COMPUTATIONAL INTELLIGENCE

PROJECT 4 DOCUMENTATION

Ferdowsi University of Mashhad
Department of Computer Engineering

SPRING 2025

نام و نام خانوادگی	شماره دانشجویی
امیرحسین افشار	۴۰۱۲۲۶۲۱۹۶
علیرضا صفار	۴۰۱۱۲۶۲۲۸۱

۱) فاز اول

فاز اول: استخراج ویژگی ها از مدل resnet۱۸

در ابتدا یک بررسی بر روی مدل resnet۱۸ که با استفاده از pytorch پیاده سازی شده، انجام می دهیم:

Layer	#Channels	Width	Height
conv1	64	112	112
bn1	64	112	112
relu	64	112	112
maxpool	64	56	56
layer1	64	56	56
layer2	128	28	28
layer3	256	14	14
layer4	512	7	7
avgpool	512	1	1
fc	1000		

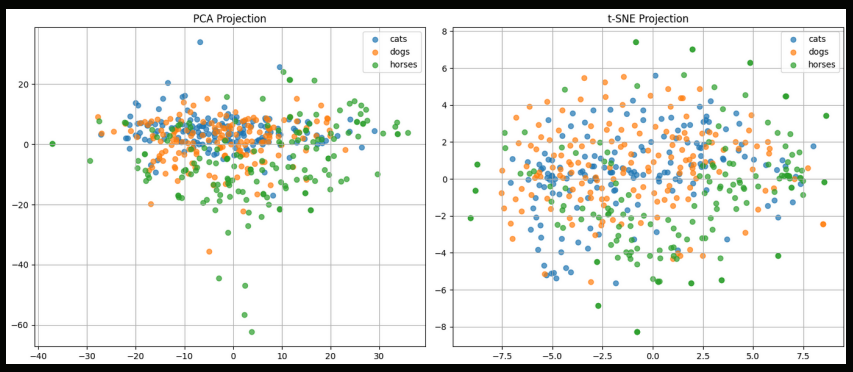
جدول ۱: بلاک های مدل resnet۱۸

بدین ترتیب، می توانیم تعداد فیچرهای هر کدام از مراحل خواسته شده را پیدا کنیم:

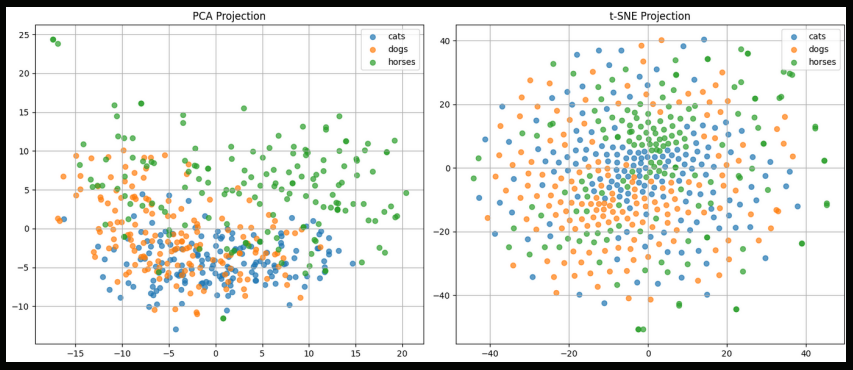
نوع فیلتر	تنظیمات	تعداد فیچرها	جزئیات بیشتر
فیلترهای ابتدایی	$112 \times 64 \times 64$	۸۰۲،۸۱۶	تا لایه maxpool قبل از لایه اول
فیلترهای میانی	$28 \times 28 \times 128$	۱۰۰،۳۵۲	تا بلاک دوم
فیلترهای سطح بالا	$1 \times 1 \times 512$	۵۱۲	تا قبل از fc

جدول ۲: ویژگی های ابتدایی، ویژگیهای میانی، ویژگی های سطح بالا

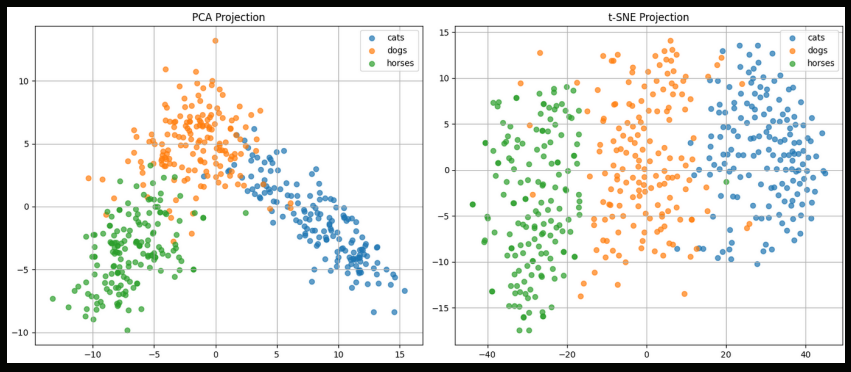
برای هر کدام از این سه دسته فیچرها، برای این که درک بهتری از نحوه پراکندگی و corrolation آنها داشته باشیم، با استفاده از PCA و t-SNE یک نمایش کلی بدست آورده ایم. بدین منظور، فیچرهای استخراج شده از مدل resnet را به شکل زیر پلات کرده ایم:



شکل ۱: ویژگی های ابتدایی



شکل ۲: ویژگی های میانی



شکل ۳: ویژگی های سطح بالا

همانطور که مشخص است، هرچه که از ویژگی های ابتدایی به ویژگی های سطح بالا عبور می کنیم، نمایش و بازنمایی بهتری از فیچرها به دست می آید که مطابق با انتظار است. بنابراین هم برای مدل های ساده فاز اول و هم برای مدل های stacked فاز دوم، انتظار داریم که برای فیچرهای سطح بالا، به دقت بالاتری دست پیدا کنیم.

فاز اول: مدل های ساده

برای پیاده سازی مدل های ساده، ۵ مدل زیر را در نظر گرفتیم

- SVM
- Logistic Regression
- Random Forest
- KNN
- Decision Tree

که برای هر کدام، پارامتر های default آنها را در نظر گرفتیم. بدین منظور، هرکدام از سه نوع ورودی را به آنها دادیم تا دقت آنها را بررسی کنیم:

Classifier	Accuracy	Precision	Recall	F1-Score
SVM	0.943	0.944	0.943	0.943
Logistic Regression	0.967	0.967	0.967	0.967
Random Forest	0.951	0.951	0.951	0.951
KNN	0.943	0.945	0.943	0.942
Decision Tree	0.820	0.825	0.820	0.821

جدول ۳: دقت طبقه بند ها به ازای فیچر های high

برای ویژگی های سطح بالا، بهترین مدل، logistic regression بود که درصد ۹۶ برای acc را داشت. سایر مدل ها نیز درصدهای تقریبا مشابهی را ارائه می دادند اما مدل درخت تصمیم، کمترین درصد را داشت که دلیل آن، اورفیت شدن سریع آن می باشد که در مقایسه با نسخه بهبود یافته آن یعنی random forrest این موضوع مشهود است.

Classifier	Accuracy	Precision	Recall	F1-Score
SVM	0.770	0.778	0.770	0.773
Logistic Regression	0.803	0.806	0.803	0.800
Random Forest	0.607	0.615	0.607	0.610
KNN	0.451	0.478	0.451	0.454
Decision Tree	0.533	0.530	0.533	0.531

جدول ۴: دقت طبقه بند ها به ازای فیچر های سطح متوسط

پس از بررسی فیچرهای سطح بالا، به جدول ۴ یعنی دقت طبقه بند ها به ازای فیچرهای سطح متوسط می رسیم که به طور میانگین دقت ها ۱۵ درصد کاهش یافته اند و بیشترین کاهش نیز متعلق به KNN است که دقتش از نصف مرحله قبلی نیز کمتر شده است. دلیل این موضوع را نیز میتوان بدین شکل توصیف کرد که چون knn به دلیل محاسبه فاصله از سایر نقاط به ازای ،k بسیار به تعداد

Classifier	Accuracy	Precision	Recall	F1-Score
SVM	0.607	0.597	0.607	0.597
Logistic Regression	0.656	0.652	0.656	0.636
Random Forest	0.672	0.673	0.672	0.660
KNN	0.500	0.588	0.500	0.434
Decision Tree	0.492	0.485	0.492	0.484

جدول ۵: دقت طبقه بند ها به ازای فیچر های سطح initial

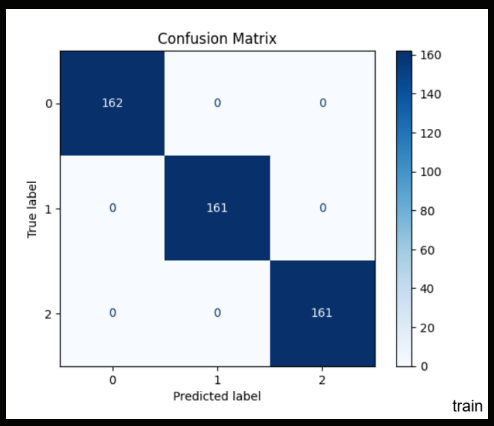
نقاط وابسته است، curse of dimentialty برایش صادق است و بنابراین بیشترین افت دقت را نیز در این مرحله داشته است.

در نهایت، به فیچر های ابتدایی میرسیم (جدول ۵) که همانطور که انتظار می رود، درصد ها نیز افت زیادی میکنند. در این دسته بندی، random forrest بهترین درصد را دارد که میتوان دلیل آن را جلوگیری از اورفیت ذاتی آن دانست. نکته قابل توجه نیز افزایش دقت مدل knn است که میتوان گفت زمانی که تعداد نقاط (data point) از حدی بالا رود، عملکرد این طبقه بند نیز غیرقابل پیش بینی خواهد شد.

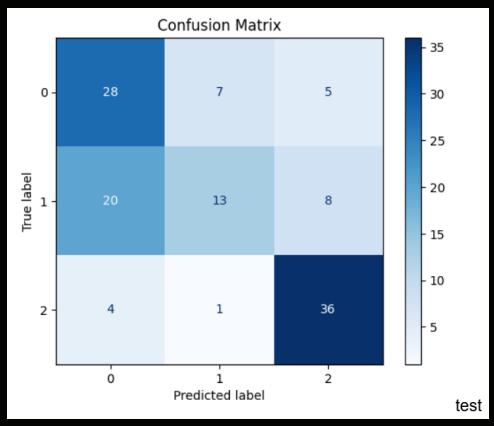
۲) فاز دوم

در فاز دوم پروژه، یک مدل یادگیری پشته‌ای stacked learner پیاده‌سازی شد. در این مرحله، ابتدا داده‌ها با استفاده از StandardScaler نرمال‌سازی شدند. سپس چهار الگوریتم طبقه‌بندی مختلف شامل SVM و درخت تصمیم و Logistic Regression و Random Forest به‌عنوان مدل‌های پایه انتخاب شدند.

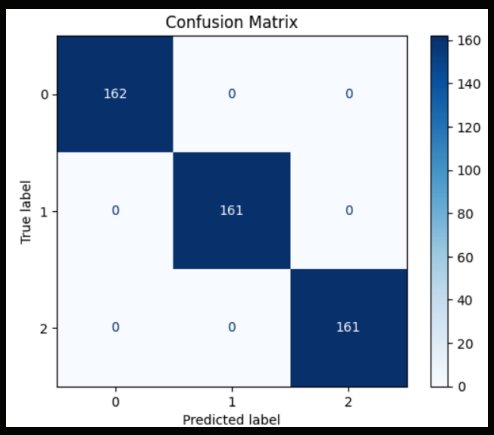
هر یک از این مدل‌ها به‌صورت مستقل روی داده‌های آموزشی آموزش داده شدند و خروجی آن‌ها برای داده‌های آموزشی و آزمایشی استخراج گردید. خروجی مدل‌های پایه به‌صورت ویژگی‌های ورودی به یک متا-مدل داده شد. برای مدل نهایی، از رگرسیون لجستیک به‌عنوان متا-مدل استفاده شد و این مدل روی ترکیب خروجی مدل‌های پایه آموزش دید. این فرآیند برای هر سه نوع مجموعه ویژگی شامل initial-features، mid-level features و high-level features به‌صورت جداگانه انجام شد. برای هر یک از این سه حالت، ارزیابی مدل نهایی با استفاده از معیارهای عملکرد روی مجموعه‌های آموزش و آزمون صورت گرفت و همان طور که در ماتریس های زیر مشخص است در هر سه مورد مدل دچار overfitting شده است.



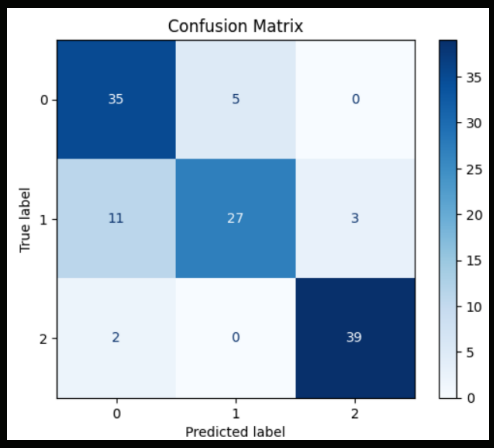
شکل ۴: مدل فیچر های ابتدایی: دقت آموزش



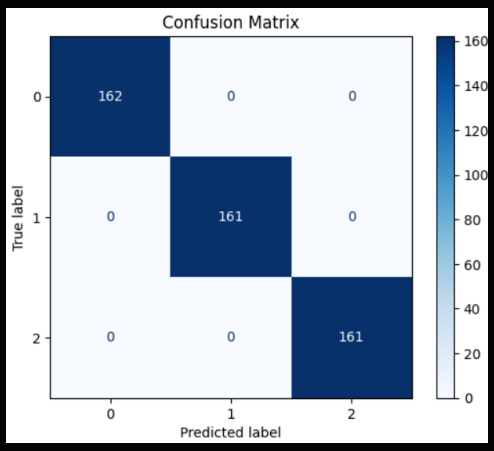
شکل ۵: مدل فیچر های ابتدایی: دقت تست



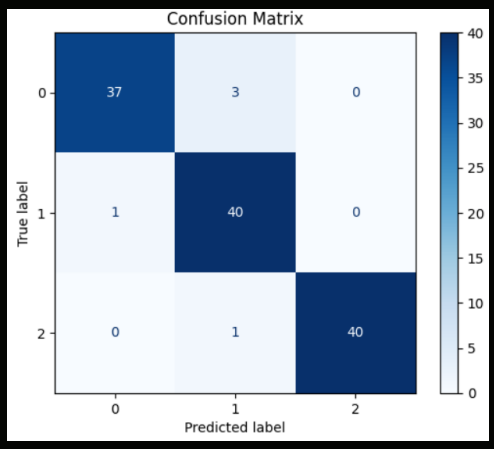
شکل ۶: مدل فیچر های میانی: دقت آموزش



شکل ۷: مدل فیچر های میانی: دقت تست



شکل ۸: مدل فیچر های بالا: دقت آموزش



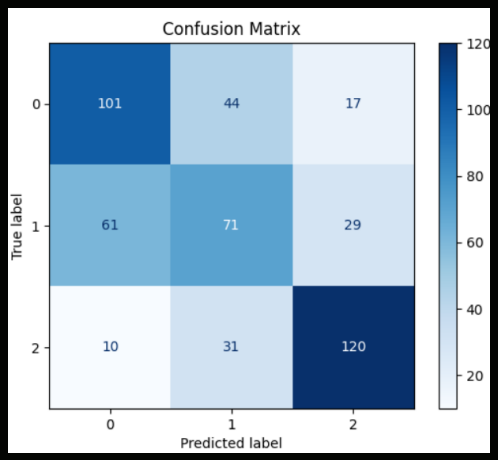
شکل ۹: مدل فیچر های بالا: دقت تست

جلوگیری از اورفیت

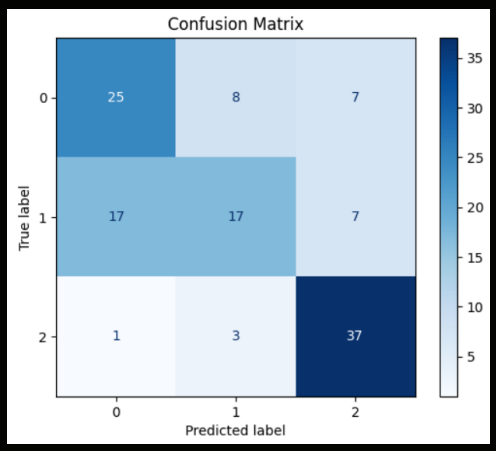
برای جلوگیری از overfitting در مدل پشته‌ای، روش پیاده‌سازی مدل بهینه‌سازی شد تا فرآیند یادگیری سطح دوم (Meta-Learner) اطلاعات نشت‌یافته از داده‌های آموزش را دریافت نکند. برای این منظور، از تکنیک K-Fold Cross Validation روی داده‌های آموزشی استفاده شد. در این رویکرد، هر مدل پایه به‌جای آموزش روی کل داده و پیش بینی روی همان داده، در هر تکرار روی K-۱ بخش آموزش دید و روی بخش باقی‌مانده (validation) پیش بینی انجام داد. خروجی‌های پیش‌بینی‌شده بر روی هاvalidation به‌عنوان ویژگی برای آموزش متا-مدل استفاده شدند.

برای داده‌های آزمایشی نیز، در هر تکرار از ،K-Fold مدل آموزش‌دیده بر روی fold مربوطه پیش‌بینی انجام داد و نهایتاً با میانگین‌گیری از نتایج تمام تکرارها، خروجی نهایی برای test تولید شد. در این ساختار، مدل‌های پایه شامل ،SVM درخت تصمیم، رگرسیون لجستیک و Random forest بودند. متا-مدل نیز همانند قبل یک رگرسیون لجستیک بود.

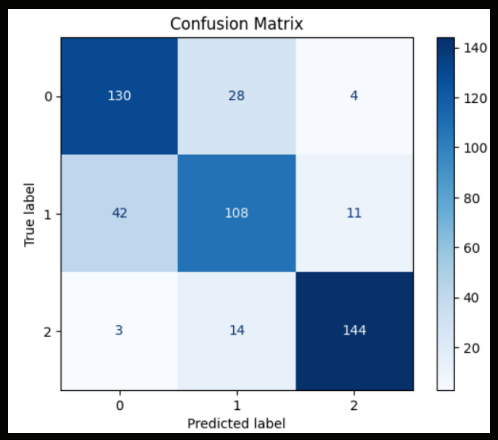
در این مرحله نیز فرآیند فوق برای هر سه مجموعه ویژگی شامل initial، mid-level features و features high-level اجرا شد. با این تفاوت که برای مجموعه features initial به‌دلیل بالا بودن تعداد ویژگی‌ها و محدودیت‌های پردازشی، مقدار n_splitsn به ۲ کاهش یافت تا زمان آموزش کاهش یابد. اما برای دو مجموعه دیگر، مقدار پیش‌فرض n_splits=۵ حفظ شد. در پایان، عملکرد مدل نهایی برای هر سه نوع ویژگی، با استفاده از معیارهای ارزیابی روی مجموعه‌های آموزش (Train) و آزمون (Test) گزارش شد و مشاهده شد که این بار مدل ها دچار overfitting نشده اند.



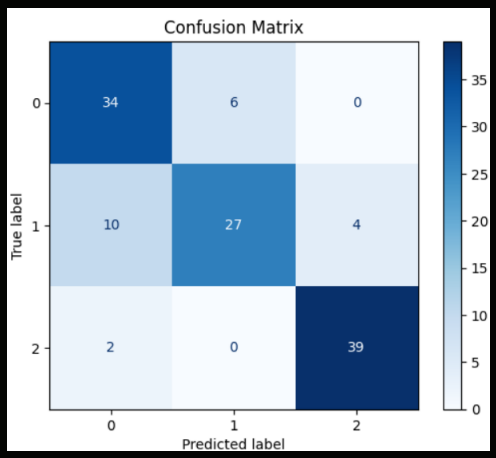
شکل ۱۰: مدل فیچر های ابتدایی با جلوگیری از اورفیت: دقت آموزش



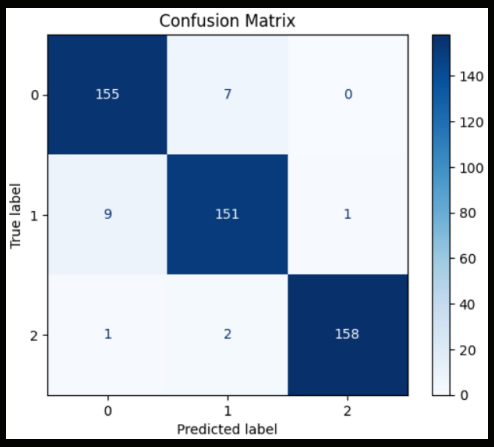
شکل ۱۱: مدل فیچر های ابتدایی با جلوگیری از اورفیت: دقت تست



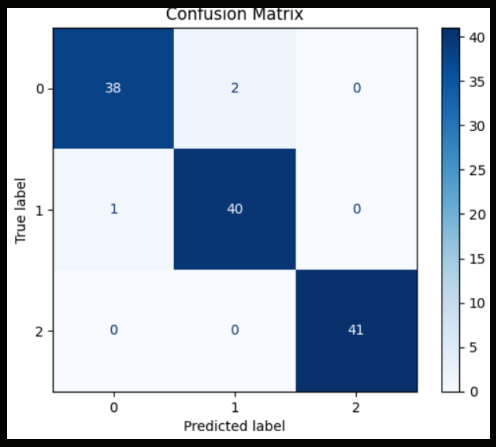
شکل ۱۲: مدل فیچر های میانی با جلوگیری از اورفیت: دقت آموزش



شکل ۱۳: مدل فیچر های میانی با جلوگیری از اورفیت: دقت تست



شکل ۱۴: مدل فیچر های بالا با جلوگیری از اورفیت: دقت آموزش



شکل ۱۵: مدل فیچر های بالا با جلوگیری از اورفیت: دقت تست

hyperparameter tuning

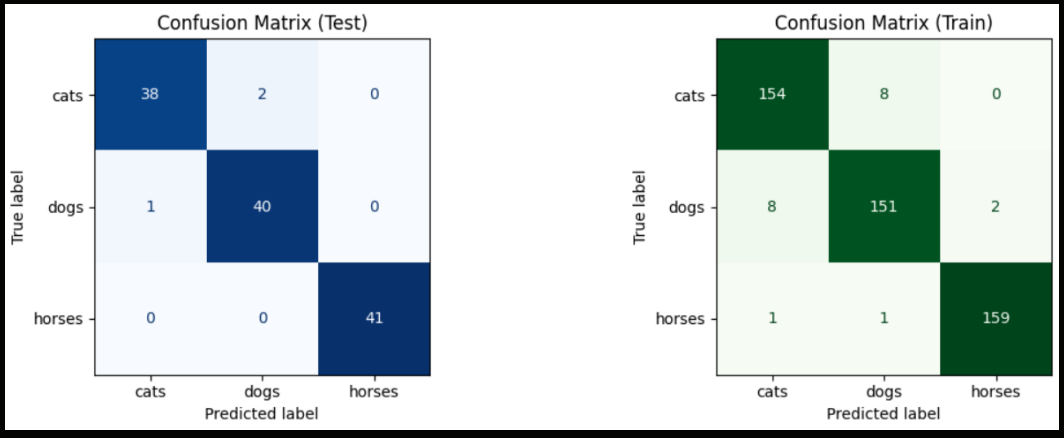
برای انجام هایپرپارامتر تیونینگ، یک لیست به شرح زیر ساخته شد که جایگشت میان حالات آن، در مجموع ۱۲۸ تا configuration را می سازد:

Algorithm	Parameter	Value
SVM	C	[0.1, 1.0]
SVM	kernel	[”rbf”, ”linear”]
DecisionTree	max_depth	[3, 5]
DecisionTree	min_samples_split	[2, 4]
LogisticRegression	C	[0.1, 1.0]
LogisticRegression	penalty	[”l2”]
RandomForest	n_estimators	[50, 100]
RandomForest	max_depth	[4, 6]

Table 6: Machine Learning Algorithm Configurations

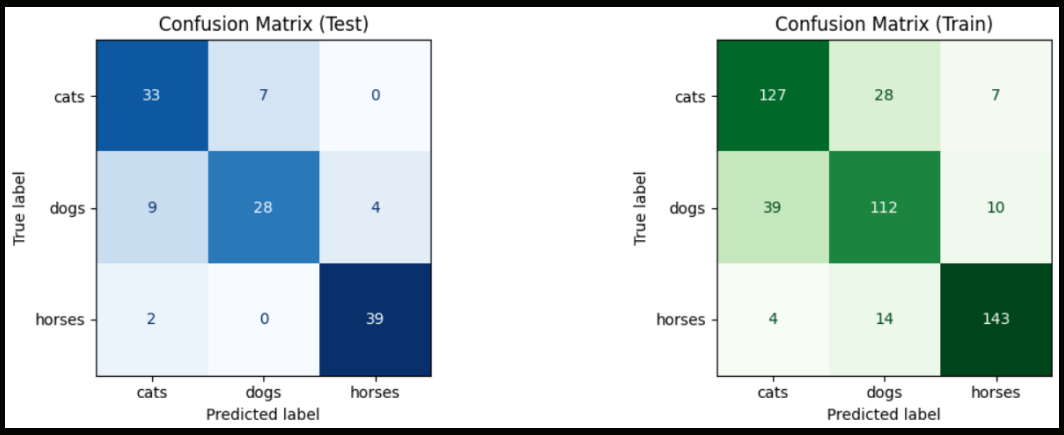
از میان موارد اشاره شده، به شکل رندوم سرچ، انتخاب شد و برای آنها عملیات train ، test انجام شد. برای فیچر های سطح بالا، ۶۰ عدد کانفیگوریشن انتخاب و تست شد (چون فیچر های سطح بالا به دلیل کم بودن تعدادشان با محدودیت پردازشی رو به رو نمیشدند.) و برای فیچر های سطح میانی، ۱۰ عدد کانفیگوریشن تست شد و همچنین، برای فیچرهای سطح پایین، مدلمان را هایپرپارامتر تیونینگ نکردیم؛ دلیل موجهی نیز برای آن داشتیم: تعداد بسیار زیاد فیچر ها، عملیات یادگیری را بسیار طولانی (حتی با استفاده از گوگل کولب) می کرد؛ همچنین، اصولا این نوع از مدل های یادگیری ماشین ساده، توانایی اجرا بر روی gpu نیز نداشتند.

نتیجه این تیونینگ، این شد که هر دو مدل، به درصد های بالاتری رسیدند:



شکل ۱۶: ویژگی های سطح بالا با تیونینگ

برای ویژگی های سطح بالا، دقت از ۹۶ به 97.5 درصد رسید.



شکل ۱۷: ویژگی های میانی با تیونینگ

برای ویژگی های میانی، acc از ۷۶ به ۸۲ درصد رسید. (شایان ذکر است، تمامی شاخص های آماری دیگر نظیر f1-score و غیره در پروژه به شکل خوانایی ذخیره شده است و برای جلوگیری از تکرار مکررات، از نوشته شدن مجدد آنها در این بخش خودداری می شود.)

ارزیابی مدل Stacked و مقایسه با نتایج فاز اول

مدل‌های پشته ای به‌عنوان راهکاری برای افزایش دقت طبقه‌بندی در این پروژه مورد استفاده قرار گرفتند. هدف از این فاز، بررسی این بود که آیا ترکیب چندین مدل یادگیری ماشین می‌تواند نسبت به مدل‌های منفرد عملکرد بهتری در سطوح مختلف ویژگی داشته باشد یا خیر. برای این منظور در سه سطح از ویژگی‌های استخراج‌شده مدل‌های Stacked ایجاد شده‌اند و با نتایج حاصل از فاز اول مقایسه شده‌اند. همانطور که اشاره شد، در این مرحله آن است که هایپرپارامتر تونینگ تنها برای نسخه‌های mid و high مدل‌های Stacked انجام شده و نسخه initial و مدل‌های منفرد فاز اول بدون tuning اجرا شده‌اند.

در سطح features high-level مدل Stacked توانسته است بهترین عملکرد را از خود نشان دهد. دقت این مدل به عدد0.9754 رسیده که نسبت به بهترین مدل منفرد در فاز اول، یعنی Regression Logistic با دقت0.967 بهبود قابل توجهی داشته است. این بهبود نه‌تنها در دقت بلکه در تمام معیارهای ،Precision Recall و F۱-score نیز دیده می‌شود. دلیل اصلی این افزایش عملکرد را می‌توان در دو عامل دانست: اول آنکه استفاده از Stacked باعث شد که ضعف‌ها و خطاهای مدل‌های منفرد تا حد زیادی جبران شود و عملکرد کلی سیستم با ترکیب دیدگاه‌های متنوع مدل‌ها بهبود یابد. این ترکیب به‌ویژه در ویژگی‌های سطح بالا، منجر به تصمیم‌گیری دقیق‌تر و پایدارتر نسبت به استفاده‌ی تنها از یک مدل شد. دوم تنظیم دقیق ابرپارامترهای مدل‌های پایه در ساختار Stacked باعث شده که مدل نهایی از نظر تعمیم‌پذیری بهبود یابد و بتواند الگوهای پیچیده‌تری را از داده‌ها استخراج کند.

درfeatures mid-level نیز مدل ترکیبی عملکرد بهتری نسبت به مدل‌های منفرد داشته است. دقت مدل Stacked در این سطح برابر با0.8197 گزارش شده در حالی که بهترین مدل فاز اول در همین سطح دقتی معادل0.803 داشته است. استفاده از مدل Stacked باعث شد که ضعف‌های مدل‌های پایه تا حدودی جبران شده و عملکرد بهتری نسبت به مدل‌های منفرد به دست آید. همچنین استفاده از tuning در این عملکرد بهتر بسیار تاثیر گذار بوده است.

اما در features initial-level برخلاف دو سطح قبل، مدل Stacked عملکرد ضعیف‌تری نسبت به بهترین مدل فاز اول داشته است. دقت این مدل برابر با0.6475 گزارش شده، در حالی که بهترین عملکرد در فاز اول Random Forest در همین سطح برابر با0.672 بوده است. این افت عملکرد را می‌توان به دلیل آن دانست که در این سطح، برای مدل Stacked هیچ‌گونه تنظیم ابرپارامتر انجام نشده است و همین باعث شده که مدل از ظرفیت کامل خود برای یادگیری استفاده نکند. این مسأله اهمیت تنظیم دقیق مدل‌ها را در معماری‌های ترکیبی به‌خوبی نشان می‌دهد. در جمع‌بندی می‌توان گفت استفاده از معماری Stacked به‌ویژه در کنار ویژگی‌های سطح بالا و با انجام Tuning مناسب، می‌تواند به‌طور مؤثری دقت طبقه‌بندی را نسبت به مدل‌های منفرد بهبود بخشد. این نتایج نشان می‌دهند که اهمیت تنظیم دقیق هایپرپارامترها و Tuning در کنار استفاده از Stacking نقشی کلیدی در بهبود عملکرد سیستم طبقه‌بندی ایفا می‌کند. این مسئله همچنین تأکید می‌کند که کیفیت ویژگی‌های استخراج‌شده و تنظیمات داخلی مدل‌ها نقش تعیین‌کننده‌ای در موفقیت یک سیستم طبقه‌بندی دارند.