Amirkabir University of Technology
(Tehran Polytechnic)

Applied Machine Learning Course

By Dr. Nazerfard

CE5501 | Spring 2024

Assignment (3)

Name: Esmaeil Khosravi

S_ID: 402131046

Email: es.khosravi@aut.ac.ir

فهرست مطالب

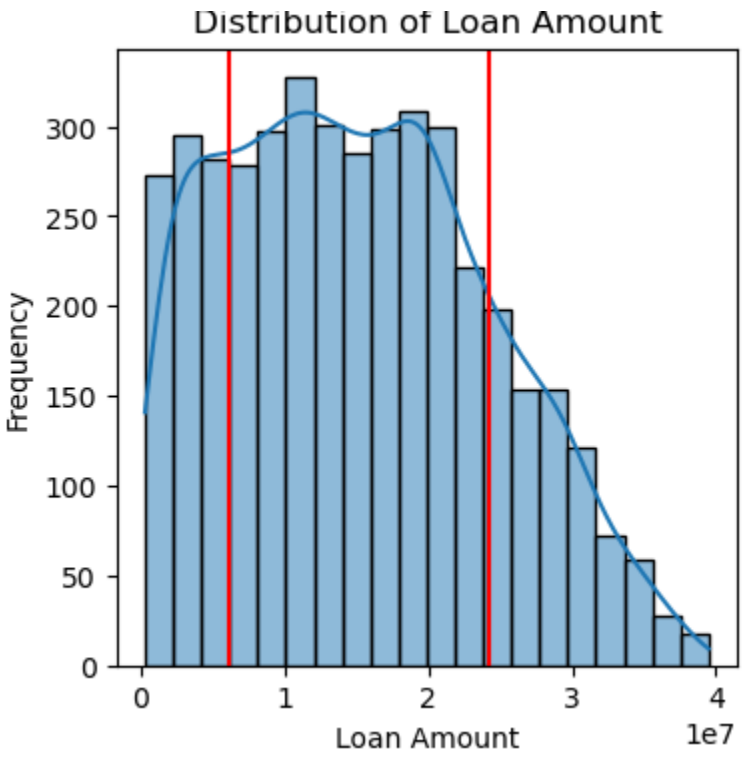
سوال ۱

قسمت a

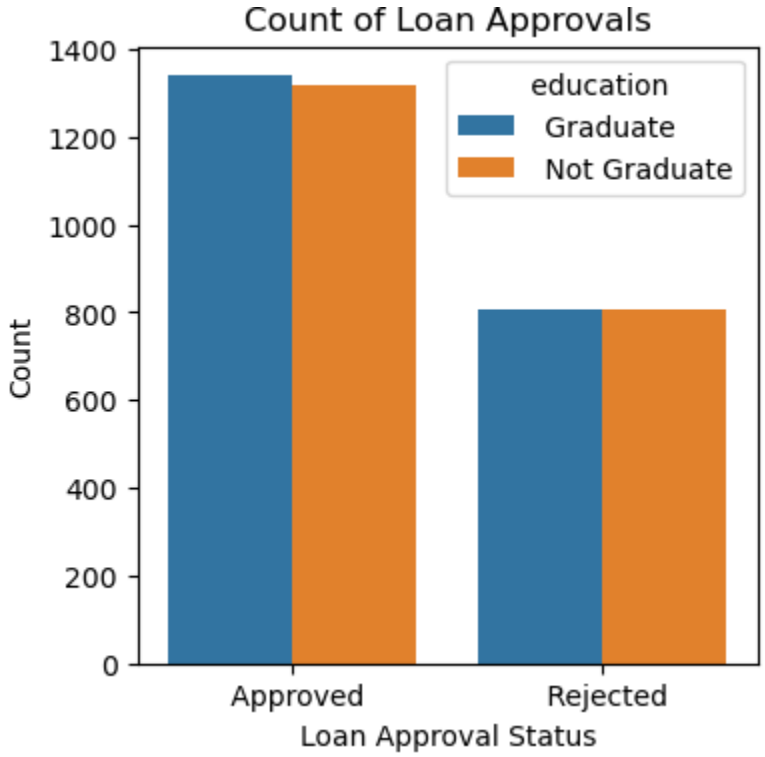
در این قسمت، داده‌های مربوط به شرایط دریافت وام از یک فایل **CSV** با متد **read_csv** بارگیری می‌شود و یک نمونه از ۱۰ مورد تصادفی از داده‌ها با متد **sample** چاپ می‌شود.

در این قسمت همچنین، تعدادی نمودار نشان داده شده است که تفسیرآنها در ادامه آمده است. همچنین نمودار های توزیع فراوانی هر ویژگی نمایش داده شده‌اند.

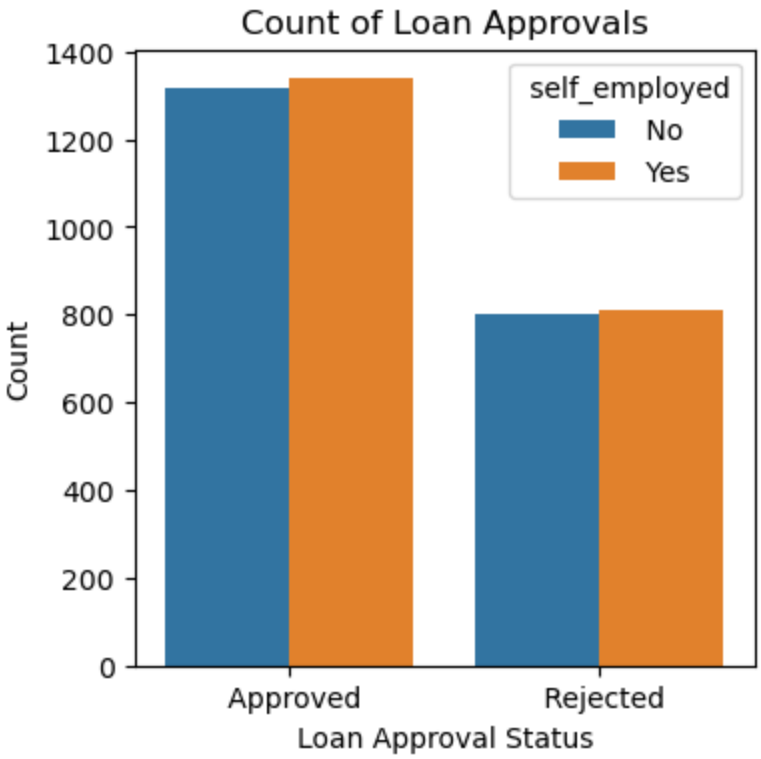
- نمودار **Distribution of Loan Amount** فراوانی مبلغ وام های دریافتی را نشان میدهد که ۶۸ درصد آنها در بازه بین ۶ میلیون تا ۲۴ میلیون بوده است. (البته این داده ها به طور مقیاس بندی شده نشان داده شده اند.) خطوط عمودی قرمز نشان دهنده فاصله از میانگین به اندازه یک انحراف معیار است که بیشتر داده ها در این بازه بوده است



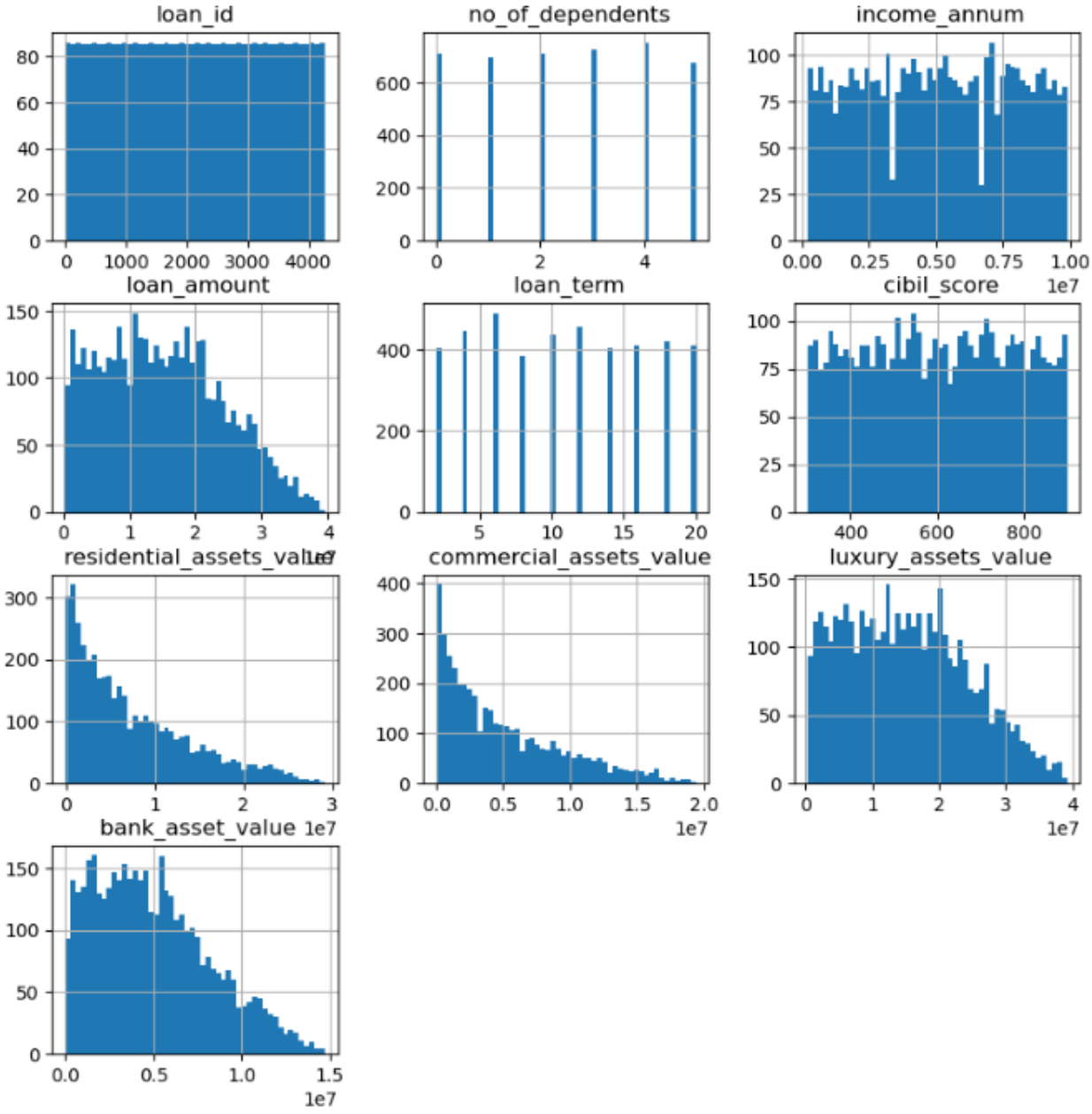
- نمودار زیر نشان می‌دهد که چه تعداد از وام ها تایید یا رد شده است همچنین تاثیر ویژگی تحصیلات با دو رنگ متفاوت روی این نمودار نشان داده شده است که نشان میدهد این عامل با تایید یا رد آن همبستگی نداشته و کمکی در پیش بینی تایید شدن یک وام نمی‌کند می توان در مهندسی ویژگی آن را حذف کرد.



- نمودار نشان میدهد که ویژگی **self_employed** تاثیری در تایید یا رد شدن آن نداشته و این عامل در هر حالت تایید یا رد شدن وام به یک میزان حضورداشته است. بنابراین می‌توان این ویژگی را در مهندسی ویژگی حذف کرد چون با ویژگی هدف همبستگی قابل ملموسی ندارد.

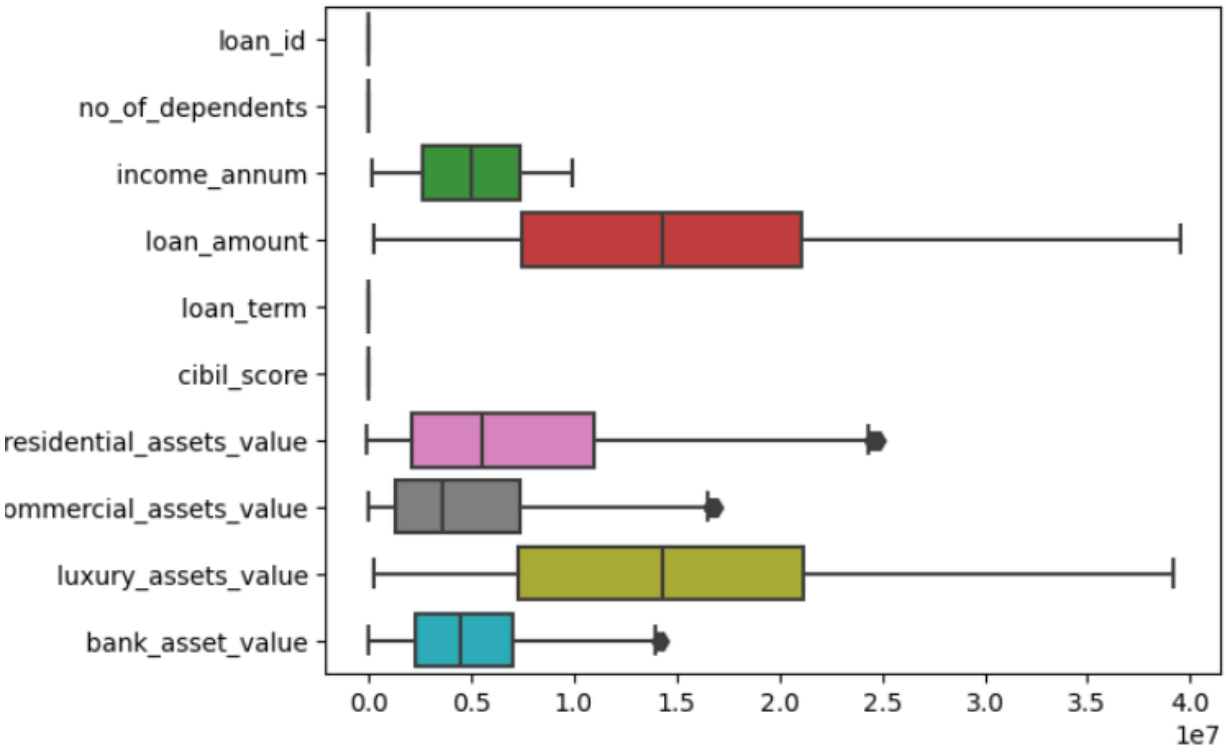


در ادامه نمودارهای بیشتری از ویژگی ها نشان داده شده است.


loan_id
no_of_dependents
income_annum
loan_amount
loan_term
cibil_score
residential_assets_value
commercial_assets_value
luxury_assets_value
bank_asset_value
1e7

قسمت b

در این قسمت پاکسازی داده ها انجام میشود. با استفاده از متد **describe** اطلاعات آماری مربوط به هر ویژگی نشان داده شده است. با استفاده از متد**shape** و **info** معلوم میشود دیتاست دارای ۴۲۶۹ قلم داده است و داده ناموجود ندارد. همچنین با متد **duplicate** معلوم میشود دیتاست دارای داده تکراری نیست. با استفاده از نمودار **box plot** وضعیت ویژگی های داری دادگان پرت نشان داده شده است یک تابع برای تشخیص بازه پرت نبودن داده ها به نام **outlier_detect** نوشته شده است که بر اساس متد **IQR** عمل میکند با استفاده از این تابع میتوان دیتای پرت را فیلتر کرد. در ادامه نمودار **box plot** دیتای فیلتر شده که داده پرت ندارد نشان داده شده است.



قسمت c

در این قسمت ویژگی های نامفید مانند **'loan_id', ' self_employed', ' education'** از دیتاست حذف شده اند تا مدل یادگیری بهتر عمل کند.

با استفاد از ماژول های **OneHotEncoder** و **StandardScaler** دادگان کیفی به کمی تبدیل شده است. همچنین دادگان کمی به بازه بین ۰ تا ۱ مقیاس دهی شده اند.

قسمت d

با استفاده از **train_test_split** داده ها به دو گروه ازمایشی و اموزشی با نسبت ۲۰ به ۸۰ تقسیم شده اند.

قسمت e

یک مدل درخت تصمیم‌گیری با پارامتر های پیش فرض بر روی داده آموزشی اموزش دیده شداست و کارایی آن با معیارهای دقت، صحت، بازخوانی و امتیاز **f1** گزارش شده است.

```
Accuracy: 0.974
Precision: 0.951
Recall: 0.981
F1 Score: 0.966
```

قسمت f

هدف کلی این قسمت، یافتن بهترین مجموعه مقادیر هایپرپارامتر برای مدل درخت تصمیم با استفاده از جستجوی گریدی و اعتبارسنجی متقابل است. این مدل برای یک وظیفه دسته‌بندی استفاده می‌شود، جایی که باید نمونه‌ها را به یک یا چند دسته اختصاص داد. کارهای انجام شده در این قسمت در ادامه آمده است.

دیکشنری **param_grid** مقادیر مختلفی را که برای هر هایپرپارامتر مدل **DecisionTreeClassifier**امتحان می‌شود، مشخص می‌کند.

هایپرپارامترها عبارتند از :

criterion: معیار تقسیم برای ساخت گره‌های درخت

entropy یا **gini**

max_depth: حداکثر عمق درخت (هیچ، ۱۰، ۲۰، ۳۰، ۴۰، ۵۰)

min_samples_split: **(**۲، ۵، ۱۰**)** حداقل تعداد نمونه برای تقسیم یک گره

min_samples_leaf: **(**۱، ۲، ۴**)** حداقل تعداد نمونه در یک گره برگ

max_features: **(** ۳، ۵، ۷، ۹**)** حداکثر تعداد ویژگی‌هایی که در هر تقسیم در نظر گرفته می‌شوند

در ادامه با استفاده از جست وجوی گریدی، بهترین مجموعه مقادیر هایپرپارامتر را که بر اساس معیارهای امتیازدهی تعریف شده اند، پیدا می‌شوند. بهترین مقادیر برای این پارامترها در ادامه نشان داده شده است.

```
{'criterion': 'entropy',
 'max_depth': 30,
 'max_features': 9,
 'min_samples_leaf': 4,
 'min_samples_split': 10}
```

در مرحله بعد یک مدل درخت تصمیم با استفاده از بهترین مقادیر هایپرپارامتر آموزش دیده شده است و کارایی آن با معیارهای ارزیابی در زیر گزارش شده است.

اموزش دیده با بهترین هایپرپارامتر ها

```
Accuracy: 0.976
Precision: 0.960
Recall: 0.978
F1 Score: 0.969
```

آموزش دیده با مقادیر پیش فرض

```
Accuracy: 0.974
Precision: 0.951
Recall: 0.981
F1 Score: 0.966
```

قسمت g

نمودار درخت تصمیم با استفاده از کتابخانه **sklearn** و متد **plot_tree** رسم شده است.

از این درخت می‌توان اطلاعات مهمی از نحوه تصمیم گیری این مدل بدست آورد. ساختار درخت تصمیم به‌گونه‌ای است که هر نود درخت نشان‌دهنده‌ی عمل دسته بندی بر اساس یک ویژگی است. در هر نود اطلاعاتی مرتبط با دسته‌بندی انجام‌شده وجود دارد. در هرنود بر اساس یک شرط نسبت به یک ویژگی خاص داده ها دسته‌بندی می‌شوند. این شرط فقط در نودهای غیربرگ به داده ها اعمال می‌شود. دو معیار برای اندازه گیری خالص بودن دسته‌بندی به نام های **gini** و **entropy** وجود دارد که نشان می‌دهد توزیع دسته‌بندی انجام‌شده نسبت به کلاس ها چه مقدار بوده است. هر چقدر مقدار این معیارها برای یک نود خاص کمتر باشد نشان دهنده این است که داده ها به یک کلاس خاص دسته‌بندی شده‌اند و در ادامه تقسیم بندی‌های کمتری از فرزندان آن نود شاهد خواهیم بود. مشخصه‌ی بعدی در یک نود، **sample** یا تعداد داده‌هایی است که شرط آن نود به آن داده‌ها اعمال خواهد شد. مشخصه بعدی **value** است که نشان می‌دهد به هر کدام از دسته ها چه تعداد داده نگاشت شده‌اند. مشخصه اخر **calss** نام دارد که نشان می‌دهد داده به کدام کلاس دسته بندی شده است.

قسمت h

درخت تصمیم مدلی است که فرضیات محدودی نسبت به داده آموزشی در نظرمی‌گیرد. بنابراین آزادی عمل بیشتری را دارد تا ساختار داده آموزشی را یاد بگیرد بنابراین بیشتر در معرض بیش‌برازش یا **overfitting** است. برای جلوگیری از آن، می‌توان با تنظیم مقادیر هایپرپارامترها

درخت تصمیم را **prune** کرد. در این قسمت یک مدل درخت تصمیم آموزش داده شده اما ساختار درخت با تنظیم هایپرپارامترهایی مانند **max_depth** ، **min_samples_split**، **max_features**، **criterion** هرس یا محدود شده است. مقادیر این پارامتر ها با استفاده از نتایج بدست امده از قسمت **f** که بهترین مقادیر برای هایپرپارامتر ها را بدست آورده، تنظیم شده است. برای مثال **max_depth** یا بیشترین عمق درخت به ۵ محدود شده است. بیشترین ویژگی های مورد استفاده برای تصمیم گیری در هر نود به میزان ۹ محدود شده است. نتایج ارزیابی کارایی این مدل در مقایسه با حالت پیش‌فرض آن در زیر آمده است.

بعد از هرس	حالت پیش فرض
Accuracy: 0.945 Precision: 0.890 Recall: 0.975 F1 Score: 0.930	Accuracy: 0.975 Precision: 0.954 Recall: 0.981 F1 Score: 0.967

با هرس ساختار درخت تصمیم شاهد کاهش کارایی مدل در همه شاخص‌های ارزیابی هستیم.

قسمت i

در این قسمت نمودار درخت تصمیم برای مدل هرس شده نشان داده شده است. تعداد سطوح درخت کاهش یافته و ساختار آن نسبت به درخت قبلی متعادل شده است. درخت جدید امکان بیش‌برازش کمی دارد و قابلیت تعمیم پذیری بالاتری دارد و می‌تواند در داده‌های جدید بهتر عمل کند.

سوال ۲

قسمت a

در این قسمت، داده‌های مربوط به دیتاست قارچ ها از یک فایل **CSV** با متد **read_csv**بارگیری می‌شود و یک نمونه از ۱۰ مورد تصادفی از داده‌ها با متد **sample** چاپ می‌شود.

قسمت b

پیش پردازش داده ها

داده تکراری

با استفاده از متد **duplicated** معلوم میشود دیتاست دارای داده تکراری نیست.

داده ناموجود

اطلاعات مربوط به ویژگی های آن با استفاده از متدهای **describe** و **info** نشان داده شده است. با استفاده از متد های **isnull** و **duplicated** معلوم می‌شود دیتاست دارای دیتای ناموجود یا تکراری نیست.

فرایند **encoding** بر روی داده های غیرعددی اعمال شده است تا این داده ها به فرمت عددی تبدیل شوند، سپس فرایند **scaling** بر روی این داده‌ها اعمال شده است.

قسمت c

با استفاده از متد **train_test_split** دیتاست به مجموعه آموزشی و آزمایشی با اندازه ۷۰ به ۳۰ تقسیم شده است.

قسمت d

در این قسمت یک مدل **logistic regression** با آموزش بر روی داده‌های اموزشی ایجاد شده است و معیارهای ارزیابی آن گزارش شده است.

```
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1-score: 1.0
```

نتایج فوق نشان می‌دهد مدل دچار بیش‌برازش شده است.

قسمت e

در این قسمت، برای مقابله با بیش‌برازش از متد اعتبارسنجی متقابل **(cross validation)** استفاده شده است. در یک نوع خاص از این متد به نام **k-fold-cross validation** داده آموزشی به **k** زیرمجموعه تقسیم می‌شود که یکی از آنها به عنوان داده آموزشی و **k-1** تای آنها برای آموزش مدل استفاده می‌شود. این فرایند به تعداد هایپرپارامتر **cv** تکرار می‌شود. بدین صورت از بیش‌برازش مدل جلوگیری یا تا حدی با آن مقابله می‌شود.

در این قسمت طبق خواسته سوال مقدار **cv** برابر با ۱۰ در نظر گرفته شده است. نتایج ارزیابی کارایی مدل در هر بار تکرار این فرایند و میانگین آن آمده است.

```
Cross-validation scores: [0.68511685 1.         1.         1.         0.99753695 1.
 1.         1.         0.93103448 1.        ]
Mean Accuracy: 0.961
```

قسمت f

در قسمت **d** مدل با مشکل بیش‌برازش روبرو است. در قسمت **f** از متد **cross validation** برای مقابله با آن استفاده شده است. نتایج آن برای ارزیابی مدل در هر بار یادگیری با داده‌های آموزشی مختلف در بالا نشان داده شده است. میانگین امتیاز مدل ۹۶ درصد می‌باشد که نشان می‌دهد تا حدودی با بیش‌برازش مقابله شده است.

سوال ۳

قسمت a

بعد از فراخوانی کتابخانه های موردنیاز، **(pandas, ...)** دیتاست با استفاده از متد **read_csv()** در یک دیتافریم به نام **data** ذخیره می‌شود و با استفاده از متد **sample** ده داده تصادفی از آن نمایش داده شده است.

قسمت b

در این مرحله پیش‌پردازش داده‌ها انجام می‌شود. با استفاده از متد های مختلف کتابخانه **NLTK** که یک کتابخانه برای پردازش داده های زبان طبیعی است، می توان نرمالسازی متن را انجام داد. در اولین مرحله از پیش‌پردازش، جداسازی کلمات (توکن) متن هر قلم داده با استفاده از ماژول **word_tokenize** از کتابخانه **NLTK** انجام شده و به همان ستون دیتا فریم اعمال شده است. در مرحله بعد، کلمات توقف که کلمات بسیار رایج هر زبانی هستند، با استفاده از ماژول **stop_words**، ازمتن دیتاست جداسازی شده است. این فرایند با تابع **remove_stopwords** انجام شده است. در مرحله بعدی از پیش‌پردازش، **lemmatization** بر روی دیتاست انجام شده است. در این فرایند هر کلمه یا توکن به ریشه سازنده آن تبدیل می‌شود. در مرحله **stemming** انجام شده است که عملکرد آن بسیار شبیه به **lemmatization** است و برای کاهش تعداد کلمات و ساده سازی آن در یک متن انجام میشود. در نهایت، کلمات فیلترشده و نرمال شده با استفاده از متد **join** و کاراکتر فاصله به هم متصل شده اند.

قسمت c

برای ساخت و آموزش یک مدل بر روی داده های متنی نیاز است که این داده ها به فرمت عددی تبدیل شوند تا کار پردازش مدل های یادگیری ماشین روی این داده‌ها بتواند انجام شود. رویکرد رایج بردازسازی از کلمات متن داده‌شده است. در مدل **CountVectorizer** تعداد رخداد هر کلمه(بعد از انجام مراحل پیش‌پردازش) به صورت بردار مدل می‌شود به‌طوریکه یک قلم داده برداری است که مولفه های آن تعداد تکرار آن کلمات در متن آن قلم داده است. در این مدل نهایتا یک ماتریس از ارتباط بین قلم داده ها و تکرار کلمات شکل می‌گیرد.

بر اساس نتایج مدل **CountVetorizer** انتخاب ویژگی امکان پذیر است به طوریکه می‌توان کلمات با تکرار بالا را به عنوان ویژگی انتخاب کرد.

در مدل **tf-idf Vectorizser** برای هر کلمه یک وزن محاسبه می‌شود که وزن بالاتر نشان‌دهنده اهمیت بالاتر است. هر چه تعداد تکرار یک کلمه در یک قلم داده خاص بیشتر و تکرار آن در کل دیتاست کمتر باشد، اهمیت (وزن) آن بیشتر است. برای انتخاب ویژگی با این روش، باید کلمات دارای با وزن بالا انتخاب شوند.

در نهایت از هردو روش برای محاسبه ماتریس ویژگی استفاده شده است.

قسمت d

در این قسمت داده ها به دو مجموعه آموزشی و آزمایشی تقسیم شده اند. از ۵۰۰۰۰ قلم داده ۴۹۹۹۹ داده در این فاز استفاده شده اند و یک قلم داده برای قسمت **f** نگه داشته شده است.

قسمت e

در این قسمت از یک الگوریتم که بر پایه قضیه بیز می‌باشد برای ساخت یک مدل دسته‌بند استفاده شده است. این مدل با هر دو روش انتخاب ویژگی **(CountVectorizer- tf-idf Vectorizer)** ساخته و آموزش دیده شده است. عملکرد مدل با معیار های ارزیابی در زیر برای هر دو روش انتهاب ویژگی آمده است.

```
Accuracy for TF_IDF vectorizer method: 0.8642
              precision    recall  f1-score   support

           0       0.85      0.88      0.87      4969
           1       0.88      0.85      0.86      5031

    accuracy                           0.86     10000
   macro avg       0.86      0.86      0.86     10000
weighted avg       0.86      0.86      0.86     10000
```

```
Accuracy for Count vectorizer method: 0.8567
              precision    recall  f1-score   support

           0       0.84      0.88      0.86      4969
           1       0.87      0.84      0.85      5031

    accuracy                           0.86     10000
   macro avg       0.86      0.86      0.86     10000
weighted avg       0.86      0.86      0.86     10000
```

صحت مدل **naïveBayes** در حدود ۸۶ درصد می‌باشد.

(صحت مدل ساخته شده با ویژگی های انتخاب شده با **tf-idf** بهتر از **countVectorization** است.)

مدل های ساخته شده با متد **joblib.dump** ذخیره شده‌اند.

قسمت f

در این قسمت مدل های ذخیره شده در قسمت قبل برای پیش بینی یک قلم داده جدید (قلم داده اخر از دیتاست که در فاز آموزش و آزمایش توسط مدل دیده نشده است) استفاده شده اند و به درستی مقدار وضعیت **lable** را پیش بینی کردند.

سوال ۴

قسمت a

در مرحله اول کتابخانه های لازم برای کار با این دیتاست فراخوانی می‌شوند. در مرحله بعدی، با استفاده از کتابخانه **urlib.request** دیتاست موردنظر دانلود شده و با استفاده از **np.asarray** به فرمت ارایه‌ای و عددی تبدیل می‌شود. فرمت ارایه‌ای موردنظر را می‌توان با متدهای کتابخانه **OpenCV** مانند **imdecode** به فرمت قابل نمایش تبدیل کرد. برای پردازش آسان‌تر این دادها ها توسط مدل یادگیری ماشین، با استفاده از متد **cvtColor** داده‌ها به فرمت خاکستری (بدون بعد سوم) تبدیل شده‌اند.

این دیتاست دارای نمونه‌های مختلف از اعداد ۰ تا ۹ می‌باشد که برای آموزش یک مدل بر اساس این نمونه‌داده‌ها، باید هر نمونه از این دیتاست استخراج شده و هر پیکسل از تصویر این نمونه‌دادها یک ویژگی خواهد بود. با توجه به بعد تصویر دیتاست عرض و ارتفاع هر نمونه‌داده ۲۰ پیکسل بدست می‌آید.

بعد دیتاست: ۳ * ۲۰۰۰ * ۱۰۰۰

از هر نمونه داده ۵ سطر در دیتاست وجود دارد و همچنین ده نوع نمونه داریم (اعداد ۰ تا ۹) بنابراین داریم:

۱۰۰۰/ ۱۰ =۱۰۰

۱۰۰/۵ = ۲۰

عرض و ارتفاع هر پیکسل از تصاویر برابر ۲۰ می‌باشد.

در چند گام نحوه جداسازی یک نمونه داده از دیتاست موردنظر نشان داده شده است و لیبل‌گذاری هر نمونه‌داده نیز در حین استخراج هر نمونه داده انجام شده است. همه نمونه داده ها بعد ازاستخراج از دیتاست در یک لیست ذخیره شده‌اند. با استفاده از متد **DataFrame**یک دیتافریم با استفاده از لیست ساخته شده که هر ستون آن نشان‌دهنده یک فیچر است. از آنجایی که هر نمونه داده (هر تصویر یک عدد) به اندازه ۲۰ * ۲۰ است، تعداد ۴۰۰ ستون یا فیچر در این دیتافریم وجود دارد.

قسمت b

در این قسمت دیتاست با استفاده از متد **train_test_split** به دو مجموعه اموزشی و تست با نسبت های ۸۰ به ۲۰ درصد تقسیم می‌شود.

قسمت c

در این قسمت نحوه تبدیل یک دسته بند باینری به یک دسته بند چند کلاسه توضیح داده می‌شود. رویکردهای مختلفی برای انجام دسته‌بندی چندکلاسه با یک دسته‌بند باینری وجود دارد. یک روش برای انجام این کار **one-versus-the-rest (OvR)** نام دارد که در آن دسته بندی داده‌ها به **N** کلاس مختلف اینگونه می‌تواند باشد که تعداد **N** دسته‌بند برای (هرکدام برای یک دسته خاص) ساخته شود. در این روش برای تشخیص دسته یک داده، هر کدام از این دسته بند ها دسته بندی را انجام می‌دهند و امتیاز بالاتر از این دسته ها از نظر معیار کارایی به عنوان دسته آن نمونه داده انتخاب می‌شود.

روش دیگر آموزش دسته‌بند برای هر جفت از داده ها می‌باشد که به روش **one-versus-one (OvO)** معروف است. (برای مثال یک دسته‌بند برای ۰ و ۱، یکی برای ۰ و ۲ و ...) در این روش، تعداد **N * N /2** دسته‌بند لازم است اگر **N** کلاس داشته باشیم. اصلی‌ترین مزیت **OvO** این است که هر دسته‌بند فقط نیاز به آموزش بخشی از مجموعه آموزشی دارد که برای دو کلاسی که باید از هم تمیز داده شود استفاده می‌شود. برخی از الگوریتم‌ها (مانند دسته‌بندهای ماشین بردار پشتیبان) با افزایش اندازه مجموعه آموزشی بهبود عملکرد خود را نشان نمی‌دهند. برای این الگوریتم‌ها **OvO** ترجیح داده می‌شود زیرا آموزش بسیاری از طبقه‌بندها بر روی مجموعه‌های آموزشی کوچک سریع‌تر است تا آموزش چندین طبقه‌بند بر روی مجموعه‌های آموزشی بزرگ. با این حال، برای اکثر الگوریتم‌های دسته‌بندی دودویی، **OvR** ترجیح داده می‌شود.

قسمت d

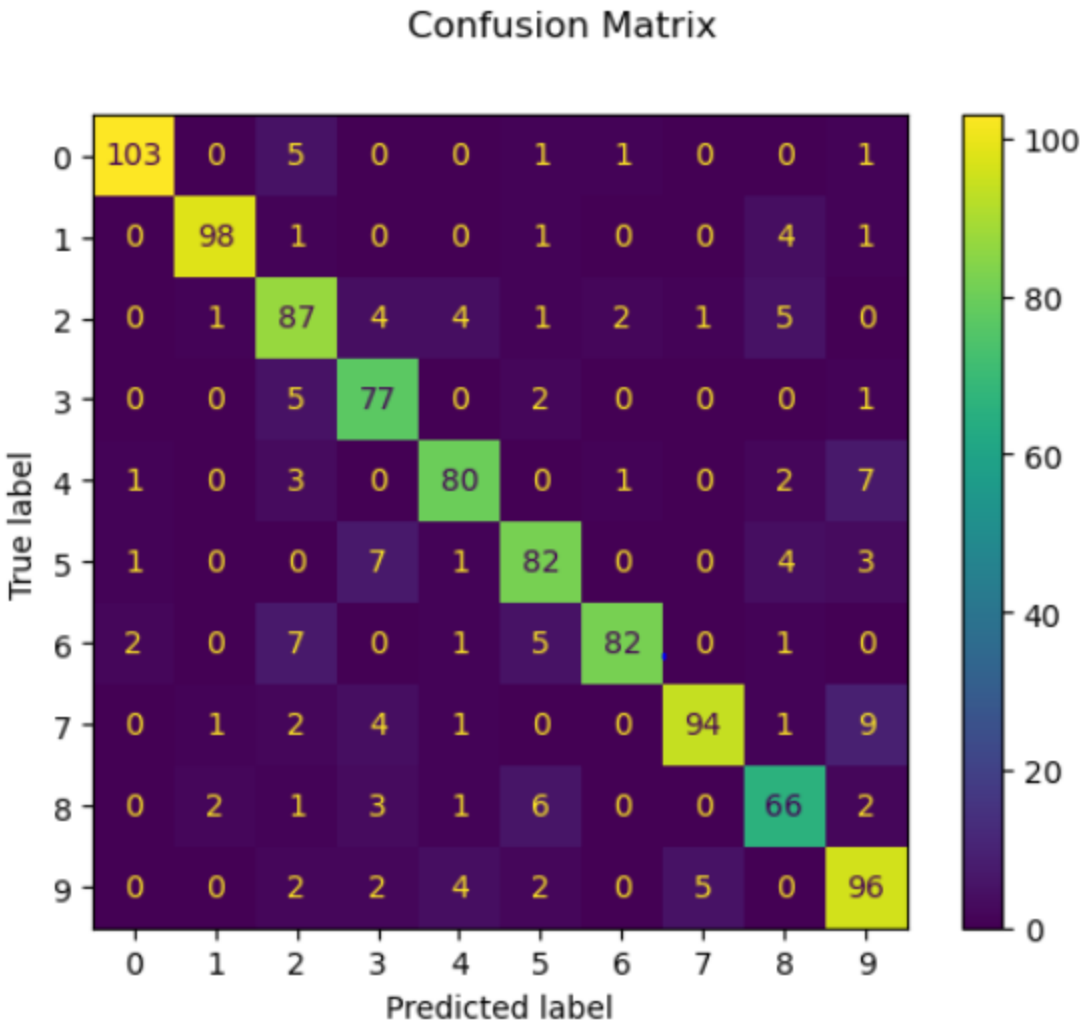
در این قسمت یک مدل **logistic regression** بر روی داده آموزشی آموزش داده شده است. با تنظیم پارامتر **multi_class** به **'multinomial'** این مدل دسته‌بندی چندکلاسه را می‌تواند انجام دهد.

قسمت e

Accuracy مدل برابر ۸۶ درصد بدست آمده است.

قسمت f

ماتریس درهم ریختگی (**confusion matrix**) برای مدل آموزش‌دیده نشان داده شده است.



تعداد دسته بندی های اشتباه برای هر کلاس از اعداد در این ماتریس قابل مشاهده است که در زیر به موارد اشاره شده است.

برای نمونه در سطر اول، ۱۰۳ مورد دسته‌بندی درستی برای کلاس ۰ انجام شده است در بقیه موارد تعداد ۵ بار به اشتباه کلاس ۲ تشخیص داده شده است با جمع مقادیر خارج از قطر اصلی می‌توان تعداد دسته‌بندی های اشتباه را برای هر کلاس تعیین کرد.

تعداد دسته بندی اشتباه برای کلاس صفر : ۸

تعداد دسته بندی اشتباه برای کلاس یک : ۷

تعداد دسته بندی اشتباه برای کلاس دو : ۱۸

تعداد دسته بندی اشتباه برای کلاس سه : ۸

تعداد دسته بندی اشتباه برای کلاس چهار : ۱۴

تعداد دسته بندی اشتباه برای کلاس پنج : ۱۶

تعداد دسته بندی اشتباه برای کلاس شش : ۱۶

تعداد دسته بندی اشتباه برای کلاس هفت : ۱۶

تعداد دسته بندی اشتباه برای کلاس هشت: ۱۵

تعداد دسته بندی اشتباه برای کلاس نه: ۱۵

بیشترین تعداد دسته‌بندی اشتباه برای کلاس چهار بوده است.