به نام خدا

مسئله : پیاده‌سازی سیستم تشخیص نفوذ براساس مقاله با عنوان **A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms** منتشر شده در سال ۲۰۱۵ است.

مشخصات پروژه:

- زبان پایتون، نسخه 3.8.2
- کتابخانه numpy
- کتابخانه matplotlib
- کتابخانه scikit-learn

توضیحات پروژه:

پیاده‌سازی پروژه شامل شش مرحله به شرح زیر است:

- مرحله یک – خواندن داده

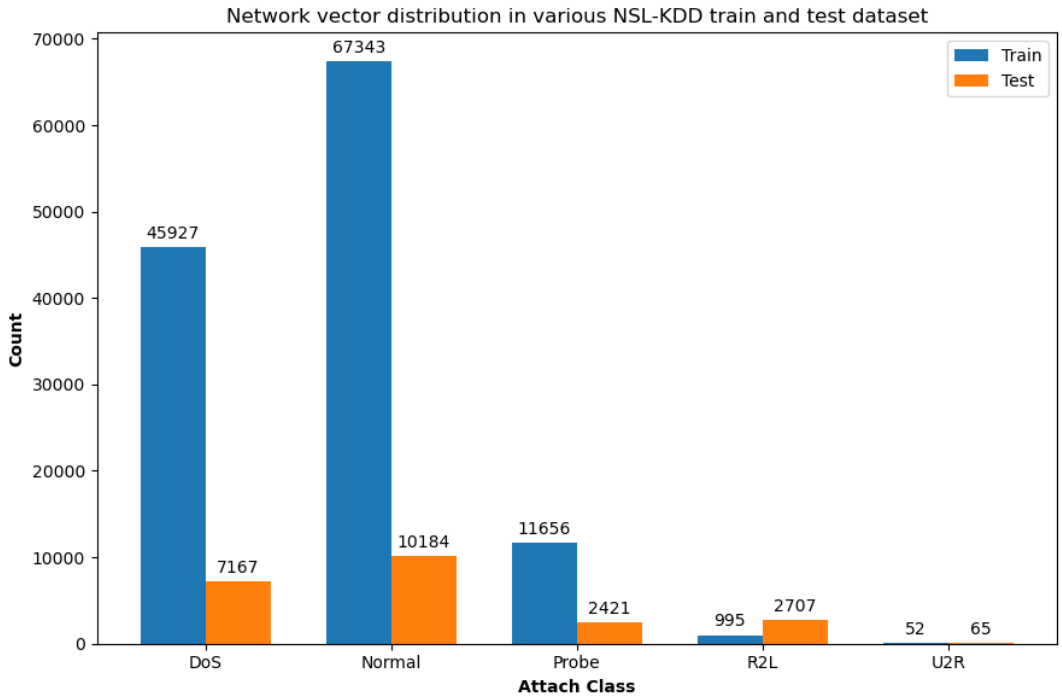
دو فایل KDDTrain+.txt ، KDDTest+.txt شامل مجموعه داده آموزشی و تست در پوشه data در مسیر پروژه قرار داده شده است. برای خواندن داده یک تابع با نام load_NSL_KDD در فایل datasets.py تعریف کردیم که تابع به عنوان ورودی مسیر فایل را می‌گیرد و خروجی نمونه های داده ، برچسب هر نمونه و نام ستون‌های نمونه را برمی‌گرداند. داخل تابع مسیر فایل چک می‌شود اگر فایلی با این مسیر موجود نبود خطا نمایش داده می‌شود وگرنه خط به خط از فایل خوانده و به وسیله کاما از هم مقادیر آن جدا می‌شود و داخل یک لیست ریخته می‌شود. در لیست یکی مانده به آخرین خانه مقدار برچسب است و بقیه خانه‌های لیست مقدار نمونه‌های ما را تشکیل می‌دهد. با توجه به مقاله برچسب‌ها بر پنج نوع (normal، Dos، Probe، R2L ، U2R) است که 4 مورد Dos ، Probe ، R2L و U2R نوع حملات را مشخص می‌کند و هر کدام از این نوع حملات خود نیز شامل زیر بخش‌های دیگری هستند. در دیتا، برچسب‌ها شامل Normal

و زیربخش‌های حملات است چون مسئله ما از نوع طبقه‌بندی با پنج کلاس است نیاز به این داریم که برای هر برچسب از نوع زیربخش، نوع اصلی آن زیر بخش یعنی یکی از چهار مورد حملات را جایگزین کنیم و این جایگزینی در پیاده‌سازی درنظرگرفته شده است.

- مرحله دو – آنالیز اکتشافی داده

برای این مرحله در پیاده‌سازی، فایل exploratory_analysis.py تعریف کردیم که نمودار ستونی توزیع نوع حملات برای داده آموزشی و تست ، نمودار ستونی توزیع پروتکل‌ها برحسب نوع حملات برای داده آموزشی و تست به صورت عکس در پوشه result ذخیره می‌کند.

نمودار ستونی توزیع نوع حملات

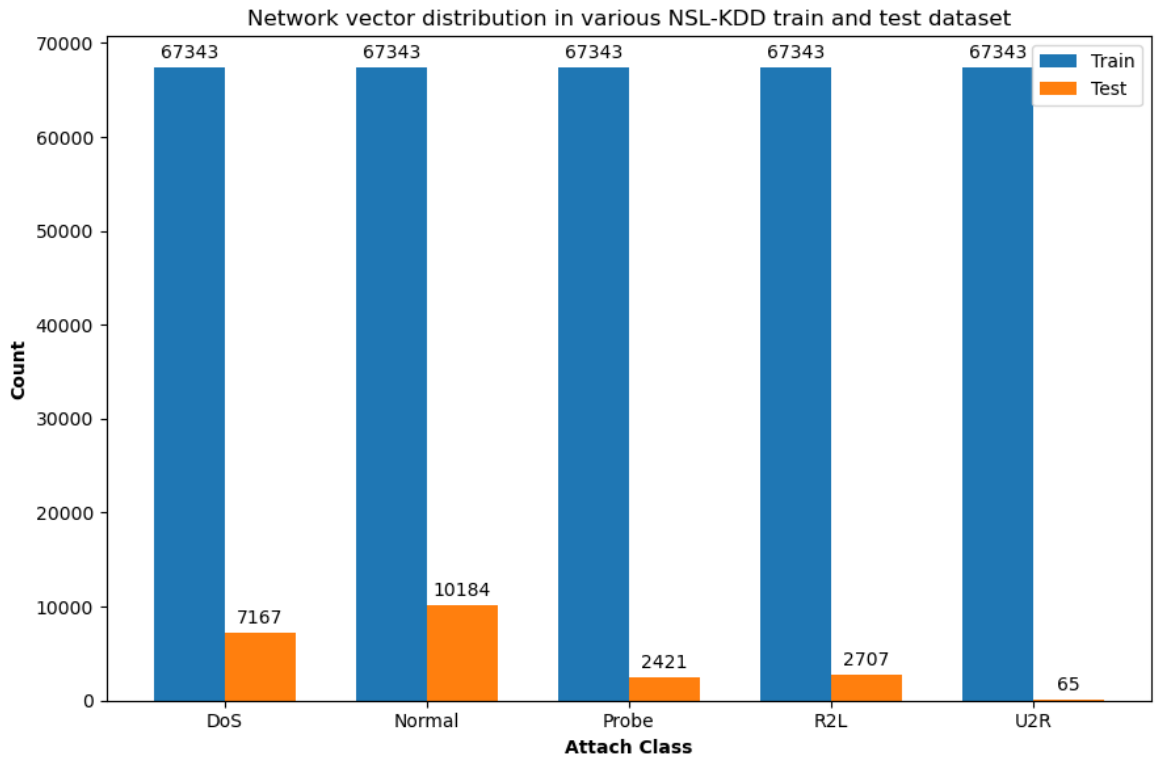


از نمودار بالا برداشت می‌شود که توزیع داده غیرمتعادل(imblanced) است و این بالانس نبودن داده در میزان دقت سیستم تاثیر خواهد گذاشت که در قسمت خروجی نتایج را مشاهد خواهید کرد.

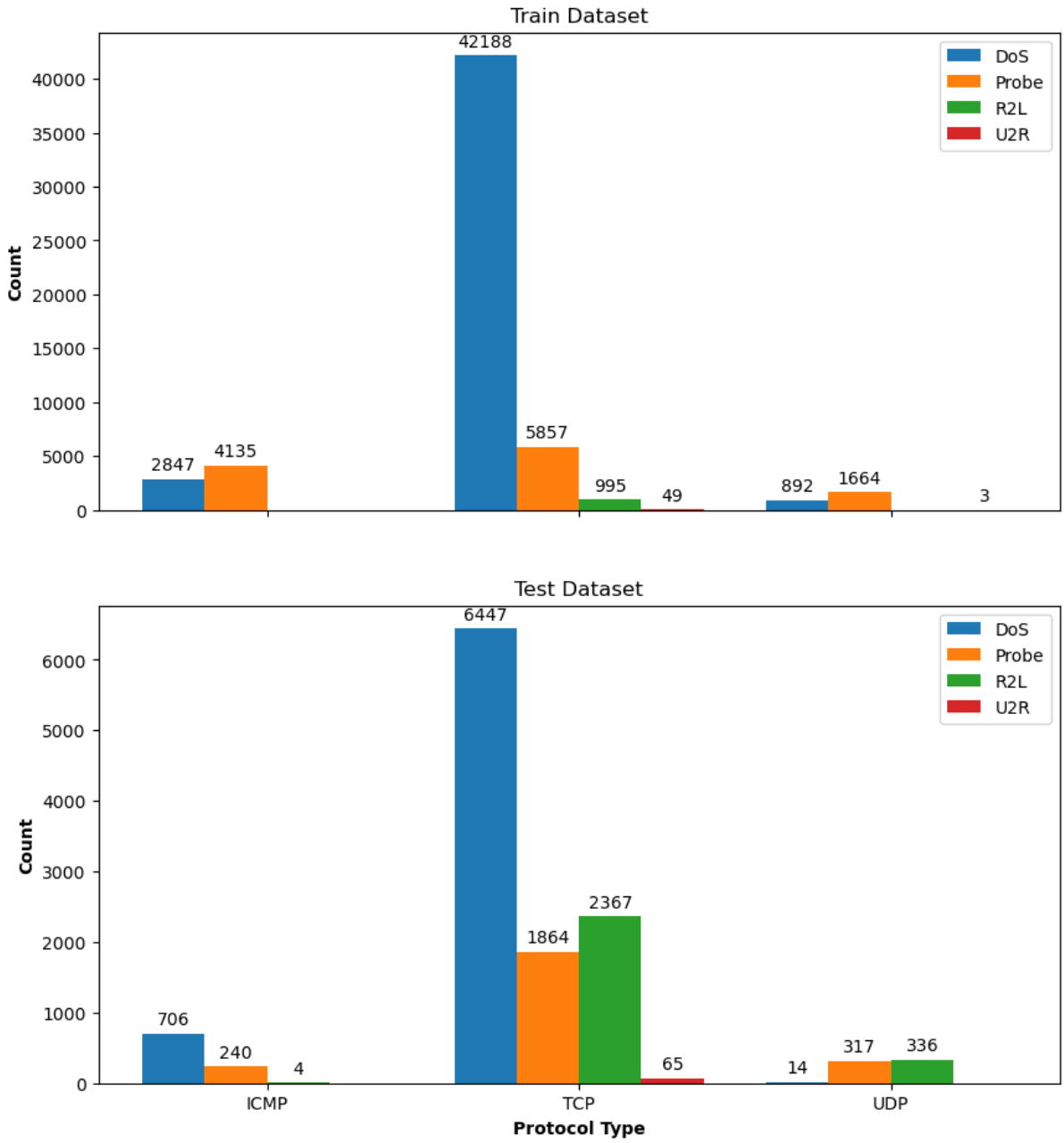
برای حل این مسئله با استفاده از روش SMOT حجم داده را افزایش دادیم تا توزیع داده بالانس شود. در پیاده‌سازی یک تابع با نام resample_data در فایل util تعریف کردم که ورودی داده و برچسب را می‌گیرد

و خروجی داده بالانس شده به همراه برچسب را برمی‌گرداند. داخل تابع فرکانس برچسب‌ها را به دست می‌آوریم برچسبی به بیشترین فرکانس را دارد جدا می‌کنیم و برای بقیه برچسب‌ها به تعداد بیشترین فرکانس از خود نمونه‌های متعلق به برچسب به صورت رندم نمونه تکراری اضافه می‌کنیم تا فرکانس آن برابر بیشترین فرکانس شود و داده بالانس شده را برمی‌گردانیم. این عملیات فقط برای داده آموزشی به دلیل آموزش مدل انجام می‌شود. نمودار ستونی داده بالانس شده را به صورت زیر است.

نمودار ستونی توزیع نوع حملات(بالانس شده)



نمودار ستونی توزیع پروتکل‌ها برحسب نوع حملات



- مرحله سه - پیش‌پردازش:

برای این مرحله دو تابع one_hot_encoding ، label_encoder و min_max_scaler را در فایل preprocessing.py تعریف کردیم.

تابع one_hot_encoding به عنوان ورودی نمونه‌های داده آموزشی و تست را دریافت می‌کند و ویژگی‌هایی از داده که به صورت گروه‌بندی(categorical) هستند را به برداری از صفر و یک تبدیل می‌کند تا برای مرحله بعد برای انجام عملیات ریاضی مشکلی نداشته باشیم. در دیتا سه ویژگی categorical وجود دارد، ویژگی پروتکل (protocol_type) ، خدمات(service) و Flag. دیتا ورودی تابع 42 ویژگی است با اعمال OneHotEncoder به 123 ویژگی می‌رسد.

ویژگی پروتکل: 3 مقدار ، ویژگی خدمات : 79 مقدار ، ویژگی flag: 11 مقدار

ویژگی‌های بدون categorical در دیتا : 42-3=39

مجموع ویژگی‌ها بعد از اعمال OneHotEncoder: 39+3+79+11=123

تابع label_encoder به عنوان ورودی برچسب‌های داده آموزشی و تست را دریافت می‌کند و خروجی برچسب‌ها به صورت عدد را برمی‌گرداند. با استفاده از روش labelEncoder به هر برچسب یک مقدار عددی انتساب داده می‌شود.

[‘Normal’ , ‘Dos’,’Probe’,’R2L’,’U2R’]

[0,1,2,3,4]

تابع min_max_scaler به عنوان ورودی نمونه را دریافت می‌کند و مقادیر هر ویژگی را از بازه اصلی به بازه[0,1] تغییر می‌دهد تا میزان تاثیر هر ویژگی در مدل یکسان باشد.

- مرحله چهار – استخراج ویژگی:

برای این مرحله تابع principal_component_analysis را در فایل feature_extraction.py تعریف کردیم. تابع ورودی داده آموزشی و تست را دریافت می‌کند و خروجی با استفاده از تابع PCA از کتابخانه Sklearn از داده به تعداد تعیین شده ویژگی استخراج می‌کند و داده با تعداد ویژگی جدید برمی‌گرداند.

- مرحله پنج – آموزش مدل:

در این مرحله تابع train_model در فایل run.py تعریف کردیم. این تابع ورودی داده آموزشی و برچسب‌های آن را می‌گیرد و خروجی مدل آموزش دیده را برمی‌گرداند. داخل تابع با توجه به مقاله سه مدل J48 یا همان Decision Tree C4.5 ، SVM و Naïve Bayes برای آموزش در نظر گرفته شده است. قطعا هر مدل نیاز به تنظیم کردن یک سری از پارامترها را دارد برای مقداردهی پارامترها دو حالت(دستی – خودکار) در نظر گرفتم. اگر مایل به تنظیم دستی پارامترها دارید در فایل config.py مدل مورد نظر را مقدار True می‌دهید و پارامترهای تعریف شده را در همان فایل مقداردهی کنید و اگر مایل به تنظیم خودکار پارامترها هستید گزینه gridsearch را در فایل config مقدار True می‌دهید و برای مدل تابع best_params_gridsearchcv فراخوانی می‌شود و بعد از محاسبه بهترین پارامتر برگشت داده می‌شود و مدل با پارامتر مورد نظر آموزش دیده می‌شود و به عنوان خروجی تابع برگشت داده می‌شود.

- مرحله شش – ارزیابی مدل:

برای این مرحله تابع evaluate در فایل run.py تعریف کردم. تابع ورودی داده آموزشی تست، برچسب آن، مدل آموزش دیده ، عنوان ماتریس درهم ریختگی عکس ، نام برچسب‌ها و مسیر ذخیره عکس ماتریس درهم ریختگی را دریافت می‌کند. داخل تابع به وسیله مدل برچسب داده تست را پیش‌بینی می‌کند. میزان دقت مدل را برای برچسب داده تست و برچسب پیش بینی شده محاسبه و نمایش می‌دهد و ماتریس درهم ریختگی را ایجاد و در مسیر مشخص شده ذخیره می‌کند.

برای راحتی کار فایل config.py تعریف کردم که نیازی به تغییر فایل‌های دیگر نباشد.

متغیرpath_train_NSL_KDD مسیر داده آموزشی

متغیر path_test_NSL_KDD مسیر داده تست

متغیر balance_data برای اینکه داده آموزشی بالانس شود یا نه اگر مایل به بالانس کردن داده هستید مقدار True را به آن انتساب دهید

متغیر normalize برای تغییر بازه اعداد داده به بازه [0,1] است.

متغیر max_feature تعداد ویژگی‌هایی که باید به وسیله pca استخراج شود(طبق مقاله 6 در نظر گرفته شده است)

متغیر model_J48 ، model_SVM ، model_NaiveBayes برای تعیین مدل برای آموزش در نظرگرفته شده است برای اجرا فقط به متغیر مورد نظر مقدار True انتساب دهید.

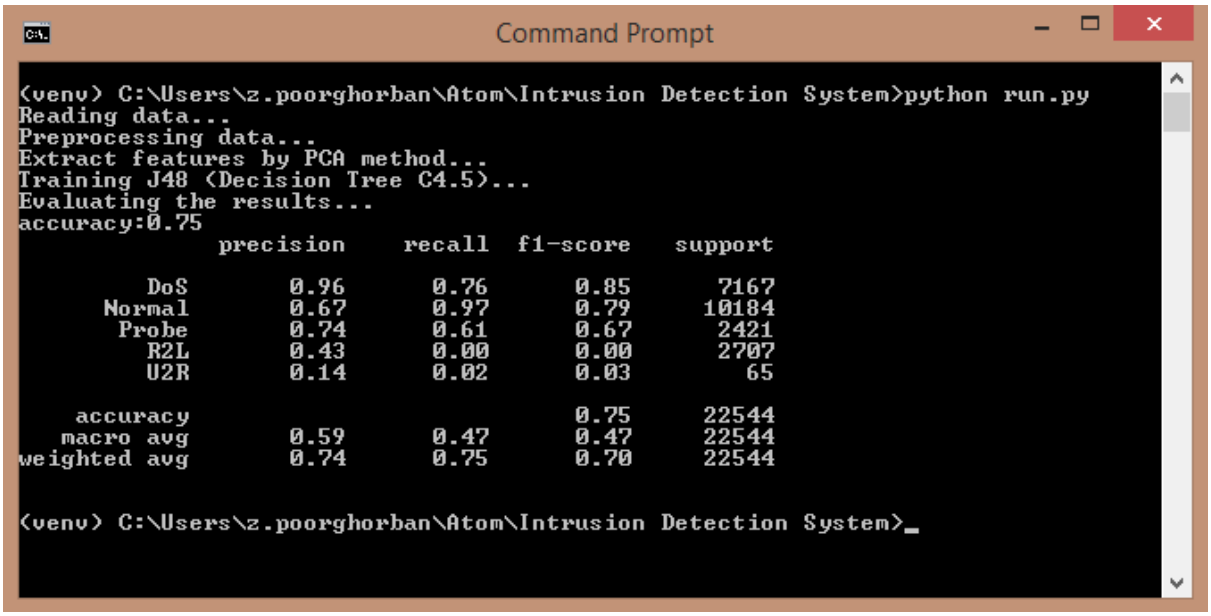
متغیرgridsearch برای تنظیم خودکار پارامترهای مدل در نظر گرفته شده است.

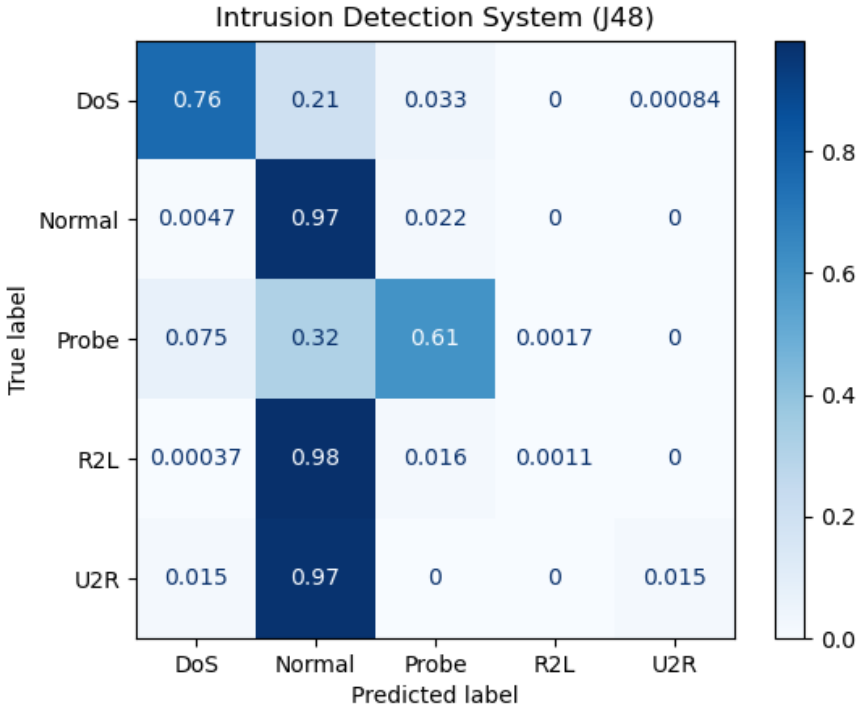
متغیر max_depth تنظیم دستی پارامتر مدل J48 است.

متغیر های kernel ، C ، gamma برای تنظیم دستی پارامترSVM است.

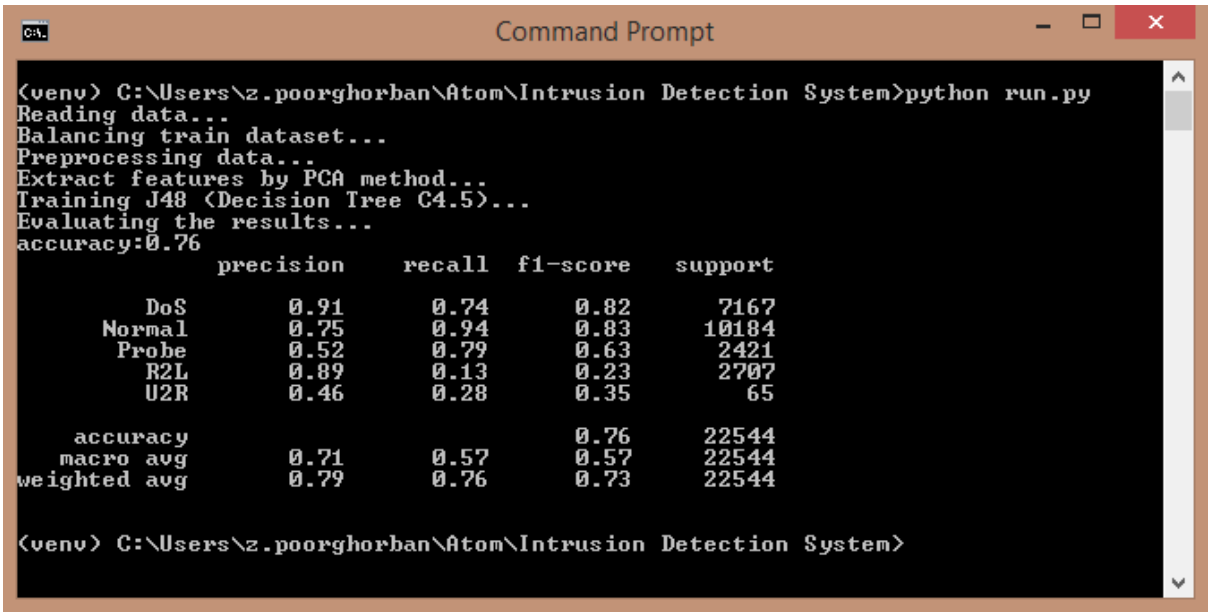
خروجی:

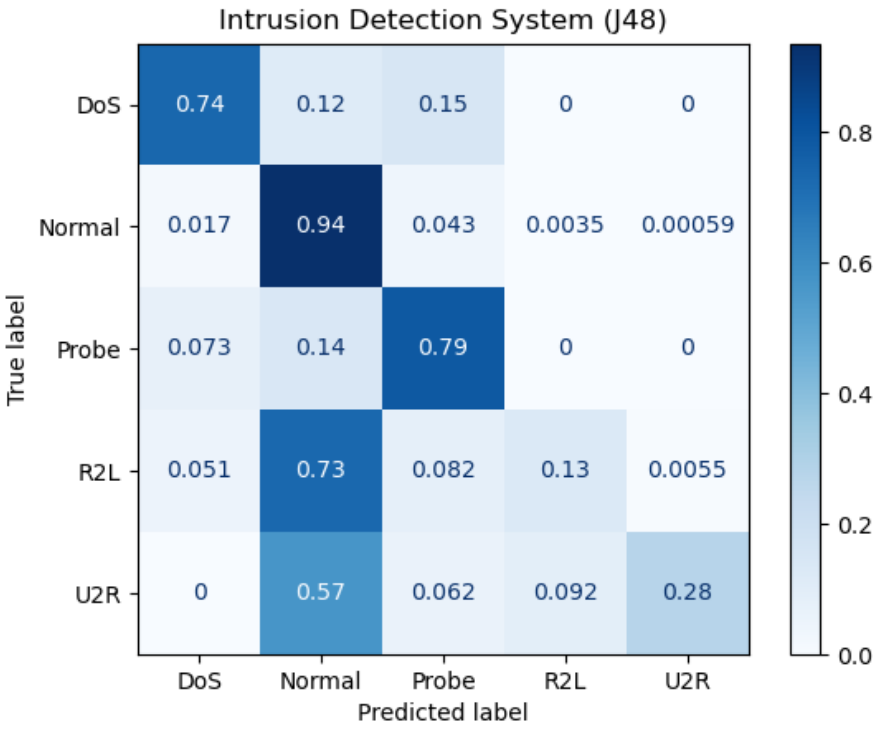
مدل J48 – داده غیرمتعادل(imbalance)



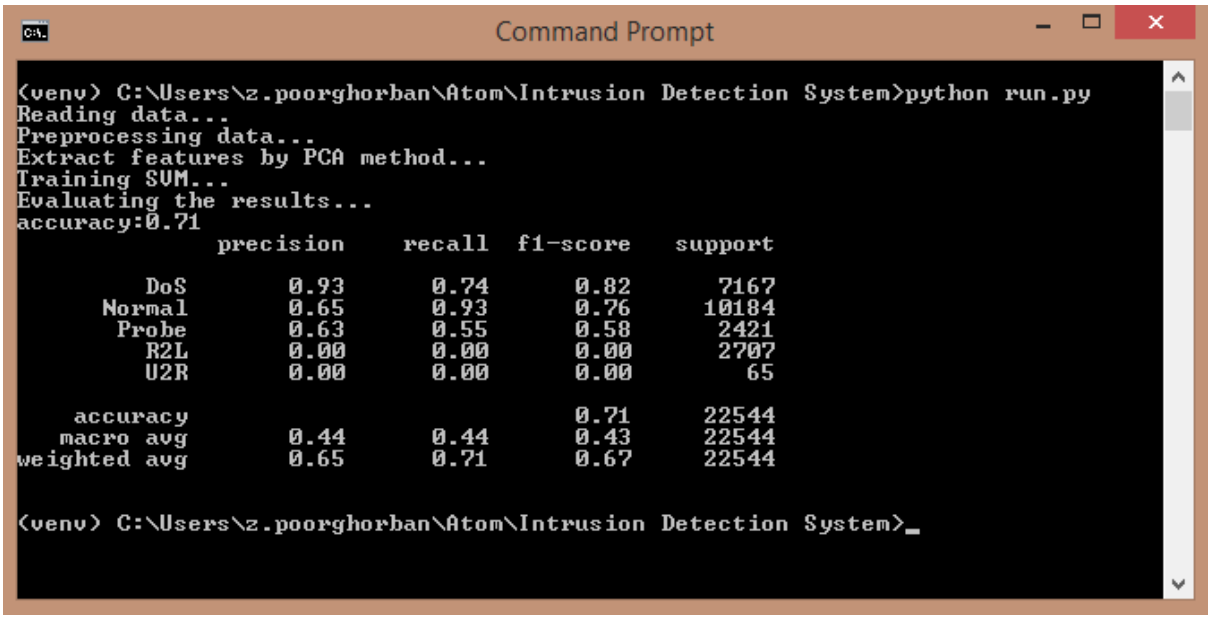


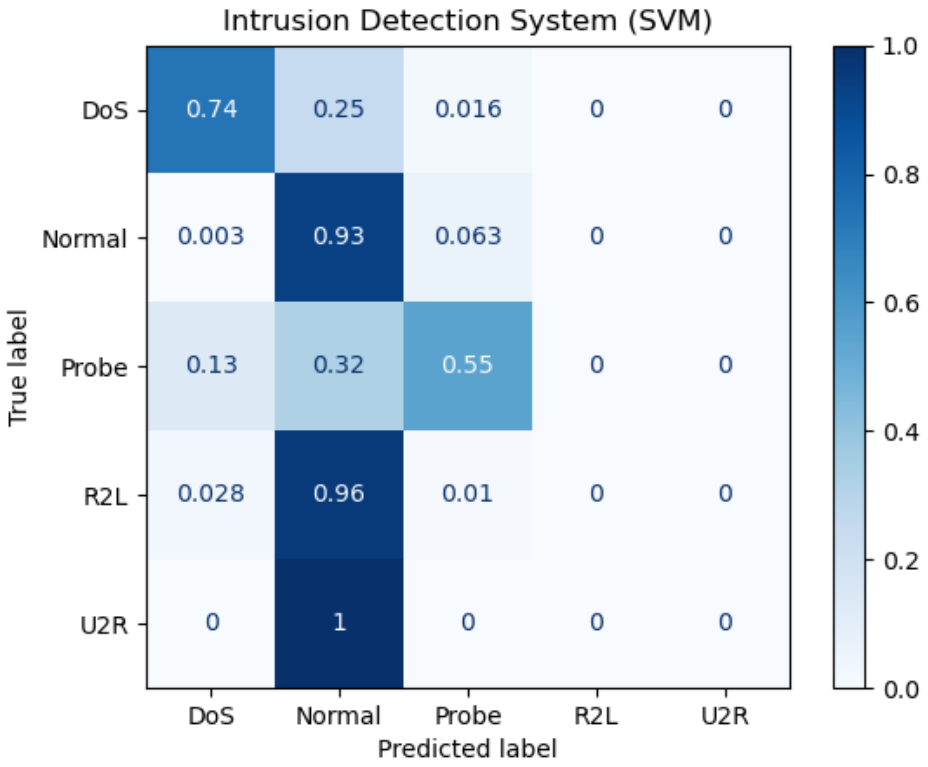
مدل J48 – داده متعادل(balance)



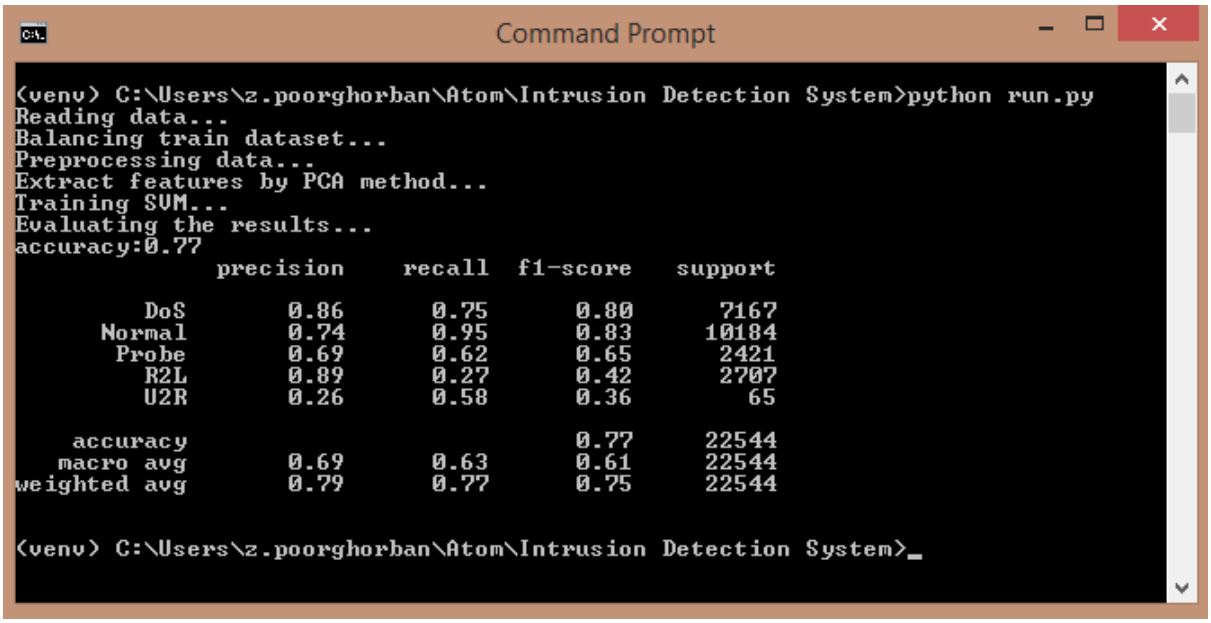


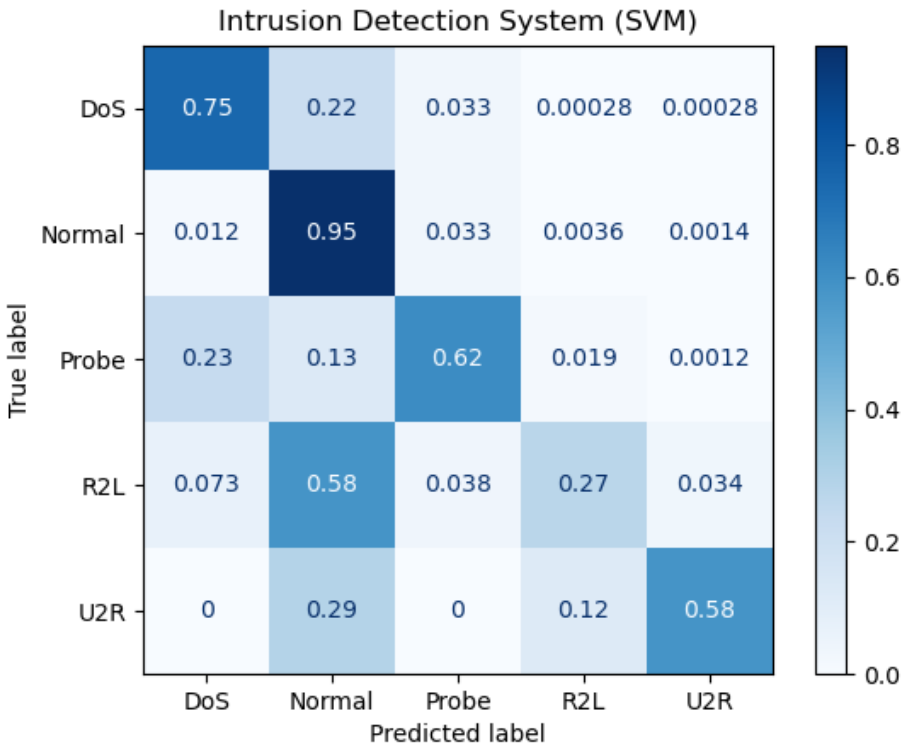
مدل SVM – داده غیر متعادل(imbalance)



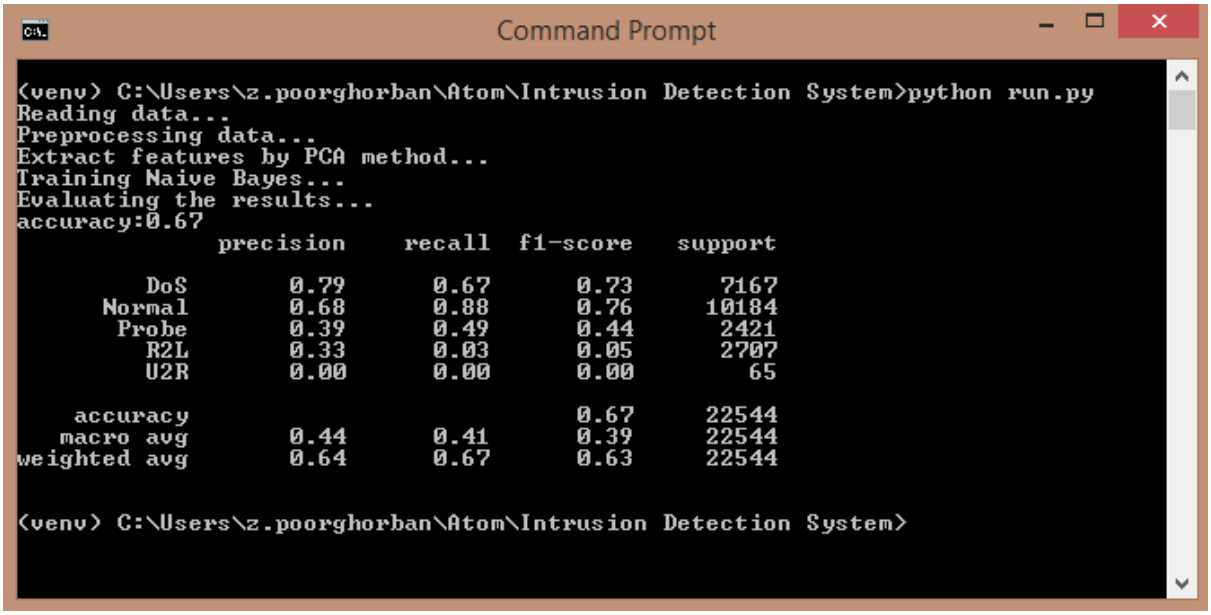


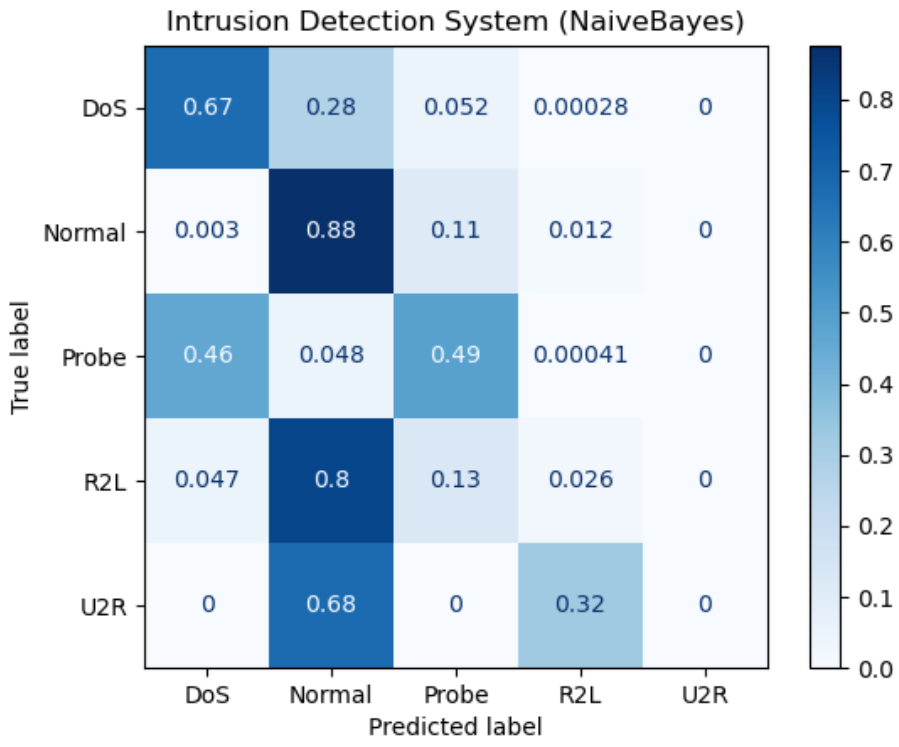
مدل SVM – داده متعادل(balance)





مدلNaive Bayes– داده غیرمتعادل(imbalance)





مدل Naive Bayes– داده متعادل(balance)