']] .values
X_train[0:5]
```

```
array([[5.4, 3. , 4.5, 1.5],
       [7.7, 2.8, 6.7, 2. ],
       [5.2, 3.4, 1.4, 0.2],
       [4.8, 3.4, 1.9, 0.2],
       [6.6, 3. , 4.4, 1.4]])
```

عکس ۲۴ مرحله پیش‌پردازش : جداسازی متغیرهای مستقل از داده‌های آموزش

```
y_train = df_train['Species'].values
y_train[0:5]
```

```
array(['Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-versicolor'], dtype=object)
```

عکس ۲۵ مرحله پیش‌پردازش : جداسازی متغیر وابسته از داده‌های آموزش

```
X_test = df_test[['SepalLengthCm', 'SepalWidthCm','PetalLengthCm', 'PetalWidthCm']] .values
X_test[0:5]
```

```
array([[4.9, 3. , 1.4, 0.2],
       [5. , 3.4, 1.5, 0.2],
       [5.8, 4. , 1.2, 0.2],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.3, 1.7, 0.5]])
```

عکس ۲۶ مرحله پیش‌پردازش : جداسازی متغیرهای مستقل از داده‌های تست

```
y_test = df_test['Species'].values
y_test[0:5]
```

```
array(['Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa'], dtype=object)
```

عکس ۲۷ مرحله پیش‌پردازش : جداسازی متغیر وابسته از داده‌های تست

Normalize Data

```
X_train = preprocessing.StandardScaler().fit(X_train).transform(X_train.astype(float))
X_test = preprocessing.StandardScaler().fit(X_test).transform(X_test.astype(float))
X_train[0:5]
```

```
array([[-0.48444978, -0.11986112,  0.45640414,  0.44097878],
       [ 2.27811507, -0.5693403 ,  1.69857623,  1.09107034],
       [-0.72467281,  0.77909726, -1.29392926, -1.24925929],
       [-1.20511887,  0.77909726, -1.01161742, -1.24925929],
       [ 0.9568884 , -0.11986112,  0.39994177,  0.31096046]])
```

عکس ۲۸ نرمالایز کردن متغیرهای مستقل آموزش و تست

Task 4

KNN

Modeling

Classifier implementing the k-nearest neighbors vote

```
from sklearn.neighbors import KNeighborsClassifier
k = 5
#Train Model
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
neigh
```

```
KNeighborsClassifier()
```

عکس ۲۹ KNN : ایجاد و آموزش مدل با k=5

Prediction

```
y_pred = neigh.predict(X_test)
y_pred[0:5]
```

```
array(['Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa'], dtype=object)
```

عکس ۳۰ KNN : پیش‌بینی مدل با داده‌های تست

Task 5

Evaluation and Confusion matrix

```
from sklearn.metrics import f1_score, recall_score, precision_score, accuracy_score, confusion_matrix
print("KNN's Accuracy:", accuracy_score(y_test, y_pred))
print("KNN's F1 score:", f1_score(y_test, y_pred, average='weighted'))
print("KNN's Recall score: ", recall_score(y_test, y_pred, average='weighted'))
print("KNN's Precision score: ", precision_score(y_test, y_pred, average='weighted'))
```

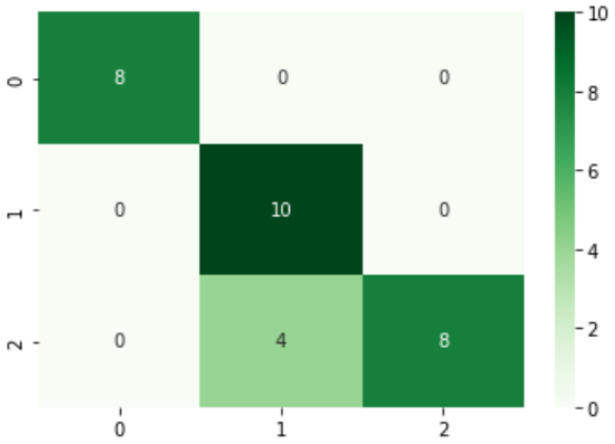
```
KNN's Accuracy: 0.8666666666666667
KNN's F1 score: 0.8644444444444445
KNN's Recall score:  0.8666666666666667
KNN's Precision score:  0.9047619047619048
```

عکس ۳۱ KNN : ارزیابی مدل با معیارهای مختلف

Confusion matrix

```
con_matrix=confusion_matrix(y_test, y_pred)
sns.heatmap(con_matrix,annot=True, cmap="Greens")
```

```
<AxesSubplot:>
```



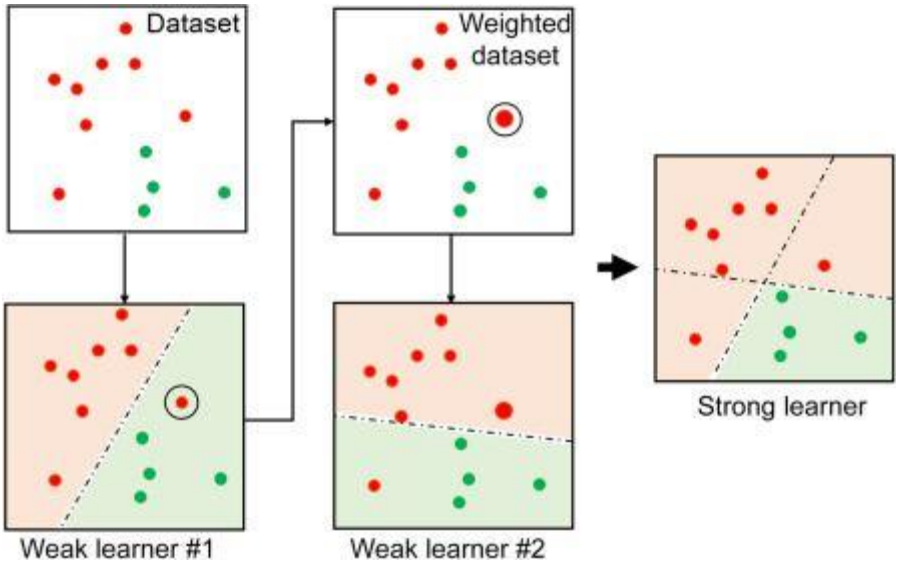
عکس ۳۲ KNN : ارزیابی مدل با ماتریس درهم‌ریختگی

۳- الگوریتم یادگیری ترکیبی (مدل آدابوست)

مدل‌های یادگیری ماشین ترکیبی یا مدل‌های جمعی یکی از روش‌های یادگیری ماشین است که در آن چندین مدل که یادگیری‌های ضعیف (weak learner) یا مدل های پایه (base models) نامیده می‌شوند برای حل یک مسئله آموزش داده می‌شوند و برای داشتن نتایج بهتر با هم ترکیب می‌شوند. زمانی که مدل‌های ضعیف به طور صحیح با یکدیگر ترکیب شوند می‌توانند مدل‌های دقیق‌تر و یا پایدارتری به وجود آورند.

الگوریتم AdaBoost تعداد n درخت تصمیم را در زمان آموزش داده ایجاد می‌کند. هنگامی که اولین درخت تصمیم ایجاد شد، نمونه‌هایی که در اولین مدل، نادرست طبقه بندی شده اند، اولویت بیشتری به خود اختصاص می‌دهند و تنها این نمونه‌ها به دومین مدل فرستاده می شوند. روش استفاده شده در ایجاد مدل های پایه، تخصیص وزن های بالاتر به نمونه هایی است که با دقت کمتری تخمین زده شده اند.

۳-۱- مراحل الگوریتم آدابوست



عکس ۳۳ الگوریتم Adaboost

۳-۲- معرفی دیتاست

دیتاست این الگوریتم، مجموعه داده‌های گل‌های ایریس می‌باشد. ویژگی‌های این مجموعه داده شامل طول کاسبرگ، عرض کاسبرگ، طول گلبرگ و عرض گلبرگ است و متغیر هدف نوع گونه‌ی گل ایریس می‌باشد.

ابتدا مدل آدابوست را برا اساس بخشی از این دیتاست آموزش می‌دهیم و سپس از آن برای پیش‌بینی نوع یک گل ایریس ناشناخته استفاده می‌کنیم.

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	85	5.4	3.0	4.5	1.5	Iris-versicolor
1	123	7.7	2.8	6.7	2.0	Iris-virginica
2	29	5.2	3.4	1.4	0.2	Iris-setosa
3	25	4.8	3.4	1.9	0.2	Iris-setosa
4	76	6.6	3.0	4.4	1.4	Iris-versicolor

عکس ۳۴ بخشی از دیتاست گل‌های ایریس همراه با ویژگی‌های آن

۳-۳- کد پایتون و خروجی‌ها

HW4 DataMining

AdaBoost

Tasks :

1- Downloading Iris Data Set

2- Obtaining information (#Rows, #Columns, Types of Columns, Missing values)

3- Pre-processing

4- AdaBoost

5- Evaluation and Confusion matrix

عکس ۳۵ فهرست بخش پیاده‌سازی الگوریتم آدابوست

Task 1

Downloading Iris Train and Iris Test

Import libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn import preprocessing
```

عکس ۳۶ دانلود دیتاست گل‌های ایریس و ایمپورت کتابخانه‌های مورد نیاز

Reading Iris_train.csv and Iris_test.csv

```
df_train = pd.read_csv("Iris_train.csv")
df_test = pd.read_csv("Iris_test.csv")
```

```
df_train.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	85	5.4	3.0	4.5	1.5	Iris-versicolor
1	123	7.7	2.8	6.7	2.0	Iris-virginica
2	29	5.2	3.4	1.4	0.2	Iris-setosa
3	25	4.8	3.4	1.9	0.2	Iris-setosa
4	76	6.6	3.0	4.4	1.4	Iris-versicolor

عکس ۳۷ خواندن دیتاست Iris_train.csv و Iris_test.csv

مجموعه داده‌های تست و آموزش به صورت جداگانه در فایل های csv قرار گرفته‌اند و در نتیجه نیازی به جداسازی (split) مجدد داده‌ها نمی‌باشد.

Task 2

Obtaining information

```
df_train.shape
```

```
(120, 6)
```

```
df_test.shape
```

```
(30, 6)
```

عکس ۳۸ نمایش تعداد سطرها و ستون‌ها مجموعه داده‌های تست و آموزش

Find columns names and their types

```
df_train.dtypes
```

```
Id                int64
SepalLengthCm     float64
SepalWidthCm      float64
PetalLengthCm     float64
PetalWidthCm      float64
Species           object
dtype: object
```

عکس ۳۹ نوع‌های داده‌ای ستون‌ها

Check Missing values

```
df_train.isna().sum() , df_train.isna().sum()
```

```
(Id              0
 SepalLengthCm    0
 SepalWidthCm     0
 PetalLengthCm    0
 PetalWidthCm     0
 Species          0
 dtype: int64,
 Id              0
 SepalLengthCm    0
 SepalWidthCm     0
 PetalLengthCm    0
 PetalWidthCm     0
 Species          0
 dtype: int64)
```

عکس ۴۰ نمایش تعداد مقادیر از دسته رفته

در این دیتافریم مقادیر از دست رفته‌ای در ستون‌ها وجود ندارد و همچنین مقادیر ناسازگار نیز یافت نمی‌شود.

Task 3

Pre-processing

X as the Feature Matrix

y as the response vector (target)

```
X_train = df_train[['SepalLengthCm', 'SepalWidthCm','PetalLengthCm', 'PetalWidthCm']] .values
X_train[0:5]
```

```
array([[5.4, 3. , 4.5, 1.5],
       [7.7, 2.8, 6.7, 2. ],
       [5.2, 3.4, 1.4, 0.2],
       [4.8, 3.4, 1.9, 0.2],
       [6.6, 3. , 4.4, 1.4]])
```

عکس ۴۱ مرحله پیش‌پردازش : جداسازی متغیرهای مستقل از داده‌های آموزش

```
y_train = df_train['Species'].values
y_train[0:5]
```

```
array(['Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-versicolor'], dtype=object)
```

عکس ۴۲ مرحله پیش‌پردازش : جداسازی متغیر وابسته از داده‌های آموزش

```
X_test = df_test[['SepalLengthCm', 'SepalWidthCm','PetalLengthCm', 'PetalWidthCm']] .values
X_test[0:5]
```

```
array([[4.9, 3. , 1.4, 0.2],
       [5. , 3.4, 1.5, 0.2],
       [5.8, 4. , 1.2, 0.2],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.3, 1.7, 0.5]])
```

عکس ۴۳ مرحله پیش‌پردازش : جداسازی متغیرهای مستقل از داده‌های تست

```
y_test = df_test['Species'].values
y_test[0:5]
```

```
array(['Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa'], dtype=object)
```

عکس ۴۴ مرحله پیش‌پردازش : جداسازی متغیر وابسته از داده‌های تست

Normalize Data

```
X_train = preprocessing.StandardScaler().fit(X_train).transform(X_train.astype(float))
X_test = preprocessing.StandardScaler().fit(X_test).transform(X_test.astype(float))
X_train[0:5]
```

```
array([[-0.48444978, -0.11986112,  0.45640414,  0.44097878],
       [ 2.27811507, -0.5693403 ,  1.69857623,  1.09107034],
       [-0.72467281,  0.77909726, -1.29392926, -1.24925929],
       [-1.20511887,  0.77909726, -1.01161742, -1.24925929],
       [ 0.9568884 , -0.11986112,  0.39994177,  0.31096046]])
```

عکس ۴۵ نرمالایز کردن متغیرهای مستقل آموزش و تست

Task 4

AdaBoost

Modeling

```
from sklearn.ensemble import AdaBoostClassifier
ada = AdaBoostClassifier(n_estimators=64, random_state=32)
ada.fit(X_train, y_train)
```

```
AdaBoostClassifier(n_estimators=64, random_state=32)
```

عکس ۴۶ آدابوست : ایجاد و آموزش مدل

تنظیم Hyper-parameter :

base_estimator: مدل مجموعه، پیش‌فرض درخت تصمیم است.

n_estimators: تعداد مدل‌هایی که باید ساخته شوند.

Learning_rate: سهم هر طبقه‌بندی‌کننده را با این مقدار کاهش می‌دهد.

random_state: دانه (seed) اعداد تصادفی، به طوری که هر بار همان اعداد تصادفی تولید می‌شود.

Prediction

```
y_pred = ada.predict(X_test)
y_pred[0:5]
```

```
array(['Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa'], dtype=object)
```

عکس ۴۷ آدابوست : پیش‌بینی مدل با داده‌های تست

Task 5

Evaluation and Confusion matrix

```
from sklearn.metrics import f1_score, recall_score, precision_score, accuracy_score, confusion_matrix
print("KNN's Accuracy:", accuracy_score(y_test, y_pred))
print("KNN's F1 score:", f1_score(y_test, y_pred, average='weighted'))
print("KNN's Recall score: ", recall_score(y_test, y_pred, average='weighted'))
print("KNN's Precision score: ", precision_score(y_test, y_pred, average='weighted'))
```

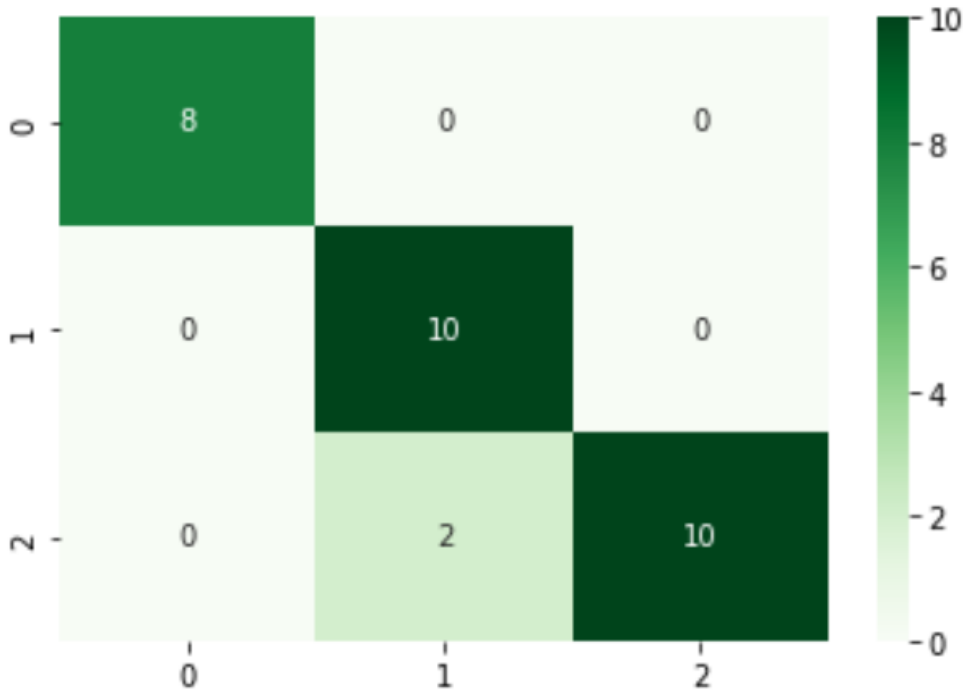
```
KNN's Accuracy: 0.9333333333333333
KNN's F1 score: 0.9333333333333333
KNN's Recall score:  0.9333333333333333
KNN's Precision score:  0.9444444444444445
```

عکس ۴۸ آدابوست: ارزیابی مدل با معیارهای مختلف

Confusion matrix

```
con_matrix=confusion_matrix(y_test, y_pred)
sns.heatmap(con_matrix,annot=True, cmap="Greens")
```

```
<AxesSubplot:>
```



عکس ۴۹ آدابوست: ارزیابی مدل با ماتریس درهم‌ریختگی