Amirkabir University of Technology
(Tehran Polytechnic)

Applied Machine Learning Course

By Dr. Nazerfard

CE5501 | Spring 2024

Assignment (4)

Name: Esmaeil Khosravi

S_ID: 402131046

Email: es.khosravi@aut.ac.ir

فهرست مطالب

سوال ۱

مهم‌ترین چالش پیاده سازی این سوال، رفع محدودیت پلتفرم لینکدین در مواجه با **scraping** داده های پزوفایل افراد است. در صورت نبودن این محدودیت، جمع آوری داده ها می توانست راحت تر صورت پذیرد. برای پیاده سازی **scraping** پروفایل افراد چندین تابع نوشته شده است که هر کدام به تفصیل توضیح داده می شود.

برای جمع آوری داده از این پلتفرم نیاز به وارد شدن یا **log in** وجود دارد. بنابراین تابعی لازم است نوشته شود که با گرفتن ادرس **sign in** لینکدین اطلاعات کاربری را وارد کرده و بتواند وارد این پلتفرم شود. یک تابع به نام **linkedin_login**ان نوشته شده است.

```
def linkedin_login(driver, username, password):
    driver.get(login_url)
    time.sleep(10)

    username_input = driver.find_element(By.ID, 'username')
    password_input = driver.find_element(By.ID, 'password')

    username_input.send_keys(username)
    password_input.send_keys(password)

    password_input.send_keys(Keys.RETURN)
    time.sleep(20)

linkedin_login(driver, linkedin_username, linkedin_password)
```

این تابع فراخوانی شده و **scraper** وارد پلتفرم می شود.

در مرحله بعدی نیاز است که افرادی که در حوزه مهندسی نرم افزار کار می کنند شناسایی شده و اطلاعات موجود در پروفایل آن ها جمع آوری شود. متاسفانه، این پلتفرم تعداد افراد قابل نمایش در قسمت جست و جو را محدود کرده است بنابراین روش دیگری لازم است و فرایند جست و جو افراد باید خارج از این پلتفرم انجام شود.

بعد از جست جوی افراد در گوگل، نیاز است لینک پروفایل این افراد جمع آوری شده و سپس اطلاعات هرکدام از آنها جمع آوری شود. در حین جست و جو برای افراد در گوکل لازم است چندین عمل مانند بروزرسانی صفحه جست و جو، کلیک بر روی دیدن نتایج بیشتر، **scroll**کردن صفحه و کلیک بر روی لینک هایی که نتایج بیشتری نشان می دهند، انجام شود که برای هر کدام از آنها تابعی نوشته شده است.

```python
def scroll_to_bottom(browser):
    last_height = browser.execute_script("return document.body.scrollHeight")
    while True:
        browser.execute_script("window.scrollTo(0, document.body.scrollHeight);")
        time.sleep(5)  # Wait for the page to load
        new_height = browser.execute_script("return document.body.scrollHeight")
        if new_height == last_height:
            break
        last_height = new_height
```

تابع **scroll**کردن صفحه

```python
def click_additional_link(driver):
    try:
        additional_link = driver.find_element(By.XPATH, '//*[@id="ofr"]/i/a')
        additional_link.click()
        time.sleep(5)  # Wait for the new results to load
    except:
        pass
```

تابع کلیک بر روی لینک دیدن نتایج تکراری

```python
def click_more_results(driver):
    while True:
        try:
            more_results_button = driver.find_element(By.XPATH, '//*[@id="botstuff"]/div/div[4]/div[4]/a[1]/h3/div/span[2]')
            more_results_button.click()
            time.sleep(5)  # Wait for the new results to load
        except:
            break
```

تابع کلیک بر روی گزینه نتایج بیشتر

```
def clic_next(driver):
    try:
        next_click = driver.find_element(By.XPATH, '//*[@id="pnnext"]/span[2]')
        next_click.click()
        time.sleep(5)
    except:
        pass
```

تابع کلیک بر روی گزینه ص بعدی

الیته هر کدام از این موارد همیشه رخ نمیدهند اما در صورت رخداد هر کدام تابعی برای آنها نوشته شده است.

در مرحله بعدی، بعد از اتمام جمع آوری لینک افراد لازم است تابعی نوشته شود که با گرفتن لینک پروفایل یک فرد، اطلاعات پروفایل آن را خوانده و در یک لیست ذخیره کند.

```
def scrape_profile(driver, url):
    driver.get(url)
    time.sleep(180)


    html_file = driver.page_source
    soup = BeautifulSoup(html_file, 'lxml')
    try:
        name = soup.find('h1', class_= 'text-heading-xlarge inline t-24 v-align-middle break-words')
    except:
        name = None

    try:
        headline = soup.find('div', class_ = 'text-body-medium break-words')
    except:
        headline = None

    try:
        city = soup.find('span', class_ = 'text-body-small inline t-black--light break-words')
    except:
        city = None


    try:
        exp = soup.find_all('div', class_= 'display-flex align-items-center mr1 t-bold')
    except:
        exp = None
```

قسمتی از کد تابع **scraper**

در نهایت، اطلاعات تعدادی از اعضا جمع آوری گردید اما مهم ترین مانع کار محدودیت های خود پلتفرم لینکداین است که به دلیل خط مشی های حریم خصوصی، اجازه جمع آوری پروفایل را بعد از تعدادی **scrap**

کردن، نمی دهد و نهایتا منجر به بسته شدن حساب کاربری می شود. در نهایت فایل **scv** دیتای جمع شده ذخیره شده است. لازم به ذکر است چندین بار تلاش شد که این محدودیت برداشته شود و دیتای بیشتری جمع اوری گردد به همین دلیل چندین نمونه از دیتای جمع اوری شده از افراد درفایل سوال یک موجود است و نهایتا خروجی با بیشترین دیتا در فایل **csv** ذخیره شده است.

سوال ۲

قسمت a

در این قسمت، داده‌ها از یک فایل **CSV** با متد **read_csv** بارگیری می‌شود و یک نمونه تصادفی ده تایی از داده‌ها با متد **sample** چاپ می‌شود. در این قسمت مراحل پیش پردازش داده ها انجام شده است. ویژگی هایی که در پیش بینی یا دسته بندی هدف تاثیری ندارند، از دیتاست حذف شده اند. ویژگی های با مقادیر **null** شناسایی شده و مقدار آنها با مقدار میانگین همه مقادیر ستون مربوطه پر شده اند برای این کار از متد **fillna** استفاده شده است. از آنجایی که ستون هدف یعنی **survived** دارای مقادیر رشته ای است نیاز است که این مقادیر به نوع عددی تبدیل شوند که با استفاده از کتابخانه **OneHotEncoder** این کار انجام شده است.

قسمت b

در این قسمت، با استفاده از کتابخانه **sklearn** مجموعه دادگان با استفاده از **train_test_split** به دو گروه ازمایشی و اموزشی با نسبت ۲۰ به ۸۰ تقسیم شده اند.

قسمت c

در این قسمت یک مدل **SVM** خطی برای دسته بندی ویژگی هدف یعنی نجات یافتن یا نیافتن مسافران آموزش دیده شده است. در نهایت مدل بعد از آموزش دیدن، بر روی داده های تست عمل پیش بینی را انجام داده که وضعیت شاخص های ارزیابی آن در زیر آمده است.

```
accuracy: 0.798
precision: 0.726
recall: 0.768
```

نتیجه نشان میدهد مدل عملکرد نسبتا خوبی را داشته است اما می توان با تنظیم بهتر هایپرپارامترها به نتیجه بهتری رسید.

قسمت d

در این قسمت سعی شده است با استفاده از روش **cross validation** و **grid search** بهترین مقادیر ممکن برای هایپرپارامتر ها در مدل آموزش دیده شده را یافت. مجموعه هایپرپارامتر های تست شده برای یافتن بهترین مقادیر در ادامه آمده است:

```
param_grid = {
    'C': [0.1, 1, 10, 100],
    'gamma': [1, 0.1, 0.01, 0.001],
    'kernel': ['rbf', 'linear', 'poly'],
    'degree':[2, 3, 4, 5]
}
```

قرار است با استفاده از روش های گفته شده بهترین ترکیب ممکن از هایپرپارامتر ها برای آموزش مدل استفاده شود.

قسمت e

نتیجه نشان میدهد بهترین مقادیر برای هایپرپارامتر ها C = 100 و kernel = linear است.

در ادامه دلیل احتمالی این مقادیر بدست آمده توجیه می شود:

پارامترC

پارامتر C در SVM میزان تعادل بین داشتن خطای کم در آموزش و خطای کم در تست را کنترل می‌کند که نوعی تنظیم (Regularization) است. مقدار بالاتر C تأکید بیشتری بر روی دسته‌بندی صحیح تمام مثال‌های آموزشی دارد که ممکن است منجر به کاهش تنظیم و احتمالاً بیشتر شدن بیش‌برازش (Overfitting) شود.

C = 100این مقدار بالا نشان‌دهنده این است که مدل برای دسته‌بندی اشتباهات به شدت جریمه می‌شود. با توجه به عدم توازن شدید کلاس‌ها (با تراکنش‌های جعلی که نادر اما حیاتی هستند)، استفاده از مقدار بالاتر C به مدل کمک می‌کند که نسبت به کلاس اقلیت (تراکنش‌های جعلی) حساس‌تر باشد و سعی کند تا حد ممکن نقاط بیشتری را به درستی دسته‌بندی کند. این امر به ویژه در سناریوهایی که هزینه عدم شناسایی تراکنش‌های جعلی بسیار بالاست، مفید است.

پارامترKernel

پارامتر kernel در SVM نوع تبدیل اعمال شده به داده‌ها را تعیین می‌کند. کرنل خطی زمانی استفاده می‌شود که داده‌ها به صورت خطی قابل تفکیک هستند یا تقریباً اینگونه هستند.

kernel = linear: با توجه به اینکه داده‌ها با استفاده از PCA پردازش شده‌اند، ابعاد داده‌ها قبلاً کاهش یافته و به شکلی تبدیل شده‌اند که بیشترین واریانس به صورت خطی نشان داده شود. PCA معمولاً با مدل‌های خطی به خوبی کار می‌کند زیرا هدف آن به حداکثر رساندن جدایی خطی در فضای ابعاد کاهش یافته است. بنابراین، استفاده از کرنل خطی منطقی است زیرا با ماهیت داده‌های تبدیل شده همخوانی دارد.

قسمت f

در این قسمت یک مدل **svm** با هایپرپارامتر های بدست آمده از قسمت قبل، ساخته و آموزش دیده شده است.

عمل دسته بندی روی داده آزمایشی انجام شده و نتایج ارزیابی آن با مدل قبلی مقایسه شده است.

```
accuracy: 0.798
precision: 0.726
recall: 0.768
```

```
accuracy: 0.803
precision: 0.73
recall: 0.783
```

نتیجه ارزیابی نشان می دهد مدل آموزش دیده شده با بهترین هایپرپارامتر های بدست امده دقت بهتری نسبت به مدل قبلی دارد.

سوال ۳

قسمت a

اگر منظور سوال از اینکه همه خروجی ها صفر باشند این باشد که مدل **svm** همه خرید ها را به عنوان مورد قانونی در نظر میگیرد، دقت آن طبق فرمول **accuracy** به صورت زیر محاسبه می شود:

در این مسله اگر قانونی بودن خرید به عنوان دسته منفی (مقدار صفر) و جعلی بودن آن به عنوان دسته مثبت (مقدار یک) شناخته شود، محاسبه **accuracy** طبق فرمول زیر بدست می آید.

TP: 0

TN: 284807

P + N : 284807 + 492 = 285299

ACC = (TP + TN) / P + N → ACC = 0 + 284807/285299 = 0.99

دقت بدست آمده برای مدل طبق پیش بینی آن بر اساس پیش بینی های درست مدل برابر ۹۹ درصد بدست آمده است.

قسمت b

با توجه به نتیجه قسمت قبل اهمیت معیارهای **false positive** و **false negative** در ارزیابی دسته بند پررنگ می شود به طوریکه اگر کارایی دسته بند فقط بر اساس **TP** و **TN** ارزیابی شود، نتیجه بدست آمده گمراه کننده است. به دلیل اینکه دقت مدل در تشخیص خرید های غیر قانونی در نظر گرفته نمی شود. به دلیل اینکه تعداد زیادی از داده ها مربوط به کلاس صفر هستند، و همه خروجی های مدل مربوط به این کلاس هستند، دقت بدست آمده طبق فرمول **accuracy** دقت واقعی مدل را در تشخیص خرید های غیرقانونی یعنی کلاس مثبت نشان نمی دهد. اگر شاخص های ارزیابی دیگری همچون **recall** یا **precision** استفاده شود مقدار صفر برای مدل بدست خواهد امد زیرا که صورت کسر در فرمول این شاخص ها تعداد پیش بینی درست برای کلاس مثبت است و از آنجاییکه این مدل همه تشخیص های خود را به کلاس صفر داده، شاخص های **precision** و **recall** برای آن صفر هستند.

Recall= TP/TP + FN = 0

Precision = 0

قسمت c

با استفاده از متد **train_test_split** دیتاست به مجموعه آموزشی و آزمایشی با اندازه ۸۰ به ۲۰ تقسیم شده است. همچنین در این قسمت، مراحل پیش پردازش داده ها نیز انجام شده است. با استفاده از **onehotencoding** ویژگی هدف از فرمت رشته ای به عددی تبدیل شده است.

قسمت d

در این قسمت یک مدل **svm** طبق خواسته سوال آموزش دیده شده است و عمل دسته بندی با استفاده از این مدل بر روی داده های آزمایشی انجام شده است.

قسمت e

در این قسمت ارزیابی مدل با استفاده از شاخص های **precion** ، **recall**، **accuracy** و **f1-score** انجام شده است. امتیاز مدل به ازای شاخص **accuracy** بیش از ۹۹ درصد و به ازای دیگر شاخص ها صفر بدست آمده است.

```
accuracy: 0.998
precision: 0.0
recall: 0.0
```

	precision	recall	f1-score	support
class 0	1.00	1.00	1.00	56847
class 1	0.00	0.00	0.00	115
accuracy			1.00	56962
macro avg	0.50	0.50	0.50	56962
weighted avg	1.00	1.00	1.00	56962

همچنین در ادامه این قسمت مدل **svm** دیگری با **kernel rbf** آموزش دیده شده است. که نتایج حاصل از ارزیابی آن به صورت زیر است:

	precision	recall	f1-score	support
class 0	1.00	1.00	1.00	56847
class 1	0.46	0.80	0.58	115
accuracy			1.00	56962
macro avg	0.73	0.90	0.79	56962
weighted avg	1.00	1.00	1.00	56962

قسمت f

برای افزایش دقت در شناسایی تراکنش‌های نامعتبر (تقلبی) و بهبود عملکرد کلی مدل، چندین روش وجود دارد. هر کدام از این روش‌ها می‌تواند بر دقت هر دو کلاس تأثیر بگذارد و ممکن است تأثیراتی بر پیچیدگی محاسباتی نیز داشته باشد. در ادامه چند روش برای حل این مشکل پیشنهاد شده است:

روش اول

مدیریت عدم تعادل کلاس‌ها

افزایش نمونه‌های کلاس اقلیت (Oversampling) استفاده از تکنیک‌هایی مانند SMOTE (تکنیک افزایش نمونه‌های مصنوعی) برای افزایش تعداد نمونه‌های تقلبی در مجموعه داده‌های آموزشی.

کاهش نمونه‌های کلاس اکثریت :(Undersampling) کاهش تعداد نمونه‌های معتبر برای ایجاد توازن در مجموعه داده.

استفاده از وزن‌های متعادل: تنظیم وزن کلاس‌ها در طول آموزش به طوری که به کلاس اقلیت اهمیت بیشتری داده شود.

تأثیر بر دقت:

افزایش دقت (Recall) برای کلاس نامعتبر (تراکنش‌های تقلبی بیشتری شناسایی می‌شوند).

ممکن است کمی دقت (Precision) برای کلاس معتبر کاهش یابد به دلیل افزایش تعداد مثبت‌های کاذب.(False Positives)

دقت کلی ممکن است کاهش یابد، اما تعادل بین دقت و بازخوانی برای کلاس نامعتبر بهبود می‌یابد.

تأثیر محاسباتی:

افزایش نمونه‌های کلاس اقلیت حجم داده‌های آموزشی را افزایش می‌دهد که ممکن است زمان محاسبات را افزایش دهد.

کاهش نمونه‌های کلاس اکثریت حجم داده‌های آموزشی را کاهش می‌دهد که ممکن است زمان محاسبات را کاهش دهد.

استفاده از وزن‌های متعادل معمولاً تغییر زیادی در زمان محاسبات ایجاد نمی‌کند.

روش دوم:

روش‌های تجمیع (Ensemble Methods): تکنیک‌هایی مانند جنگل تصادفی **(Random Forests)**، افزایش گرادیانی **(Gradient Boosting)** و **XGBoost** اغلب بر روی مجموعه داده‌های نامتوازن عملکرد بهتری دارند.

تنظیم ابرپارامترها: استفاده از جستجوی شبکه‌ای یا جستجوی تصادفی برای یافتن پارامترهای بهینه برای مدل انتخاب شده.

تأثیر بر دقت:

بهبود شناسایی هر دو کلاس با استفاده از قدرت چندین الگوریتم.

دقت و بازخوانی برای کلاس نامعتبر می‌تواند هر دو بهبود یابند، که منجر به بهبود امتیاز F1 می‌شود.

تأثیر محاسباتی:

روش‌های تجمیع و تنظیم ابرپارامترها پیچیدگی محاسباتی و زمان آموزش را افزایش می‌دهند.

روش سوم:

تکنیک‌های تشخیص ناهنجاری:

شناسایی تراکنش‌های تقلبی به عنوان ناهنجاری و استفاده از روش‌های تشخیص ناهنجاری مانند جنگل ایزوله (Isolation Forest) یا SVM تک‌کلاسی.(One-Class SVM)

تأثیر بر دقت:

افزایش دقت و بازخوانی برای کلاس نامعتبر.

ممکن است دقت کلی کاهش یابد اگر بسیاری از تراکنش‌های معتبر به عنوان ناهنجاری شناسایی شوند.

تأثیر محاسباتی:

این روش‌ها بسته به اندازه مجموعه داده و الگوریتم انتخاب شده می‌توانند محاسبات سنگینی داشته باشند.

سوال ۴

قسمت a

بعد از لود کردن دیتاست و تقسیم آن به دو گروه آموزشی و آزمایشی مدل دسته بند **Random Forest** لود می شود. برای پیدا کردن بهترین هایپرپارامتر ها از روش **grid search** استفاده شده است. بعد از پیدا کردن بهترین هایپرپارامتر ها، مدل نهایی با استفاده از آنها آموزش داده شده است. هایپرپارامتر های زیر به عنوان ورودی روش **grid search** تعریف شده اند.

```
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_features': ['sqrt', 'log2', None],
    'max_depth': [None, 10, 20]
}
```

در نهایت بعد از اجرای این روش، مقادیر زیر برای هر کدام از هایپرپارامتر ها بدست آمده اند.

```
Train Accuracy: 1.0
Test Accuracy: 0.7532467532467533
Best Model Parameters: {'max_depth': None, 'max_features': 'sqrt', 'n_estimators': 200}
```

همانطور که دیده می شود دقت مدل بر روی داده های اموزشی ۱ و بر روی داده های آزمایشی برابر ۰.۷۵ است.

قسمت b

بهترین مقادیر بدست آمده برای هایپرپارامتر ها به صورت زیر است:

```
Train Accuracy: 1.0
Test Accuracy: 0.7532467532467533
Best Model Parameters: {'max_depth': None, 'max_features': 'sqrt', 'n_estimators': 200}
```

مقادیر مختلف هایپرپارامترها بر روی کارایی مدل اثر میگذارد. برای نمونه، اگر مقدار **max_depth** برابر **None** باشد، یعنی درخت محدود به عمق خاصی نیست و می تواند جزییات زیادی از داده های آموزشی را دربر

بگیرد. به همین دلیل با محدود نبودن عمق درخت دقت آن افزایش می یابد اما استعداد آن برای بیش برازش زیاد می شود. هایپرپارامتر بعدی **max_features** یا حداکثر فیچر های دخیل در فرایند آموزش است. تنظیم این پارامتر به مقدار **sqrt** بدین معنی است که برای هر تقسیم در درخت، به تعداد جذر تعداد کل ویژگی ها ویژگی انتخاب می شود. این هایپرپارامتر منجر به تعادل بین بایاش و واریانس می شود و همچنین سرعت فرایند آموزش را افزایش می دهد هایپرپارامتر بعدی **n-estimatore** است که تعداد کل درختان جنگل را تعیین می کند، هر چه تعداد درخت بیشتر باشد، عملکرد مدل بهتر می شود اما می تواند زمان آموزش و هزینه های محاسباتی مدل را افزایش دهد.

در نهایت می توان اینطور نتیجه گیری کرد که

- **max_depth=None** اجازه می‌دهد درختان بدون محدودیت رشد کنند، که ممکن است منجر به درختان بسیار دقیقی شود که روی داده‌های آموزشی بیش از حد تطابق یافته‌اند.
- **'max_features='sqrt** به تعادل بین جذب اطلاعات کافی برای هر تقسیم و جلوگیری از تطابق بیش از حد با در نظر نگرفتن تعداد زیادی ویژگی کمک می‌کند.
- **n_estimators=200** به این معنی است که مدل دارای تعداد زیادی درخت است، که معمولاً عملکرد را با میانگین‌گیری از مدل‌های متعدد بهبود می‌بخشد، اما با هزینه زمان محاسباتی بیشتر.

در عمل، انتخاب این پارامترها باید با استفاده از تکنیک‌های اعتبارسنجی مانند اعتبارسنجی متقابل (**cross-validation**) هدایت شود تا اطمینان حاصل شود که مدل به خوبی روی داده‌های دیده نشده تعمیم می‌یابد.

قسمت c

روش‌های تجمیع (**Ensemble Methods**) در یادگیری ماشین به تکنیک‌هایی اشاره دارند که در آن‌ها چندین مدل (که اغلب به آن‌ها یادگیرنده‌های پایه یا **base learners** گفته می‌شود) ترکیب می‌شوند تا عملکرد کلی مدل بهبود یابد. هدف این روش‌ها افزایش دقت، پایداری و قابلیت اطمینان پیش‌بینی‌ها است. در این قسمت از سه روش تجمیع استفاده شده و نتایج ارزیابی آنها در ادامه آمده است:

- روش **Gradient Boosting** یکی از روش‌های تجمیع (**Ensemble**) در یادگیری ماشین است که با ترکیب چندین مدل ساده به صورت ترتیبی، یک مدل قوی‌تر و دقیق‌تر می‌سازد. این روش به طور خاص برای کاهش خطاهای مدل‌های قبلی با تمرکز بر نمونه‌هایی که به درستی پیش‌بینی نشده‌اند، طراحی شده است. نتایج ارزیابی مدل با استفاده از این روش به صورت زیر است:

```
Gradient Boosting Classifier:
Train Accuracy: 1.0
Test Accuracy: 0.7532467532467533
Best Parameters: {'max_depth': 20, 'max_features': 'sqrt', 'n_estimators': 50}
```

- **AdaBoostClassifier (Adaptive Boosting)** یک الگوریتم تجمیع (**Ensemble**) است که با ترکیب چندین مدل ساده به یک مدل قوی‌تر، عملکرد پیش‌بینی را بهبود می‌بخشد. **AdaBoost** به ویژه برای طبقه‌بندی دودویی طراحی شده است و اساس کار آن بر افزایش وزن نمونه‌هایی است که به درستی توسط مدل‌های قبلی پیش‌بینی نشده‌اند. نتایج حاصل از بکارگیری این مدل در ادامه آمده است:

```
AdaBoost Classifier:
Train Accuracy: 0.8156424581005587
Test Accuracy: 0.7619047619047619
Best Parameters: {'learning_rate': 0.1, 'n_estimators': 200}
```

- **VotingClassifier** یکی از روش‌های تجمیع (**Ensemble**) در یادگیری ماشین است که از ترکیب چندین مدل مختلف برای بهبود دقت پیش‌بینی استفاده می‌کند. این روش به جای ترکیب مدل‌ها به صورت ترتیبی (مانند **Boosting**) یا موازی (مانند **Bagging**)، پیش‌بینی‌های نهایی مدل‌ها را با استفاده از یک روش رأی‌گیری ساده ترکیب می‌کند. نتیجه حاصل از آموزش مدل به این روش در ادامه نشان داده شده است:

```
Voting Classifier:
Train Accuracy: 0.8026070763500931
Test Accuracy: 0.7489177489177489
```