

---

# سوال اول: گزارش پروژه تشخیص تقلب در تراکنش‌های کارت اعتباری با استفاده از شبکه‌های عصبی چندلایه (MLP)

---

## ۱. مقدمه

در این پروژه، هدف طراحی و پیاده‌سازی یک سیستم هوشمند برای شناسایی تراکنش‌های جعلی کارت‌های اعتباری با استفاده از شبکه‌های عصبی چندلایه (MLP) است. با توجه به ماهیت داده‌ها که با عدم تعادل شدید کلاس‌ها مواجه است، چالش اصلی، ساخت مدلی است که بتواند با دقت بالا تراکنش‌های نادر اما بسیار مهم تقلبی را شناسایی کند. در این گزارش، مراحل مختلف از پیش‌پردازش داده‌ها تا ساخت، ارزیابی و مقایسه مدل‌های مختلف به تفصیل شرح داده خواهد شد.

---

## ۲. پیش‌پردازش و بررسی دادگان

### ۲-۱. خلاصه‌ای از داده‌ها

مجموعه داده Credit Card Fraud Detection از وبسایت Kaggle بارگذاری شد. این مجموعه داده شامل تراکنش‌های انجام شده در طی دو روز است. به دلیل حفظ حریم خصوصی، اکثر ویژگی‌ها (V1 تا V28) با استفاده از تحلیل مؤلفه‌های اصلی (PCA) تبدیل شده‌اند. تنها ویژگی‌هایی که تبدیل نشده‌اند Time (زمان) و Amount (مبلغ تراکنش) هستند. ستون هدف، Class، مشخص‌کننده نوع تراکنش است که مقدار ۱ برای تراکنش تقلبی و ۰ برای تراکنش سالم است.

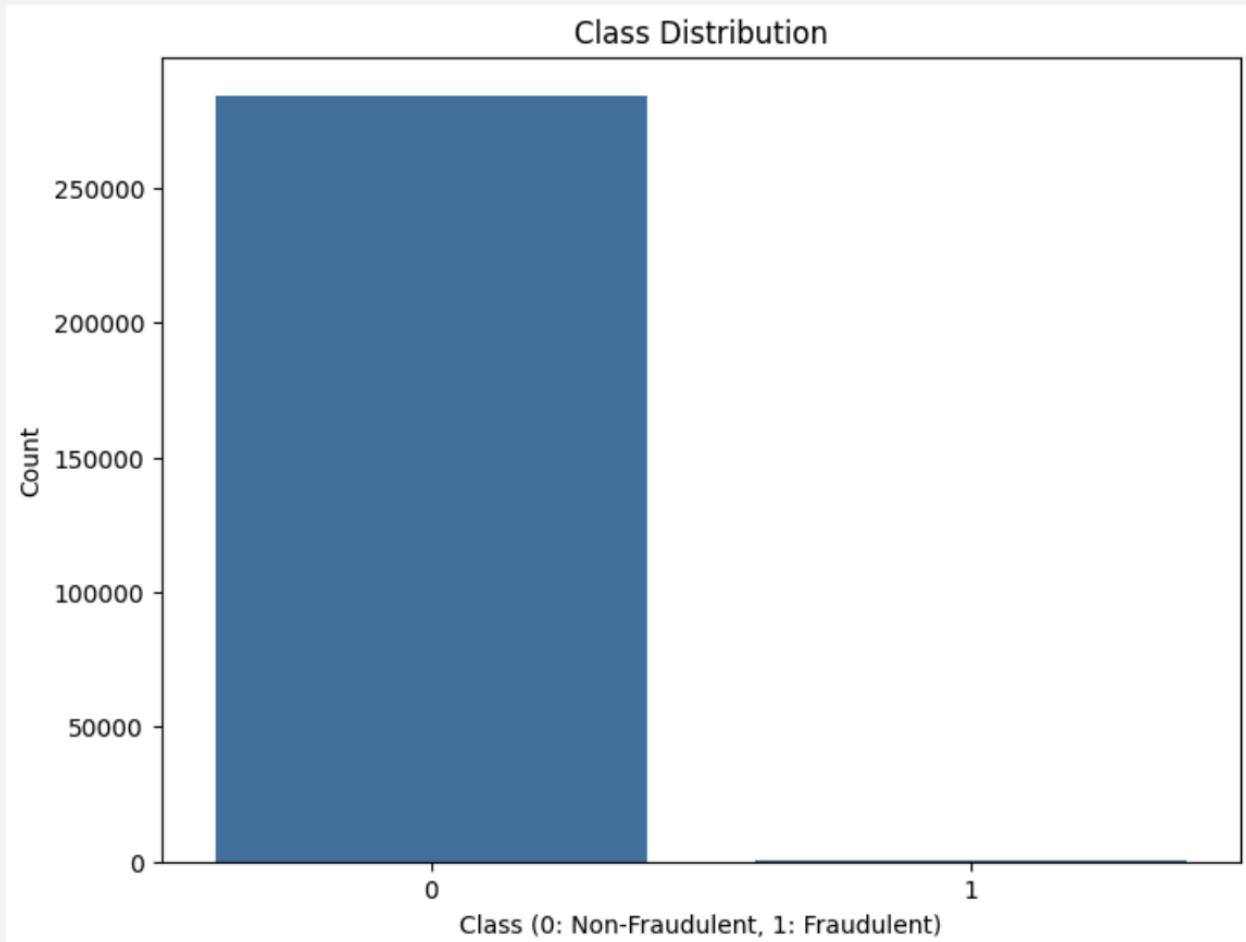
- تعداد کل تراکنش‌ها: ۲۸۴,۸۰۷
- تعداد ویژگی‌ها: ۳۰ (شامل Time, Amount و ۲۸ ویژگی PCA)
- داده گمشده (Missing Value): خوشبختانه هیچ داده گمشده‌ای در مجموعه داده وجود ندارد.

### ۲-۲. توزیع کلاس‌ها و عدم تعادل داده

پس از بررسی، مشخص شد که داده‌ها به شدت نامتوازن هستند:

- تراکنش‌های سالم (کلاس ۰): ۲۸۴,۳۱۵ (٪۹۹.۸۲۷)
- تراکنش‌های تقلبی (کلاس ۱): ۴۹۲ (٪۰.۱۷۳)

نمودار میله‌ای زیر این عدم تعادل شدید را به وضوح نشان می‌دهد:



### ۴-۳. چالش عدم تعادل کلاس

عدم تعادل کلاس‌ها یک چالش بزرگ در مدل‌سازی است، زیرا:

یک مدل "تنبل" می‌تواند با پیش‌بینی همیشگی کلاس اکثربیت (تراکنش سالم)، به دقت بالای ۹۹٪ دست یابد. اما چنین مدلی کاملاً بی‌فایده است، زیرا هدف اصلی ما، یعنی شناسایی تقلب، را انجام نمی‌دهد. مدل باید یاد بگیرد که الگوهای بسیار نادر کلاس اقلیت را تشخیص دهد، که این کار را بسیار دشوار می‌کند.

### ۴-۴. نرمال‌سازی و تقسیم داده‌ها

- نرمال‌سازی: ویژگی‌های Amount و Time دارای مقیاس متفاوتی نسبت به سایر ویژگی‌ها بودند. برای جلوگیری از تأثیرگذاری نامتناسب این ویژگی‌ها بر مدل، تمام ویژگی‌ها با استفاده از StandardScaler نرمال‌سازی شدند. این کار باعث می‌شود میانگین هر ویژگی صفر و انحراف معیار آن یک شود.
- تقسیم داده: مجموعه داده به دو بخش آموزش (۷۰٪) و تست (۳۰٪) تقسیم شد. برای حفظ نسبت کلاس‌ها در هر دو مجموعه، از تقسیم‌بندی طبقه‌ای (Stratified Splitting) استفاده شد. این کار تضمین می‌کند که مدل در هر دو مرحله آموزش و تست، با درصد مشابهی از داده‌های تقلبی مواجه شود.

---

### ۳. پیاده‌سازی یک شبکه MLP ساده (یک لایه مخفی)

یک شبکه عصبی پایه با یک لایه مخفی طراحی و در چهار حالت مختلف ارزیابی شد. معماری پایه شامل یک لایه ورودی، یک لایه مخفی با ۶۴ نورون و تابع فعال‌ساز ReLU و یک لایه خروجی با یک نورون و تابع فعال‌ساز Sigmoid بود.

نتایج مدل‌های ساده:

1. MLP ساده (بدون منظم‌سازی):

F1-Score: 0.7887 ○

تحلیل: این مدل پایه عملکرد معقول دارد اما نمودار زیان (Loss) آن نشان‌دهنده شروع بیش‌برازش (Overfitting) است؛ زیان اعتبارسنجی (Validation Loss) در حال افزایش است در حالی که زیان آموزش (Train Loss) به کاهش ادامه می‌دهد.

2. MLP با منظم‌سازی L2:

F1-Score: 0.7836 ○

تحلیل: افزودن منظم‌سازی L2 باعث افزایش دقت (Precision) شد (از ۰.۸۲ به ۰.۸۷)، به این معنی که هشدارهای غلط کمتری تولید کرد. اما بازخوانی (Recall) کاهش یافت (از ۰.۷۵ به ۰.۷۰)، یعنی تعداد بیشتری از تقلب‌ها را از دست داد. این نشان می‌دهد که منظم‌سازی به تنهایی راه حل کاملی نیست.

3. Dropout با MLP:

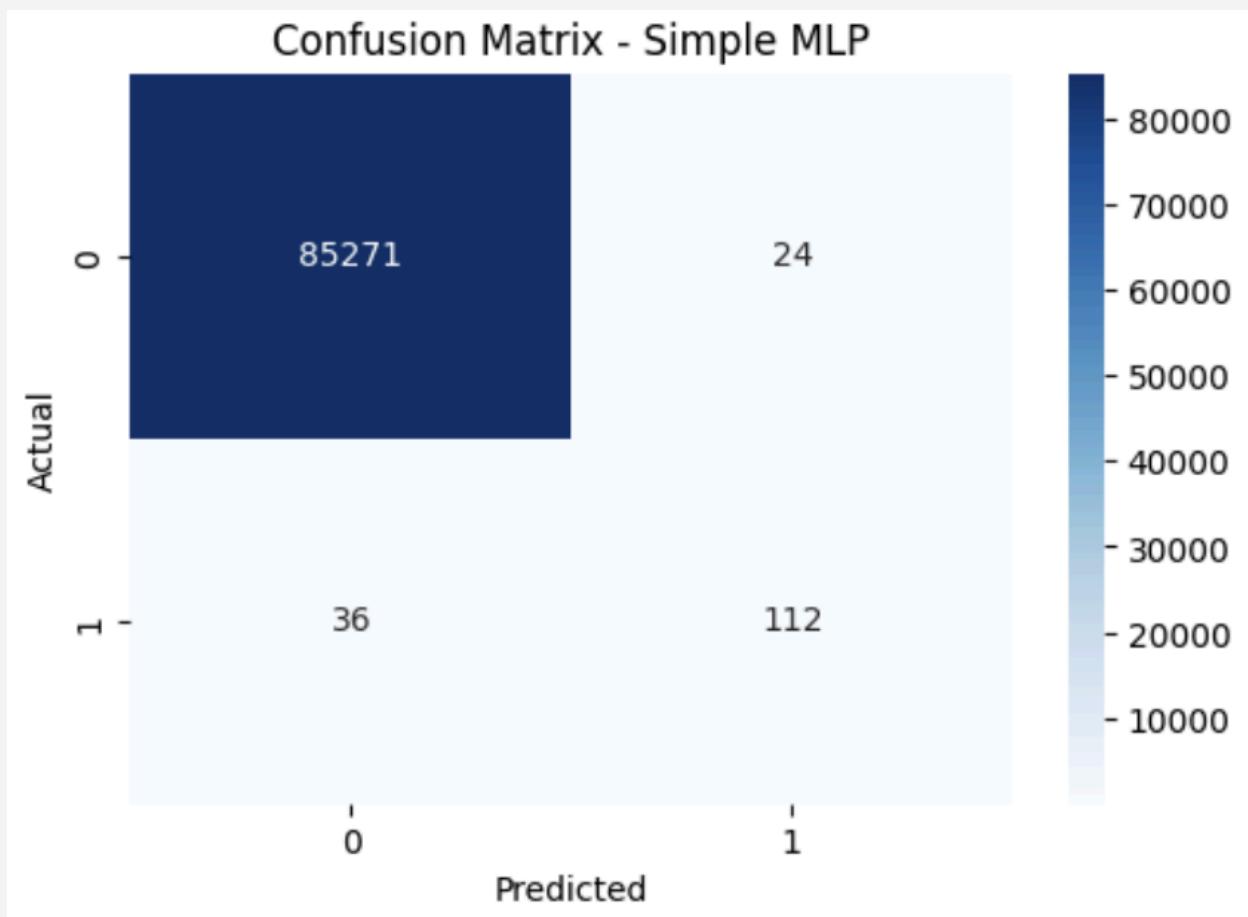
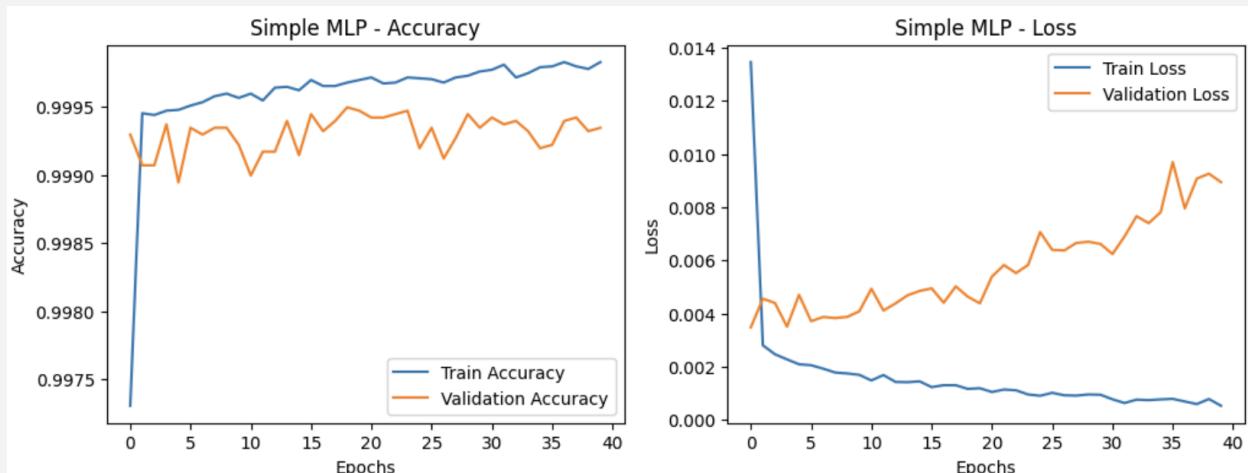
F1-Score: 0.8015 ○

تحلیل: این مدل بهترین عملکرد را در میان مدل‌های تک‌لایه داشت. Dropout با غیرفعال کردن تصادفی نورون‌ها در حین آموزش، از بیش‌برازش جلوگیری کرده و به تعمیم‌پذیری بهتر مدل کمک می‌کند. این مدل به تعادل خوبی بین دقت (۰.۸۹) و بازخوانی (۰.۷۲) دست یافت.

4. MLP با Dropout و L2:

F1-Score: 0.7704 ○

تحلیل: ترکیب این دو روش منظم‌سازی در این معماری خاص، منجر به کاهش عملکرد شد. بازخوانی به شدت افت کرد (۰.۶۶) که نشان می‌دهد مدل بیش از حد محدود شده و در شناسایی موارد مثبت ضعیف عمل کرده است.

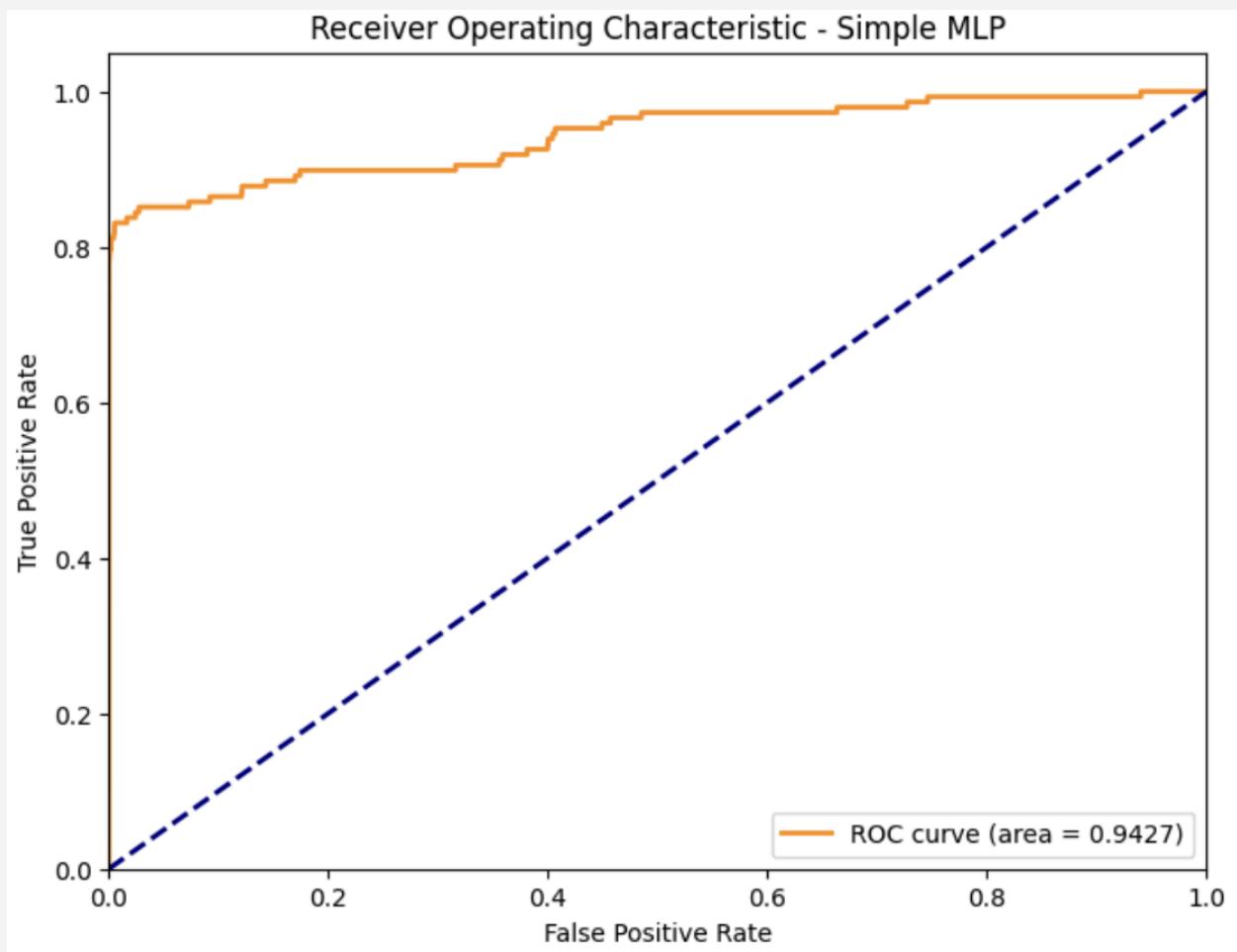


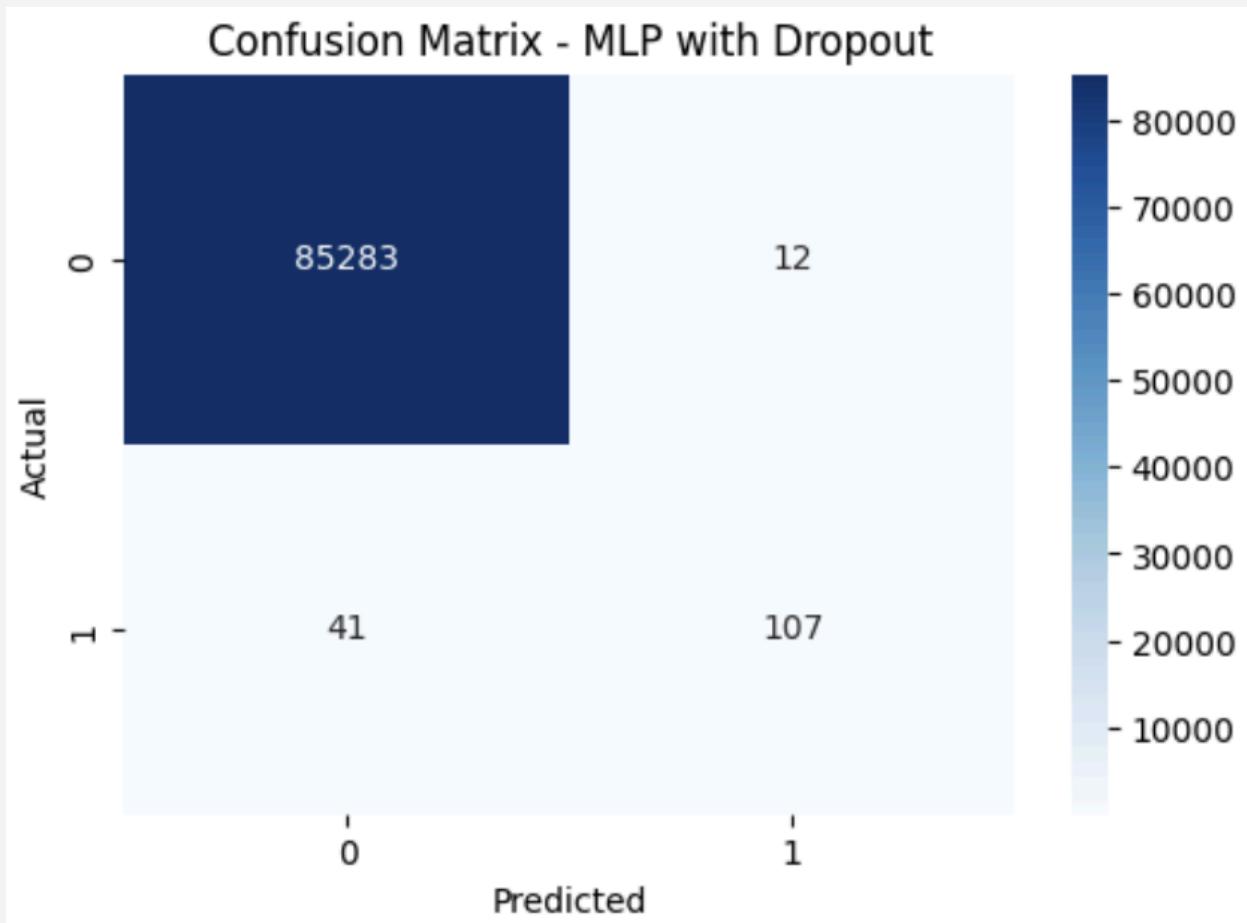
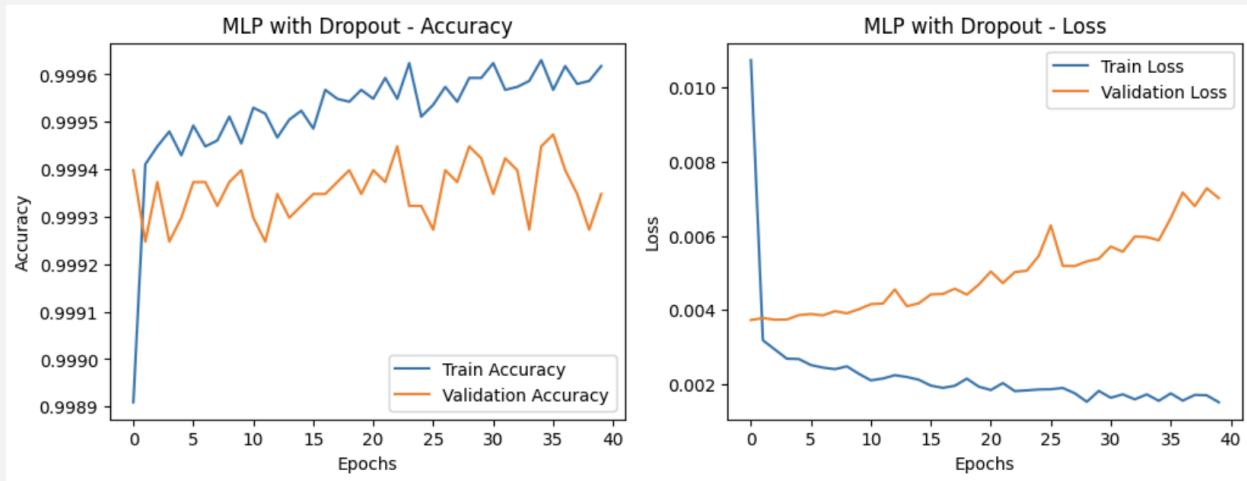
Accuracy: 0.9993

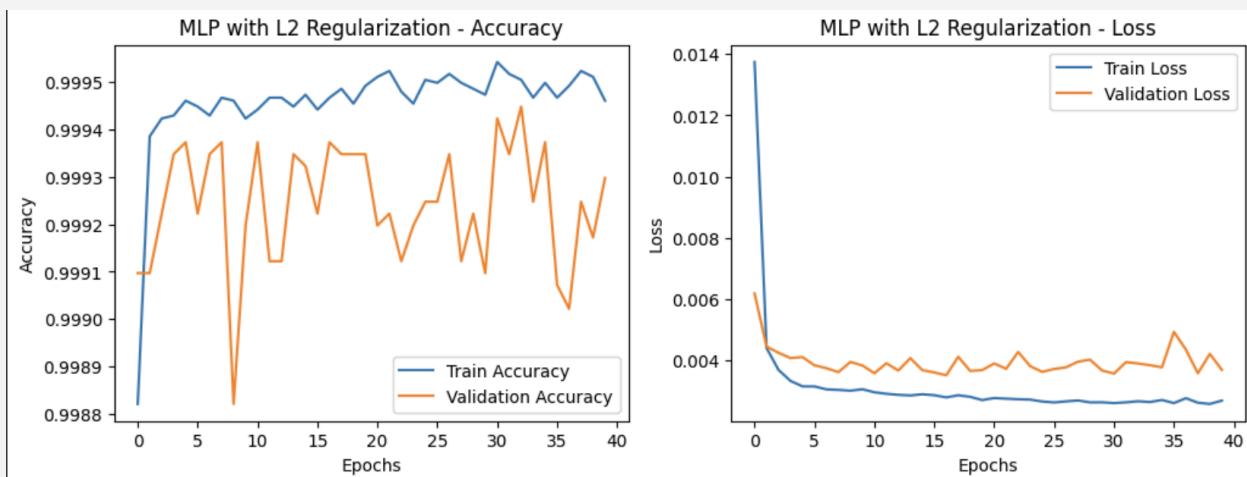
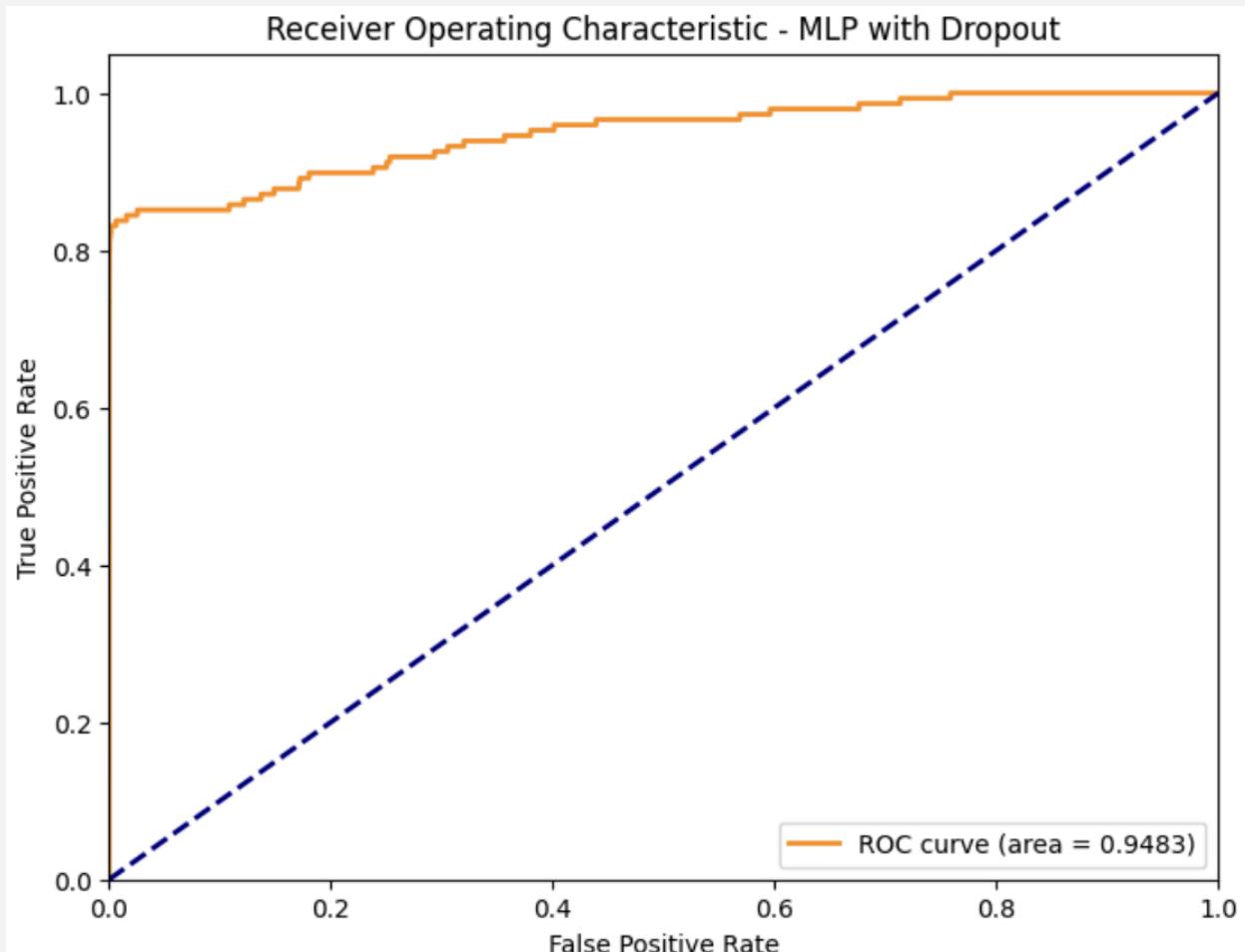
Precision: 0.8235

Recall: 0.7568

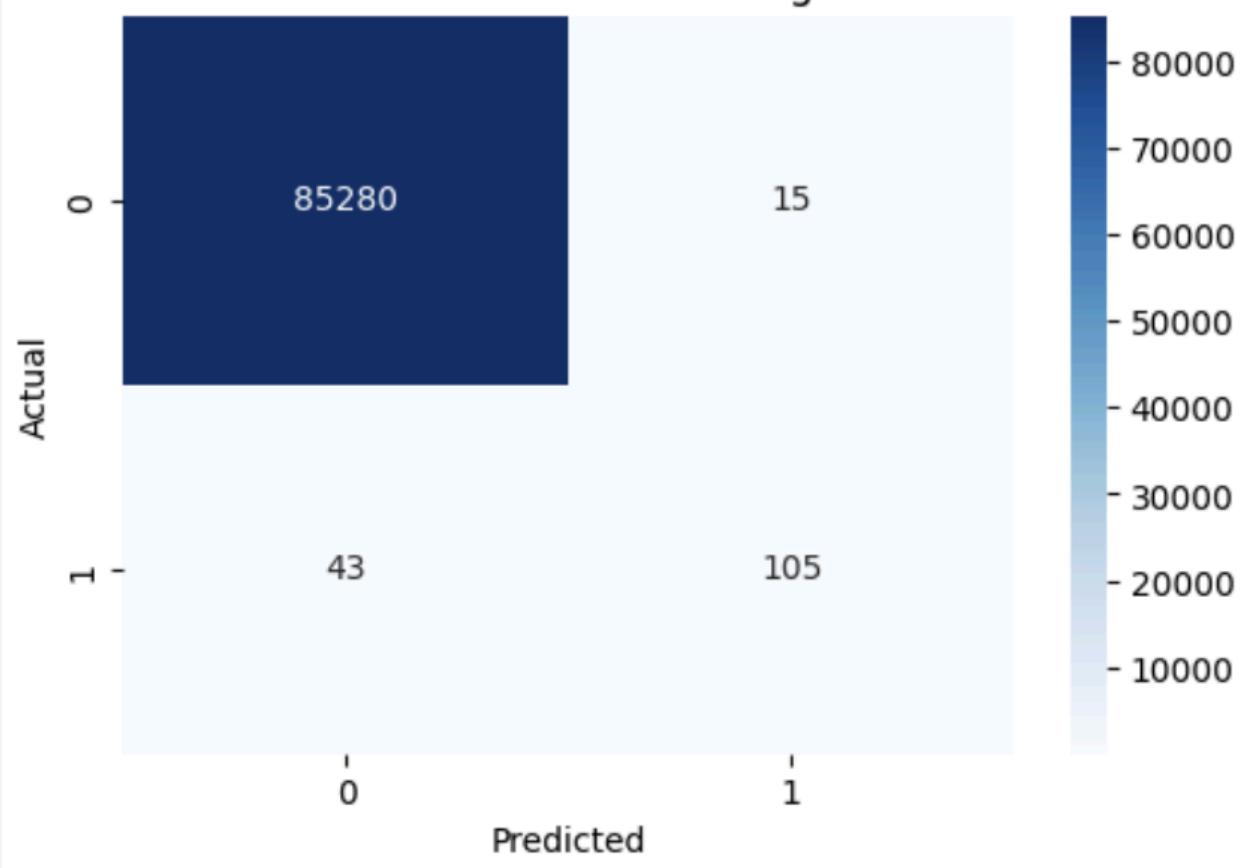
F1-Score: 0.7887

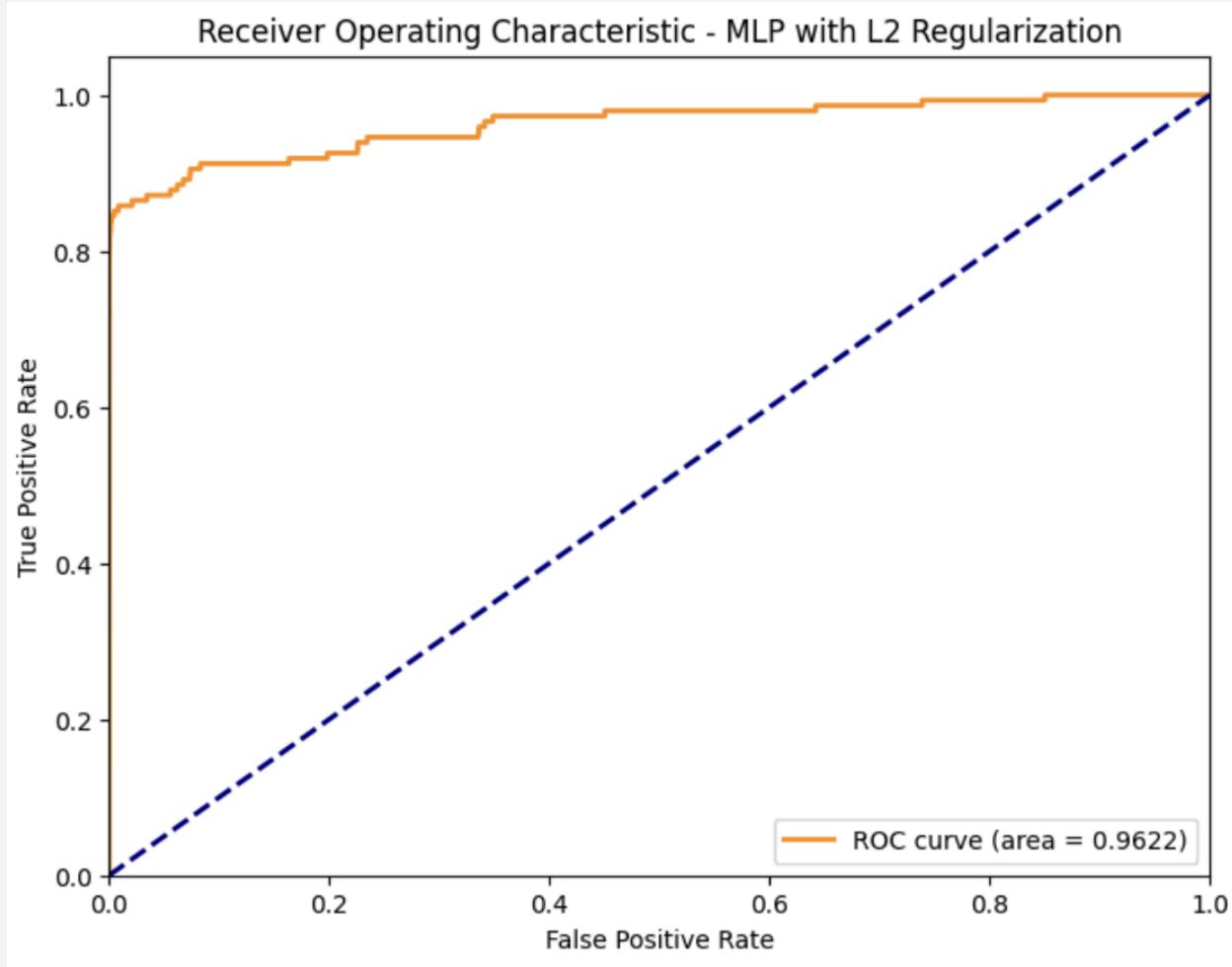






Confusion Matrix - MLP with L2 Regularization





#### ۴. طراحی یک شبکه عصبی عمیقتر (Deep MLP)

برای بررسی تأثیر افزایش پیچیدگی، یک شبکه عمیقتر با دو لایه مخفی طراحی شد.

- معماری:
- لایه مخفی اول: ۱۲۸ نورون با تابع ReLU
- لایه مخفی دوم: ۶۴ نورون با تابع ReLU
- منظمسازی: استفاده از ۲۰٪ (Dropout) پس از هر لایه مخفی و منظمسازی L2.

نتایج:

- F1-Score: 0.7905 ○
- دقت (Precision): 0.7905 ○
- بازخوانی (Recall): 0.7905 ○

مقایسه با مدل‌های قبلی:

این مدل عمیق توانست به بالاترین بازخوانی (Recall) در میان تمام مدل‌ها دست یابد. این مدل موفق شد ۱۱۷ تراکنش تقلیب را شناسایی کند که از بهترین مدل تکلایه (۱۰۷ مورد) بیشتر است. با این حال، این بهبود در بازخوانی به قیمت کاهش دقت تمام شد (تعداد هشدارهای غلط از ۱۲ به ۳۱ افزایش یافت). F1-Score آن کمی پایین‌تر از بهترین مدل تکلایه (MLP با Dropout) بود. این نشان می‌دهد که افزودن لایه لزوماً به معنای عملکرد کلی بهتر نیست، بلکه می‌تواند توانایی مدل را در یک معیار خاص (مانند بازخوانی) تقویت کند.

## ۵. تحلیل ماتریس آشفتگی و معیارهای ارزیابی

- دقت (Accuracy): این معیار درصد کل پیش‌بینی‌های صحیح را نشان می‌دهد. در داده‌های نامتوازن مانند این پروژه، معیار مناسبی نیست. زیرا اگر مدل همیشه "تراکنش سالم" را پیش‌بینی کند، دقت آن ۹۹.۸٪ خواهد بود، اما هیچ تقلیبی را تشخیص نخواهد داد.
- Precision (دقت کلاس مثبت): از میان تمام هشدارهای تقلب صادر شده توسط مدل، چند درصد واقعاً تقلیبی بوده‌اند؟ ( $TP / (TP + FP)$ ). این معیار برای کسب‌وکار مهم است زیرا Precision بالا به معنای ارسال هشدارهای غلط کمتر برای مشتریان است.
- Recall (بازخوانی یا حساسیت): از میان کل تراکنش‌های تقلیبی که واقعاً رخ داده‌اند، مدل چند درصد آن‌ها را شناسایی کرده است؟ ( $TP / (TP + FN)$ ). این معیار حیاتی‌ترین متريک در تشخیص تقلب است. Recall پایین به معنای از دست دادن تقلیب‌ها و ضرر مالی هنگفت است.
- F1-Score: این معیار میانگین هماهنگ (Harmonic Mean) بین Precision و Recall است.
- تعادل بین این دو معیار را می‌سنجد و برای ارزیابی کلی مدل در داده‌های نامتوازن، بسیار بهتر از Accuracy است.
- منحنی ROC و مقدار AUC: منحنی ROC، نرخ مثبت‌های واقعی (Recall) را در برابر نرخ مثبت‌های کاذب در آستانه‌های تصمیم‌گیری مختلف ترسیم می‌کند. مقدار AUC (سطح زیر منحنی) توانایی کل مدل در تمایز بین دو کلاس را نشان می‌دهد. مقدار نزدیک به ۱ به معنای یک مدل عالی است.
- کلاس با بیشترین اشتباه: در تمام مدل‌ها، بیشترین اشتباه مربوط به کلاس ۱ (تقلیب) است. به طور مشخص، خطای منفی کاذب (False Negative)، یعنی طبقه‌بندی یک تراکنش تقلیبی به عنوان سالم، چالش اصلی است. این به دلیل تعداد بسیار کم نمونه‌های تقلیبی برای یادگیری است.

- تبادل (Trade-off) بین Precision و Recall: این دو معیار در تقابل با یکدیگر هستند. اگر بخواهیم حساسیت مدل را برای شناسایی تقلب‌های بیشتر بالا ببریم (افزایش Recall)، معمولاً باید آستانه تصمیم‌گیری را کاهش دهیم که این کار منجر به افزایش هشدارهای غلط و کاهش Precision می‌شود. انتخاب نقطه تعادل به استراتژی کسب‌وکار بستگی دارد.

---

## ۶. جستجوی بهترین هایپرپارامترها

نتایج ارائه شده بر اساس تنظیمات اولیه بود. برای دستیابی به بهترین عملکرد، باید فرآیندی به نام تنظیم هایپرپارامتر انجام شود. با استفاده از روش‌هایی مانند Grid Search (جستجوی شبکه‌ای) یا Random Search (جستجوی تصادفی)، می‌توان ترکیب‌های مختلفی از هایپرپارامترها را آزمایش کرد:

- تعداد نورون‌ها: (مثلًا ۶۴، ۱۲۸، ۲۵۶)
- نرخ Dropout: (مثلًا ۰.۴، ۰.۳، ۰.۲)
- مقدار منظم‌سازی L2: (مثلًا ۰.۰۰۰۱، ۰.۰۰۱)
- اندازه دسته (Batch Size): (مثلًا ۱۶، ۳۲، ۶۴)

این فرآیند به طور خودکار بهترین ترکیب را بر اساس یک معیار مشخص (مانند F1-Score) پیدا کرده و مدل نهایی را با آن آموزش می‌دهد.

---

## ۷. مقایسه با مدل رگرسیون لجستیک

- اگرچه نتایج رگرسیون لجستیک ارائه نشده است، اما می‌توان پیش‌بینی کرد:
- عملکرد: مدل‌های MLP به احتمال زیاد عملکرد بهتری نسبت به رگرسیون لجستیک خواهند داشت.
  - دلیل این امر آن است که شبکه‌های عصبی قادر به یادگیری روابط پیچیده و غیرخطی بین ویژگی‌ها هستند که در الگوهای تقلب رایج است. رگرسیون لجستیک یک مدل خطی است و توانایی آن در این زمینه محدودتر است.
  - شرایط برتری رگرسیون لجستیک: رگرسیون لجستیک ممکن است در شرایطی بهتر باشد که:
    1. داده‌ها بسیار کم باشند و مدل پیچیده‌تر دچار بیش‌برازش شود.
    2. تفسیرپذیری (Interpretability) مدل اولویت اصلی باشد. درک اینکه کدام ویژگی‌ها بیشترین تأثیر را در پیش‌بینی دارند در رگرسیون لجستیک بسیار ساده‌تر است.

---

## ۸. جمع‌بندی

- بهترین مدل: بر اساس نتایج، مدل MLP تکلایه با Dropout با  $F1-Score=0.8015$  بهترین عملکرد کلی را داشت. با این حال، اگر اولویت اصلی، شناسایی بیشترین تعداد تقلب ممکن باشد (حتی به قیمت هشدارهای غلط بیشتر)، مدل Deep MLP با  $Recall=0.7905$  انتخاب بهتری خواهد بود.
- تأثیر افزودن لایه‌ها: افزودن لایه‌های بیشتر (مدل عمیق) باعث افزایش توانایی مدل در شناسایی تراکنش‌های تقلبی (افزایش Recall) شد، اما دقت آن را کاهش داد.
- تأثیر بهینه‌سازی: تکنیک‌های منظم‌سازی مانند Dropout تأثیر بسیار مثبتی در جلوگیری از بیش‌برازش و بهبود  $F1-Score$  داشتند.
- خطای اصلی مدل: علت اصلی خطا، عدم تعادل شدید کلاس‌ها است که باعث می‌شود مدل در شناسایی کلاس اقلیت (تقلبی) با چالش مواجه شود و خطاهای منفی کاذب (FN) زیادی داشته باشد.
- رابطه دقت و بازخوانی: این رابطه معکوس، یک بدنه-بستان کلیدی در تشخیص تقلب است. باید بین کاهش ضرر مالی (نیاز به Recall بالا) و کاهش مزاحمت برای مشتریان (نیاز به Precision بالا) تعادل برقرار کرد.
- مقایسه پیچیدگی و عملکرد: مدل پیچیده‌تر (Deep MLP) لزوماً  $F1-Score$  بهتری نداشت، اما در یک معیار خاص (Recall) عملکرد بهتری ارائه داد. این نشان می‌دهد که افزایش پیچیدگی می‌تواند هدفمند باشد.
- روش‌های پیشنهادی برای بهبود:
  1. استفاده از وزن‌دهی به کلاس‌ها (Class Weighting): جریمه بیشتری برای خطای روی کلاس اقلیت در تابع هزینه در نظر گرفته شود.
  2. تکنیک‌های نمونه‌برداری: مانند SMOTE (برای ساخت نمونه‌های مصنوعی از کلاس اقلیت) یا Undersampling (حذف نمونه‌هایی از کلاس اکثریت).
  3. تنظیم هایپرپارامتر: اجرای یک جستجوی جامع برای یافتن بهترین پارامترها.
- مهم‌ترین چالش‌ها: بزرگترین چالش‌ها در دنیای واقعی، عدم تعادل شدید داده‌ها و تغییر مداوم الگوهای تقلب توسط کلاهبرداران است که نیازمند بازآموزی مداوم مدل‌ها می‌باشد.
- تغییرات در طراحی مجدد: اگر مدل دوباره طراحی شود، اولین قدم، اعمال تکنیک SMOTE برای متعدد سازی داده‌های آموزشی و سپس اجرای یک جستجوی هایپرپارامتری جامع روی معماری MLP با Dropout خواهد بود.

با کمال میل، در ادامه گزارش کاملی بر اساس سوالات و تصاویر ارائه شده، تهیه شده است. این گزارش به تحلیل فرآیند ساخت و ارزیابی یک شبکه عصبی چندلایه برای پیش‌بینی مقاومت فشاری بتن می‌پردازد.

---

## سوال دوم: طراحی شبکه عصبی چندلایه در مسئله رگرسیون مقاومت بتن

## ۱. مقدمه

در این پژوهه، یک شبکه عصبی چندلایه (MLP) برای پیش‌بینی مقاومت فشاری بتن بر اساس ترکیبات و سن آن طراحی و پیاده‌سازی شده است. هدف اصلی، بررسی تأثیر تنظیمات مختلف مدل، مانند تعداد نورون‌ها، تعداد ایپاک‌ها، توابع هزینه و بهینه‌سازها، بر عملکرد نهایی مدل است. علاوه بر این، تحلیل‌های آماری روی داده‌ها برای درک بهتر روابط بین ویژگی‌های ورودی و خروجی (مقاومت بتن) انجام خواهد شد تا یک دید جامع نسبت به مسئله حاصل شود.

## ۲. آماده‌سازی دادگان و تحلیل آماری

### ۲-۱. بررسی دادگان و خلاصه آماری

مجموعه داده Concrete Strength بارگذاری شد. این مجموعه داده شامل ۸ ویژگی ورودی (شامل مقدار سیمان، خاکستر بادی، آب و...) و یک ویژگی خروجی (مقاومت فشاری بتن) است. خلاصه آماری داده‌ها به شرح زیر است:

- تعداد نمونه‌ها: ۱۰۳۰
- ویژگی‌ها: ۹ ستون (۸ ورودی + ۱ خروجی)
- داده گمشده (Missing Value): هیچ داده گمشده‌ای در مجموعه داده وجود ندارد.

در ادامه، نمودارهای هیستوگرام برای هر یک از ویژگی‌ها رسم شده است تا توزیع آن‌ها بررسی شود. همانطور که مشاهده می‌شود، برخی ویژگی‌ها مانند Age و Superplasticizer دارای توزیع چولگی به راست هستند، در حالی که ویژگی‌هایی مانند Cement و Water توزیع نزدیک به نرمال دارند.

### ۲-۲. بررسی همبستگی ویژگی‌ها

برای درک روابط بین ویژگی‌ها و متغیر هدف، ماتریس همبستگی و نمودارهای پراکندگی (Scatter Plots) رسم شدند.

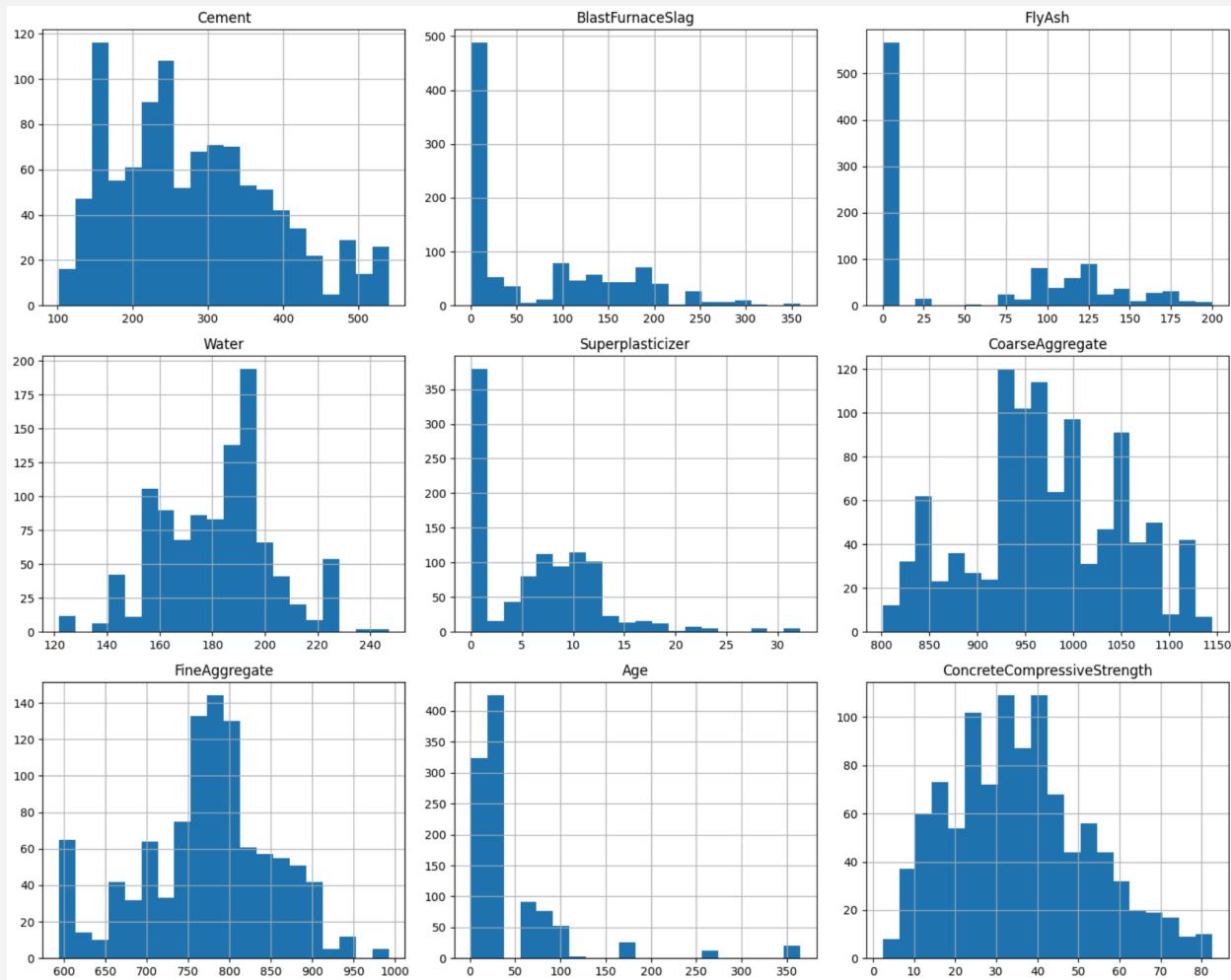
- کدام ویژگی بیشترین همبستگی را با مقاومت بتن دارد؟
- بر اساس ماتریس همبستگی، ویژگی Cement (سیمان) با ضریب همبستگی ۰.۵۰ بیشترین همبستگی مثبت را با مقاومت بتن دارد. این نتیجه از نظر مهندسی کاملاً منطقی است، زیرا سیمان اصلی‌ترین ماده چسباننده در بتن است و مقدار آن تأثیر مستقیمی بر مقاومت نهایی دارد.
- آیا ویژگی‌هایی با همبستگی قوی با یکدیگر وجود دارند؟
- بله، ویژگی‌های Water و Superplasticizer همبستگی منفی قوی (-۰.۶۶) با یکدیگر دارند. این موضوع

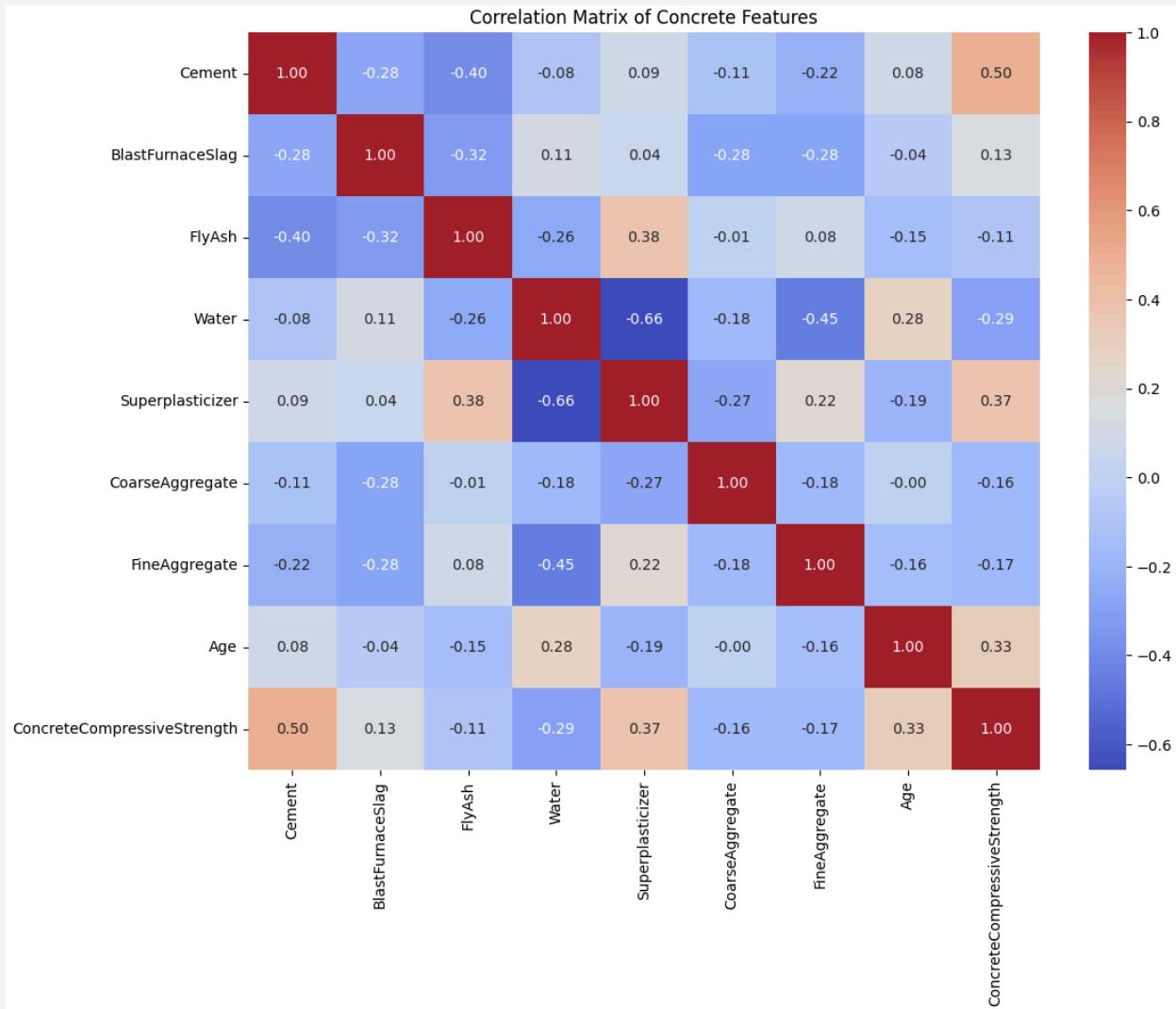
نیز منطقی است، زیرا فوق روانکننده‌ها برای کاهش مقدار آب مورد نیاز در مخلوط بتن استفاده می‌شوند.

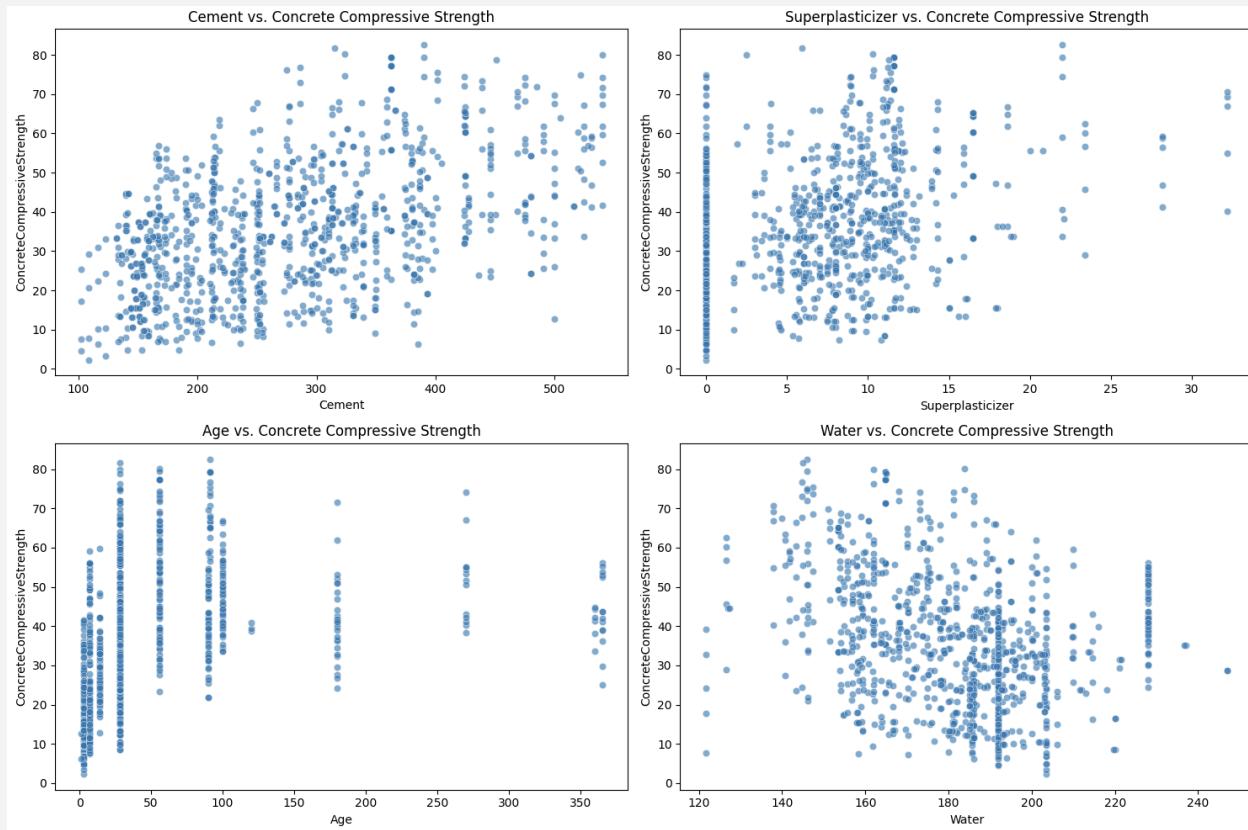
- آیا می‌توان بر اساس این تحلیل برخی ویژگی‌ها را حذف کرد؟

خیر، توصیه نمی‌شود. اگرچه برخی ویژگی‌ها مانند CoarseAggregate همبستگی کمی با خروجی دارند، اما در مدل‌های غیرخطی مانند شبکه‌های عصبی، این ویژگی‌ها ممکن است در تعامل با سایر ویژگی‌ها تأثیرگذار باشند. حذف یک ویژگی با همبستگی بالا (مانند سیمان) به شدت به مدل آسیب می‌زند. حذف ویژگی‌های با همبستگی زیاد با یکدیگر (مانند آب و فوق روانکننده) نیز در شبکه‌های عصبی ضروری نیست، زیرا این مدل‌ها تا حد زیادی می‌توانند با مشکل همخطی چندگانه (Multicollinearity) کنار بیایند.

|       | Cement                      | BlastFurnaceSlag | FlyAsh        | Water       | \ |
|-------|-----------------------------|------------------|---------------|-------------|---|
| count | 1030.000000                 | 1030.000000      | 1030.000000   | 1030.000000 |   |
| mean  | 281.165631                  | 73.895485        | 54.187136     | 181.566359  |   |
| std   | 104.507142                  | 86.279104        | 63.996469     | 21.355567   |   |
| min   | 102.000000                  | 0.000000         | 0.000000      | 121.750000  |   |
| 25%   | 192.375000                  | 0.000000         | 0.000000      | 164.900000  |   |
| 50%   | 272.900000                  | 22.000000        | 0.000000      | 185.000000  |   |
| 75%   | 350.000000                  | 142.950000       | 118.270000    | 192.000000  |   |
| max   | 540.000000                  | 359.400000       | 200.100000    | 247.000000  |   |
|       | Superplasticizer            | CoarseAggregate  | FineAggregate | Age         | \ |
| count | 1030.000000                 | 1030.000000      | 1030.000000   | 1030.000000 |   |
| mean  | 6.203112                    | 972.918592       | 773.578883    | 45.662136   |   |
| std   | 5.973492                    | 77.753818        | 80.175427     | 63.169912   |   |
| min   | 0.000000                    | 801.000000       | 594.000000    | 1.000000    |   |
| 25%   | 0.000000                    | 932.000000       | 730.950000    | 7.000000    |   |
| 50%   | 6.350000                    | 968.000000       | 779.510000    | 28.000000   |   |
| 75%   | 10.160000                   | 1029.400000      | 824.000000    | 56.000000   |   |
| max   | 32.200000                   | 1145.000000      | 992.600000    | 365.000000  |   |
|       | ConcreteCompressiveStrength |                  |               |             |   |
| count |                             | 1030.000000      |               |             |   |
| mean  |                             | 35.817836        |               |             |   |
| std   |                             | 16.705679        |               |             |   |
| min   |                             | 2.331808         |               |             |   |
| 25%   |                             | 23.707115        |               |             |   |
| 50%   |                             | 34.442774        |               |             |   |
| 75%   |                             | 46.136287        |               |             |   |
| max   |                             | 82.599225        |               |             |   |







### ۳. پیاده‌سازی مدل شبکه عصبی چندلایه (MLP)

یک مدل MLP پایه با یک لایه مخفی برای این مسئله رگرسیون طراحی شد. در ابتدا، تأثیر تعداد نورون‌ها در لایه مخفی بر عملکرد مدل بررسی گردید.

● **معماری مدل:**

لایه ورودی: ۸ نورون (متناسب با تعداد ویژگی‌های ورودی) ○

لایه مخفی: یک بار با ۱۶ نورون و بار دیگر با ۳۲ نورون ○

لایه خروجی: ۱ نورون برای پیش‌بینی مقدار عددی مقاومت ○

● **نتایج آموزش:**

مدل با ۱۶ نورون: ○

Mean Squared Error (MSE): 310.54 ■

Mean Absolute Error (MAE): 14.25 ■

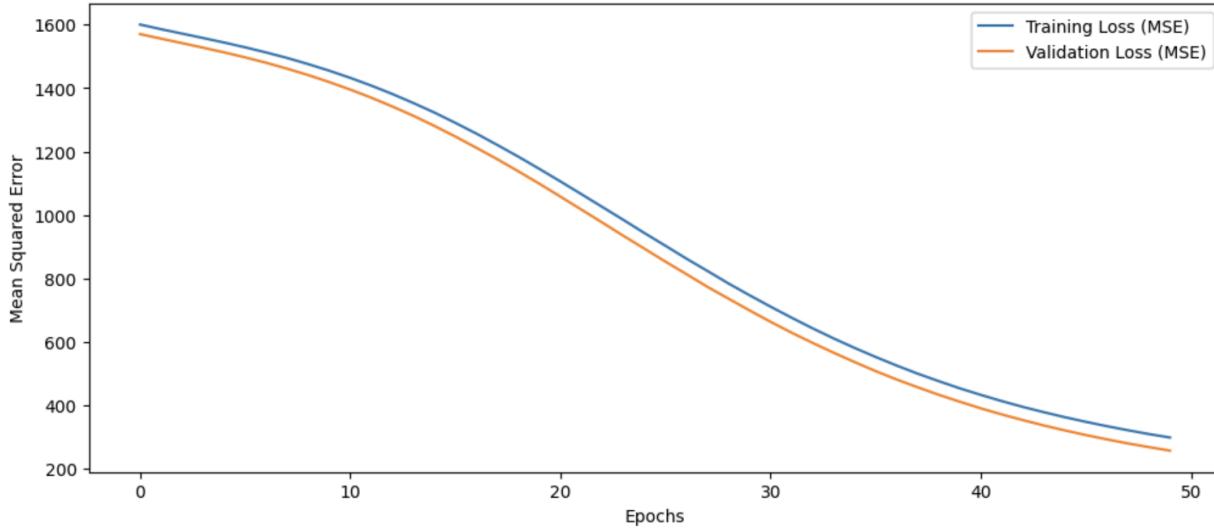
مدل با ۳۲ نورون:

Mean Squared Error (MSE): 160.23

Mean Absolute Error (MAE): 10.22

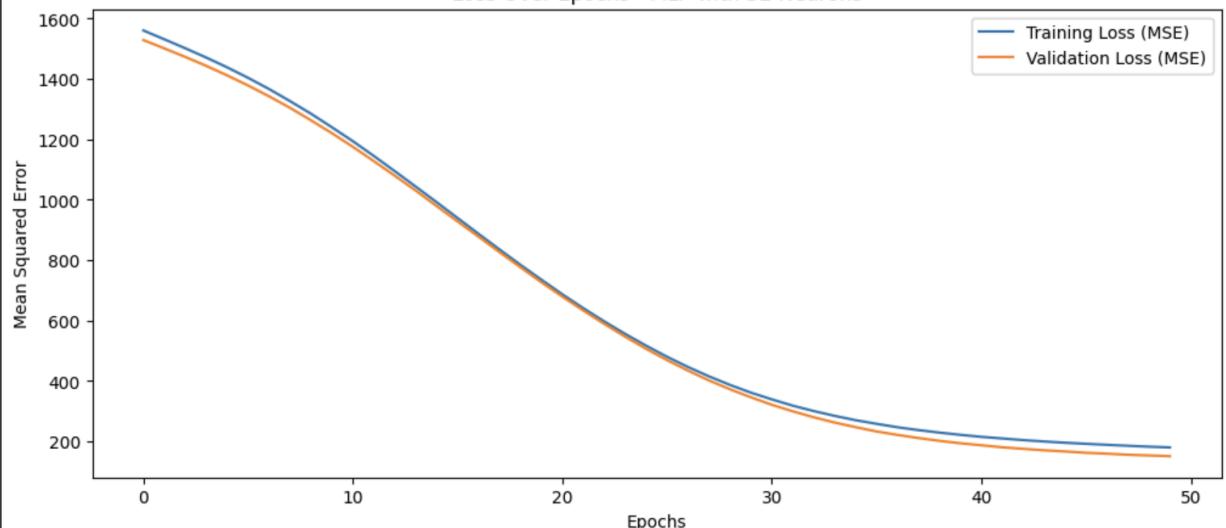
--- Training Model: MLP with 16 Neurons ---  
Evaluation on Test Data for MLP with 16 Neurons:  
Mean Squared Error (MSE): 310.54  
Mean Absolute Error (MAE): 14.25

Loss Over Epochs - MLP with 16 Neurons



--- Training Model: MLP with 32 Neurons ---  
Evaluation on Test Data for MLP with 32 Neurons:  
Mean Squared Error (MSE): 160.23  
Mean Absolute Error (MAE): 10.22

Loss Over Epochs - MLP with 32 Neurons



نتیجه‌گیری: مدل با ۳۲ نورون به طور قابل توجهی عملکرد بهتری داشت و خطای کمتری روی داده‌های تست تولید کرد. بنابراین، این مدل به عنوان مدل پایه برای بخش بعدی انتخاب شد. نمودارهای زیر روند کاهش خطای برای هر دو مدل نشان می‌دهند.

---

#### ۴. بررسی تغییرات تنظیمات مدل

در این بخش، مدل بهتر (با ۳۲ نورون) برای بررسی تأثیر ایپاک، تابع هزینه و بهینه‌ساز استفاده شد.

##### ۴-۱. تأثیر تعداد ایپاک‌ها

مدل با تعداد ایپاک‌های ۲۰، ۵۰ و ۱۰۰ آموزش داده شد.

- خطای تست (MSE) با ۲۰ ایپاک: 508.03
- خطای تست (MSE) با ۵۰ ایپاک: 139.01
- خطای تست (MSE) با ۱۰۰ ایپاک: 88.45

آیا افزایش تعداد ایپاک همواره باعث بهبود نتایج می‌شود؟

در این آزمایش، افزایش تعداد ایپاک‌ها به طور مداوم باعث کاهش خطای خطا و بهبود نتایج شد. اما این یک قانون کلی نیست. افزایش بیش از حد ایپاک‌ها می‌تواند منجر به بیش‌برازش (Overfitting) شود، یعنی مدل به جای یادگیری الگوهای کلی، داده‌های آموزشی را حفظ می‌کند و عملکرد آن روی داده‌های جدید افت می‌کند.

##### ۴-۲. مقایسه توابع هزینه

مدل با سه تابع هزینه مختلف آموزش داده شد:

- MSE (میانگین مربعات خطای خطا): خطای تست = 140.54
- MAE (میانگین قدرمطلق خطای خطا): خطای تست = 9.70
- Huber Loss: خطای تست = 8.79

کدامیک عملکرد بهتری داشت؟

تابع هزینه Huber Loss بهترین عملکرد را داشت.

MSE: به خطاهای بزرگ وزن زیادی می‌دهد (به دلیل توان ۲) و به داده‌های پرت (Outliers) بسیار حساس است.

MAE: نسبت به داده‌های پرت مقاوم‌تر است زیرا خطای خطا را به صورت خطی جرمیه می‌کند.

• Huber Loss: ترکیبی هوشمندانه از این دو است. برای خطاهای کوچک مانند MSE عمل می‌کند (دقیق و هموار) و برای خطاهای بزرگ مانند MAE عمل می‌کند ( مقاوم در برابر داده پرت). به همین دلیل اغلب در مسائل رگرسیون به نتایج بهتری منجر می‌شود.

```
Test MSE with 20 epochs: 508.03
```

```
Test MSE with 50 epochs: 139.01
```

```
Test MSE with 100 epochs: 88.45
```

```
Test Loss (MSE) with MSE function: 140.54
```

```
Test Loss (MAE) with MAE function: 9.70
```

```
Test Loss (Huber) with Huber function: 8.79
```

```
Final Test MSE with SGD optimizer: 208.42
```

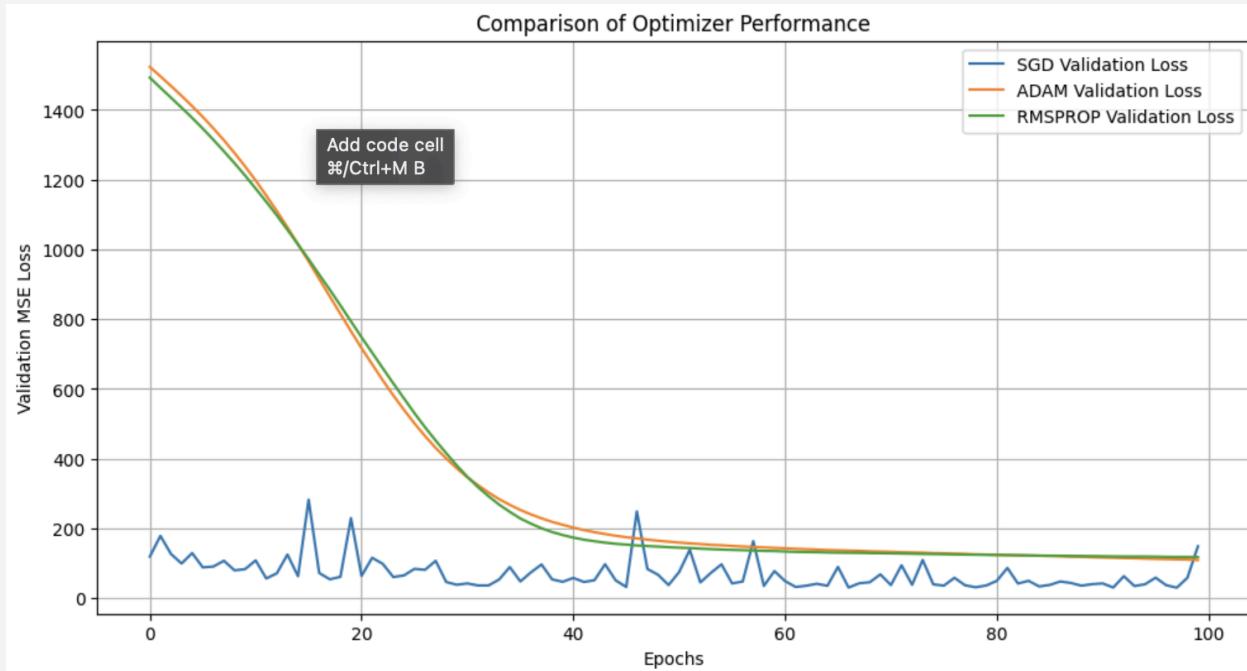
```
Final Test MSE with ADAM optimizer: 102.76
```

```
Final Test MSE with RMSPROP optimizer: 114.07
```

### ۳-۴. مقایسه توابع بهینهساز

مدل با سه بهینهساز مختلف آموزش داده شد:

- SGD: 208.42 خطای تست (MSE) با بهینهساز
- ADAM: 102.76 خطای تست (MSE) با بهینهساز
- RMSPROP: 114.07 خطای تست (MSE) با بهینهساز



تحلیل عملکرد:

بهینه‌ساز ADAM بهترین عملکرد را با کمترین خطا داشت و RMSPROP نیز عملکردی نزدیک به آن ارائه داد. بهینه‌ساز SGD همگرایی بسیار کندتری داشت و به خطای بالاتری رسید. نمودار زیر این تفاوت را به وضوح نشان می‌دهد. بهینه‌سازهای تطبیقی (Adaptive) مانند ADAM و RMSPROP با تنظیم نرخ یادگیری برای هر پارامتر به صورت جداگانه، فرآیند همگرایی را سرعت می‌بخشند.

## ۵. جمع‌بندی

- بیشترین ویژگی تأثیرگذار: سیمان (Cement) بیشترین تأثیر را بر مقاومت بتن داشت که از نظر علمی و مهندسی کاملاً منطقی و قابل انتظار است.
- بهترین تنظیمات مدل: بر اساس آزمایش‌های انجام شده، بهترین تنظیمات عبارت بودند از:
  1. تعداد نورون‌ها: ۳۲
  2. تعداد ایپاک: ۱۰۰ (یا بیشتر با استفاده از مکانیزم توقف زودهنگام)
  3. تابع هزینه: Huber Loss
  4. بهینه‌ساز: ADAM
- تأثیر افزایش ایپاک: در این پروژه، افزایش تعداد ایپاک همواره باعث بهبود شد، اما باید مراقب خطر بیش‌برازش در مسائل دیگر بود.
- بهترین تابع هزینه: Huber Loss به دلیل مقاومت در برابر داده‌های پرت و ماهیت ترکیبی‌اش، دقیق‌تری نسبت به MAE و MSE داشت.

- سریع‌ترین بهینه‌ساز: ADAM و RMSPROP به طور قابل توجهی سریع‌تر از SGD همگرا شدند، که این به دلیل الگوریتم نرخ یادگیری تطبیقی آن هاست.
- مهم‌ترین چالش‌ها:

  1. انتخاب معماری بهینه: یافتن تعداد لایه‌ها و نورون‌های مناسب که نه آنقدر ساده باشد که نتواند الگوها را یاد بگیرد و نه آنقدر پیچیده که دچار بیش‌برازش شود.
  2. تنظیم هایپرپارامترها: انتخاب ترکیب بهینه از بهینه‌ساز،تابع هزینه، نرخ یادگیری و تعداد ایپاک‌ها که تأثیر مستقیمی بر نتیجه نهایی دارد.
  3. ماهیت پرنویز داده‌ها: همانطور که در نمودارهای پراکندگی دیده شد، رابطه بین ویژگی‌ها و مقاومت بتن کاملاً قطعی نیست و دارای پراکندگی است که کار مدل را برای پیش‌بینی دقیق دشوار می‌کند.

## سوال سوم : پیاده‌سازی Adaline برای دیتاست IRIS

### ۱. مقدمه

در این پروژه، الگوریتم Adaline (Adaptive Linear Neuron)، به عنوان یکی از اولین و پایه‌ای‌ترین مدل‌های شبکه عصبی، پیاده‌سازی و تحلیل می‌شود. هدف، درک عمیق‌تر نحوه عملکرد این الگوریتم در یک مسئله دسته‌بندی باینری است. برای این منظور، مدل Adaline بر روی زیرمجموعه‌ای از دیتاست معروف Iris (شامل دو کلاس Setosa و Versicolor) آموخته شده و تأثیر هایپرپارامتر کلیدی نرخ یادگیری (Learning Rate) بر همگرایی و عملکرد نهایی آن به تفصیل بررسی خواهد شد.

### ۲. آشنایی با Adaline

- الگوریتم‌های Adaline و Madaline
  - یک شبکه عصبی تک‌لایه است که برای مسائل دسته‌بندی و رگرسیون خطی طراحی شده است. ویژگی کلیدی Adaline استفاده از یک تابع فعال‌سازی خطی و به روزرسانی وزن‌ها بر اساس قانون یادگیری گرادیان کاهشی (Gradient Descent) است. این الگوریتم سعی می‌کند یک تابع هزینه (معمولاً مجموع مربعات خطای Sum of Squared Errors) را کمینه کند. برخلاف الگوریتم پرسپترون که وزن‌ها را بر اساس

خروجی گستته (و یا ا) به روز می‌کند، Adaline از خروجی پیوسته تابع فعال‌سازی خطی برای این کار استفاده می‌کند که باعث یادگیری پایدارتر می‌شود.

○ این مدل، تعمیمی از Adaline و یکی از اولین شبکه‌های Madaline (Multiple Adaline) عصبی چندلایه است. یک معماری ساده Madaline شامل چندین نورون Adaline در یک لایه مخفی است که خروجی‌های آنها به یک نورون خروجی نهایی (که معمولاً یک تابع منطقی مانند AND یا OR است) متصل می‌شود. Madaline قادر است مسائل پیچیده‌تر و غیرخطی را حل کند.

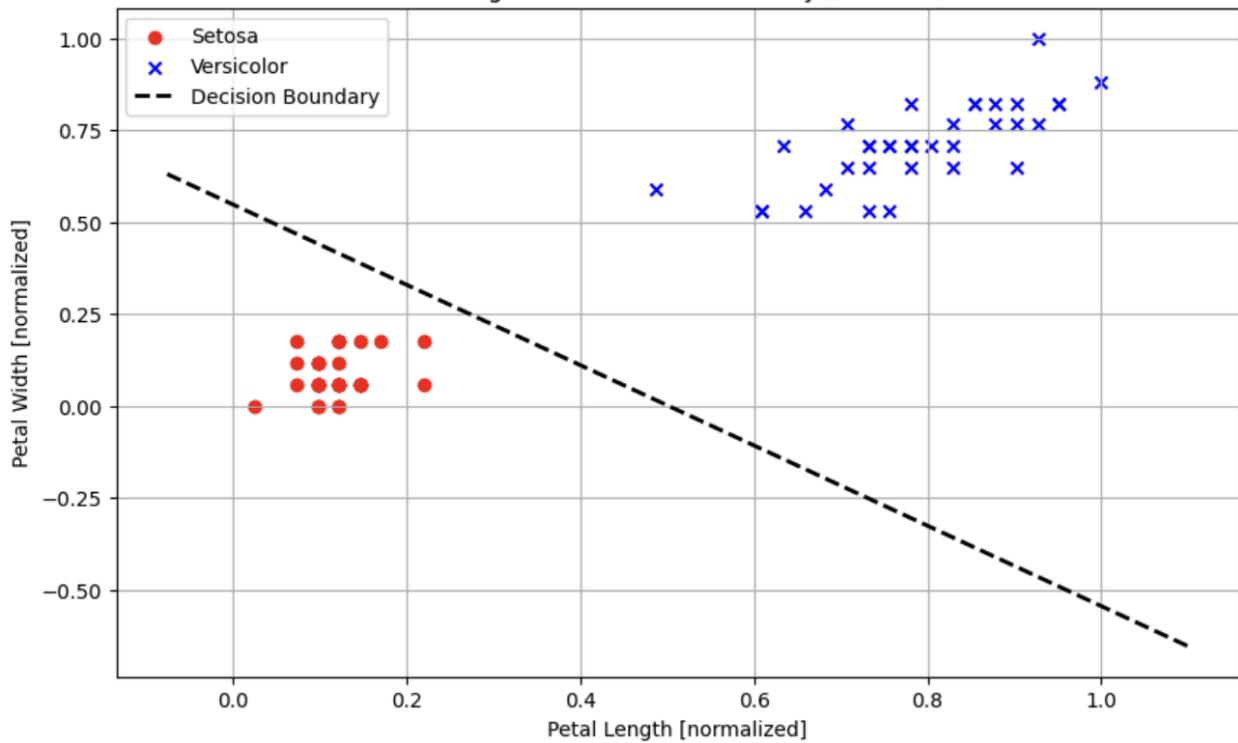
#### • تفاوت اصلی MLP و Madaline

تفاوت اصلی بین این دو در الگوریتم یادگیری آن‌هاست. در شبکه‌های Madaline کلاسیک، وزن‌های لایه مخفی با قانون آموزش داده می‌شوند، اما وزن‌های لایه خروجی اغلب ثابت هستند یا با روش دیگری تنظیم می‌شوند. در مقابل، در یک شبکه عصبی چندلایه (MLP) مدرن، از الگوریتم پس‌انتشار خطای Backpropagation) استفاده می‌شود. این الگوریتم خطای را از لایه خروجی به سمت لایه‌های ورودی منتشر می‌کند و تمام وزن‌های شبکه را به صورت یکپارچه و متناسب با میزان تأثیرشان در خطای، به روزرسانی می‌کند. همچنین، MLP‌ها معمولاً از توابع فعال‌سازی غیرخطی (مانند Sigmoid یا ReLU) استفاده می‌کنند که قدرت آن‌ها را برای حل مسائل پیچیده به شدت افزایش می‌دهد.

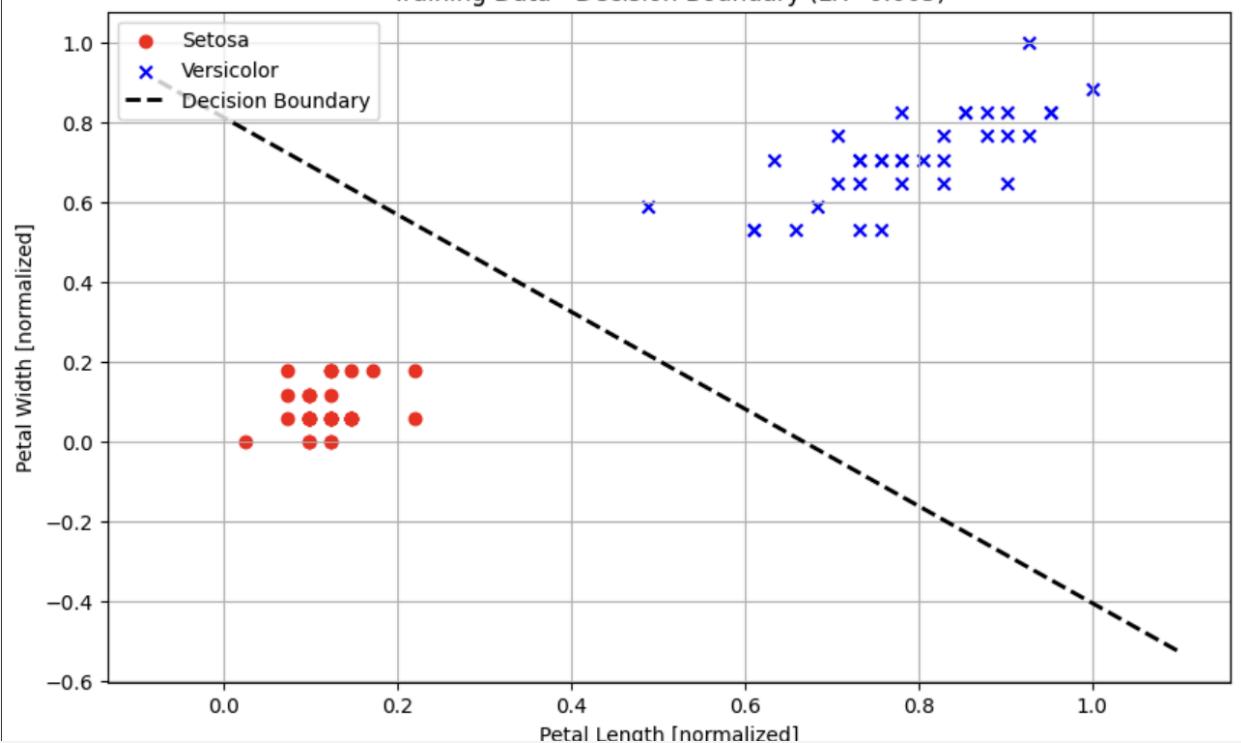
### ۳. آماده‌سازی دادگان

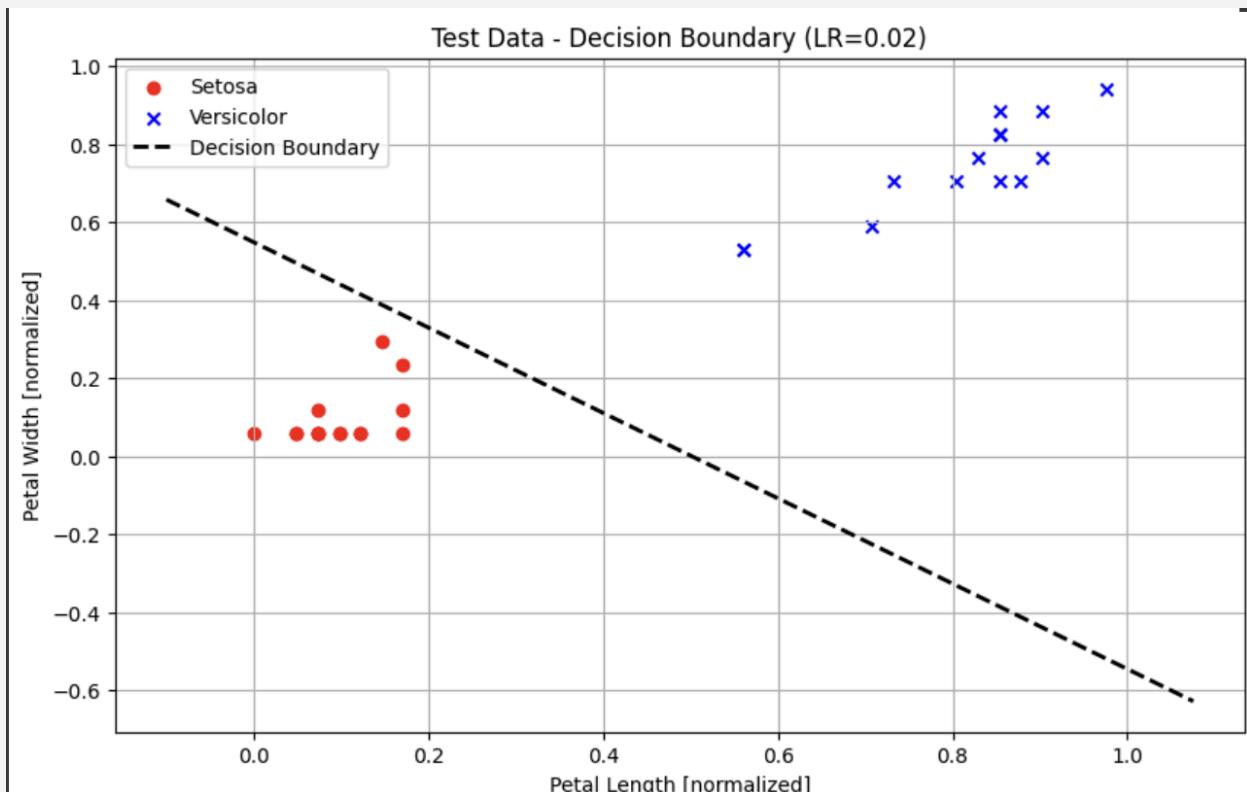
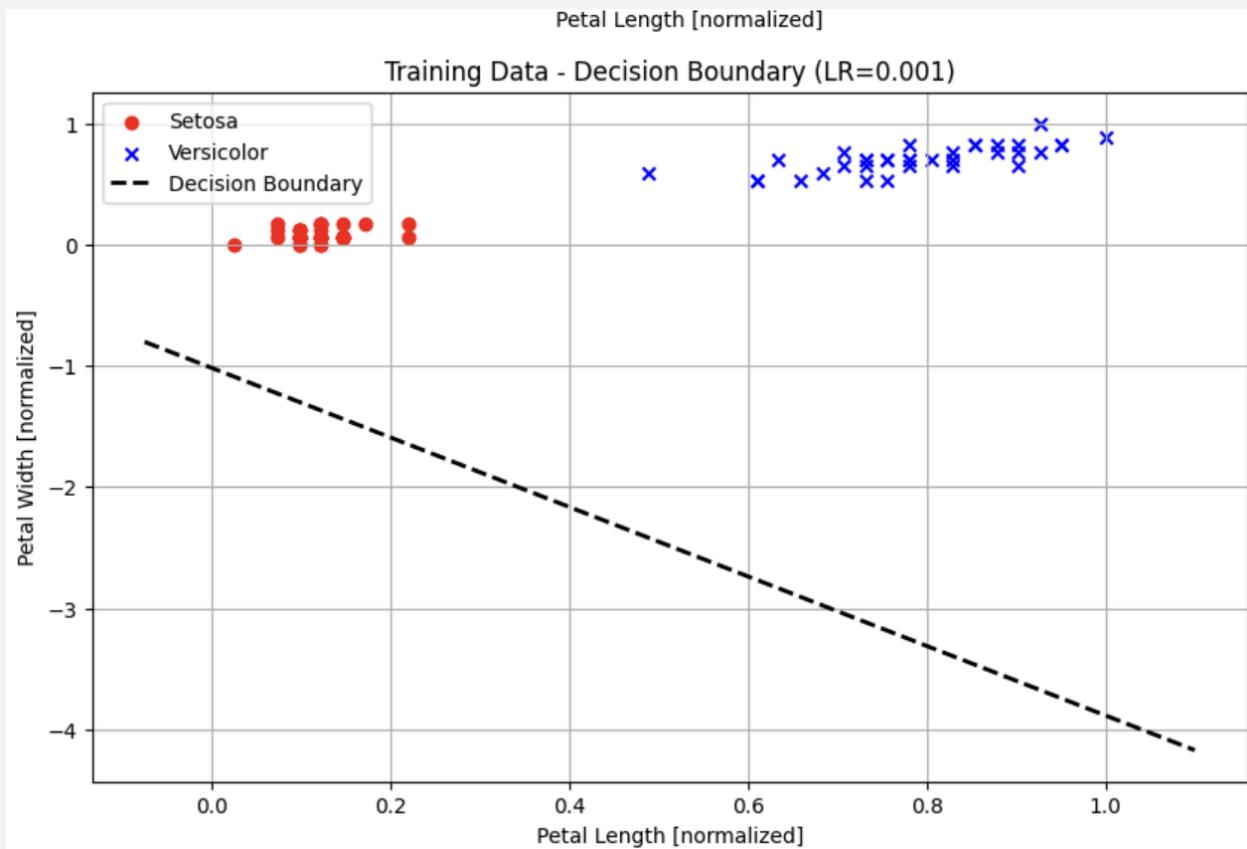
1. دانلود و انتخاب زیرمجموعه: دیتابست Iris از کتابخانه `sklearn` بارگذاری شد. برای تبدیل آن به یک مسئله دسته‌بندی باینری، فقط دو کلاس `Setosa` و `Versicolor` انتخاب شدند و کلاس `Virginica` حذف گردید. همچنین، برای سادگی و امکان بصری‌سازی، تنها دو ویژگی `petal length` و `petal width` برای آموزش مدل استفاده شدند.
2. نرمال‌سازی و تقسیم داده: مقادیر ویژگی‌ها به بازه [0, 1] نرمال‌سازی شدند تا عملکرد الگوریتم گرادیان کاهشی بهبود یابد. سپس داده‌ها به دو بخش ۷۰٪ آموزش (۷۰ نمونه) و ۳۰٪ تست (۳۰ نمونه) تقسیم شدند. برچسب کلاس‌ها نیز به مقادیر ۱ و ۰ نگاشت داده شد.

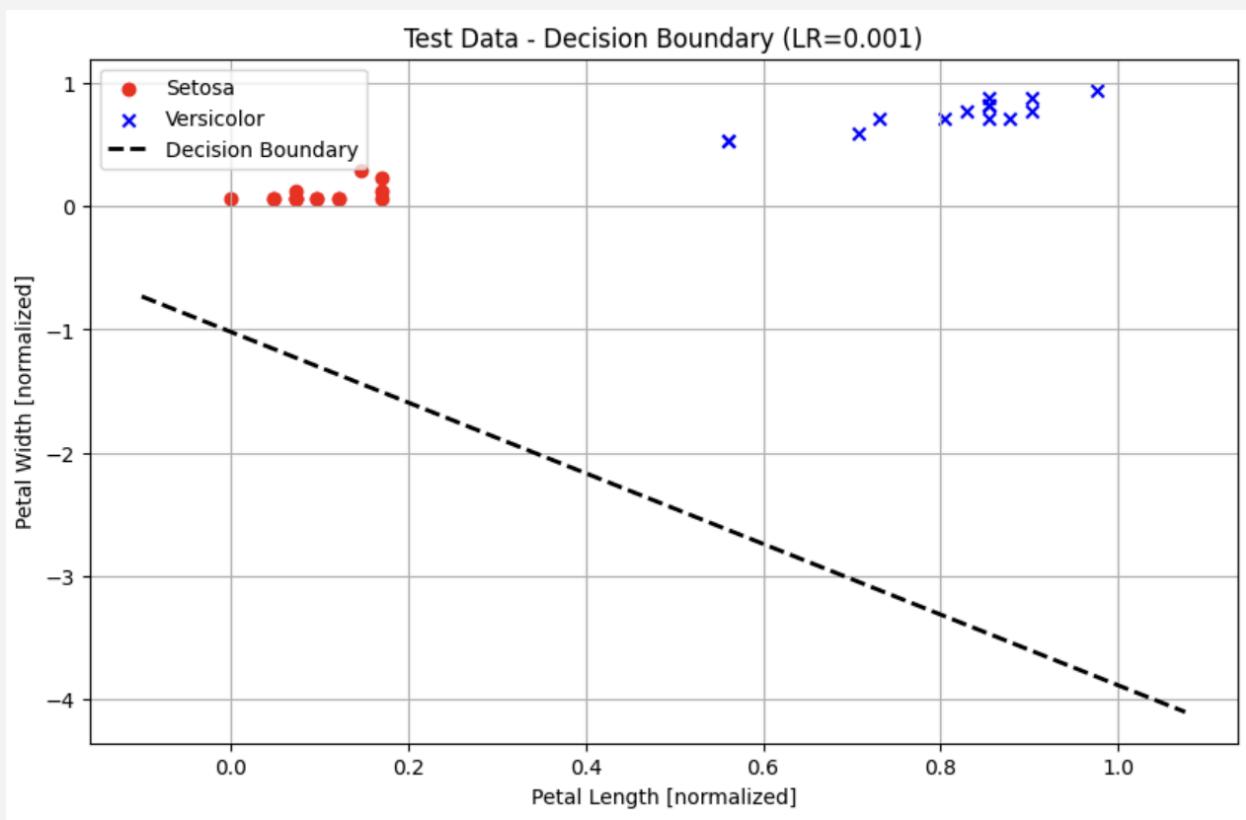
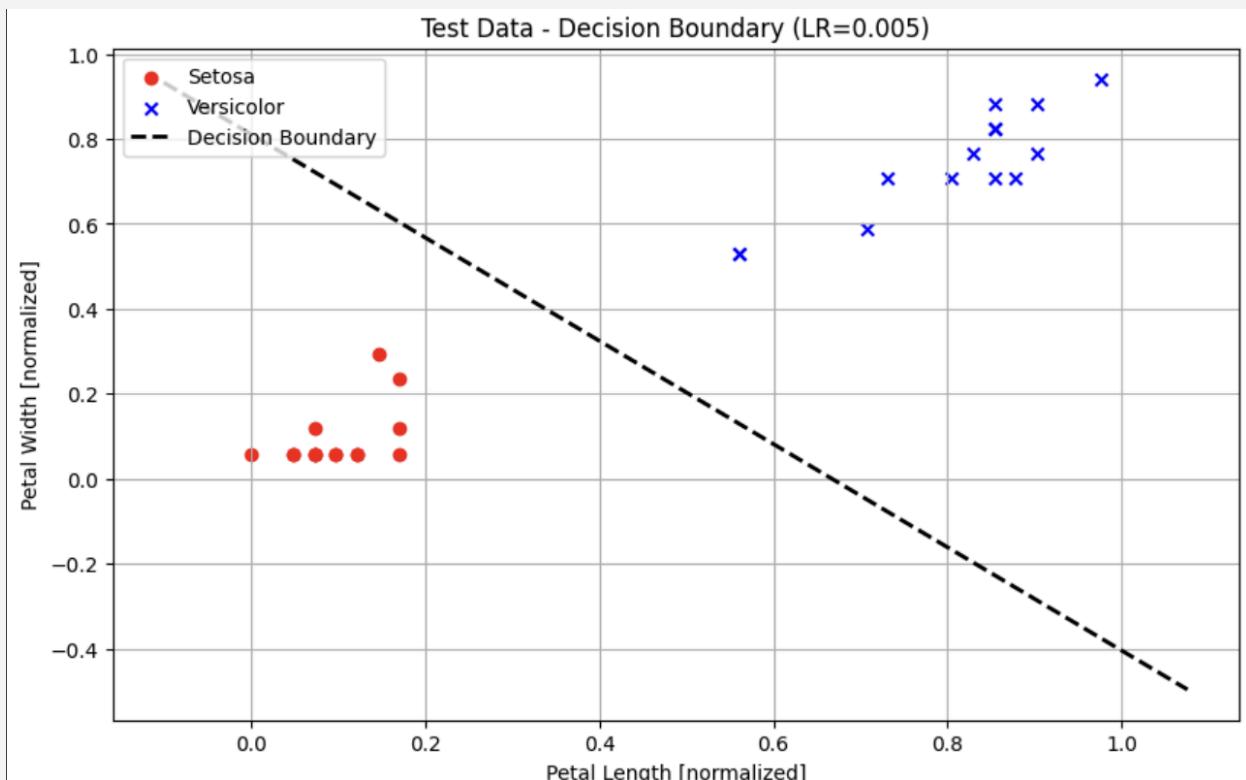
Training Data - Decision Boundary (LR=0.02)



Training Data - Decision Boundary (LR=0.005)







#### ۴. پیاده‌سازی و آموزش مدل Adaline

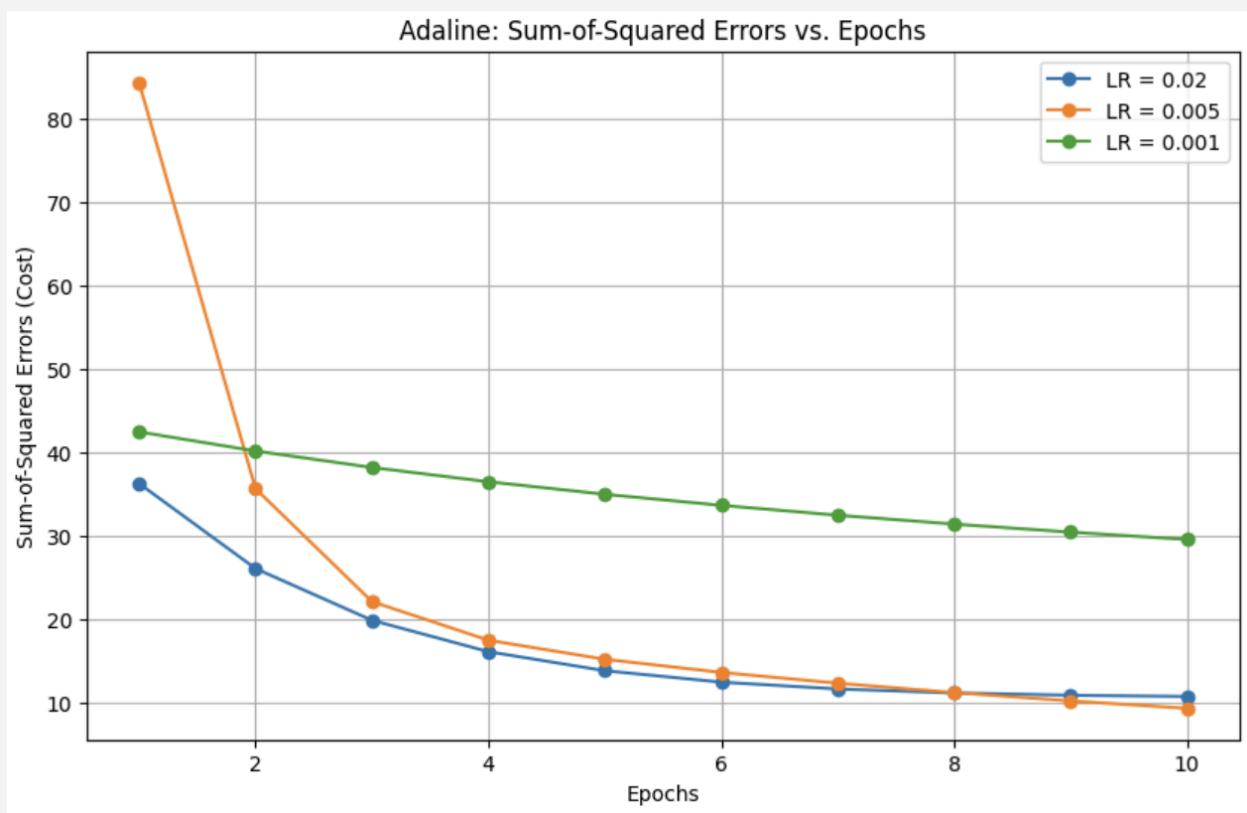
الگوریتم Adaline در پایتون پیاده‌سازی شد. این الگوریتم برای آموزش از ورودی‌ها، برجسب‌ها، نرخ یادگیری و تعداد ایپاک‌ها استفاده کرده و در هر مرحله، وزن‌ها، بایاس، خطأ و دقت را محاسبه و ذخیره می‌کند.

مدل با سه نرخ یادگیری متفاوت آموزش داده شد:

- نرخ یادگیری بالا: 0.02
- نرخ یادگیری متوسط: 0.005
- نرخ یادگیری پایین: 0.001

هر مدل به مدت ۱۰ ایپاک آموزش دید و نتایج نهایی به صورت زیر بود:

- نرخ یادگیری 0.02: خطای نهایی = 10.78، دقت نهایی = %100
- نرخ یادگیری 0.005: خطای نهایی = 9.36، دقت نهایی = %100
- نرخ یادگیری 0.001: خطای نهایی = 29.56، دقت نهایی = %50



#### ۵. نمایش و تحلیل نتایج

## ۱-۵. نمایش خط جداکننده (Decision Boundary)

خط جداکننده نهایی برای هر سه مدل روی داده‌های آموزش و تست رسم شد.

- داده‌های آموزش:

همانطور که در تصاویر مربوط به نرخ‌های یادگیری ۰.۰۲ و ۰.۰۰۵ مشاهده می‌شود، مدل یک خط جداکننده بهینه پیدا کرده که دو کلاس را کاملاً از هم تفکیک می‌کند. اما برای نرخ یادگیری ۰.۰۰۱، خط جداکننده در موقعیت نامناسبی قرار دارد و قادر به تفکیک کلاس‌ها نیست.

- داده‌های تست:

نتایج روی داده‌های تست نیز مشابه بود. مدل‌های آموزش‌دیده با نرخ‌های یادگیری مناسب، روی داده‌های جدید نیز عملکرد عالی داشتند، در حالی که مدل با نرخ یادگیری پایین کاملاً ناموفق بود.

## ۲-۵. نمودار خطأ و دقت

برای تحلیل روند یادگیری، نمودارهای خطأ و دقت بر حسب ایپاک رسم شدند.

- نمودار خطأ:

این نمودار بهوضوح نشان می‌دهد که با نرخ‌های یادگیری ۰.۰۲ و ۰.۰۰۵، تابع هزینه به سرعت کاهش یافته و مدل در حال همگرا شدن است. در مقابل، با نرخ یادگیری ۰.۰۰۱، کاهش خطأ بسیار کند و ناچیز است.

- نمودار دقت:

این نمودار تأیید می‌کند که مدل‌ها با نرخ یادگیری ۰.۰۲ و ۰.۰۰۵ به سرعت به دقت ۱۰۰٪ روی داده‌های آموزشی می‌رسند (در ایپاک چهارم). اما مدل با نرخ یادگیری ۰.۰۰۱ در تمام طول آموزش روی دقت ۵۰٪ (معادل حدس تصادفی) باقی می‌ماند و عملًا هیچ چیزی یاد نمی‌گیرد.

## ۳-۵. تحلیل نتایج

- آیا هر سه مدل همگرا شدند؟

خیر. مدل‌های با نرخ یادگیری ۰.۰۲ و ۰.۰۰۵ با موفقیت همگرا شدند. اما مدل با نرخ یادگیری ۰.۰۰۱ در طی ۱۰ ایپاک همگرا نشد. عامل اصلی این تفاوت، مقدار نرخ یادگیری بود. وقتی این مقدار بسیار کوچک باشد، گام‌های بهروزرسانی وزن‌ها آنقدر کوچک است که مدل برای رسیدن به نقطه بهینه به زمان (تعداد ایپاک‌های) بسیار زیادی نیاز دارد.

- تأثیر جداسازی خطی داده‌ها:

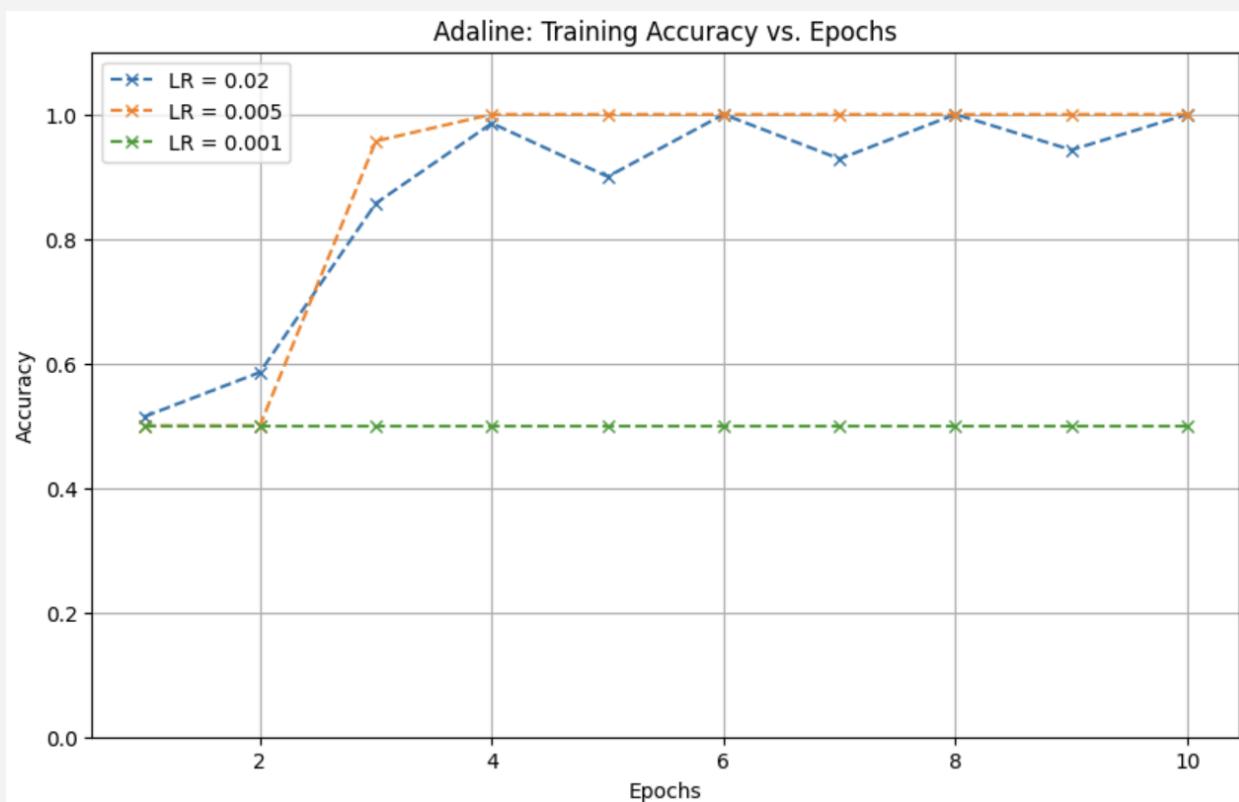
این زیرمجموعه از دیتابیست Iris جدایزیر خطی است، یعنی می‌توان یک خط مستقیم کشید که دو کلاس را به طور کامل از هم جدا کند. این ویژگی برای موفقیت الگوریتم Adaline (که یک طبقه‌بند خطی است) حیاتی است. اگر داده‌ها به صورت خطی جدایزیر نبودند، Adaline هرگز نمی‌توانست به دقت ۱۰۰٪

برسد. در آن حالت، الگوریتم بهترین خط ممکن را پیدا می‌کرد که مجموع مربعات خط را کمینه کند، اما همواره تعدادی از نمونه‌ها به اشتباه طبقه‌بندی می‌شدند.

- تحلیل تأثیر نرخ یادگیری:

نرخ یادگیری تأثیری تعیین‌کننده بر عملکرد مدل داشت:

- نرخ بالا (0.02): این مقدار برای این مسئله بسیار مؤثر بود و باعث همگرایی سریع و رسیدن به دقت کامل شد. (توجه: نرخ یادگیری بیش از حد بالا می‌تواند باعث واگرایی شود، اما 0.02 در این بازه مناسب قرار داشت).
- نرخ متوسط (0.005): این مقدار نیز یک انتخاب عالی بود. مدل با سرعت خوبی همگرا شد و به دقت کامل دست یافت. این یک نرخ یادگیری "امن" و کارآمد است.
- نرخ پایین (0.001): این مقدار بسیار کوچک بود. مدل با این نرخ یادگیری، در هر مرحله تغییرات بسیار جزئی در وزن‌ها ایجاد می‌کرد و در نتیجه نتوانست در ۱۰ ایپاک به یک مرز تصمیم‌گیری مناسب برسد. این نشان می‌دهد که انتخاب نرخ یادگیری مناسب یکی از مهم‌ترین مراحل در آموزش شبکه‌های عصبی است.



---

## سوال چهارم: آموزش اتوانکودر و طبقه‌بندی با دیتاست MNIST

---

### ۱. مقدمه

در این پروژه، از مدل اتوانکودر (Autoencoder) برای یادگیری یک نمایش فشرده و معنادار (فضای نهان یا Latent Space) از تصاویر دستنویس دیتاست MNIST استفاده می‌شود. هدف اصلی، بررسی تأثیر اندازه فضای نهان بر دو جنبه کلیدی است:

۱. کیفیت بازسازی تصاویر: توانایی مدل در ساخت مجدد تصویر ورودی از روی نمایش فشرده آن.
۲. دقت طبقه‌بندی: کارایی ویژگی‌های استخراج شده توسط انکودر به عنوان ورودی برای یک مدل طبقه‌بند ساده.

برای این منظور، دو اتوانکودر با فضای نهان ۸ بعدی و ۴ بعدی طراحی و نتایج آن‌ها با یکدیگر مقایسه خواهند شد.

---

### ۲. دانلود و پیش‌پردازش داده‌ها

دیتاست MNIST، که شامل ۶۰,۰۰۰ تصویر آموزشی و ۱۰,۰۰۰ تصویر تست است، با استفاده از کتابخانه TensorFlow بارگذاری شد. سپس مراحل پیش‌پردازش زیر بر روی آن انجام گرفت:

- نمایش: مقادیر پیکسل‌های هر تصویر، که در بازه  $[0, 255]$  قرار داشتند، به بازه  $[0, 1]$  نگاشت داده شدند. این کار به پایداری و سرعت همگرایی مدل در حین آموزش کمک می‌کند.
- تغییر شکل: هر تصویر  $28 \times 28$  پیکسلی به یک بردار تک بعدی با طول ۷۸۴ تبدیل شد تا به عنوان ورودی به لایه‌های کاملاً متصل (Dense) مدل قابل استفاده باشد.

---

### ۳. طراحی و پیاده‌سازی مدل

فرآیند مدل‌سازی در دو بخش اصلی انجام شد:

بخش اول: اتوانکودرها

دو مدل اتوانکودر با معماری مشابه اما اندازه فضای نهان متفاوت طراحی شدند.

1. اتوانکودر با فضای نهان ۸ بعدی (AE-8):

○ انکودر: (ReLU)  $\rightarrow$  8 (ReLU) 128  $\leftarrow$  784 ○

(Decoder): 8  $\rightarrow$  128 (ReLU)  $\rightarrow$  784 (Linear) ○ رمزگشا

2. اتوانکودر با فضای نهان ۴ بعدی (AE-4):

○ انکودر: (ReLU)  $\rightarrow$  4 (ReLU) 128  $\leftarrow$  784 ○

(Decoder): 4  $\rightarrow$  128 (ReLU)  $\rightarrow$  784 (Linear) ○ رمزگشا

هر دو مدل باتابع خطای MSE و به مدت ۲۰ ایپاک آموزش داده شدند.

#### بخش دوم: طبقه‌بندها

در این مرحله، از بخش انکودر آموزش دیده در مرحله قبل به عنوان یک استخراج‌کننده ویژگی ثابت (فریز شده) استفاده شد. یک طبقه‌بند ساده به خروجی هر انکودر متصل و سپس آموزش داده شد.

1. طبقه‌بند با انکودر ۸ بعدی (Classifier-8):

○ انکودر فریز شده: (خروجی ۸ بعدی)

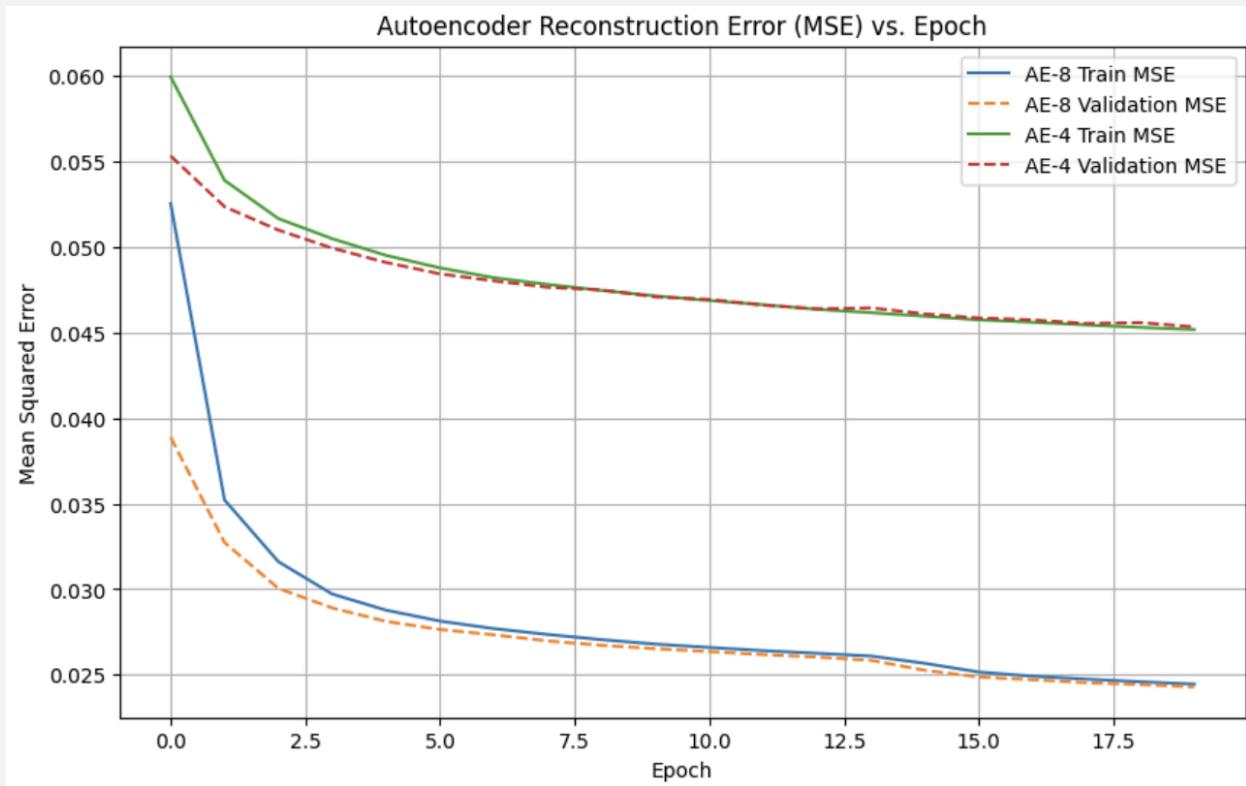
○ لایه طبقه‌بند: (ReLU)  $\rightarrow$  10 (Softmax) 4  $\leftarrow$  8 ○

2. طبقه‌بند با انکودر ۴ بعدی (Classifier-4):

○ انکودر فریز شده: (خروجی ۴ بعدی)

○ لایه طبقه‌بند: (Softmax) 10  $\leftarrow$  4 ○

این دو مدل با تابع خطای Cross-Entropy و به مدت ۲۰ اپاک آموزش دیدند. در طول این آموزش، وزن‌های انکودرها تغییری نکردند.



## ۴. نتایج و تحلیل

### ۴-۱. نتایج

- تعداد پارامترهای مدل‌ها:

خلاصه پارامترهای مدل‌ها به شرح زیر است:

| مدل | کل پارامترها | پارامترهای قابل آموزش |

| ---: | ---: | ---: |

| Autoencoder-8 | 205,612 | 205,612 |

| Autoencoder-4 | 203,568 | 203,568 |

| Classifier-8 | 101,772 | 86 |

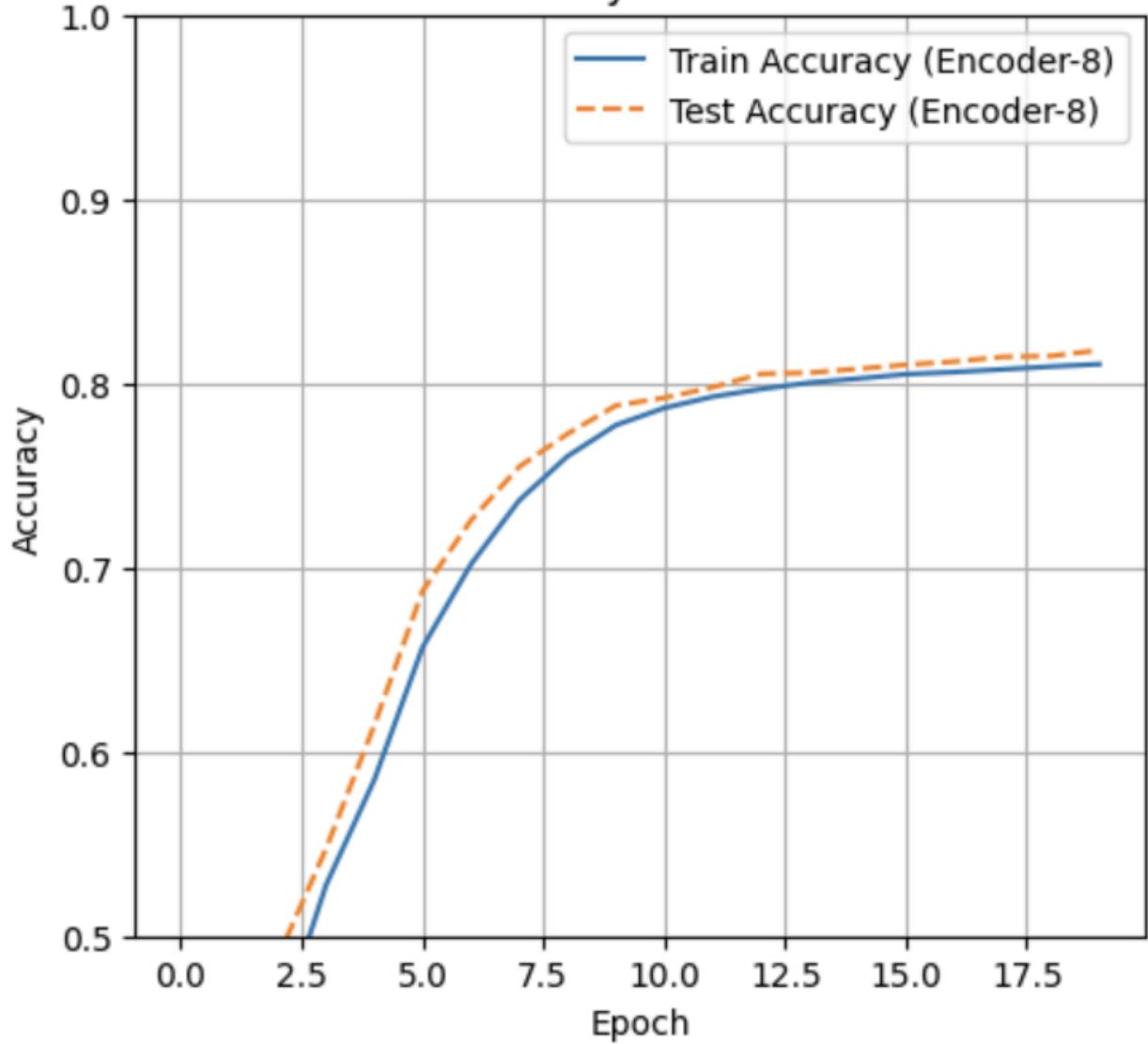
| Classifier-4 | 101,148 | 50 |

نکته قابل توجه، تعداد بسیار کم پارامترهای قابل آموزش در مدل‌های طبقه‌بند است که نشان‌دهنده کارایی روش استفاده از انکودر فریز شده است.

نمودار خطای بازسازی (MSE) :

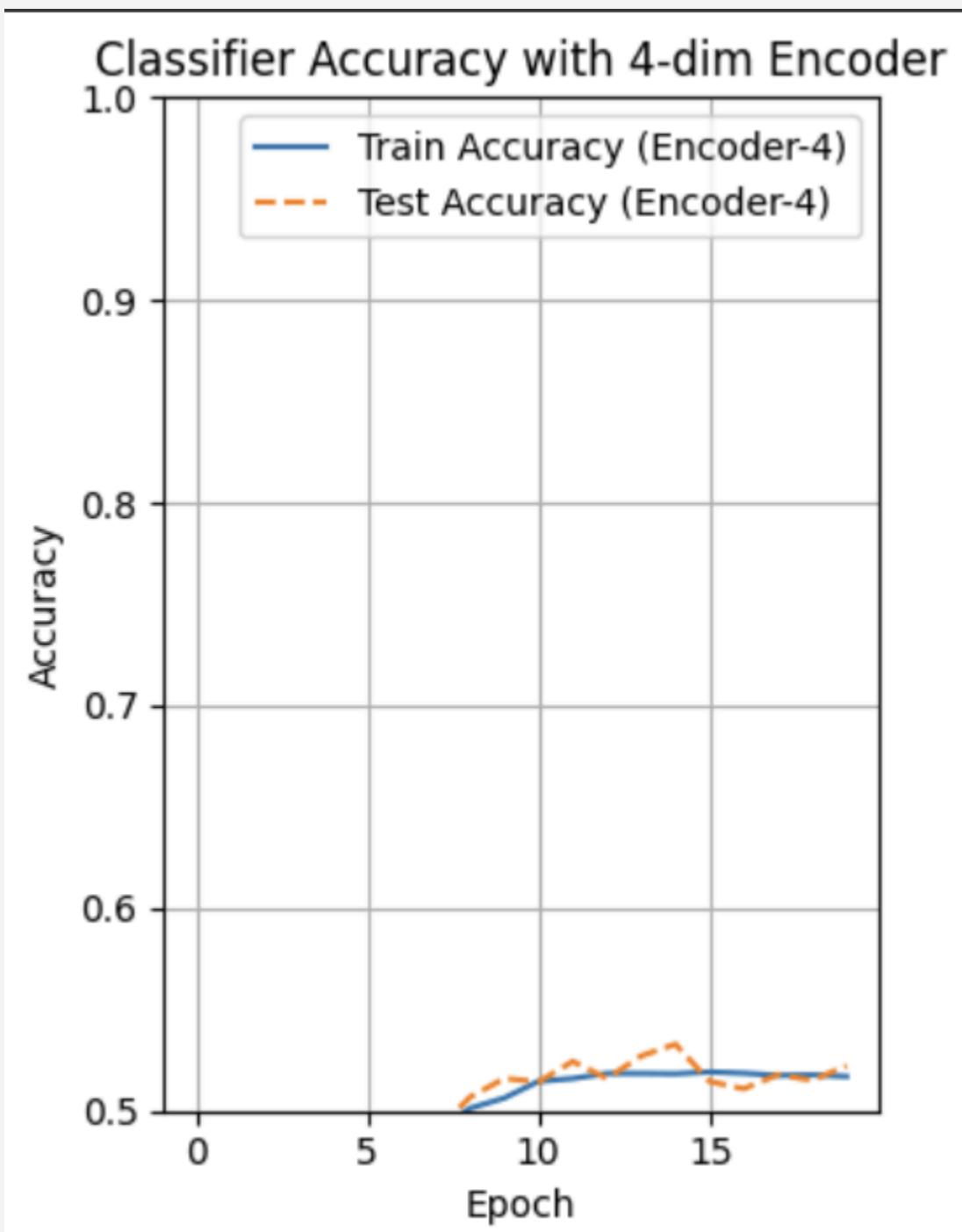
این نمودار خطای بازسازی را برای هر دو اتوانکودر در طول آموزش نشان می‌دهد.

### Classifier Accuracy with 8-dim Encoder



نمودارهای دقت طبقه‌بندی:

این دو نمودار، روند افزایش دقت را برای هر دو مدل طبقه‌بند بر روی داده‌های آموزش و تست نمایش می‌دهند.



این تصاویر به صورت بصری کیفیت بازسازی هر دو مدل را مقایسه می‌کنند.

## ٤-٢. تحليل و مقاييسه

Autoencoder with 8-dim latent space:

Model: "autoencoder\_8"

| Layer (type)                | Output Shape | Param # |
|-----------------------------|--------------|---------|
| input_layer_21 (InputLayer) | (None, 784)  | 0       |
| encoder_8 (Functional)      | (None, 8)    | 101,512 |
| decoder_8 (Functional)      | (None, 784)  | 102,288 |

Total params: 611,402 (2.33 MB)

Trainable params: 102,288 (399.56 KB)

Non-trainable params: 101,512 (396.53 KB)

Optimizer params: 407,602 (1.55 MB)

Autoencoder with 4-dim latent space:

Model: "autoencoder\_4"

| Layer (type)                | Output Shape | Param # |
|-----------------------------|--------------|---------|
| input_layer_23 (InputLayer) | (None, 784)  | 0       |
| encoder_4 (Functional)      | (None, 4)    | 100,996 |
| decoder_4 (Functional)      | (None, 784)  | 101,776 |

Total params: 608,318 (2.32 MB)

Trainable params: 101,776 (397.56 KB)

Non-trainable params: 100,996 (394.52 KB)

Optimizer params: 405,546 (1.55 MB)

Classifier with 8-dim frozen encoder:

Model: "classifier\_8"

| Layer (type)           | Output Shape | Param # |
|------------------------|--------------|---------|
| encoder_8 (Functional) | (None, 8)    | 101,512 |
| dense_51 (Dense)       | (None, 4)    | 36      |
| dense_52 (Dense)       | (None, 10)   | 50      |

Total params: 101,772 (397.55 KB)

Trainable params: 86 (344.00 B)

Non-trainable params: 101,512 (396.53 KB)

Optimizer params: 174 (700.00 B)

Classifier with 4-dim frozen encoder:

Model: "classifier\_4"

| Layer (type)           | Output Shape | Param # |
|------------------------|--------------|---------|
| encoder_4 (Functional) | (None, 4)    | 100,996 |
| dense_53 (Dense)       | (None, 10)   | 50      |

Total params: 101,148 (395.11 KB)

Trainable params: 50 (200.00 B)

Non-trainable params: 100,996 (394.52 KB)

Optimizer params: 102 (412.00 B)

• مقایسه خطای بازسازی:

مدل با ۸ نوروں (AE-8) به طور قابل توجهی تصاویر را بهتر بازسازی کرد. نمودار MSE نشان می‌دهد که خطای نهایی این مدل (حدود ۰.۰۲۴) بسیار کمتر از مدل ۴ نوروی (حدود ۰.۰۴۵) است. تصاویر بازسازی شده نیز این موضوع را تأیید می‌کنند؛ ارقام بازسازی شده توسط AE-8 واضح‌تر و به نمونه اصلی شبیه‌تر هستند، در حالی که خروجی AE-4 بسیار تار و نامشخص است.

چرا؟ یک فضای نهان ۸ بعدی ظرفیت بیشتری برای نگهداری اطلاعات کلیدی و جزئیات تصویر دارد. کاهش ابعاد به ۴، باعث از دست رفتن حجم زیادی از اطلاعات مفید شده و مدل را در بازسازی دقیق تصویر ناتوان می‌کند.

• مقایسه دقت طبقه‌بندی:

مدل طبقه‌بند متصل به انکودر ۸ بعدی عملکرد بسیار بهتری داشت و به دقت نهایی حدود ۸۲٪ روی داده‌های تست رسید. در مقابل، مدل متصل به انکودر ۴ بعدی کاملاً ناموفق بود و دقت آن حول ۵۲٪ باقی ماند که تفاوت چندانی با حدس تصادفی (۱۰٪) ندارد.

چرا؟ فضای نهان ۸ بعدی توانسته بود ویژگی‌های تمایزبخش بین ارقام مختلف را حفظ کند. اما فشرده‌سازی شدید به ۴ بعد، این ویژگی‌های حیاتی را از بین برده و یک نمایش بی‌فایده برای طبقه‌بندی ایجاد کرده بود. این یک نمونه کلاسیک از \*\*بده-بستان بین میزان فشرده‌سازی و حفظ اطلاعات\*\* است.

• مقایسه تعداد پارامترها:

تفاوت تعداد کل پارامترها بین دو اتوانکودر و همچنین دو طبقه‌بند ناچیز بود، زیرا بخش عمدی پارامترها در لایه‌های مشترک قرار داشتند. این نشان می‌دهد که عامل اصلی تفاوت عملکرد، ظرفیت فضای نهان بود و نه تعداد کل پارامترها.

• نشانه‌های بیشبرازش (Overfitting):

در نمودار دقت طبقه‌بند ۸ بعدی، فاصله کمی بین دقت آموزش و تست وجود دارد که نشان‌دهنده بیشبرازش جزئی است، اما این پدیده شدید نیست. برای کاهش آن می‌توان از تکنیک‌هایی مانند Dropout در لایه‌های طبقه‌بند یا منظم‌سازی L2 استفاده کرد.

• پیشنهاد برای بهبود عملکرد (امتیازی):

نتایج نشان داد که حتی مدل بهتر (۸ بعدی) نیز دقت بالایی کسب نکرد. برای بهبود، می‌توان معماری اتوانکودر را قدرتمندتر کرد.

همانطور که در نتایج "مدل امتیازی" با دقت ۹۵.۱۶٪ مشاهده می‌شود، با یک مدل بهتر می‌توان به نتایج عالی دست یافت. تصاویر بازسازی شده در این مدل بسیار واضح هستند.

پیشنهاد: به جای لایه‌های کاملاً متصل، از اتوانکودر کانولوشنی (Convolutional Autoencoder) استفاده شود. لایه‌های کانولوشنی برای استخراج ویژگی از داده‌های تصویری بسیار کارآمدتر هستند و می‌توانند نمایش بسیار غنی‌تر و مفیدتری را در فضای نهان ایجاد کنند که هم به بازسازی بهتر و هم به دقت بالاتر در طبقه‌بندی منجر می‌شود.

