بسم الله الرحمن الرحیم

دانشکده مهندسی برق و کامپیوتر

یادگیری ماشین - تمرین سوم

سید مهدی رضوی

استاد : آقای دکتر توسلی پور - آقای دکتر ابوالقاسمی

اردیبهشت ماه ۱۴۰۳

فهرست مطالب

فهرست تصاویر

۱ تمرین اول

حالت اول :

$$P(x > 2) = P(x = 3) + P(x = 4) + P(x = 5)$$
$$P(x > 2) = \binom{5}{3}(0.7)^3(0.3)^2 + \binom{5}{4}(0.7)^4(0.3)^1 + \binom{5}{5}(0.7)^5(0.3)^0 = 0.8369199999999999$$

حالت دوم :

$$P(x > 5) = P(x = 6) + P(x = 7) + P(x = 8) + P(x = 9)$$
$$P(x > 5) = \binom{9}{6}(0.7)^6(0.3)^3 + \binom{9}{7}(0.7)^7(0.3)^2 + \binom{9}{8}(0.7)^8(0.3)^1 + \binom{9}{9}(0.7)^9(0.3)^0 = 0.729659098$$

حالت سوم :

برای این مثال ۱۰۰ را مجاز از بی‌نهایت در نظرگرفته‌ایم .

$$N = 100, p = 0.7$$
$$P(x > 50) = 0.9999909653138044$$

حالت چهارم :

$$P(x > 2) = P(x = 3) + P(x = 4) + P(x = 5)$$
$$P(x > 2) = \binom{5}{3}(0.5)^3(0.5)^2 + \binom{5}{4}(0.5)^4(0.5)^1 + \binom{5}{5}(0.5)^5(0.5)^0 = 0.5$$

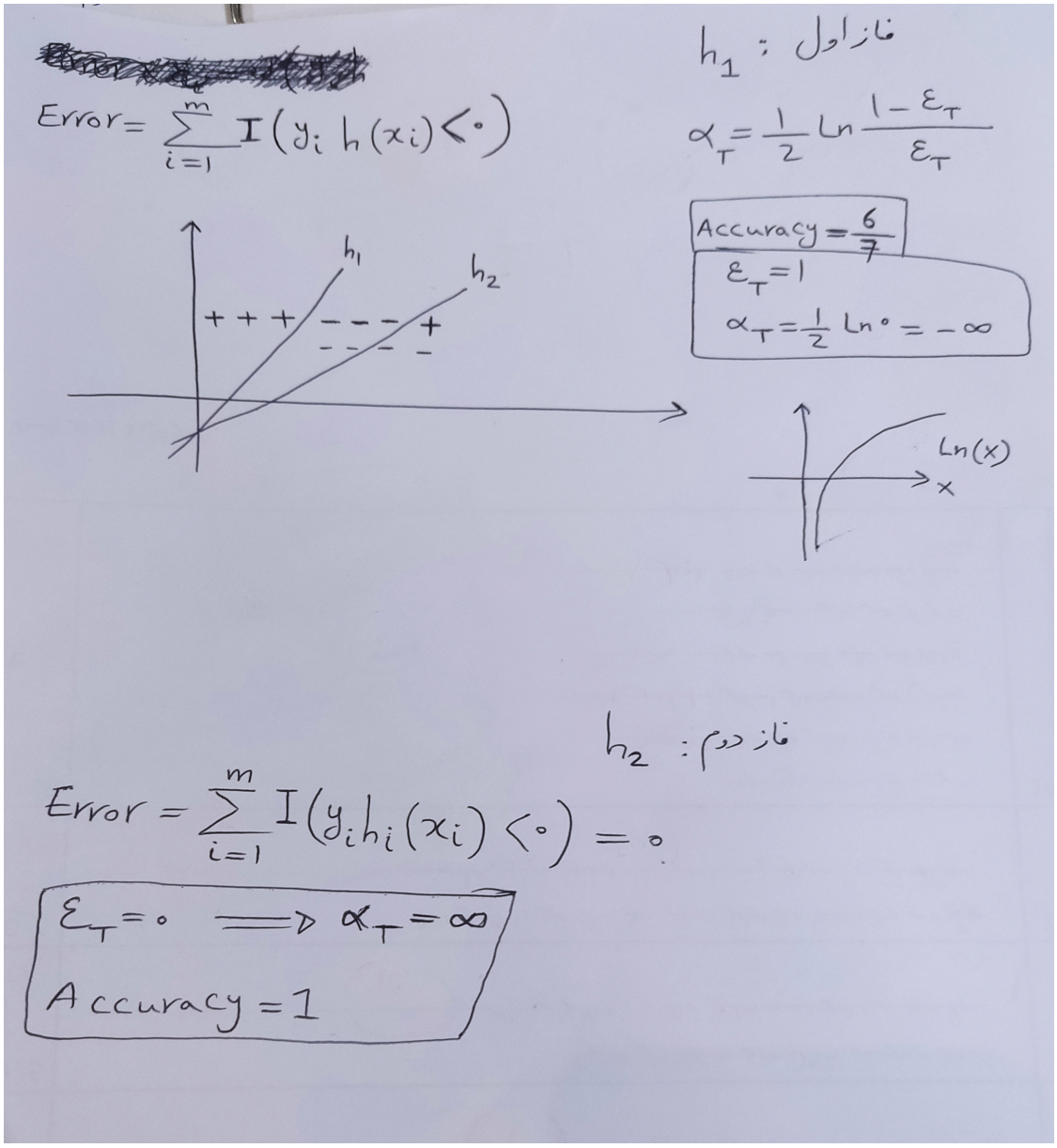
نتیجه‌گیری :

با افزایش حجم جامعه آماری در صورتی که میزان احتمال موفقیت یا همان تشخیص در این مساله بزرگتراز یک دوم باشد ، احتمال تشخیص صحیح جرم در نهایت افزایش خواهد یافت.
اما در صورتی که احتمال موفقیت هر متغیر یا هر قاضی یک دوم باشد ، عدم قطعیت به اوج خود می‌رسد .
(مساله انتروپی هم به ازای دو حالت برای هر متغیر یعنی تشخیص و عدم‌تشخیص به ازای احتمال یک دوم بیشترین حالت را به خود خواهد گرفت.)

۲ تمرین دوم

بله، الگوریتم AdaBoost برای ترکیب چندین طبقه‌بند ضعیف مانند Decision Stump به گونه‌ای طراحی شده است که دقت کلی سیستم را بهبود بخشد.
به طور خاص، AdaBoost وزن‌های بیشتری به نمونه‌هایی که به اشتباه طبقه‌بندی شده‌اند اختصاص می‌دهد و طبقه‌بندهای جدید را آموزش می‌دهد تا بر روی این نمونه‌های مشکل‌دار تمرکز کنند. با انجام این فرآیند، AdaBoost قادر است به دقت بالاتری نسبت به طبقه‌بندهای ضعیف فردی دست یابد.

اگر خطای وزن‌دار آموزشی در یک دور بین 0 و 0.5 باشد می‌دانیم که طبقه‌بند ضعیف بهتر از حد حدس تصادفی عمل می‌کند اما ممکن است هنوز خطا وجودداشته‌باشد.
اگر خطای وزن‌دار آموزشی در یک دور برابر با صفر باشد، به این معناست که طبقه‌بند ضعیف تمام نمونه‌ها را به درستی در آن دور دسته‌بندی کرده است. این سناریو بسیار ایده‌آل است اماممکن است در عمل به دلیل پیچیدگی داده‌ها همچین خطایی قابل دستیابی نباشد.
در AdaBoost ، هدف اصلاح عملکرد طبقه‌بندهای ضعیف به صورت تکراری با تنظیم وزن‌های نمونه‌های آموزش است. الگوریتم ادامه می‌دهد تاطبقه‌بندهای ضعیف ، یا به تعداد تعریف شده از تکرارها برسند یا به شرط توقف برسد و الگوریتم متوقف شود . (مثلا به حد مشخصی از خطا)
با این حال، به دست آوردن خطای وزن‌دار دقیقاً برابر با ۰ برای همه دورها همواره قابل دستیابی نیست، به خصوص اگر داده‌ها نویز داشته باشند یا پیچیده باشند.
در عمل، الگوریتم AdaBoost به کمینه کردن خطای وزن‌دار آموزشی در چندین دور با اختصاص وزن بیشتر به نمونه‌های اشتباه دسته‌بندی شده در هر تکرار، می‌پردازد. مدل نهایی ترکیبی از این یادگیرنده‌های ضعیف است، با اختصاص وزن بالاتر به طبقه‌بندهایی که عملکرد بهتری دارند.
بنابراین، تضمین نشده است که خطای وزن‌دار در پس از تعداد مشخصی از دورها در AdaBoost دقیقاً برابر با ۰ باشد. تعداد دورهای لازم برای رسیدن به خطای ۰ به پاسخگو بودن به پیچیدگی داده، کیفیت یادگیرنده‌های ضعیف و سایر عوامل وابسته است.

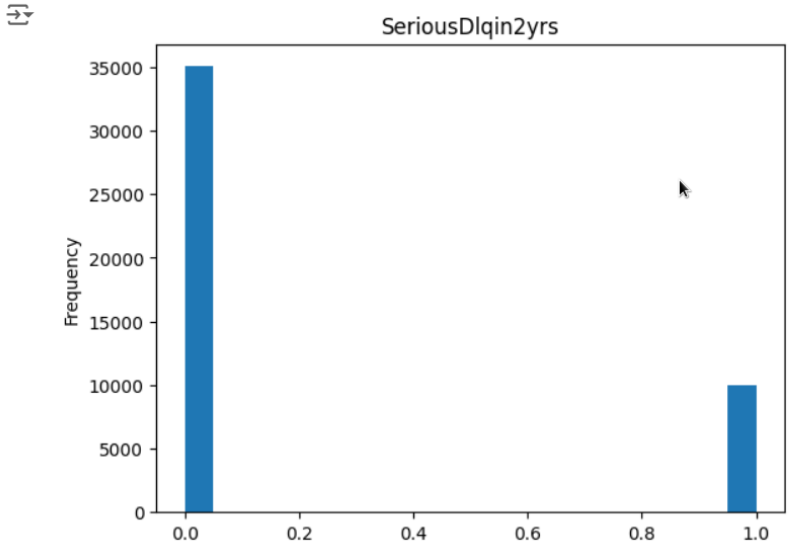


شکل ۱: مراحل تقویت طبقه‌بندهای ضعیف و محاسبه خطا در الگوریتم AdaBoost

۳ تمرین سوم

```python
# @title SeriousDlqin2yrs

from matplotlib import pyplot as plt
data['SeriousDlqin2yrs'].plot(kind='hist', bins=20, title='SeriousDlqin2yrs')
plt.gca().spines[['top', 'right',]].set_visible(True)
```



شکل ۲: نمودار توزیع مقادیر متغیر هدف در این مجموعه داده

همانطور که از توزیع داده‌ها نیز مشخص است ، میزان مشتری‌های بدحساب تا حدود 3.5 برابر مشتری‌های خوش‌حساب بانک است.

Confidence Level

```
import pandas as pd
import numpy as np

data_filtered = data[data['SeriousDlqin2yrs'] == 0]

n_bootstraps = 1000
confidence_level = 0.90

def bootstrap_confidence_interval(data, n_bootstraps, ci):
    bootstrap_samples = np.random.choice(data, size=(n_bootstraps, len(data)), replace=True)
    bootstrap_stats = np.mean(bootstrap_samples, axis=1)
    lower_bound = np.percentile(bootstrap_stats, (1 - ci) / 2 * 100)
    upper_bound = np.percentile(bootstrap_stats, (1 + ci) / 2 * 100)
    return lower_bound, upper_bound


age_confidence_interval = bootstrap_confidence_interval(data_filtered['age'].values, n_bootstraps, confidence_level)

print(f"The {confidence_level*100}% confidence interval for the 'age' column is: {age_confidence_interval}")
```

```
The 90.0% confidence interval for the 'age' column is: (52.60791163627023, 52.85553272255044)
```

شکل ۳: بازه میانگین سنی مشتریان بدحساب با اطمینان ۹۰ درصد

با اطمینان ۹۰ درصد می‌توانیم بیان کنیم که سن یک مشتری بدحساب در بازه فوق خواهد بود.

```python
# Set the target and features
X = data.drop('SeriousDlqin2yrs', axis=1)
y = data['SeriousDlqin2yrs']

# Define the parameters for grid search
param_grid = {
    'max_features': [1, 2, 4],
    'min_samples_leaf': [3, 5, 7, 9],
    'max_depth': [5, 10, 15]
}

# Create the Random Forest classifier
rf = RandomForestClassifier(n_estimators=100, class_weight='balanced')

# Create the stratified 5-fold cross-validation
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Perform grid search with cross-validation
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, scoring='roc_auc', cv=cv)
grid_search.fit(X, y)

# Get the best parameters and best score
best_params = grid_search.best_params_
best_score = grid_search.best_score_

print("Best parameters:", best_params)
print("Best ROC AUC score:", best_score)
```

```
Best parameters: {'max_depth': 10, 'max_features': 2, 'min_samples_leaf': 9}
Best ROC AUC score: 0.8352605392260977
```

شکل ۴: محاسبه پارامترهای بهینه جنگل تصمیم با استفاده از الگوریتم GridSearch

بر اساس نتایج متغیرهای بهینه برای الگوریتم **Forest Random** می‌توانیم نتیجه بگیریم که این مدل با عمق حداکثر ۱۰ و حداکثر دو ویژگی و حداقل ۹ نمونه در هر برگ، بهترین عملکرد را داشته است.
امتیاز **ROC AUC** برابر با **0.835** نشان می‌دهد که این مدل قادر به تفکیک موثر بین کلاس‌های مثبت و منفی در مجموعه داده است، با امتیاز بالاتر بهترین عملکرد را نشان می‌دهد.
به صورت کلی به نظر می‌رسد تعادل مناسبی بین پیچیدگی و تعمیم‌پذیری در این مدل فراهم شده‌است. و این مساله به قابلیت خوب پیش‌بینی این مدل کمک کرده‌است.

```
for i in range(len(feature_importances_df)):
  print(feature_importances_df.iloc[i])
```

```
Feature       NumberOfDependents
Importance              0.013791
Name: 6, dtype: object
Feature       MonthlyIncome
Importance         0.052535
Name: 5, dtype: object
Feature       DebtRatio
Importance     0.065666
Name: 2, dtype: object
Feature            age
Importance    0.079082
Name: 0, dtype: object
Feature       NumberOfTime60-89DaysPastDueNotWorse
Importance                                0.183924
Name: 4, dtype: object
Feature       NumberOfTime30-59DaysPastDueNotWorse
Importance                                0.235483
Name: 1, dtype: object
Feature       NumberOfTimes90DaysLate
Importance                   0.369517
Name: 3, dtype: object
```

شکل ۵: نمایش میزان اهمیت هریک از ویژگی‌ها در ساختن جنگل تصمیم

بر اساس کتابخانه استفاده شده ، تاثیر ویژگی NumberOfDependents در ساختن جنگل تصمیم از سایر ویژگی‌ها کمتر است.

```
  estimator = estimator.set_params(**cloned_parameters)
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py:674: FutureWarning: Par
  estimator = estimator.set_params(**cloned_parameters)
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py:674: FutureWarning: Par
  estimator = estimator.set_params(**cloned_parameters)
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_search.py:905: FutureWarning: Paramet
  clone(base_estimator).set_params(**self.best_params_)
Best parameters: {'max_samples': 0.5, 'max_features': 2, 'base_estimator__C': 1}
Best ROC AUC score: 0.7688710209338883
```

شکل ۶: محاسبه پارامترهای بهینه جنگل تصمیم با استفاده از الگوریتم RandomizedSearch و طبقه‌بند پایه Logistic Regression

بر اساس نتایج متغیرهای بهینه برای Bagging با استفاده از Logistic Regression به عنوان طبقه‌بند پایه، می‌توان نتیجه گرفت که مدل بهترین عملکرد را با حداکثر ۵۰ درصد از نمونه‌ها و حداکثر ۲ ویژگی داشته است. پارامتر تنظیم C برای تخمین‌گر پایه Logistic Regression به ۱ تنظیم شده بود. امتیاز ROC AUC برابر با 0.769 نشان می‌دهد که مدل قادر به تفکیک موثر بین کلاس‌های مثبت و منفی در مجموعه داده است، اگرچه این امتیاز کمتر از مدل‌های دیگر مانند جنگل تصمیم است.

به طور کلی، این پارامترها نشان می‌دهند که مدل Bagging با استفاده از Logistic Regression به عنوان طبقه‌بند پایه، به یک رویکرد متعادل در مورد پیچیدگی و عموم‌سازی دست یافته است. به نظر می‌رسد که این مدل عملکرد متوسطی در پیش‌بینی متغیر هدف بر اساس مجموعه داده دارد.

۴ تمرین چهارم

در ابتدا در تابع init مقادیر پیش‌فرض برای baseestimator و nestimators تعیین شده و یک LabelEncoder جهت تبدیل برچسب‌ها به اعداد صحیح ایجاد می‌شود.
در تابع fit داده‌های ورودی X و برچسب‌های y به اعداد صحیح تبدیل می‌شوند و وزن‌های نمونه‌ها مقداردهی اولیه می‌شوند. سپس برای تعداد nEstimators ، یک مبنای تصمیم جدید (learner) از baseEstimator ساخته شده و روی داده آموزش (X و yEncoded) با وزن‌های نمونه‌ها آموزش داده می‌شود. سپس پیش‌بینی‌های مبنای تصمیم بر روی داده‌های آموزش انجام شده و خطا مبنای تصمیم محاسبه می‌شود. وزن جدید برای مبنای تصمیم و خطا محاسبه شده و به لیست learners و learnerWeights اضافه می‌شود. در نهایت، وزن‌های نمونه‌ها بر اساس خطا و وزن مبنای تصمیم به روزرسانی می‌شوند.
در تابع predict ، پیش‌بینی‌های هر learners بنای تصمیم بر روی داده‌ها انجام شده و وزن‌های هر مبنای تصمیم به تابع weightedvotes اضافه می‌شود. در نهایت، پیش‌بینی نهایی بر اساس وزن‌های جمع شده بر روی هر کلاس انجام می‌شود و با استفاده از inversetransform اعداد صحیح به برچسب‌ها تبدیل می‌شوند.

```python
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_mat = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

# Print the results
print("Accuracy:", accuracy)
print("Confusion Matrix:\n", conf_mat)
print("Classification Report:\n", class_report)
```

```
X_train shape : (105, 4)
y_train shape : (105,)
X_test shape : (45, 4)
Accuracy: 1.0
Confusion Matrix:
 [[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

شکل ۷: ارزیابی مدل با استفاده از مدل MultiClassBoosting

با توجه به تعداد داده محدود این دیتاست می‌بینیم که مدل با حداکثر دقت آموزش دیده‌است. به نظر می‌رسد که تعداد ایپاک‌ها برای آموزش مدل برای دیتاست ما بسیار مناسب بوده‌است. تمامی نمونه‌ها به درستی طبقه‌بندی شده‌اند و همچنین تمام نمونه‌های درست درجای مناسب خود قرارگرفته‌اند.

۵ تمرین پنجم

$$\arg\max_i I(Y, X_i) = \arg\min_i H(Y|X_i)$$

فشارخون

$$H\left(Y \mid X_1\right) = \frac{6}{14}\left[-\frac{3}{6}\log\frac{3}{6} - \frac{3}{6}\log\frac{3}{6}\right] + \frac{8}{14}\left[-\frac{6}{8}\log\frac{6}{8} - \frac{2}{8}\log\frac{2}{8}\right] = 0.568125172$$

سطح کلسترول

$$H\left(Y \mid X_2\right) = \frac{5}{14}\left[-\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5}\right] + \frac{4}{14}\left[-\frac{4}{4}\log\frac{4}{4}\right] + \frac{5}{14}\left[-\frac{2}{5}\log\frac{2}{5} - \frac{3}{5}\log\frac{3}{5}\right]$$
$$H\left(Y \mid X_2\right) = 0.208775181$$

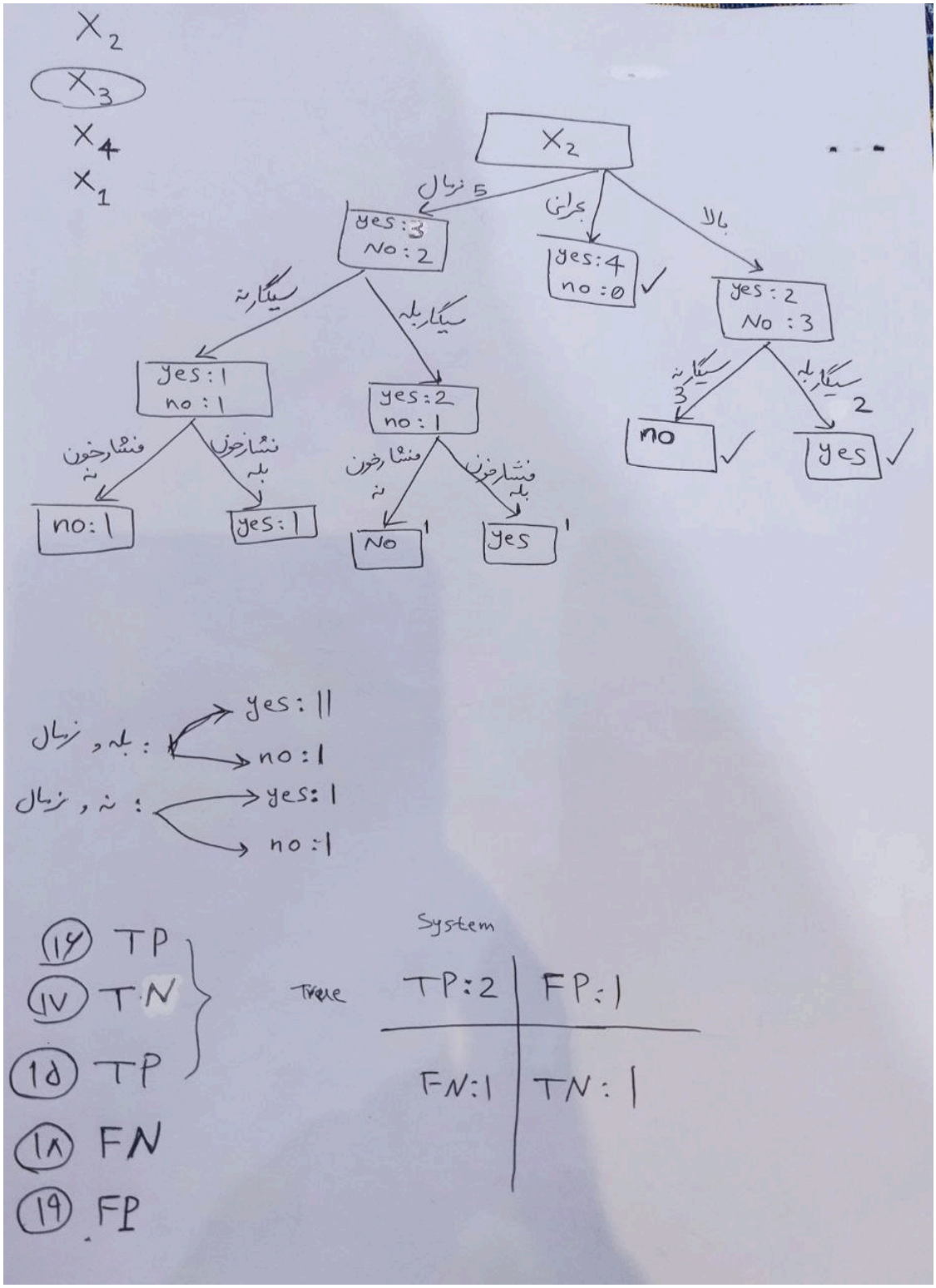
مصرف سیگار

$$H\left(Y \mid X_3\right) = \frac{1}{2}\left[-\frac{3}{7}\log\frac{3}{7} - \frac{4}{7}\log\frac{4}{7}\right] + \frac{1}{2}\left[-\frac{6}{7}\log\frac{6}{7} - \frac{1}{7}\log\frac{1}{7}\right] = 0.237347238$$

وزن

$$H\left(Y \mid X_4\right) = \frac{6}{14}\left[-\frac{4}{6}\log\frac{4}{6} - \frac{2}{6}\log\frac{2}{6}\right] + \frac{4}{14}\left[-\frac{2}{4}\log\frac{2}{4} - \frac{2}{4}\log\frac{2}{4}\right] + \frac{4}{14}\left[-\frac{3}{4}\log\frac{3}{4} - \frac{1}{4}\log\frac{1}{4}\right]$$
$$H\left(Y \mid X_4\right) = 0.473963125$$

درخت تصمیم به دلیل تمایل به ایجاد ساختارهای پیچیده و عمیق که دقیقاً با داده‌های آموزش مطابقت دارند، مستعد برای overfitting است.
این به معنای این است که درخت تصمیم ممکن است بر روی داده‌های آزمون ناکارآمد باشد. این مسئله به خصوص زمانی رخ می‌دهد که اجازه داده شود درخت بدون هیچ محدودیتی رشد کند.
دو روش برای جلوگیری از overfitting در درخت تصمیم وجود دارد:
۱. قالب‌بندی: قالب‌بندی شامل کاهش سایز درخت با حذف گره‌هایی است که بهبود معناداری در دقت پیش‌بینی نمی‌دهند. این کار باعث می‌شود درخت از پیچیدگی زیاد و overfitting داده‌های آموزش جلوگیری کند.
۲. تعیین حداکثر عمق یا حداقل تعداد نمونه‌ها در هر گره: با تعیین محدودیت‌ها بر روی حداکثر عمق درخت یا حداقل تعداد نمونه‌های مورد نیاز برای تقسیم گره، می‌توان از رشد بیش از حد و پیچیدگی درخت جلوگیری کرد. این پارامترها به کنترل پیچیدگی درخت کمک کرده و توانایی عمومی سازی آن را بهبود می‌بخشند.



شکل ۸: درخت تصمیم حاصل از الگوریتم ID3

۶ تمرین ششم

```
# Calculate accuracy
accuracy = accuracy_score(test_data['Recidivism - Return to Prison numeric'], predictions)
print('Accuracy:', accuracy)

# Confusion matrix
conf_matrix = confusion_matrix(test_data['Recidivism - Return to Prison numeric'], predictions)
print('Confusion Matrix:')
print(conf_matrix)

# Classification report
class_report = classification_report(test_data['Recidivism - Return to Prison numeric'], predictions)
print('Classification Report:')
print(class_report)
```

```
Accuracy: 0.37341977309562396
Confusion Matrix:
[[ 870 1850]
 [  83  282]]
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.32      0.47      2720
           1       0.13      0.77      0.23       365

    accuracy                           0.37      3085
   macro avg       0.52      0.55      0.35      3085
weighted avg       0.82      0.37      0.44      3085
```

شکل ۹: نتایج ارزیابی مدل درخت تصمیم ID3

تابع id3 را تعریف کرده‌ایم که یک درخت تصمیم‌گیری id3 را بر اساس دیتافریم ورودی و ستون هدف می‌سازیم. این تابع به صورت بازگشتی فراخوانی می‌شود و در هر مرحله ویژگی‌ای که بیشترین افزایش اطلاعات را دارد را به عنوان ویژگی تقسیم‌بندی انتخاب می‌کند.
تابع افزایش اطلاعات بر اساس انتروپی ویژگی‌ها تعریف شده‌است.

با توجه به معیار میانگین دقت‌ها و به‌خاطرسپاری از درخت تصمیم متوجه‌خواهیم‌شد که این درخت‌ها با توجه به این‌که قانون‌محور هستند ، معمولا دقت‌های قابل‌توجهی دارند و میزان به‌خاطرسپاری درخت تصمیم به مراتب کمتر است.
اعداد و ارقام بالا به این موضوع تاکید می‌کند.
همچنین حریصانه‌بودن الگوریتم نیز بر این موضوع تصدیق خواهد کرد.