به نام خداوند جان و خرد

دانشگاه تهران

دانشکدگان فنی

دانشکده مهندسی برق و کامپیوتر

یادگیری ماشین

تمرین شماره ۳

نام و نام خانوادگی: علی خرم فر

شماره دانشجویی: ۸۱۰۱۰۲۱۲۹

اردیبهشت‌ماه ۱۴۰۳

فهرست مطالب

۱_ پاسخ سوال ۱

پاسخ به صورت کامل در فایل زیپ پیوست شده است.

۲_پاسخ سوال ۲

پاسخ به صورت کامل در فایل زیپ پیوست شده است.

۳_ پاسخ سوال ۳

ابتدا کتابخانه‌ها و پکیج‌های موردنیاز را ایمپورت کرده و سپس فایل دیتاست را وارد می‌کنیم.

Preparing Dataset

```
data = pd.read_csv('/content/gdrive/MyDrive/Colab Notebooks/ML HW3/credit_scoring_sample.csv', sep=";")
data.head()
```

	SeriousDlqin2yrs	age	NumberOfTime30-59DaysPastDueNotWorse	DebtRatio	NumberOfTimes90DaysLate	NumberOfTime60-89DaysPastDueNotWorse	MonthlyIncome	NumberOfDependents
0	0	64	0	0.249908	0	0	8158.0	0.0
1	0	58	0	3870.000000	0	0	NaN	0.0
2	0	41	0	0.456127	0	0	6666.0	0.0
3	0	43	0	0.000190	0	0	10500.0	2.0
4	1	49	0	0.271820	0	0	400.0	0.0

همانطور که در تصویر بالا مشخص است برای خواندن این دیتاست لازم است که مقدار جداکننده را ; قرار دهیم تا بتوان دیتافریم را به شکل نرمال در کتابخانه pandas بارگذاری کرد. در این سوال ستون‌های زیر ارائه شده اند:

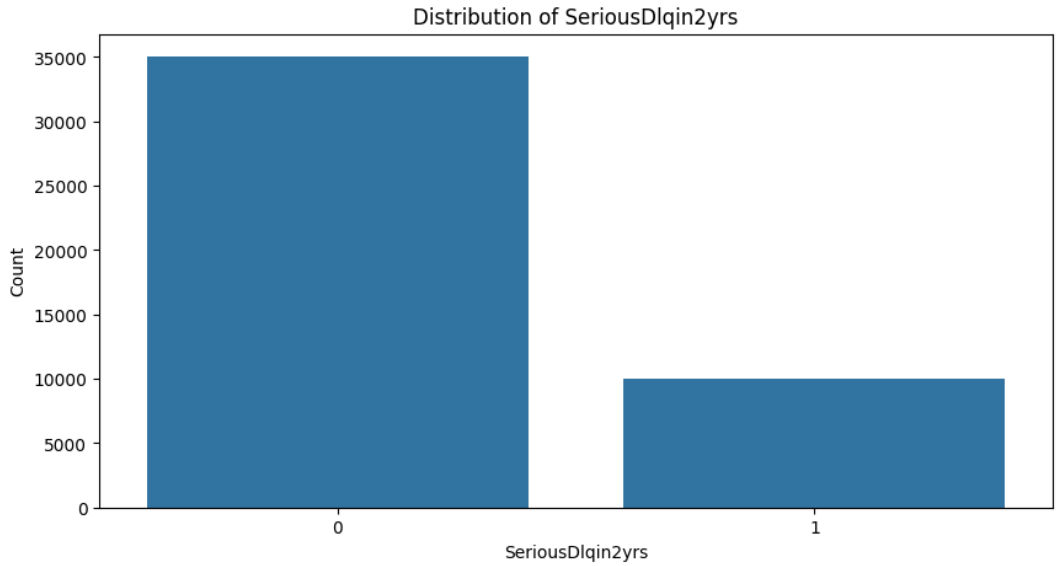
feature	Description
Age	سن مشتری
DebtRatio	مجموع پرداختی وام/درصد درآمد ماهانه کل
NumberOfTime30-59DaysPastDueNotWorse	تعداد مواردی که باز پرداخت مشتری در طول ۲ سال گذشته ۳۰ تا ۵۹ روز (نه بیشتر) عقب افتاده است.
NumberOfTimes90DaysLate	تعداد مواردی که باز پرداخت مشتری ۹۰ روز یا بیشتر عقب افتاده است.
NumberOfTime60-89DaysPastDueNotWorse	تعداد مواردی که باز پرداخت مشتری در طول ۲ سال گذشته ۶۰ تا ۸۹ روز (نه بیشتر) عقب افتاده است.
NumberOfDependents	تعداد افراد وابسته به مشتری
SeriousDlqin2yrs	مشتری بدهی را ظرف مدت ۹۰ روز پرداخت نکرده است.

مقدار target نیز برابر با SeriousDlqin2yrs است.

هدف از این سوال این است که بتوانیم با کمک ویژگی‌های ارائه شده در ستون‌های مختلف بتوانیم پیش‌بینی کنیم که مقدار SeriousDlqin2yrs برابر صفر است یا یک.

۱-۳_ پیش‌پردازش داده‌ها

ابتدا توزیع ستون هدف را رسم کرده و آن را بررسی می‌کنیم:



با توجه به نمودار توزیع ستون هدف (SeriousDlqin2yrs)، می‌توان مشاهده کرد که تعداد نمونه‌های مقدار ۰ بیشتر از نمونه‌های مقدار ۱ است. این عدم توازن در داده‌ها ممکن است بر عملکرد مدل تاثیر بگذارد.

برای جایگزین کردن مقادیر NULL با میانه نیز ابتدا تعداد این نمونه‌ها را برای هر ستون خروجی گرفتیم:

Replace Nan with Median

```
data.isna().sum()
```

```
SeriousDlqin2yrs                        0
age                                     0
NumberOfTime30-59DaysPastDueNotWorse    0
DebtRatio                               0
NumberOfTimes90DaysLate                 0
NumberOfTime60-89DaysPastDueNotWorse    0
MonthlyIncome                        8643
NumberOfDependents                   1117
dtype: int64
```

همانطور که مشخص است تعداد زیادی از داده‌ها مثلا ستون درآمد NULL دارد. بهترین کار برای این جایگزینی استفاده از میانه است. زیرا که داده‌های Outlier در این ویژگی تاثیر زیادی در میانگین خواهندداشت.

پس از جایگزینی مجددا خروجی گرفته شد که مشخص شد این بخش به درستی انجام شده است:

```
nan_columns = data.columns[data.isna().any()].tolist()
for column in nan_columns:
    median = data[column].median()
    data[column].fillna(median, inplace=True)
```

```
data.isna().sum()
```

```
SeriousDlqin2yrs                      0
age                                   0
NumberOfTime30-59DaysPastDueNotWorse    0
DebtRatio                             0
NumberOfTimes90DaysLate               0
NumberOfTime60-89DaysPastDueNotWorse    0
MonthlyIncome                         0
NumberOfDependents                    0
dtype: int64
```

Bootstrapping _۳-۲

در این روش، نمونه‌های تصادفی با جایگزینی از داده‌های موجود گرفته می‌شوند تا نمونه‌های جدیدی تلولید شوند. در این بخش، بازه اطمینان ۹۰ درصد برای میانگین سنی افراد بد حساب محاسبه شد. برای اینکار ابتدا افرادی که بدحساب هستند و ستون هدف در آن‌ها ۱ است را جدا کردیم و سپس ستون سن را از آن استخراج کردیم.

بازه اطمینان هم به این معنی است که اگر ما بگوییم بازه اطمینان ۹۰ درصد برای یک میانگین برابر (A, B) است، به این معنی است که ما ۹۰ درصد اطمینان داریم که میانگین واقعی جامعه در این بازه قرار دارد. نتیجه به دست آمده به شرح زیر است:

90% Confidence Interval for age of bad customers = (45.73058547775783 , 46.126835228406144)

این خروجی نشان می‌دهد که با ۹۰ درصد اطمینان می‌توان گفت میانگین سن مشتریان بدحساب در بازه ۴۵.۷۳ تا ۴۶.۱۲ قرار دارد.

Random Forest _۳-۳

برای پیش‌بینی بازپرداخت بدهی مشتریان، از مدل Random Forest استفاده شد. این مدل از چندین درخت تصمیم تشکیل شده که به صورت تصادفی ساخته می‌شوند و نتایج آن‌ها با هم ترکیب شده تا یک تصمیم نهایی گرفته شود.

ابتدا داده‌های ویژگی و ستون هدف را جداسازی کرده و پارامترهای ارائه شده را آماده می‌کنیم:

```
X = data.drop(columns=['SeriousDlqin2yrs'])
y = data['SeriousDlqin2yrs']

parameter = {
    'max_features': [1, 2, 4],
    'min_samples_leaf': [3, 5, 7, 9],
    'max_depth': [5, 10, 15],
}
```

max_features: تعداد ویژگی انتخاب شده.

min_samples_leaf: حداقل تعداد نمونه در برگ.

max_depth: حداکثر عمق درخت.

مدل random forest را تعریف کرده و برای اینکه بر عدم توازن داده‌ها غلبه کنیم، مقدار 'class_weight='balanced قرار داده می‌شود.

برای یافتن بهترین ترکیب پارامترها از روش Grid Search با استفاده از ۵--fold Stratified Cross Validation و معیار ROC AUC به عنوان امتیازدهی استفاده کرده و در نهایت مدل با کمک داده‌های ارائه شده آموزش داده می‌شود.

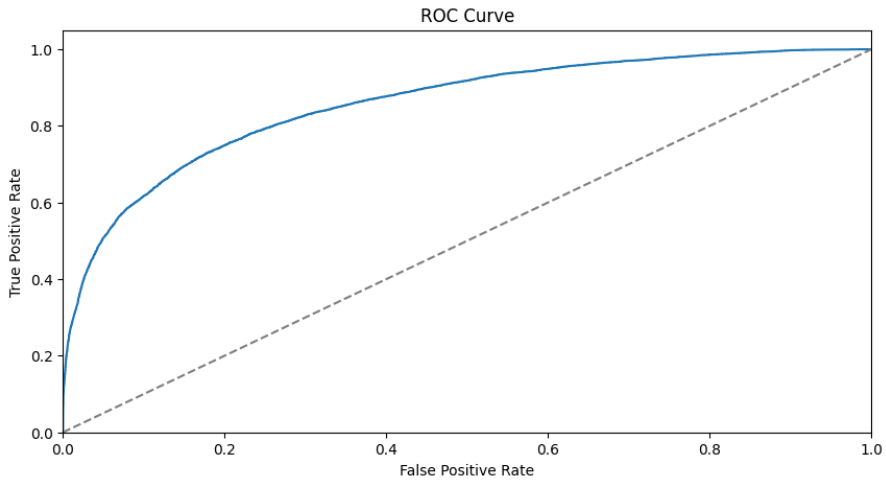
بهترین پارامترهایی که توسط Grid Search خروجی گرفته شد:

{'max_depth': 10, 'max_features': 1, 'min_samples_leaf': 9}

خروجی ROC AUC :

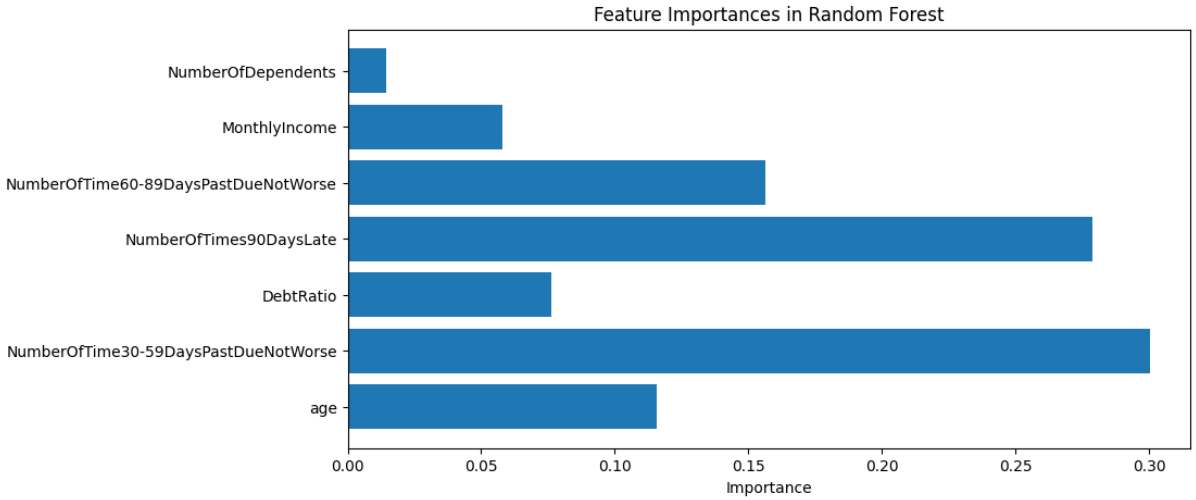
ROC AUC: 0.8354

نتایج نشان می‌دهد که مدل Random Forest با پارامترهای بهینه شده، دقت قابل قبولی در پیش‌بینی بازپرداخت بدهی مشتریان دارد. امتیاز ROC AUC ، نشان‌دهنده توانایی مدل در جدا کردن خوش‌حساب‌ها از بدحساب‌ها با دقت نسبتا خوبی است.



۳-۴_ ضعیف‌ترین ویژگی

در این بخش، اهمیت ویژگی‌های مختلف در مدل Random Forest بررسی شد. نمودار زیر اهمیت هر ویژگی را در مدل نشان می‌دهد:



از نمودار بالا، می‌توان نتیجه گرفت که تاخیرهای مشتری در بازپرداخت بدهی‌ها تاثیر بسیار زیادی بر پیش‌بینی خوش‌حساب یا بد حساب بودن مشتری دارند. تعداد دفعات تاخیر بین ۳۰ تا ۵۹ روز بیشترین اهمیت را در مدل دارد.

ضعیف ترین ویژگی هم NumberOfDependents است که همان تعداد وابستگان فرد است و مشاهده می‌کنیم که تعداد افراد وابسته به مشتری کمترین تاثیر را در مدل داشته است. این موضوع نشان می‌دهد که وجود یا عدم وجود افراد وابسته تاثیر چندانی در بدحساب بودن فرد ندارد.

۳-۵_ Bagging

در این قسمت از پرسش از مدل Bagging با استفاده از Logistic Regression به عنوان طبقه‌بند پایه استفاده شد. در این روش با ایجاد چندین نمونه Bootstrap از داده‌های آموزشی و آموزش Classifier های جداگانه بر روی هر نمونه، نتایج را ترکیب کرده تا پیش‌بینی‌های نهایی را به دست آوریم.

پارامترهای تنظیم شده برای Bagging :

```
    'max_features': [2, 3, 4],
    'max_samples': [0.5, 0.7, 0.9],
    'base_estimator__C': [0.0001, 0.001, 0.01, 1, 10, 100]
```

max_features: تعداد ویژگی‌های انتخاب شده در هر نمونه.

max_samples: درصد نمونه‌های انتخاب شده برای هر Bootstrap.

base_estimator__C: پارامتر مدل Logistic Regression.

در این بخش از سوال برای یافتن بهترین پارامتر ها از روش Randomized Search استفاده شد.

بهترین پارامترها:

max_samples: 0.9

max_features: 2

base_estimator__C: 1

خروجی ROC AUC :

ROC AUC: 0.8098

در مقایسه با مدل Random Forest که امتیاز ROC AUC آن ۰.۸۳۵۵ بود، مدل Bagging عملکرد پایین‌تری دارد. با این حال، هر دو مدل توانسته‌اند دقت قابل قبولی داشته باشند.

تحلیل پارامترهای بهینه

max_samples: 0.9

مقدار بالا نشان می‌دهد که در هر نمونه برداری با جایگزینی، ۹۰ درصد از داده‌های آموزشی برای ایجاد هر مدل پایه Logistic Regression انتخاب می‌شوند. استفاده از ۹۰ داده‌ها در هر نمونه‌گیری باعث می‌شود که هر مدل پایه مجموعه مشابهی از نمونه‌ها را ببینند. انتخاب این مقدار به مدل اجازه می‌دهد که از تنوع کافی در داده‌ها برای آموزش برخوردار باشد، بدون اینکه تمام داده‌ها در هر بار نمونه‌گیری استفاده شوند. این می‌تواند به بهبود دقت مدل و کاهش واریانس کمک کند.

max_features: 2

باتوجه به مقدار این پارامتر در هر بار، تنها دو ویژگی به صورت تصادفی انتخاب می‌شوند. انتخاب تعداد کمی از ویژگی‌ها باعث کاهش همبستگی بین مدل‌های پایه شده و عملکرد مدل کلی بهتر می‌شود. زیرا مدل‌های پایه خطاهای مختلفی خواهند داشت که میانگین آن‌ها باعث بهبود عملکرد مدل نهایی می‌شود. در مدل رگرسیون لجستیک هرچه داده‌ها همبستگی بیشتری داشته باشند، کیفیت پایین‌تر خواهدبود، به همین دلیل فقط ۲ ویژگی انتخاب می‌شود و در نهایت مدل عمکلرد بهتری خواهد داشت.

base_estimator__C: 1

این پارامتر به عنوان Regularization مدل‌های پایه Logistic Regression تنظیم می‌شود. هرچه که مقدار جریمه Regularization بیشتر باشد، وزن‌های با مقدار بزرگ جریمه بیشتری می‌شوند. این مورد در رگرسیون لجستیک برعکس است. مقدار برابر با ۱ به این معناست که مدل Logistic Regression با یک پارامتر منظم‌سازی معقولی شده است.این امر برای مقابله با عدم توازن داده‌ها مفید است و به مدل کمک

می‌کند تا روابط پیچیده‌تری را بین ویژگی‌ها و هدف یاد بگیرد بدون اینکه دچار بیش‌برازش شود. مقدار زیاد باعث می‌شود Regularization بیش از حد بوده و مدل روی داده‌ها به خوبی fit نشود. مقدار کم هم باعث بیش‌برازش می‌شود. پس مقدار انتخابی مقدار خوب و معقولی است.

۴_پاسخ سوال ۴

باتوجه به صورت مسئله هدف از این قسمت پیاده‌سازی یک طبقه‌بند AdaBoost برای مجموعه داده ارائه شده است. برای طبقه‌بند پایه هم یک درخت تصمیم با عمق ۱ استفاده می‌کنیم.

۴-۱_ کلاس تکمیل شده

در بخش اول این سوال کد اولیه کلاس SimpleMultiClassBoosting را توسعه دادیم تا به یک نسخه کامل‌تر و قابل اجرا برسیم. در کد اولیه، ساختار کلی کلاس و توابع اصلی آن تعریف شده بود، اما جزییات عملکرد و منطق اصلی الگوریتم پیاده‌سازی نشده بود. تغییرات و اضافات زیر به کد اصلی اضافه شدند تا کلاس به‌طور کامل عمل کند. در این کلاس از LabelEncoder برای تبدیل برچسب‌های چندکلاسه به مقادیر صحیح استفاده شد تا بتوانیم آن‌ها را در محاسبات و وزن‌دهی استفاده کنیم. همچنین وزن‌های نمونه‌ها به‌طور یکنواخت و مساوی تنظیم شد تا همه نمونه‌ها در ابتدا تاثیر یکسانی داشته باشند.

در هر تکرار، یک کپی از مدل پایه ایجاد و با استفاده از وزن‌های فعلی آموزش داده شد پس از آن پیش‌بینی ها و خطا محاسبه شد. همچین یک شرط برای توقف اضافه شد که اگر خطای یادگیرنده بدتر از حدس تصادفی باشد، الگوریتم متوقف شود.

در نهایت پیش‌بینی تمامی مدل‌ها جمع‌آوری و یک رای وزن‌دار از هر مدل برای هر نمونه محاسبه شد. پس از آن کلاس نهایی با بیشترین رای به‌عنوان پیش‌بینی نهایی انتخاب شد. در آخر هم پیش‌بینی‌های نهایی به برچسب‌های اصلی تبدیل شدند تا خروجی کلاس قابل تفسیر باشد.

۴-۲_ حل مسئله طبقه‌بندی

برای حل این مسئله طبقه‌بندی ابتدا داده‌ها را بارگذاری کرده و آنها را به دو بخش آموزش و آزمون تقسیم می‌کنیم.

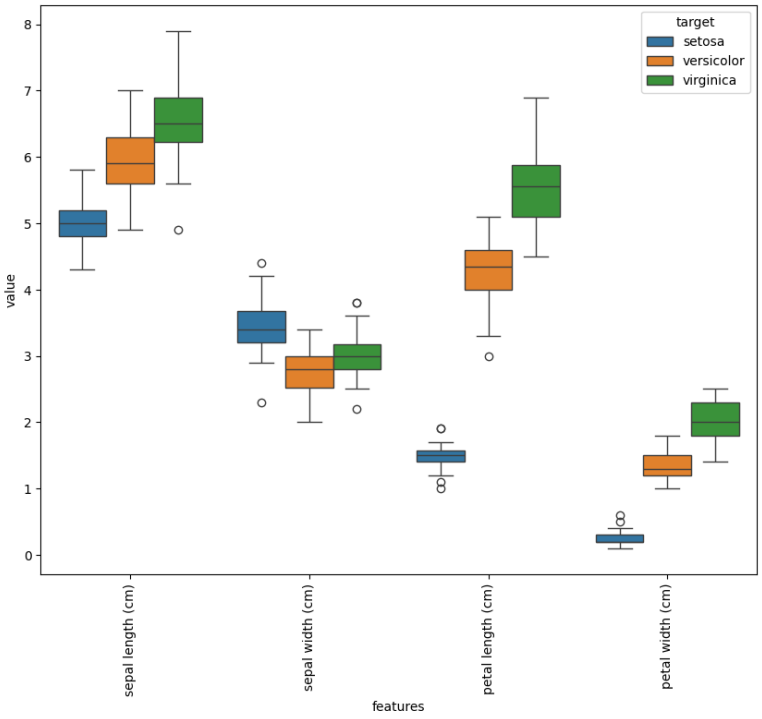
Load Dataset

```
data = load_iris()
X = data.data
y = data.target

# 70-30 split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

مجموعه داده‌ی iris یکی از معروف ترین دیتاست‌ها در یادگیری ماشین است. این دیتاست شامل اطلاعاتی درباره‌ی سه گونه‌ی مختلف گل iris است که شامل اندازه‌های کاسبرگ و گلبرگ مختلف می‌شود.

برای اینکه درک بهتری از این ۳ دسته گل داشته باشیم، نمودار boxplot آن را رسم می‌کنیم:



باتوجه به نمودار بالا می‌توان متوجه شد که برخی ویژگی‌ها به خوبی می‌توانند گونه‌های مختلف را از هم تفکیک کنند. که در ادامه یک طبقه‌بند بر روی این ویژگی‌ها پیاده‌سازی می‌کنیم.

برای این کار یک طبقه‌بند AdaBoost با درخت تصمیم به عنوان طبقه‌بند پایه ایجاد می‌کنیم. تعداد طبقه‌بندهای پایه نیز ۵۰ در نظر می‌گیریم. سپس مدل را آموزش می‌دهیم.

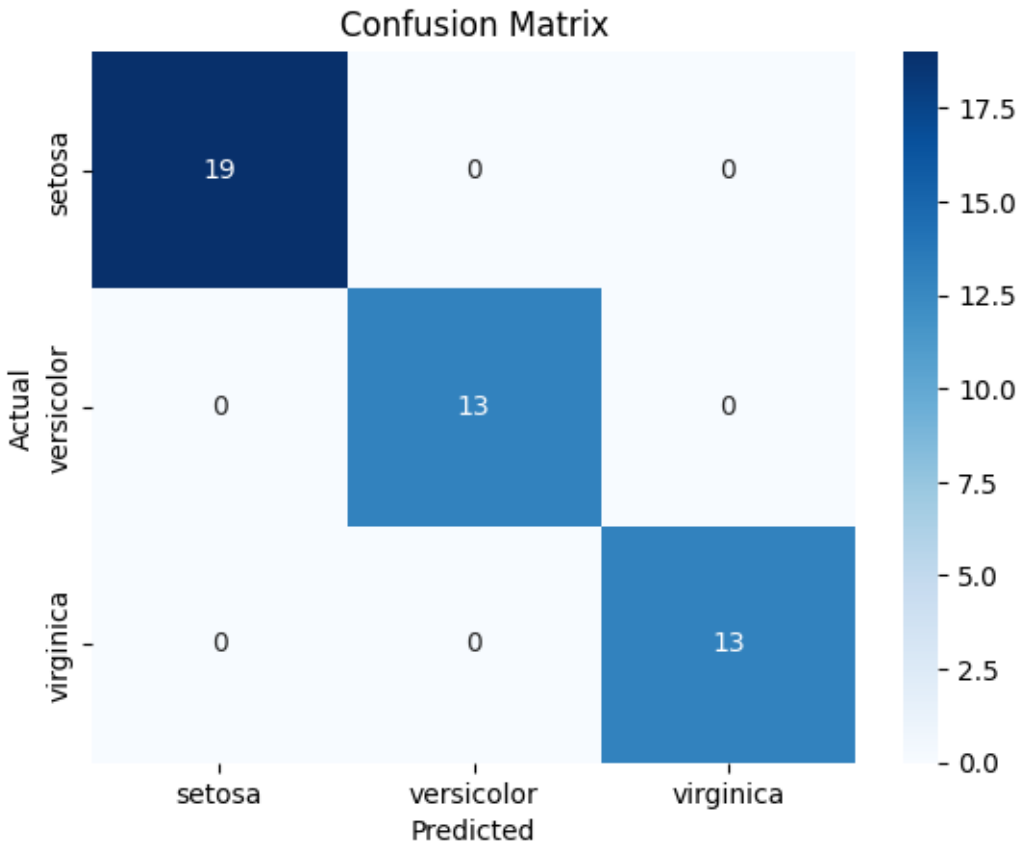
Train Model

```
base_classifier = DecisionTreeClassifier(max_depth=1)
boosting_model = SimpleMultiClassBoosting(base_estimator = base_classifier , n_estimators=50)
boosting_model.fit(X_train, y_train)
```

خروجی مدل بر روی داده‌های آزمون به صورت زیر است :

```
Accuracy: 1.0
Confusion Matrix:
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
```

دقت مدل بر روی داده‌های آزمون برابر با ۱ است. این مقدار نشان می‌دهد که مدل توانسته تمامی نمونه‌های داده‌ی آزمون را به درستی طبقه‌بندی کرده و هیچ خطایی در پیش‌بینی‌ها نداشته باشد. ماتریس آشفتگی را به صورت heatmap نیز در شکل زیر مشاهده می‌کنیم:



همانطور که مشاهده می‌شود تمامی داده‌ها به صورت درست پیش‌بینی شده اند.

۵_پاسخ سوال ۵

پاسخ به صورت کامل در فایل زیپ پیوست شده است.

۶_پاسخ سوال ۶

در این سوال قصد داریم الگوریتم ID3 را بدون استفاده از کتابخانه پیاده‌سازی کرده و از آن برای طبقه‌بندی دیتاست مربوط به زندانیان استفاده کنیم. ابتدا دیتاست را بارگذاری کرده و چند سطر اول آن را خروجی می‌گیریم :

	Fiscal Year Released	Recidivism Reporting Year	Race - Ethnicity	Age At Release	Convicting Offense Classification	Convicting Offense Type	Convicting Offense Subtype	Main Supervising District	Release Type	Part of Target Population	Recidivism - Return to Prison numeric
0	2010	2013	White	<45	D Felony	Violent	Other	3JD	Parole	Yes	1
1	2010	2013	White	>45	D Felony	Other	Other	3JD	Parole	Yes	1
2	2010	2013	White	<45	D Felony	Other	Other	5JD	Parole	Yes	1
3	2010	2013	White	>45	Other Felony	Drug	Trafficking	3JD	Parole	Yes	1
4	2010	2013	Black	<45	D Felony	Drug	Trafficking	3JD	Parole	Yes	1

ستون هدف در این دیتاست Recidivism - Return to Prison numeric است که همان بازگشت به زندان در سوال مطرح شده است .

۱-۶_ بررسی داده‌ها

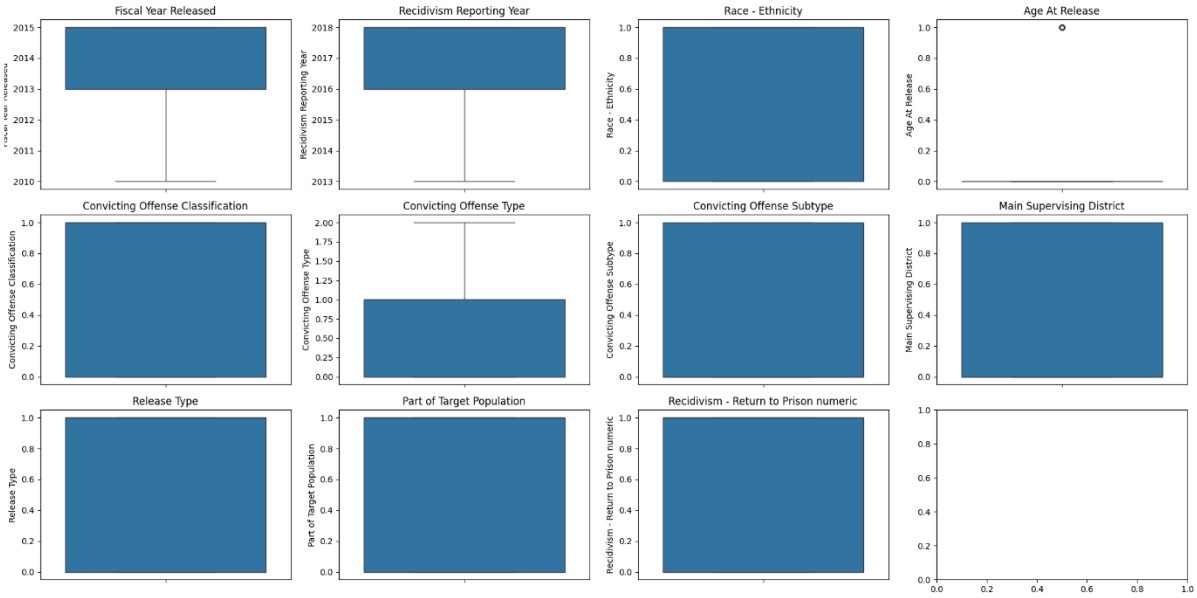
همانطور که مشاهده می‌شود بیشتر داده‌های این دیتاست Categorical هستند. برای اینکه درک بهتری از این داده‌ها داشته باشیم ابتدا مشخصات ستون‌ها را خروجی میگیریم:

```
print(dataset.columns)
```

```
Index(['Fiscal Year Released', 'Recidivism Reporting Year', 'Race - Ethnicity',
       'Age At Release', 'Convicting Offense Classification',
       'Convicting Offense Type', 'Convicting Offense Subtype',
       'Main Supervising District', 'Release Type',
       'Part of Target Population', 'Recidivism - Return to Prison numeric'],
      dtype='object')
```

این فایل شامل ویژگی‌های مختلفی از جمله سال آزاد شدن، سال گزارش بازگشت، نژاد، سن در زمان آزاد شدن، نوع جرم و ... می‌باشد.

برای درک بهتر پراکندگی داده‌ها نمودار boxplot آن‌ها را رسم می‌کنیم:



پیش از اجرای الگوریتم درخت تصمیم ID3 برای طبقه‌بندی داده‌ها، نیاز به پیش‌پردازش داده‌ها داریم.

نیاز به پیش‌پردازش خاصی نبود و پس از مشاهده تفاوت‌ها در نتیجه، تصمیم گرفته شد که پیش‌پردازش خاصی انجام نشود.

۶-۲_ پیاده‌سازی مدل

ابتدا توابع لازم برای اجرای الگوریتم id3 را پیاده‌سازی می‌کنیم.

فرمول محاسبه آنتروپی به صورت زیر است:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

همچنین فرمول Information gain به صورت زیر است :

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

هردو فرمول بالا به صورت زیر پیاده‌سازی شدند:

```
def entropy(y):
    unique, counts = np.unique(y, return_counts=True)
    probabilities = counts / len(y)
    ent = -np.sum(probabilities * np.log2(probabilities))
    return ent

def information_gain(X, y, feature):
    total_entropy = entropy(y)
    values, counts = np.unique(X[feature], return_counts=True)
    weighted_entropy = np.sum([(counts[i] / np.sum(counts)) * entropy(y[X[feature] == values[i]]) for i in range(len(values))])
    info_gain = total_entropy - weighted_entropy
    return info_gain
```

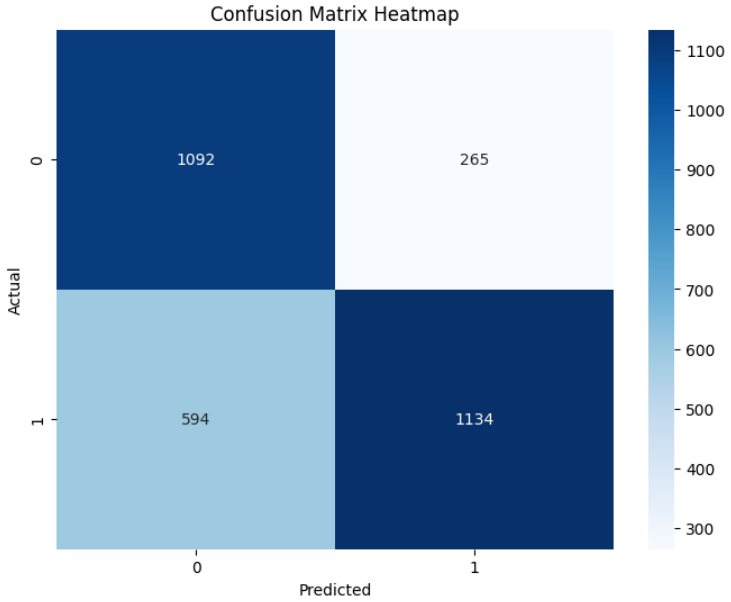
سپس درخت تصمیم با استفاده از الگوریتم ID3 پیاده‌سازی شد. توابع مختلفی پیاده سازی شدند. متد predict_one و predict توضیحات خاصی لازم ندارند. تابع build_tree به صورت بازگشتی درخت تصمیم را می‌سازد.به این صورت که در هر مرحله بهترین ویژگی برای تقسیم داده‌ها با استفاده از کسب اطلاعات انتخاب می‌شود و اگر عمق درخت به حداکثر عمق مجاز برسد یا تمامی نمونه‌ها به یک کلاس تعلق داشته باشند فرآیند تقسیم متوقف می‌شود.

۶-۳_ آموزش مدل

پس از آموزش مدل نتایج زیر حاصل شد:

```
Accuracy: 0.7215559157212318
Confusion Matrix:
[[1092  265]
 [ 594 1134]]
Precision: 0.810578984989278
Recall: 0.65625
```

دقت مدل برابر ۷۲.۱۵ حاصل شد که قابل قبول است. در شکل زیر هیت‌مپ ماتریس آشفتگی را مشاهده می‌کنیم:



همانگونه که مشاهده می‌شود، بیشتر داده‌هایی که به اشتباه طبقه‌بندی شده‌اند مربوط به کلاس ۱ هستند. یکی از مزایای درخت تصمیم بحث تفسیرپذیر بودن آن است. اگر بخواهیم علت اینکه مدل برای کلاس ۱ ضعیف عمل کرده را پیدا کنیم می‌توانیم بر روی ویژگی‌های مختلف بررسی دقیق‌تری داشته و علت را متوجه شویم.