



تمرین سوم هوش مصنوعی و یادگیری ماشین (دسته‌بندی)

نگارش:

مهربد ظریفی

۸۱۰۶۰۳۱۱۵

استاد درس:

دکتر مسعود شریعت‌پناهی

اردیبهشت ۱۴۰۴

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ

فهرست

۱	فهرست
۱	- الف) بررسی داده‌های خام
۱	۱-۱- الف-۱: ساختار کلی داده‌ها
۴	۱-۲- الف-۲: بررسی مقادیر گمشده (MISSING VALUES) یا مقادیر ناموجود در داده‌ها
۵	۱-۳- الف-۳: بررسی همبستگی بین ویژگی‌ها و وضعیت ابزار فرز
۶	۱-۴- الف-۴: تحلیل تصویری و آماری ویژگی‌ها نسبت به FAILURE TYPES
۶	۱-۴-۱- تحلیل توزیع داده‌ها
۸	۱-۴-۲- بررسی بصری ارتباط با خروجی
۹	۱-۴-۳- تحلیل آماری (آزمون ANOVA)
۱۱	۲- ب) پیش‌پردازش داده‌ها
۱۱	۲-۱- ب-۱: بررسی و اصلاح داده‌های ناقص و پرت
۱۱	۲-۱-۱- ب-۱-۱: بررسی و اصلاح مقادیر گمشده (Missing Values)
۱۴	۲-۱-۲- شناسایی و حذف داده‌های پرت (Outliers)
۱۴	۲-۱-۳- نتیجه‌ی نهایی
۱۵	۲-۲- ب-۱: نرمال‌سازی و استانداردسازی ویژگی‌های عددی
۱۵	۲-۲-۱- تعریف مفاهیم
۱۵	۲-۲-۲- آیا در این تمرین نیاز به مقیاس‌بندی بود؟
۱۵	۲-۲-۳- چرا استانداردسازی و نرمال‌سازی؟
۱۶	۲-۲-۴- پیاده‌سازی در کد:
۱۸	۳- ج) دسته‌بندی دوگانه یا دودویی (BINARY CLASSIFICATION)
۱۸	۳-۱- ج-۱: ساخت ستون برچسب دودویی
۱۸	۳-۲- ج-۲: بررسی توزیع کلاس‌های دودویی
۱۹	۳-۳- ج-۳: تأثیر عدم توازن داده‌ها بر عملکرد مدل
۲۰	۳-۴- ج-۴: متوازن‌سازی داده‌ها با استفاده از SMOTE

فهرست

۲۱	۵-۳- ج-۵: آموزش مدل‌های طبقه‌بندی روی داده‌های متوازن شده
۲۲	۶-۳- ج-۶: ارزیابی عملکرد مدل‌ها با استفاده از ماتریس آشفتگی، دقت و CLASSIFICATION REPORT
۲۵	۷-۳- ج-۷: تنظیم بهینه‌ی هایپرپارامترها با استفاده از GRIDSEARCHCV
۲۶	۸-۳- ج-۸: مقایسه نهایی عملکرد مدل‌ها با شاخص‌های تکمیلی
۲۹	۹-۳- ج-۹: جمع‌بندی بخش ج
۳۱	۴- د) دسته‌بندی چندگانه بر اساس ستون FAILURE TYPES
۳۱	۴-۱-۱: تقسیم داده و آموزش مدل‌های چند کلاسه
۳۲	۴-۲-۲: ارزیابی عملکرد مدل‌های چند کلاسه
۳۶	۴-۳-۳: تنظیم پارامترها با GRIDSEARCHCV
۳۶	۴-۴-۴: مقایسه نهایی مدل‌ها با شاخص‌های تکمیلی
۳۹	۵- جمع‌بندی نهایی تمورین
۳۹	۵-۱-۱: مرحله ۱: بررسی داده و پیش‌پردازش
۳۹	۵-۲-۲: مرحله ۲: دسته‌بندی دودویی (بخش ج)
۴۰	۵-۳-۳: مرحله ۳: دسته‌بندی چندگانه (بخش د)
۴۱	۵-۴-۴: اهمیت ویژگی‌ها (FEATURE IMPORTANCE)
۴۱	۵-۵-۵: نتیجه‌گیری کلی

فصل اول

الف) بررسی داده‌های خام

در این تمرین، هدف ارزیابی عملکرد روش‌های یادگیری ماشین در دسته‌بندی دودویی و چندکلاسه است. به عنوان مطالعه‌ی موردنی، مسأله‌ی پایش سلامت و تشخیص عیب ابزار برش در دستگاه فرز انتخاب شده است. مجموعه‌داده‌ی استفاده‌شده شامل ۱۰۰۰ نمونه‌ی ثبت‌شده از وضعیت ابزار برش می‌باشد که در فایل milling_machine.csv قرار دارد.

در این دادگان، ستون‌های اول تا پنجم به ترتیب عبارتند از:

- دمای محیط (Air Temperature)
- دمای فرآیند (Process Temperature)
- سرعت چرخشی ابزار فرز (Rotational Speed)
- گشتاور وارد بر محور ابزار (Torque)
- مدت زمان قرارگیری در معرض سایش (Tool Wear)

ستون ششم نیز وضعیت نهایی ابزار را نشان می‌دهد و شامل شش برچسب مختلف است که ترکیبی از حالت سالم یا نوع خاصی از خرابی می‌باشد. این ستون در ادامه به دو صورت تحلیل می‌شود: دسته‌بندی دودویی (سالم/معیوب) و دسته‌بندی چندکلاسه (نوع خرابی).

۱-۱-الف-۱: ساختار کلی داده‌ها

برای آشنایی اولیه با ساختار داده‌ها، از توابع info() و describe() کتابخانه pandas استفاده شد. خروجی این توابع نشان داد که:

- مجموعه‌داده دارای ۶ ستون اصلی و ۱۰۰۰ سطر می‌باشد.
- تمام ویژگی‌های عددی دارای نوع داده‌ی float64 هستند.

- تنها ستون غیر عددی، ستون Failure Types است که از نوع object بوده و برچسب کلاس را مشخص می‌کند.
- برخی از ستون‌ها دارای مقدارهای گمشده هستند که در بخش پیش‌پردازش به آن‌ها پرداخته خواهد شد.
- مقادیر آماری مانند میانگین، میانه، کمینه و بیشینه برای هر ویژگی عددی استخراج شده که در ادامه به تحلیل توزیع این ویژگی‌ها کمک خواهد کرد.

◆ اطلاعات کلی دیتا:

تعداد سطرها: ۱۰,۰۰۰

تعداد ستون‌ها: ۶

نوع داده‌ها: پنج ویژگی عددی (float64) و یک ستون متند (object)

تعداد مقادیر گمشده در ستون‌ها:

Air Temp (°C): 35 ○

Process Temp (°C): 10 ○

Tool Wear (Seconds): 7 ○

Failure Types: 9 ○

```
<'class 'pandas.core.frame.DataFrame'>
```

RangeIndex: 10000 entries, 0 to 9999

:Data columns (total 6 columns)

Column	Non-Null Count	Dtype	#
--------	----------------	-------	---

Air Temp (°C)	9965	non-null	float64	.
---------------	------	----------	---------	---

Process Temp (°C)	9990	non-null	float64	1
-------------------	------	----------	---------	---

Rotational Speed (RPM)	10000	non-null	float64	2
------------------------	-------	----------	---------	---

Torque (Nm)	10000	non-null	float64	3
-------------	-------	----------	---------	---

Tool Wear (Seconds)	9993	non-null	float64	4
---------------------	------	----------	---------	---

Failure Types	9991	non-null	object	5
---------------	------	----------	--------	---

(1)dtypes: float64(5), object

memory usage: 468.9+ KB

◆ خلاصه آماری ویژگی‌های عددی:

ویژگی	بیشینه	کمینه	انحراف معیار	میانگین
Air Temp (°C)	28.52	7.72	20.00	49.99
Process Temp (°C)	80.81	15.55	60.00	119.97
Rotational Speed (RPM)	1401.91	968.45	0.04	2999.95
Torque (Nm)	47.00	26.75	0.02	89.99

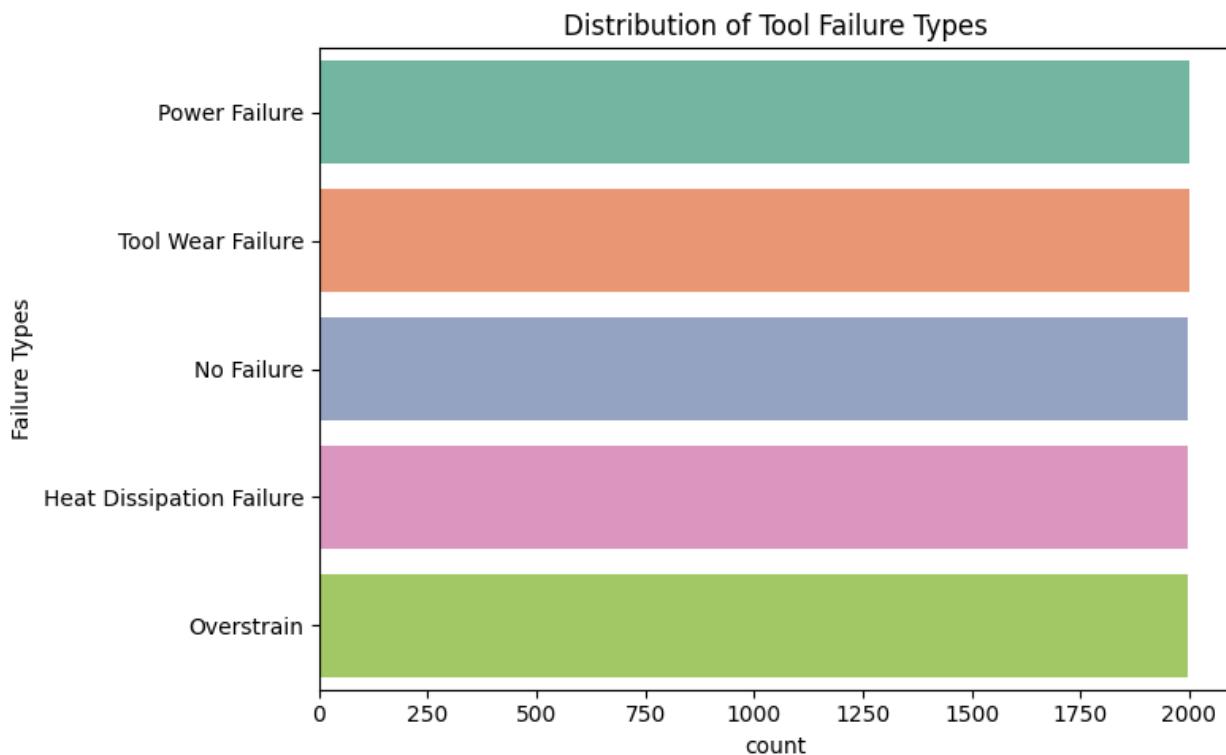
ویژگی	بیشینه	کمینه	انحراف معیار	میانگین
Tool Wear (Seconds)	11393.14	9023.34	3.47	35999.57

جدول: آمار توصیفی ویژگی‌های عددی ◆

شاخص آماری	Air Temp (°C)	Process Temp (°C)	Rotational Speed (RPM)	Torque (Nm)	Tool Wear (Seconds)
تعداد داده‌ها (count)	9965	9990	10000	10000	9993
میانگین (mean)	28.52	80.81	1401.91	47.00	11393.14
انحراف معیار (std)	7.72	15.55	968.45	26.75	9023.34
کمترین مقدار (min)	20.00	60.00	0.05	0.02	3.47
چارک اول (25%)	23.18	68.09	423.67	18.09	5023.03
میانه (50%)	26.21	76.55	1377.05	54.98	8995.17
چارک سوم (75%)	29.38	92.83	2307.97	67.26	15024.83
بیشترین مقدار (max)	49.99	119.97	2999.95	89.99	35999.57

این اطلاعات اولیه کمک می‌کنند تا دیدی کلی نسبت به نوع داده‌ها، محدوده‌ی مقادیر و نیاز به پاکسازی یا پیش‌پردازش‌های بعدی به دست آید.

همچنین، با رسم نمودار فراوانی برای ستون Failure Types، توزیع کلاس‌های مختلف ابزار فرز مشاهده شد. همان‌طور که در شکل مشاهده می‌شود، تعداد داده‌ها در تمام کلاس‌ها تقریباً یکسان است و داده‌ها دچار عدم توازن جدی (class imbalance) نیستند.



۲-الف-۲: بررسی مقادیر گمشده (Missing Values) یا مقادیر ناموجود در داده‌ها

یکی از مراحل مهم در پیش‌پردازش داده‌ها، شناسایی و مدیریت مقادیر ناموجود (Missing Values) است. این مقادیر در صورت عدم رسیدگی مناسب، می‌توانند باعث ایجاد خطأ در مدل‌سازی و کاهش دقت پیش‌بینی شوند. با استفاده از توابع `isnull()` و `sum()` در کتابخانه `pandas`، تعداد و درصد مقادیر گمشده برای هر ستون محاسبه شد. نتایج آن در جدول زیر آورده شده است:

ویژگی	درصد مقادیر گمشده	تعداد مقادیر گمشده (%)
Air Temp (°C)	35	0.35%
Process Temp (°C)	10	0.10%
Rotational Speed (RPM)	0	0.00%
Torque (Nm)	0	0.00%
Tool Wear (Seconds)	7	0.07%
Failure Types	9	0.09%

همان‌طور که از جدول پیداست:

- بیشترین تعداد مقادیر گمشده مربوط به ویژگی "دماهی هوا" با ۳۵ مقدار گمشده است (معادل 0.35% از کل داده‌ها).
- ویژگی‌های "سرعت چرخشی" و "گشتاور" فاقد هرگونه مقدار گمشده هستند.

- با توجه به نسبت پایین مقادیر گمشده در کل ویژگی‌ها، تصمیم گرفته شد این مقادیر در مرحله‌ی پیش‌پردازش، با استفاده از میانگین برای ویژگی‌های عددی و مد برای ستون برچسب جایگزین شوند.

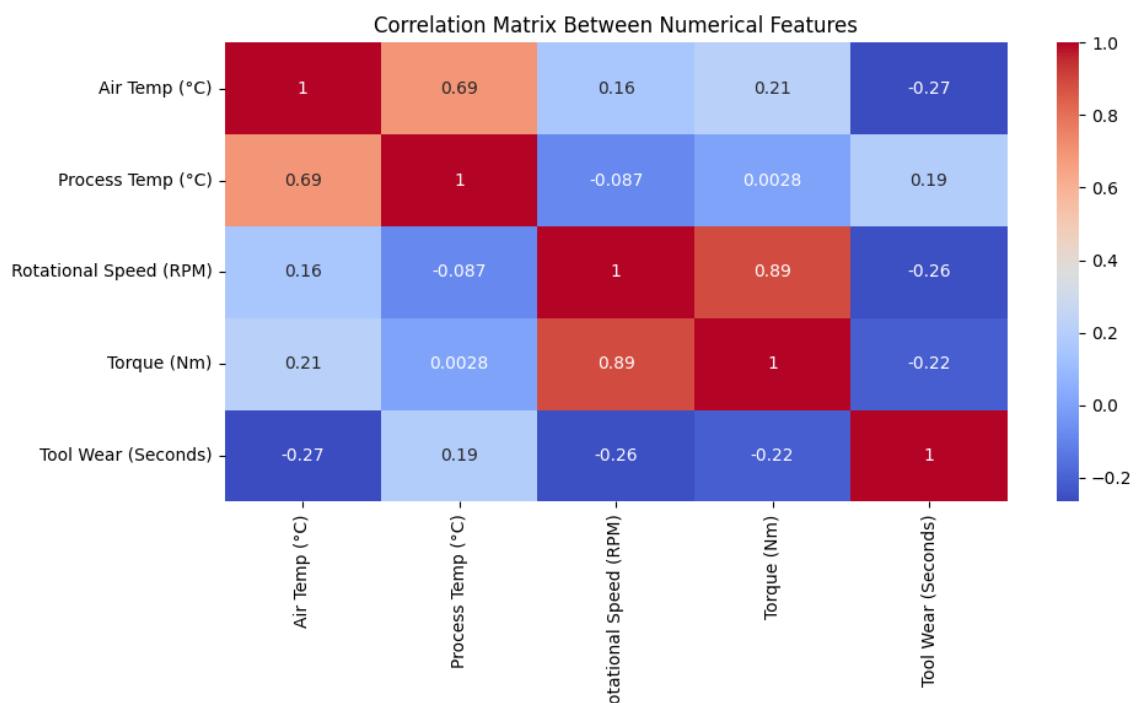
۱-۳-۳: بررسی همبستگی بین ویژگی‌ها و وضعیت ابزار فرز

برای تحلیل ارتباط بین ویژگی‌های عددی مجموعه‌داده، از ضریب همبستگی پیرسون (Pearson Correlation Coefficient) استفاده شد. این ضریب میزان همبستگی خطی بین دو متغیر عددی را در بازه‌ی [-۱, ۱] نشان می‌دهد. این تحلیل با استفاده از تابع `corr()` انجام گردید و نتایج آن در قالب یک ماتریس همبستگی (Correlation Matrix) به صورت بصری نمایش داده شد.

ماتریس همبستگی بین ویژگی‌های عددی:

نقشه‌ی حرارتی (Heatmap) زیر، میزان همبستگی بین پنج ویژگی عددی مجموعه داده را نمایش می‌دهد. رنگ‌های تیره‌تر به معنای همبستگی بیشتر (مثبت یا منفی) هستند.

برای نمایش روابط بین ویژگی‌ها، از ماتریس همبستگی به صورت Heatmap استفاده شد که در شکل زیر قابل مشاهده است:



تحلیل خلاصه‌ای از مشاهدات کلیدی از این ماتریس همبستگی:

- بالاترین همبستگی مثبت بین ویژگی‌های Rotational Speed و Torque مشاهده شد (ضریب ۰.۸۹).
- ویژگی‌های Process Temp و Air Temp نیز دارای همبستگی مثبت نسبتاً بالایی هستند (۰.۶۹).
- ویژگی Tool Wear (Seconds) با بیشتر ویژگی‌ها رابطه‌ای نسبتاً منفی یا بسیار ضعیف دارد.

- با Air Temp: ضریب -۰.۲۷
- با Rotational Speed: -0.26

○ با -0.22 Torque:

نتیجه:

- ویژگی Tool Wear که به صورت غیرمستقیم به وضعیت ابزار (خرابی یا سلامت) مرتبط است، دارای همبستگی ضعیف اما قابل بررسی با برخی ویژگی‌ها می‌باشد.
- از آنجا که Failure Types یک ویژگی طبقه‌ای (Categorical) است، برای تحلیل رابطه‌ی آن با ویژگی‌های عددی، از روش غیرمستقیم استفاده شد:

 - ابتدا همبستگی بین ویژگی‌ها محاسبه شد.
 - سپس سه ویژگی با بیشترین همبستگی با ویژگی Tool Wear (Seconds) که مستقیماً بر خرابی ابزار اثرگذار است، انتخاب شدند.
 - در مراحل بعدی (بخش الف-۴)، ارتباط مستقیم هر ویژگی با خروجی طبقه‌بندی (Failure Types) با استفاده از آزمون آماری ANOVA و نمودارهای تصویری تحلیل خواهد شد.

۱-۴-الف-۴: تحلیل تصویری و آماری ویژگی‌ها نسبت به Failure Types

برای بررسی دقیق‌تر رابطه بین ویژگی‌های عددی و برچسب خروجی (Failure Types)، سه ویژگی دارای بیشترین ارتباط از نظر همبستگی (طبق بند الف-۳) انتخاب شدند:

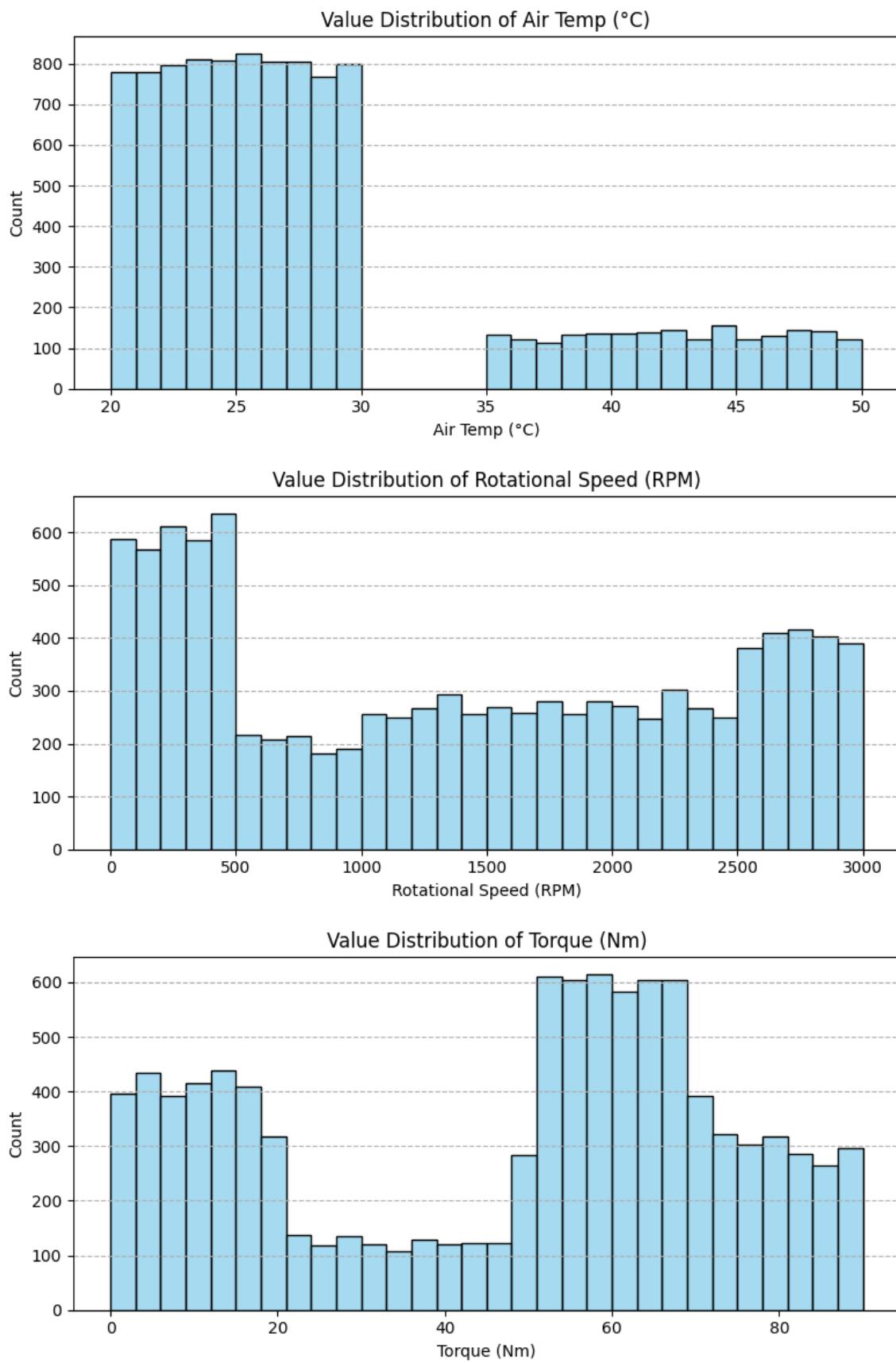
- Air Temp ($^{\circ}\text{C}$)
- Rotational Speed (RPM)
- Torque (Nm)

۱-۴-۱- تحلیل توزیع داده‌ها

برای درک بهتر نحوه توزیع مقادیر هر ویژگی، هیستوگرام تعداد مشاهدات برای مقادیر مختلف رسم شد. نتایج نشان داد: Air Temp در بازه ۲۰ تا ۳۰ دارای تمرکز بالای نمونه‌ها است و پس از آن کاهش ناگهانی دارد. Rotational Speed توزیع نسبتاً دوجهته دارد؛ با تراکم بالا در مقادیر کم (زیر ۵۰۰) و زیاد (بیش از ۲۵۰۰). Torque نیز دارای دو قله است، که نشان‌دهنده استفاده از ابزار در دو وضعیت بارگذاری متفاوت است.

◆ سه ویژگی با بیشترین همبستگی با Tool Wear: ['Air Temp ($^{\circ}\text{C}$)', 'Rotational Speed (RPM)',

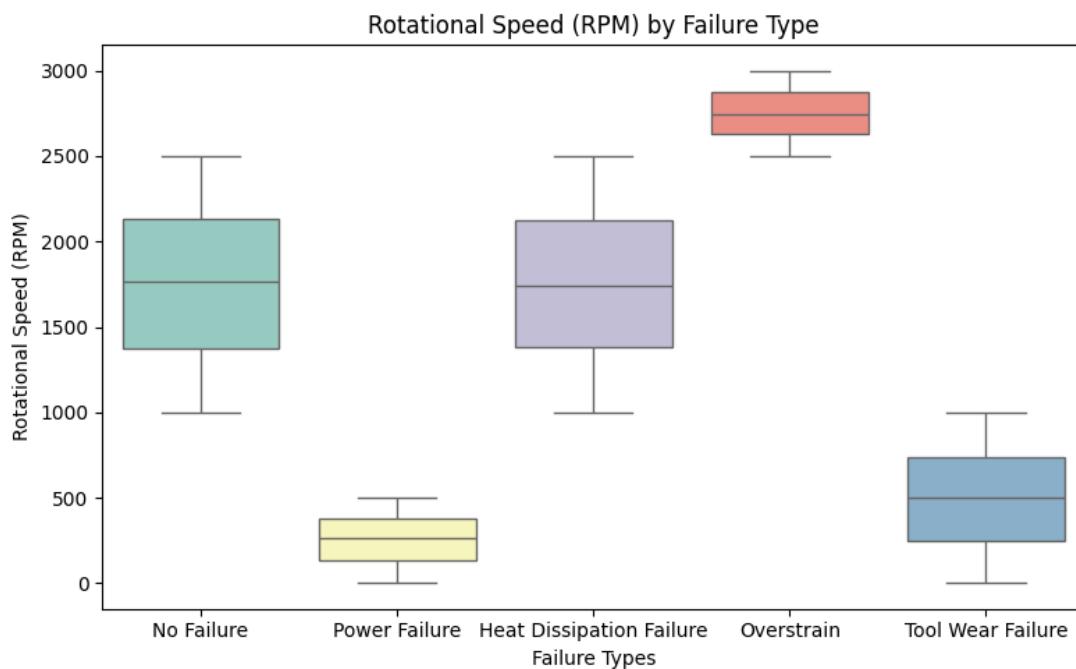
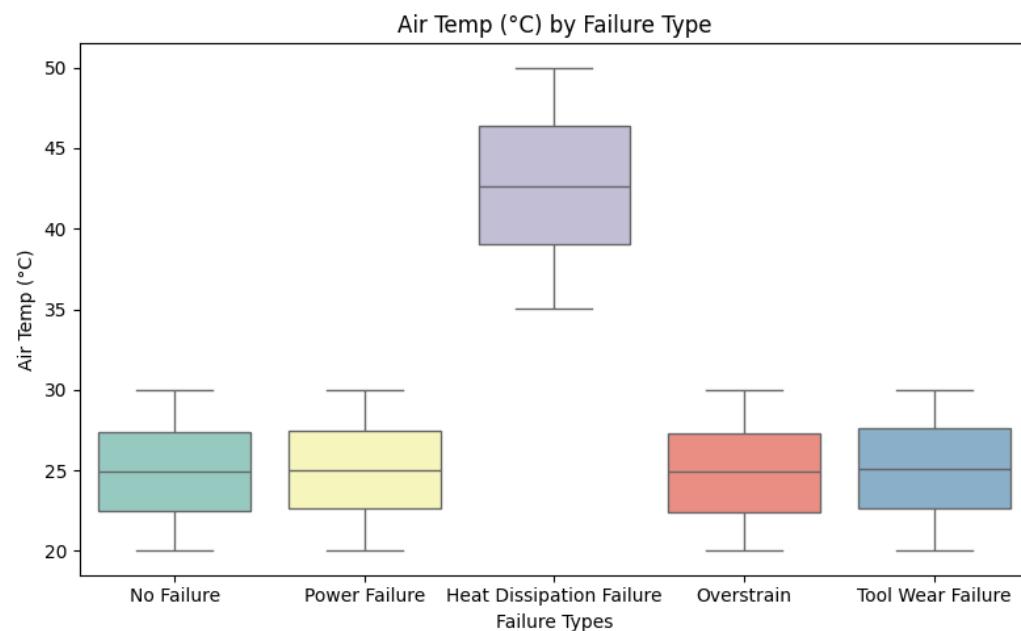
'Torque (Nm)']

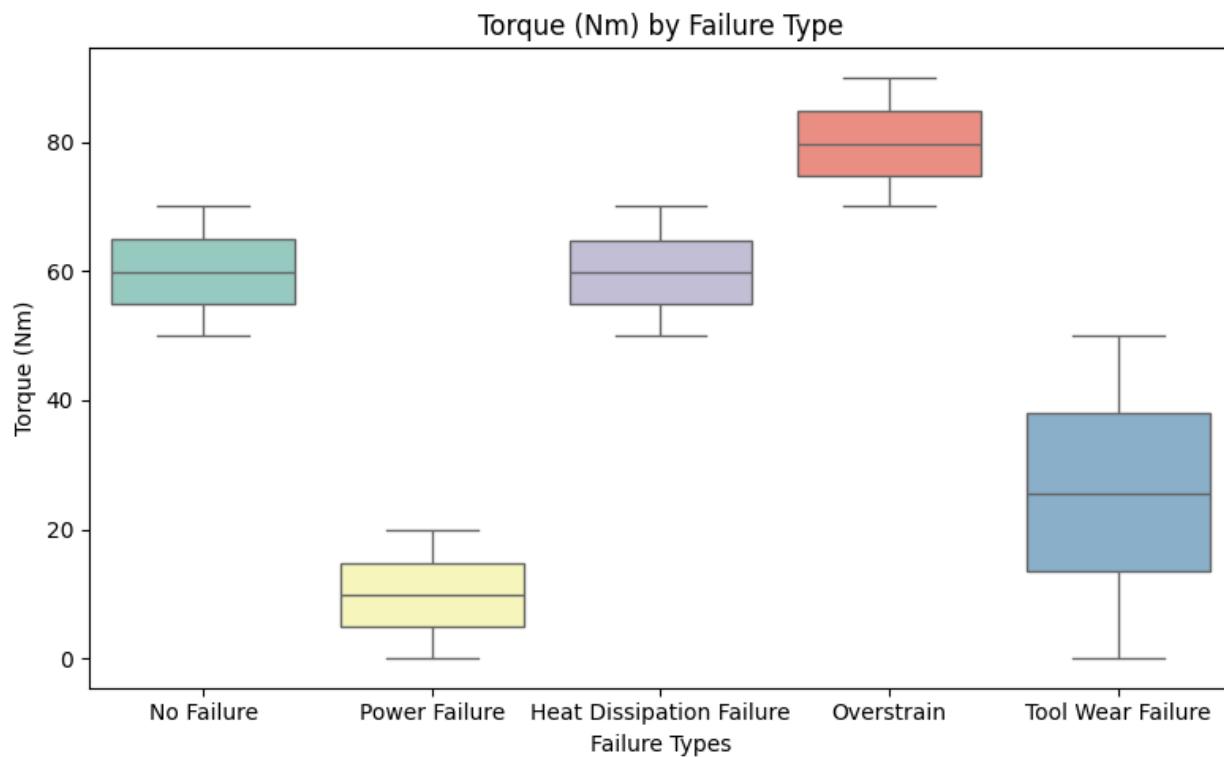


۱-۴-۲- بررسی بصری ارتباط با خروجی

برای بررسی رابطه بین این ویژگی‌ها و Failure Types، از نمودارهای Boxplot استفاده شد. مقایسه میان کلاس‌های مختلف نشان داد:

- دمای هوا در کلاس Heat Dissipation Failure به طور معناداری بیشتر از سایر کلاس‌ها است.
- سرعت چرخشی در کلاس Overstrain به بالاترین مقدار می‌رسد، در حالی که در Power Failure کمترین مقدار را دارد.
- گشتاور در Tool Wear Failure نسبت به بقیه کلاس‌ها پراکندگی بسیار بیشتری دارد و میانگین پایین‌تری دارد.





۱-۴-۳- تحلیل آماری (آزمون ANOVA)

برای بررسی معناداری تفاوت میانگین‌های این ویژگی‌ها در کلاس‌های مختلف، از آزمون ANOVA یک‌طرفه استفاده شد. نتایج به شرح زیر است:

ویژگی	Mقدار F آماره	P	نتیجه
Air Temp (°C)	12053.23	0.0000	✓ تفاوت معنادار بین گروه‌ها
Rotational Speed (RPM)	21233.75	0.0000	✓ تفاوت معنادار بین گروه‌ها
Torque (Nm)	24114.56	0.0000	✓ تفاوت معنادار بین گروه‌ها

نتایج این آزمون‌ها نشان می‌دهد که هر سه ویژگی مورد بررسی، به صورت آماری دارای تفاوت معنادار در کلاس‌های مختلف خروجی هستند و می‌توان آن‌ها را ویژگی‌های تأثیرگذار در مدل‌های طبقه‌بندی در نظر گرفت.

فصل دوم

۲

ب) پیش پردازش داده‌ها

پیش‌پردازش یکی از مراحل کلیدی در یادگیری ماشین است که تأثیر مستقیم بر دقت، سرعت و پایداری مدل دارد. در این بخش، ابتدا به بررسی و اصلاح مقادیر گمشده و سپس به حذف داده‌های پرت و نرمال‌سازی ویژگی‌های عددی خواهیم پرداخت.

۱-۱-۱: بررسی و اصلاح داده‌های ناقص و پرت

در این مرحله از پیش‌پردازش، ابتدا به بررسی و اصلاح مقادیر گمشده (Missing Values) پرداخته شد. سپس داده‌های پرت (Outliers) که می‌توانند بر عملکرد مدل‌های یادگیری ماشین تأثیر منفی داشته باشند، شناسایی و حذف شدند.

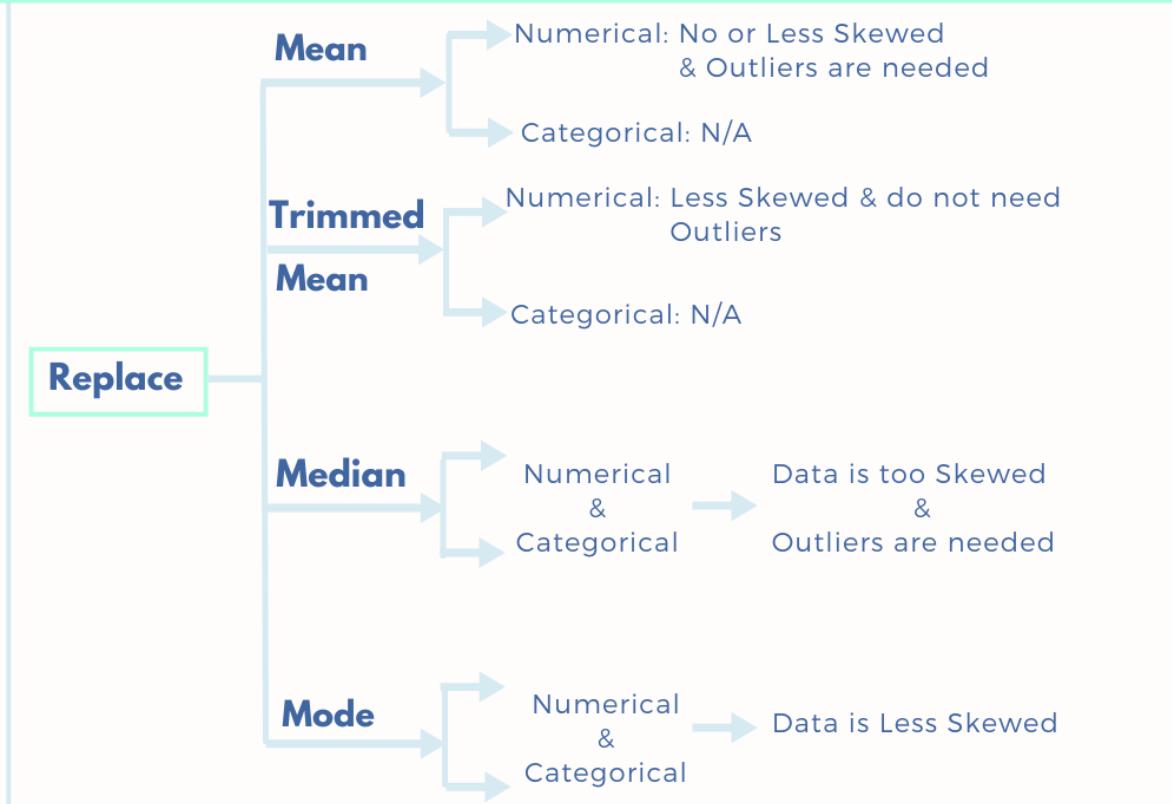
۱-۱-۱-۱: بررسی و اصلاح مقادیر گمشده (Missing Values)

وجود مقادیر گمشده (Missing Values) در مجموعه داده‌های واقعی، مسئله‌ای بسیار رایج است. این مقادیر ممکن است

ناشی از خطا در فرآیند ثبت داده، عدم پاسخ‌دهی کاربران، یا نقص در منابع اولیه داده باشند. باقی ماندن این مقادیر بدون مدیریت مناسب، می‌تواند باعث ایجاد خطا در الگوریتم‌های یادگیری ماشین، اختلال در تحلیل آماری و کاهش کیفیت پیش‌بینی مدل‌ها شود. به همین دلیل، جایگزینی صحیح و دقیق این مقادیر یکی از مهم‌ترین مراحل در پیش‌پردازش داده‌ها محسوب می‌شود.

روش‌های متعددی برای جایگزینی داده‌های گمشده وجود دارد. ساده‌ترین روش‌ها شامل جایگزینی با میانگین (Mean) و جایگزینی با میانه (Median Imputation) هستند. در روش اول، مقدار گمشده هر ویژگی با میانگین سایر مقادیر همان ستون جایگزین می‌شود. این روش سریع و پیاده‌سازی آن آسان است، اما فرض می‌کند که داده‌ها به صورت متقارن توزیع شده‌اند. در حضور چولگی یا داده‌های پرت، میانگین ممکن است نماینده‌ی خوبی برای مقدار مرکزی نباشد و موجب انحراف در تحلیل شود. در این شرایط، جایگزینی با میانه توصیه می‌شود، چرا که میانه نسبت به مقادیر دورافتاده حساسیت کمتری دارد. با این حال، هر دو روش صرفاً از اطلاعات موجود در همان ستون استفاده می‌کنند و ساختار بین‌ستونی داده را در نظر نمی‌گیرند. برای ویژگی‌های طبقه‌ای (categorical)، معمولاً از جایگزینی با پرتکرارترین مقدار (Mode) استفاده می‌شود. در این روش، مقدار گمشده با مقداری که در آن ستون بیشترین تکرار را دارد جایگزین می‌گردد. این روش در ویژگی‌هایی که یک مقدار غالب دارند مناسب است، اما اگر توزیع مقادیر متنوع یا چندحالته باشد، ممکن است دقت پایینی داشته باشد.

Impute NaN with Mean, Median & Mode



در مقابل، روش‌های پیشرفته‌تری نیز وجود دارند که با در نظر گرفتن رابطه بین ویژگی‌ها، مقدار گمشده را تخمین می‌زنند. از جمله این روش‌ها می‌توان به رگرسیون چندمتغیره یا (Multiple Imputation by Chained Equations) MICE اشاره کرد.

این روش‌ها تلاش می‌کنند با مدل‌سازی ارتباط آماری بین ویژگی‌ها، مقادیر گمشده را به صورت شرطی بر اساس سایر مقادیر موجود پیش‌بینی کنند. اگرچه این رویکردها دقیق‌تر بالایی دارند، اما پیاده‌سازی آن‌ها پیچیده‌تر است، تنظیم پارامترهای بیشتری نیاز دارد، و در پروژه‌هایی با زمان محدود یا منابع محاسباتی محدود ممکن است مناسب نباشند.

طبقه‌بندی خروجی (Failure Types) دارای تعداد محدودی مقادیر گمشده بودند. جدول زیر خلاصه‌ای از تعداد و درصد مقادیر گمشده در هر ستون را نشان می‌دهد:

ویژگی	درصد مقادیر گمشده	تعداد مقادیر گمشده (%)
Air Temp (°C)	35	0.35%
Process Temp (°C)	10	0.10%
Rotational Speed (RPM)	0	0.00%
Torque (Nm)	0	0.00%
Tool Wear (Seconds)	7	0.07%
Failure Types	9	0.09%

با توجه به اینکه هیچ‌کدام از ستون‌ها بیش از ۱٪ مقدار گمشده نداشتند، تصمیم گرفته شد که به جای حذف نمونه‌ها، از روش جایگزینی برای تکمیل داده‌ها استفاده شود.

روش انتخاب شده برای جایگزینی مقادیر گمشده:

در این تمرین، از جایگزینی ساده استفاده شد:

- برای ویژگی‌های عددی، مقدار میانگین (Mean) جایگزین شد. دلیل این انتخاب، سادگی، پایداری و حفظ توزیع آماری داده‌ها در شرایطی با مقادیر گمشده کم است.
- برای ویژگی طبقه‌ای (Failure Types)، مقدار مدل (Mode) جایگزین شد. مد متداول‌ترین برچسب در ستون است و می‌تواند منطقی‌ترین انتخاب برای تکمیل داده‌های گمشده‌ی طبقه‌ای باشد.

دلیل انتخاب این روش:

- مقادیر گمشده بسیار اندک بودند.
- ساختار داده‌ها متعادل و بدون پراکندگی غیرعادی بود.
- روش‌های ساده سرعت بالایی دارند و از ایجاد پیچیدگی محاسباتی غیرضروری جلوگیری می‌کنند.
- در تمرین‌های طبقه‌بندی (classification)، دقت عددی مانند تمرین‌های رگرسیون اهمیت کمتری دارد.

پیاده‌سازی در کد:

```
for col in numeric_cols:
    df[col].fillna(df[col].mean(), inplace=True)

for col in object_cols:
    df[col].fillna(df[col].mode()[0], inplace=True)
```

با اجرای این مرحله، تمامی مقادیر گمشده در مجموعه داده با مقادیر جایگزین مناسب پر شدن و داده‌ها آماده ورود به مرحله بعدی یعنی حذف داده‌های پرت (Outlier Detection) شدند.

مقایسه با KNNImputer

در برخی پروژه‌های یادگیری ماشین، بهویژه زمانی که مقدار زیادی داده گمشده وجود دارد یا رابطه قوی بین ویژگی‌ها برقرار است، از روش‌های پیچیده‌تر مانند KNNImputer برای پر کردن داده‌های گمشده استفاده می‌شود. این روش با استفاده از نزدیکی نمونه‌ها (بر اساس K همسایه‌ی نزدیک‌تر) مقدار گمشده را تخمین می‌زند.

این روش در تمرین قبلی (رگرسیون) نیز استفاده شد تا از ساختار محلی داده‌ها برای تخمین دقیق‌تر استفاده شود. با اینکه KNNImputer در شرایطی که روابط پیچیده بین ویژگی‌ها وجود دارد مفید است، اما در این تمرین استفاده از آن مناسب نبود زیرا:

- تعداد مقادیر گمشده در هر ستون بسیار ناچیز (کمتر از ۵٪) بود.
- ویژگی‌های عددی رفتار نسبتاً مستقل و ساده‌ای داشتند.
- ویژگی‌ها از لحاظ توزیع آماری نسبتاً متوازن بودند.
- هدف مدل ما طبقه‌بندی است، نه تخمین دقیق مقادیر عددی.

بنابراین برای جلوگیری از پیچیدگی محاسباتی و کاهش زمان اجرا، در این تمرین از روش جایگزینی ساده با میانگین (برای ویژگی‌های عددی) و مد (برای ویژگی‌های طبقه‌ای) استفاده شد.

این تصمیم باعث حفظ ساختار اصلی داده‌ها و کاهش احتمال overfitting ناشی از برآورد بیش از حد پارامترها می‌شود.

۱-۲- شناسایی و حذف داده‌های پرت (Outliers):

علاوه بر مقادیر گمشده، داده‌های پرت (یا داده‌های بسیار متفاوت از بقیه) نیز می‌توانند باعث بروز خطا در یادگیری و کاهش تعمیم‌پذیری مدل شوند. برای شناسایی و حذف این داده‌ها، از روش Z-Score استفاده شد. در این روش، داده‌ای که مقدار آن‌ها بیشتر از ۳ انحراف معیار با میانگین فاصله دارند، به عنوان outlier در نظر گرفته شده و حذف می‌شوند.

پیاده‌سازی در کد:

```
from scipy.stats import zscore
```

```
z_scores = np.abs(zscore(df[numerical_cols]))
df = df[(z_scores < 3).all(axis=1)]
```

با اجرای این کد، کلیه داده‌ای که در حداقل یکی از ویژگی‌های عددی دارای انحراف بیش از ۳ بوده‌اند، حذف شدند.

۱-۳- نتیجه‌ی نهایی:

پس از تکمیل داده‌های گمشده و حذف داده‌های پرت، داده‌ها به شکل قابل اعتمادی آماده ورود به مرحله بعدی (نرمال‌سازی) شدند، شکل نهایی مجموعه‌داده به صورت زیر است:

- تعداد نهایی سطرها: ۱۰,۰۰۰
 - تعداد ستون‌ها: ۶
- (تعداد سطرها ممکن است بسته به حذف‌های بعدی تغییر کند)

۲-۲- ب-۲: نرمال‌سازی و استانداردسازی ویژگی‌های عددی

در یادگیری ماشین، تفاوت در مقیاس مقادیر ویژگی‌ها می‌تواند موجب اختلال در عملکرد بسیاری از الگوریتم‌ها شود. به همین دلیل، یکی از گام‌های مهم در پیش‌پردازش داده‌ها، مقیاس‌بندی ویژگی‌های عددی (Feature Scaling) است.

۲-۲-۱- تعریف مفاهیم:

- استانداردسازی (Standardizing): در این روش، مقدار هر ویژگی با کم کردن میانگین و تقسیم بر انحراف معیار تبدیل به داده‌ای با میانگین صفر و واریانس یک می‌شود.

$$X_{standard} = \frac{\mu - X}{\sigma}$$

- نرمال‌سازی (Normalizing): داده‌ها در بازه $[0, 1]$ فشرده می‌شوند. این روش برای مدل‌هایی که به فاصله‌ها حساس هستند (مثل KNN یا شبکه‌های عصبی) رایج است.

۲-۲-۲- آیا در این تمرین نیاز به مقیاس‌بندی بود؟

بله؛ چون در این تمرین از مدل‌هایی مانند:

- K-Nearest Neighbors
- Support Vector Machine
- Logistic Regression

استفاده شد، که به مقیاس عددی ویژگی‌ها حساس‌اند، بنابراین انجام مقیاس‌بندی ضروری بود. همچنین، چون داده‌های ما دارای توزیع نسبتاً نرمال بودند، از استانداردسازی (StandardScaler) استفاده شد.

۲-۲-۳- چرا استانداردسازی و نه نرمال‌سازی؟

در این تمرین، به جای نرمال‌سازی (که داده‌ها را به بازه $[0, 1]$ می‌برد)، از استانداردسازی (StandardScaler) استفاده شد.

دلیل این تصمیم:

۱. توزیع ویژگی‌ها نزدیک به نرمال بود، و در چنین شرایطی StandardScaler عملکرد بهتری دارد.
۲. الگوریتم‌هایی مانند SVM و KNN که در این تمرین استفاده شدند، به واریانس و فاصله‌ها حساس هستند و عملکرد بهتری با داده‌هایی با میانگین صفر و واریانس یک دارند.
۳. نرمال‌سازی بیشتر برای الگوریتم‌هایی مانند شبکه‌های عصبی یا الگوریتم‌هایی مبتنی بر نرخ یادگیری (مثل Gradient Descent) کاربرد دارد که در این تمرین استفاده نشده‌اند. بنابراین، StandardScaler انتخاب منطقی و مناسب‌تری برای این تمرین بود.

۴-۲-۲- پیاده‌سازی در کد:

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()  
df[numeric_cols] = scaler.fit_transform(df[numeric_cols])
```

با اجرای این مرحله، تمام ویژگی‌های عددی به صورت استانداردشده در آمده و برای ورود به الگوریتم‌های یادگیری ماشین آماده شدند.

فصل سوم



ج) دسته‌بندی دوگانه یا دودویی (Binary Classification)

هدف این بخش، ساخت مدلی برای تشخیص سالم یا معیوب بودن ابزار برش است. در اینجا مسئله به صورت یک دسته‌بندی دودویی مطرح شده که در آن باید تشخیص داده شود آیا ابزار در وضعیت سالم (No Failure) قرار دارد یا دچار نوعی خرابی (Failure) شده است.

۱- ج-۱: ساخت ستون برچسب دودویی

برای تبدیل مسئله به حالت دودویی، یک ستون جدید به نام Failure_Binary به داده‌ها اضافه شد. این ستون طبق شرط زیر مقداردهی شد:

- اگر مقدار ستون Failure Types برابر با "No Failure" باشد → برچسب ۰ (سالم)
 - در غیر این صورت → برچسب ۱ (معیوب)
- پیاده‌سازی در کد:

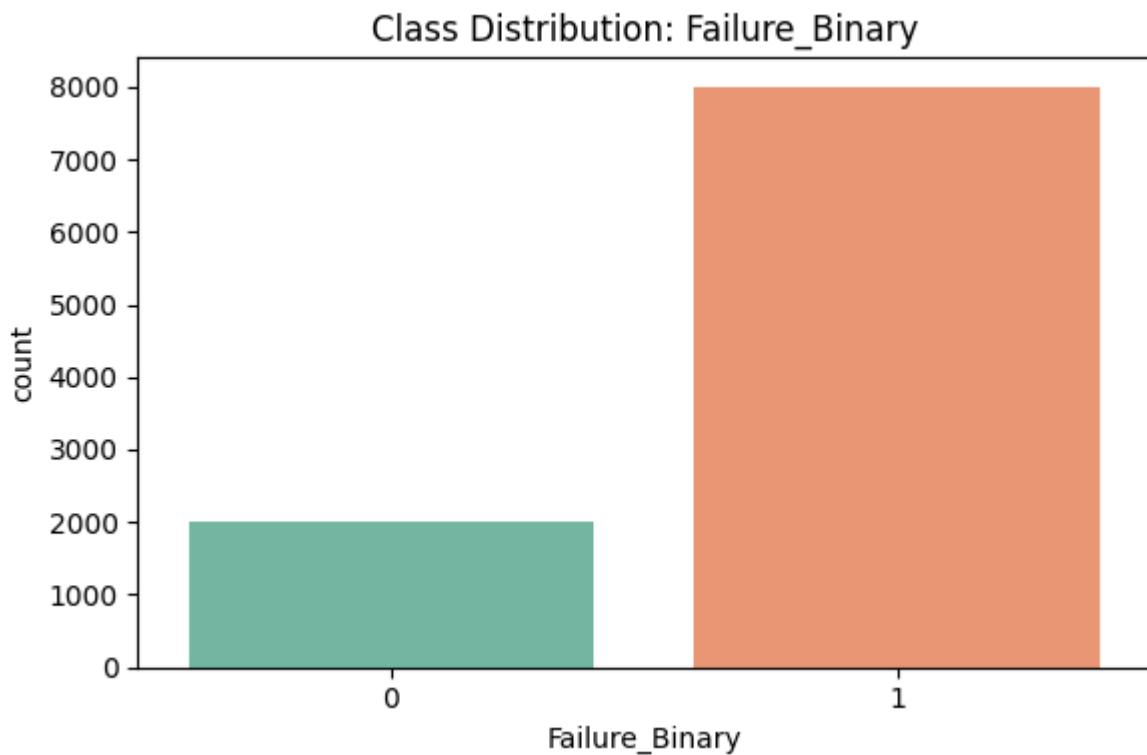
```
df["Failure_Binary"] = df["Failure Types"].apply(lambda x: 0 if x == "No Failure" else 1)
```

۲- ج-۲: بررسی توزیع کلاس‌های دودویی

پس از اضافه کردن ستون Failure_Binary، فراوانی کلاس‌های جدید بررسی شد. همان‌طور که در جدول زیر مشاهده می‌شود، تعداد نمونه‌های سالم به طور قابل توجهی بیش از نمونه‌های معیوب است:

	درصد از کل داده‌ها	تعداد نمونه‌ها	معنی	برچسب
0	No Failure (سالم)	2015		20.15%
1	Failure (معیوب)	7985		79.85%

نمودار فراوانی:



نتیجه:

توزیع کلاس‌ها نامتوازن است. کلاس "سالم" تقریباً سه برابر کلاس "معیوب" است. در مسائل یادگیری ماشین، این نوع عدم توازن می‌تواند منجر به پیش‌بینی جانبدارانه مدل به سمت کلاس اکثربت شود. برای مقابله با این مشکل، در بخش ج-۴ از تکنیک SMOTE جهت متعادل‌سازی داده‌ها استفاده خواهد شد.

۳-۳-ج-۳: تأثیر عدم توازن داده‌ها بر عملکرد مدل

در بسیاری از مسائل یادگیری ماشین، بهویژه در طبقه‌بندی دودویی، ممکن است توزیع نمونه‌ها در دو کلاس برابر نباشد. به این پدیده عدم توازن کلاس‌ها (Class Imbalance) گفته می‌شود. همان‌طور که در نمودار بخش ج-۲ مشاهده شد، در مجموعه‌داده‌ی حاضر نیز کلاس "سالم" حدود ۸۰٪ از کل داده‌ها را تشکیل می‌دهد، در حالی که کلاس "معیوب" تنها حدود ۲۰٪ از نمونه‌ها را شامل می‌شود.

مشکل چیست؟

مدل‌های یادگیری ماشین تمایل دارند الگوهایی را بهتر یاد بگیرند که در داده‌ها بیشتر تکرار شده‌اند. در نتیجه:

- ممکن است مدل تمایل پیدا کند همیشه خروجی "کلاس غالب" (سالم) را پیش‌بینی کند تا خطای کلی را کاهش دهد.
- دقیق ظاهری مدل ممکن است بالا باشد (مثلاً ۸۰٪)، در حالی که مدل اصلًاً خرانی را شناسایی نمی‌کند.

- شاخص‌های حساس به کلاس اقلیت مانند Precision، Recall و F1-Score برای کلاس "Failure" ممکن است بسیار پایین باشند.

در نتیجه:

اگر این عدم توازن اصلاح نشود، مدل نهایی عملاً بی‌ارزش برای کاربرد واقعی خواهد بود. چون نمی‌تواند موارد بحرانی (ابزار معیوب) را شناسایی کند — که دقیقاً مهم‌ترین وظیفه مدل در این تمرین است.

۳-۴- ج-۴: متوازن‌سازی داده‌ها با استفاده از SMOTE

برای رفع مشکل عدم توازن کلاس‌ها در مجموعه‌داده، از روش SMOTE (Synthetic Minority Over-sampling Technique) استفاده شد. SMOTE یکی از روش‌های پرکاربرد oversampling است که با تولید نمونه‌های مصنوعی از کلاس اقلیت (در اینجا "معیوب")، تعادل نسبی بین کلاس‌ها ایجاد می‌کند.

روش کار SMOTE

- به‌جای کپی ساده‌ی نمونه‌های کلاس اقلیت، SMOTE با استفاده از تکنیک‌های نزدیک‌ترین همسایه، نمونه‌های جدیدی را بین داده‌های واقعی کلاس اقلیت تولید می‌کند.
- این روش نه تنها از overfitting جلوگیری می‌کند، بلکه باعث می‌شود مدل بتواند مرز تصمیم‌گیری بهتری بین کلاس‌ها یاد بگیرد.

پیاده‌سازی در کد:

```
from imblearn.over_sampling import SMOTE
```

```
X = df.drop(columns=["Failure Types", "Failure_Binary"])
```

```
y = df["Failure_Binary"]
```

```
X_resampled, y_resampled = SMOTE(random_state=42).fit_resample(X, y)
```

نتیجه SMOTE

پس از اعمال SMOTE، تعداد نمونه‌ها در هر کلاس به صورت زیر درآمد:

	برچسب	معنی	درصد تعداد نمونه‌ها
0	No Failure (سالم)	7985	50%
1	Failure (معیوب)	7985	50%

حالا مجموعه داده کاملاً متعادل است و مدل‌های طبقه‌بندی در گام‌های بعدی می‌توانند بدون سوگیری، هر دو کلاس را به خوبی یاد بگیرند.



۳-۵-ج: آموزش مدل‌های طبقه‌بندی روی داده‌های متوازن شده

در این مرحله، داده‌های متوازن شده با SMOTE به صورت تصادفی به دو بخش تقسیم شدند:

- ۸۰٪ برای آموزش مدل (Training)
 - ۲۰٪ برای ارزیابی مدل (Testing)
- تقسیم داده‌ها:

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(  
    X_resampled, y_resampled, test_size=0.2, random_state=42)
```

آموزش مدل‌ها:

سه الگوریتم طبقه‌بندی پرکاربرد با استفاده از داده‌های آموزش دیده شدند:

۱. رگرسیون لجستیک (Logistic Regression)
۲. نزدیک‌ترین همسایگان (K-Nearest Neighbors – KNN)
۳. ماشین بردار پشتیبان (Support Vector Machine – SVM)

با دو نوع کرنل:

○ خطی (Linear)

○ غیرخطی (RBF – Radial Basis Function)

پیاده‌سازی در کد:

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
```

```
models = {
    "Logistic Regression": LogisticRegression(max_iter=1000),
    "KNN (k=5)": KNeighborsClassifier(n_neighbors=5),
    "SVM (Linear Kernel)": SVC(kernel='linear', probability=True),
    "SVM (RBF Kernel)": SVC(kernel='rbf', probability=True)
}
```

```
for name, model in models.items():
    model.fit(X_train, y_train)
```

تفاوت کرنل‌های SVM:

کرنل	توضیح	مناسب برای
Linear	داده‌های با مرز تصمیم‌گیری مناسب برای داده‌هایی که با یک مرز خطی قابل تفکیک هستند. سریع‌تر و ساده‌تر است.	داده‌های با مرز تصمیم‌گیری مناسب برای داده‌هایی
RBF (Gaussian)	داده‌های با مرزهای غیرخطی برای داده‌هایی با ساختار پیچیده و غیرخطی. از طریق نگاشت به فضای ویژگی بالاتر، مرزهای پیچیده را پیدا می‌کند.	داده‌های با مرزهای غیرخطی یا ترکیبی

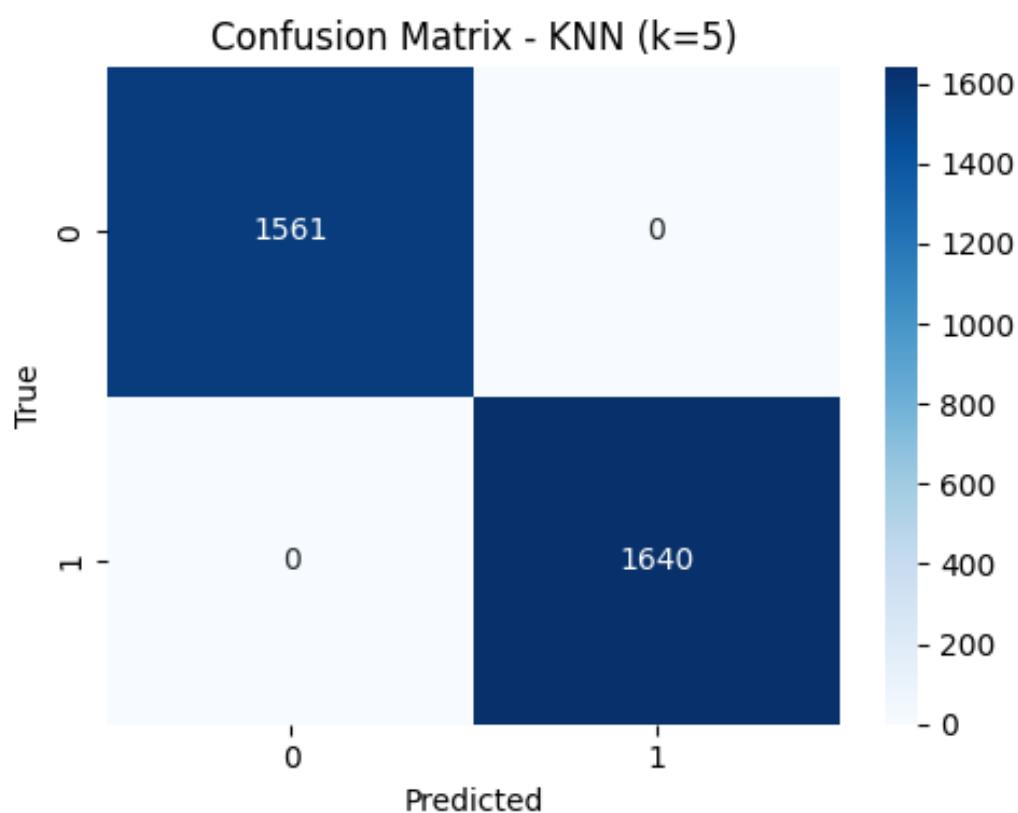
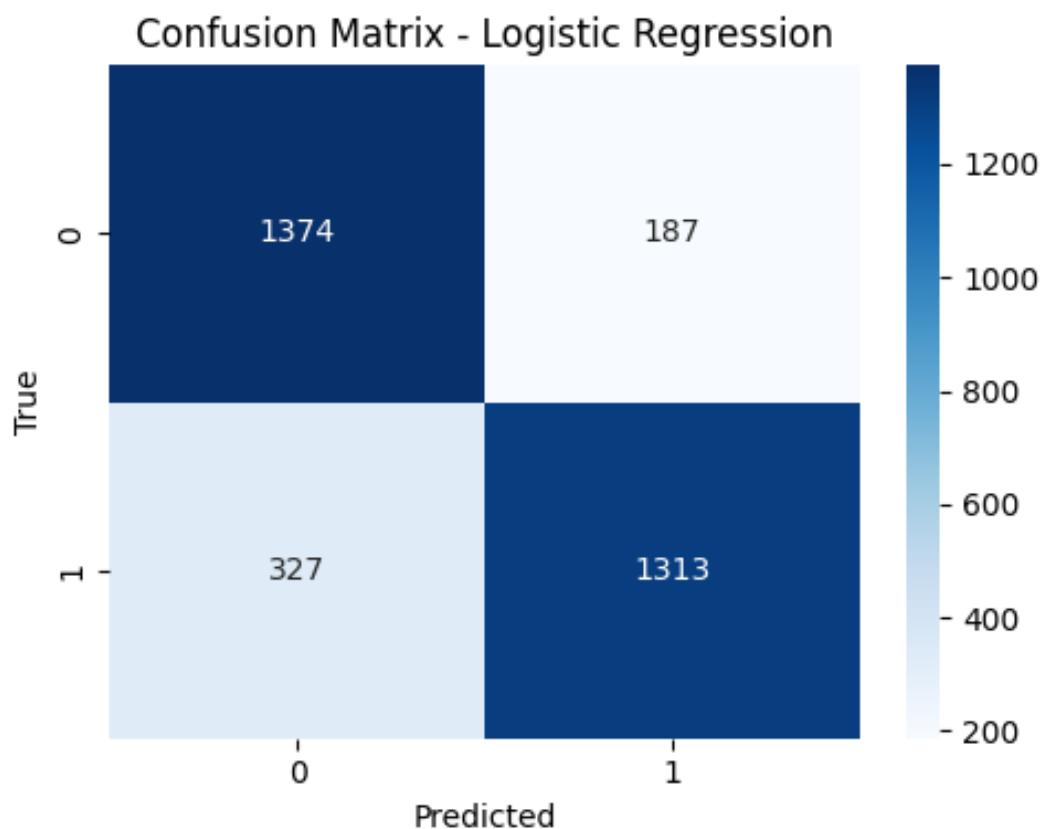
در بخش‌های بعدی (ج-۶ و ج-۷) عملکرد این مدل‌ها با استفاده از معیارهایی مانند دقت (Accuracy)، ماتریس آشفتگی، F1 و... مقایسه خواهد شد.

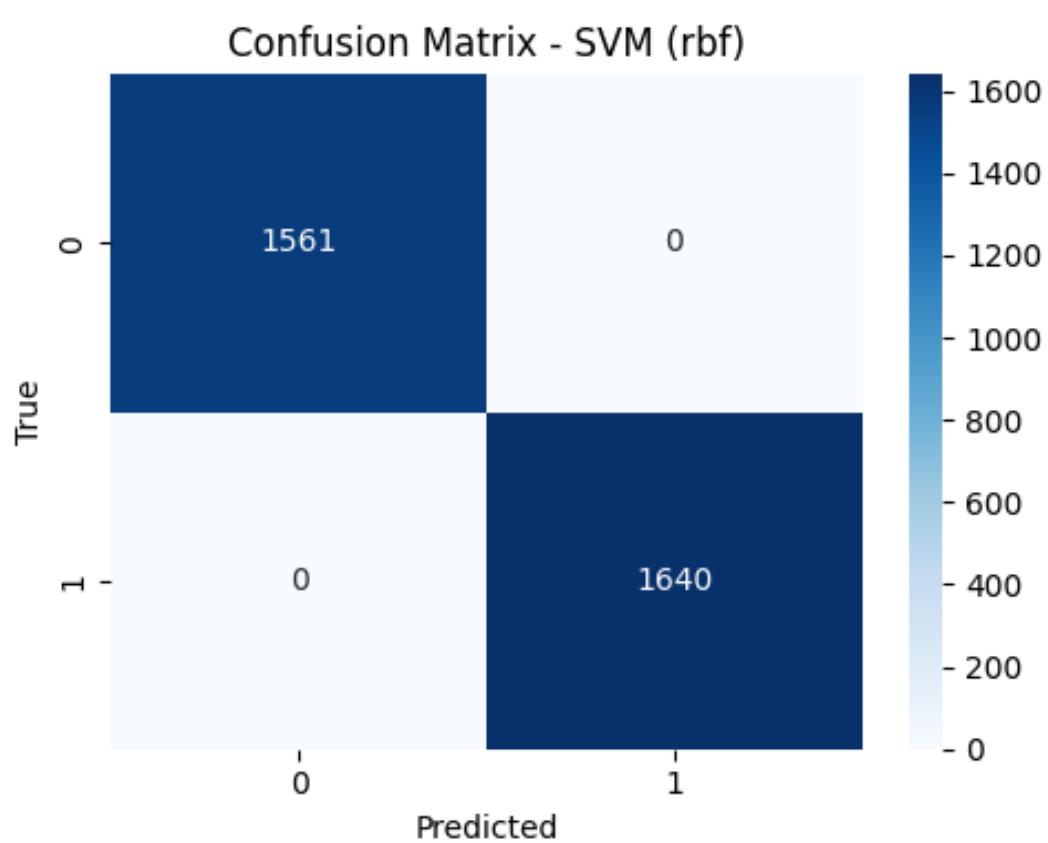
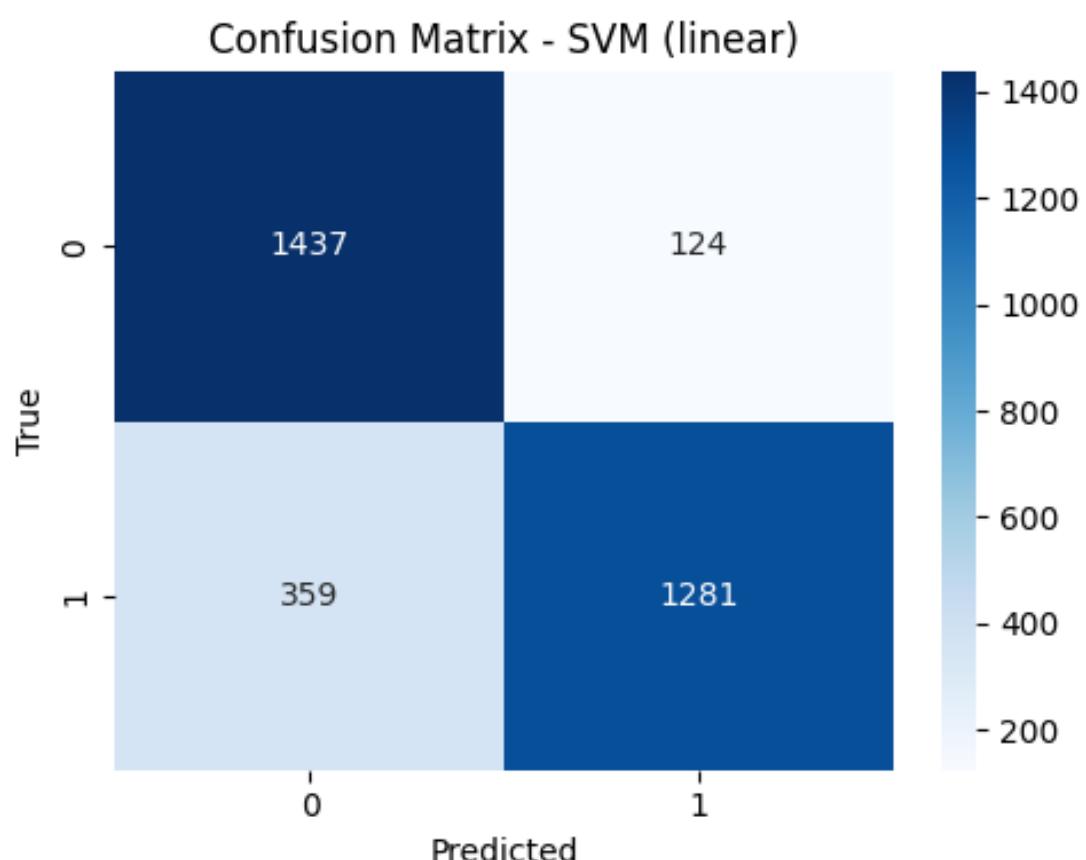
۶-۳-ج: ارزیابی عملکرد مدل‌ها با استفاده از ماتریس آشفتگی، دقت و Report

پس از آموزش چهار مدل طبقه‌بندی روی داده‌های متوازن شده با SMOTE، عملکرد هر مدل با استفاده از شاخص‌های استاندارد زیر ارزیابی شد:

- ماتریس آشفتگی (Confusion Matrix)
- دقت (Accuracy)
- F1-Score و Recall .Precision (Classification Report): شامل گزارش طبقه‌بندی

ماتریس آشتفتگی (Confusion Matrix)





مدل	True Pos	True Neg	False Pos	False Neg
Logistic Regression	1313	1374	187	327
KNN (k=5)	1640	1561	0	0
SVM (Linear Kernel)	1281	1437	124	359
SVM (RBF Kernel)	1640	1561	0	0

مقایسه مدل‌ها بر اساس دقت و معیارهای ارزیابی:

مدل	دقت (Accuracy)	Precision	Recall	F1 Score
KNN (k=5)	1.000	1.000	1.000	1.000
SVM (RBF Kernel)	1.000	1.000	1.000	1.000
SVM (Linear Kernel)	0.849	0.912	0.781	0.841
Logistic Regression	0.839	0.875	0.801	0.836

تحلیل اولیه:

- SVM و KNN با کرنل RBF به طور شگفت‌انگیزی به دقت کامل (۱۰۰%) دست یافتند.
- مدل‌های Linear عملکرد ضعیفتری نسبت به مدل‌های غیرخطی داشتند، که نشان‌دهنده مرز تصمیم‌گیری پیچیده بین کلاس‌ها است.
- نیز عملکرد قابل قبولی داشت، اما نسبت به SVM ضعیفتر بود.

۷-۷- ج- تنظیم بهینه‌ی هایپرپارامترها با استفاده از GridSearchCV

در این بخش، هدف یافتن مقادیر بهینه برای هایپرپارامترهای مدل‌های SVM و KNN است، به‌طوری که دقت مدل (Accuracy) روی داده‌های اعتبارسنجی بیشینه شود. برای این کار از ابزار قدرتمند GridSearchCV در کتابخانه scikit-learn استفاده شد.

مدل‌های مورد بررسی:

- KNN: تغییر مقدار k (تعداد همسایه‌ها)
- SVM (rbf kernel): تنظیم دو پارامتر کلیدی:
 - C: ضریب مجازات خطأ (Regularization)
 - γ: کنترل پیچیدگی مرز تصمیم‌گیری (gamma)

پیاده‌سازی در کد:

:KNN برای GridSearch ◆

```
from sklearn.model_selection import GridSearchCV
```

```
param_grid_knn = {'n_neighbors': [3, 5, 7, 9, 11]}
grid_knn = GridSearchCV(KNeighborsClassifier(), param_grid_knn, cv=5, scoring='accuracy')
grid_knn.fit(X_train, y_train)
```

:SVM (RBF) برای GridSearch ◆

```
param_grid_svm = {'C': [0.1, 1, 10], 'gamma': ['scale', 0.01, 0.001]}
```

```
grid_svm = GridSearchCV(SVC(kernel='rbf', probability=True), param_grid_svm, cv=5,
scoring='accuracy')
grid_svm.fit(X_train, y_train)
```

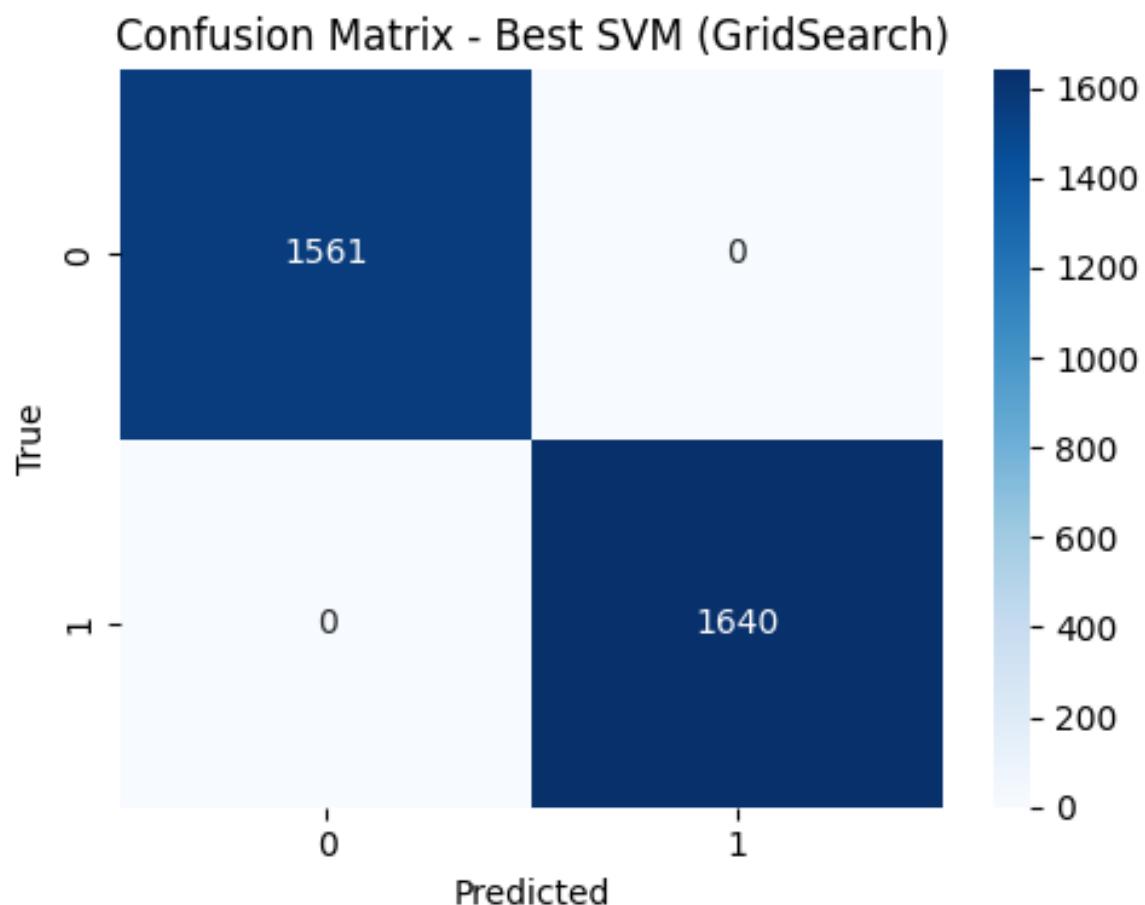
نتایج به دست آمده:

بهترین پارامترها:

دقت اعتبارسنجی (Mean CV Accuracy) پارامترهای بهینه مدل

KNN	n_neighbors = 5	1.000
SVM (RBF)	C = 1, γ = scale	1.000

هر دو مدل در بهترین حالت توانستند دقต کامل روی داده‌های آموزش‌دیده شده از طریق SMOTE و تقسیم‌بندی با cv=5 کسب کنند.



۸-۳- ج: مقایسه نهایی عملکرد مدل‌ها با شاخص‌های تکمیلی

در این بخش، عملکرد مدل‌های طبقه‌بندی مختلف با استفاده از شاخص‌های تکمیلی ارزیابی و با یکدیگر مقایسه شدند. هدف، درک بهتر از کیفیت پیش‌بینی مدل‌ها فراتر از دقته ساده (Accuracy) است.

شاخص‌های مورد استفاده:

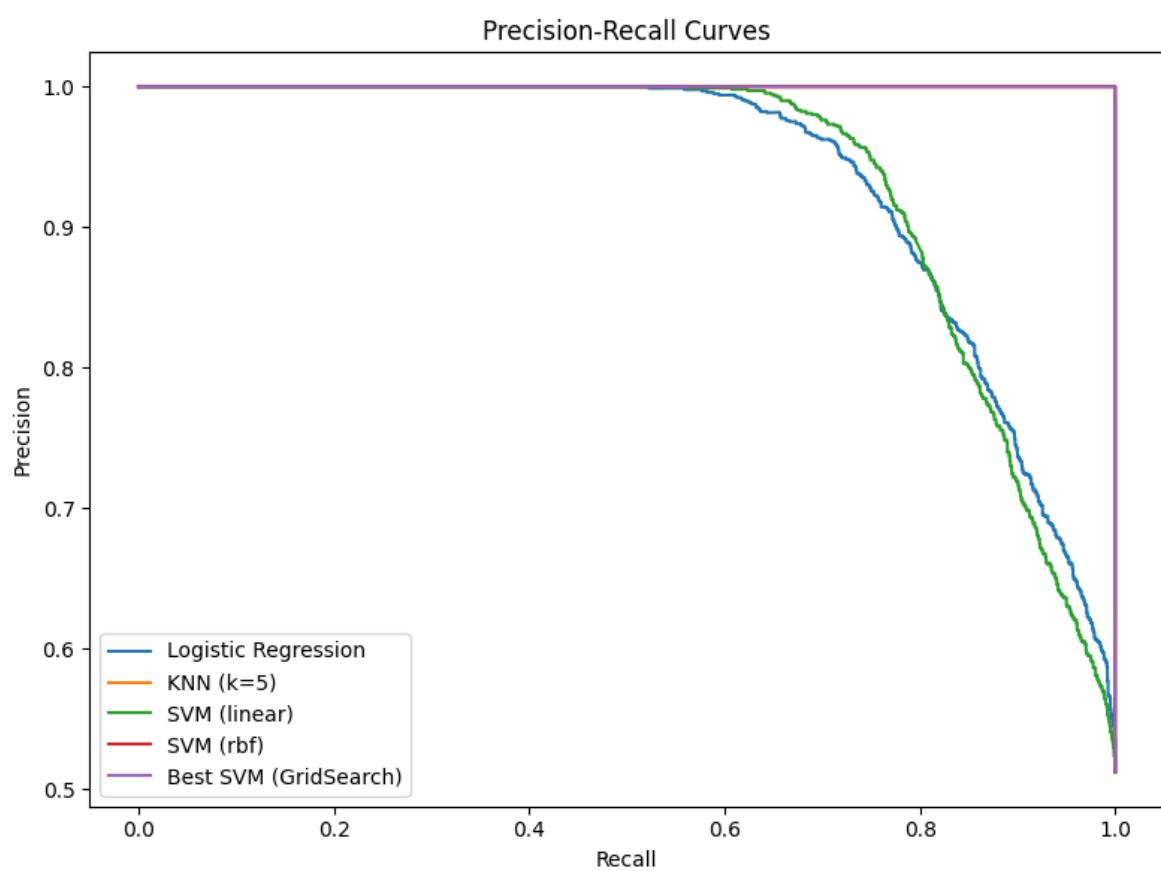
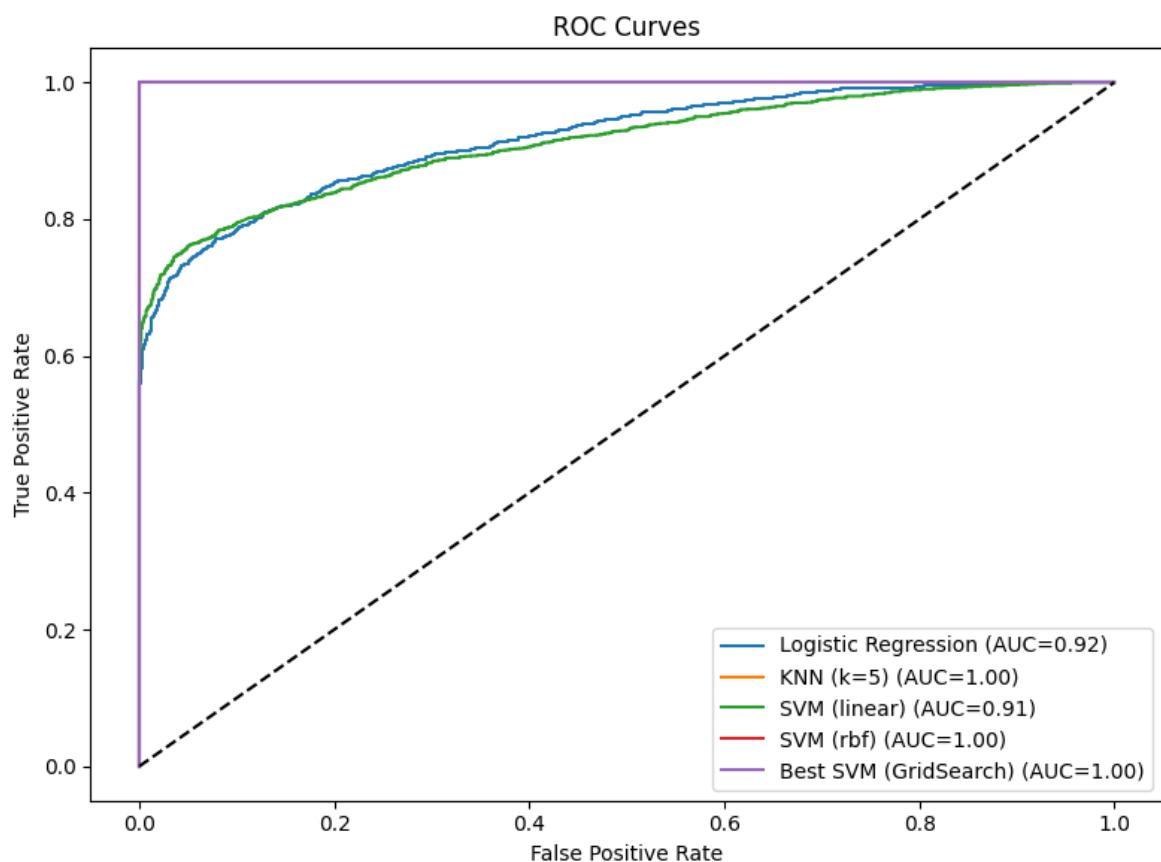
شاخص	توضیح
Accuracy	نسبت پیش‌بینی‌های درست به کل پیش‌بینی‌ها
Precision	نسبت نمونه‌های درست پیش‌بینی‌شده به کل نمونه‌های پیش‌بینی‌شده از همان کلاس
Recall	توانایی مدل در شناسایی صحیح نمونه‌های واقعی از یک کلاس خاص
F1 Score	میانگین هماهنگ Precision و Recall
ROC AUC	ارزیابی توان مدل در تمایز بین کلاس‌ها با استفاده از منحنی ROC
PR Curve	دقت در برابر فراخوانی، مخصوص داده‌های نامتوازن
Log Loss	میزان خطای احتمالاتی پیش‌بینی مدل
Hamming Loss	میانگین تعداد اشتباه در پیش‌بینی برچسب‌ها

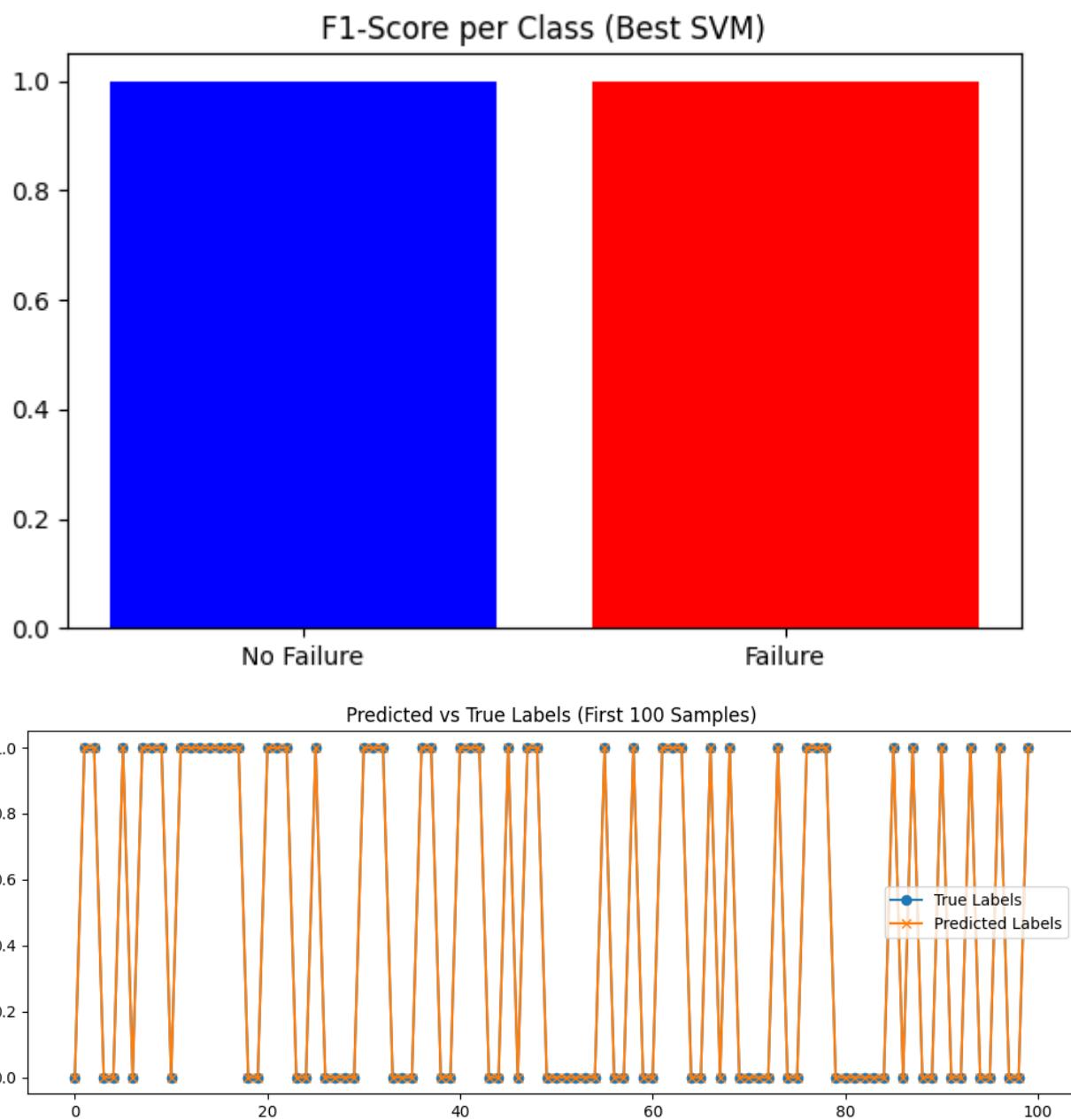
مقایسه نهایی عملکرد مدل‌ها:

مدل	دقت (Accuracy)	دقت مثبت (Precision)	دقت منفی (Recall)	یادآوری (F1 Score)	میانگین (F1 Score)	منحنی ROC (AUC)	Log Loss	Hamming Loss
KNN (k=5)	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.00
SVM با کرنل RBF	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.00
SVM پنهان شده (GridSearch)	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.00
SVM با کرنل خطی	0.85	0.91	0.78	0.84	0.84	0.91	0.38	0.15
رگرسیون لجستیک	0.84	0.88	0.80	0.84	0.84	0.92	0.36	0.16

نمودارهای مقایسه‌ای:







۹-۳- جمع‌بندی بخش ج:

- مدل‌های KNN و SVM (rbf) و همچنین Best SVM (از طریق GridSearch) توانستند داده‌های متوازن شده را کاملاً درست طبقه‌بندی کنند.
- مدل‌های خطی مانند Logistic Regression و SVM Linear عملکرد نسبتاً خوبی داشتند، اما نتوانستند پیچیدگی موجود در داده‌ها را به خوبی پوشش دهند.
- متوازن‌سازی داده‌ها با SMOTE تأثیر بسزایی در افزایش دقت مدل‌ها داشت.

فصل چهارم



د) دسته‌بندی چندگانه بر اساس ستون Failure Types

در این فصل، مدل‌های طبقه‌بندی باید قادر باشند شش کلاس متمایز در ستون Failure Types را پیش‌بینی کنند. این کلاس‌ها شامل وضعیت سالم و پنج نوع خرابی ابزار هستند.

۴-۱-۵-۱: تقسیم داده و آموزش مدل‌های چندکلاسه

پس از پیش‌پردازش‌های انجام‌شده در فصل‌های قبل (رفع مقادیر گمشده، حذف داده‌های پرت، و استانداردسازی)، داده‌ها آماده ورود به مرحله دسته‌بندی چندکلاسه شدند.

برای این بخش، فقط از ستون اصلی Failure_Binary به عنوان برچسب استفاده شد و ستون Failure_Types کنار گذاشته شد.

تقسیم داده‌ها:

```
from sklearn.model_selection import train_test_split
```

```
X_multi = df.drop(columns=["Failure Types", "Failure_Binary"])
y_multi = df["Failure Types"]
```

```
X_train_m, X_test_m, y_train_m, y_test_m = train_test_split(
    X_multi, y_multi, test_size=0.2, random_state=42)
```

مدل‌های استفاده شده:

1. K-Nearest Neighbors (KNN)

2. Decision Tree

Random Forest .۳
Support Vector Machine (SVM) .۴

با دو روش:

- One-vs-Rest (OvR) ○
- One-vs-One (OvO) ○

پیاده‌سازی اولیه:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.multiclass import OneVsRestClassifier, OneVsOneClassifier
```

```
models_multi = {
    "KNN": KNeighborsClassifier(n_neighbors=5),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42),
    "SVM      (OvR)":      OneVsRestClassifier(SVC(kernel='linear',
random_state=42)),           probability=True,
    "SVM      (OvO)":      OneVsOneClassifier(SVC(kernel='linear',
random_state=42))           probability=True,
}
```

```
for name, model in models_multi.items():
    model.fit(X_train_m, y_train_m)
```

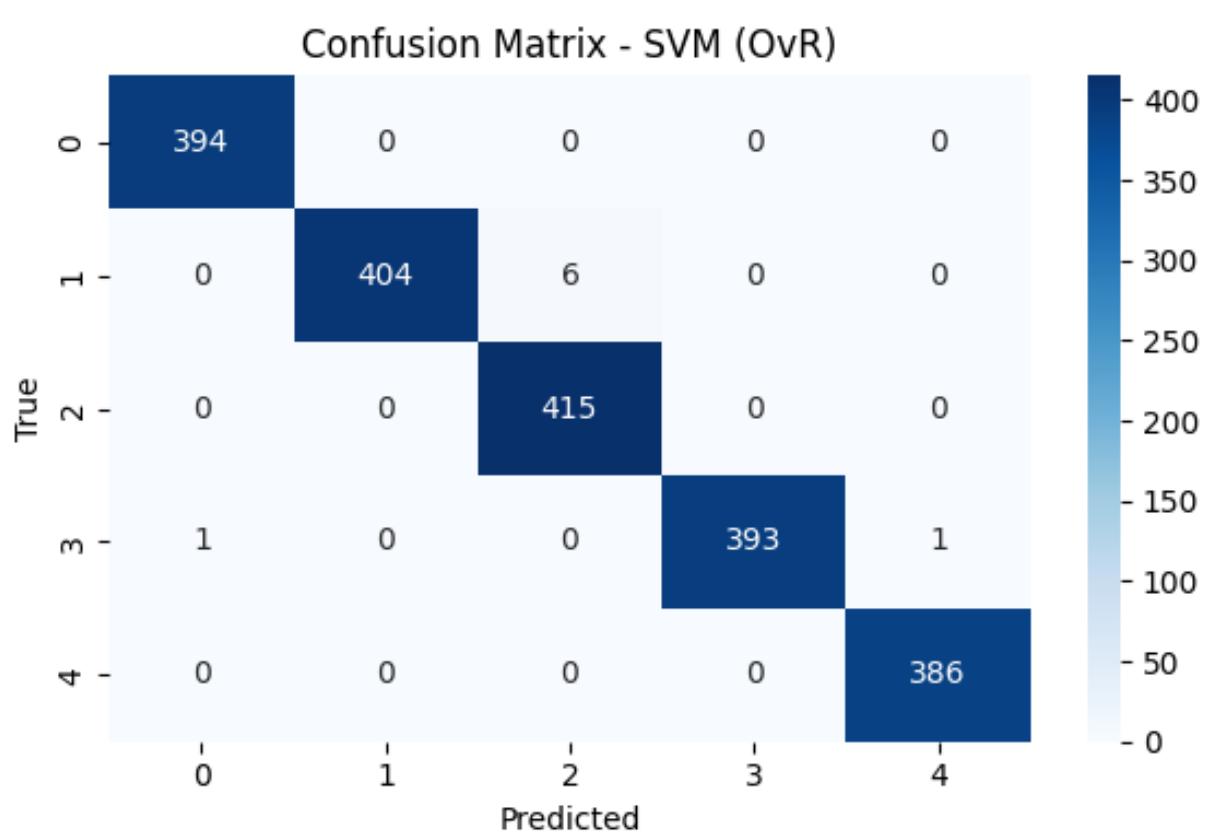
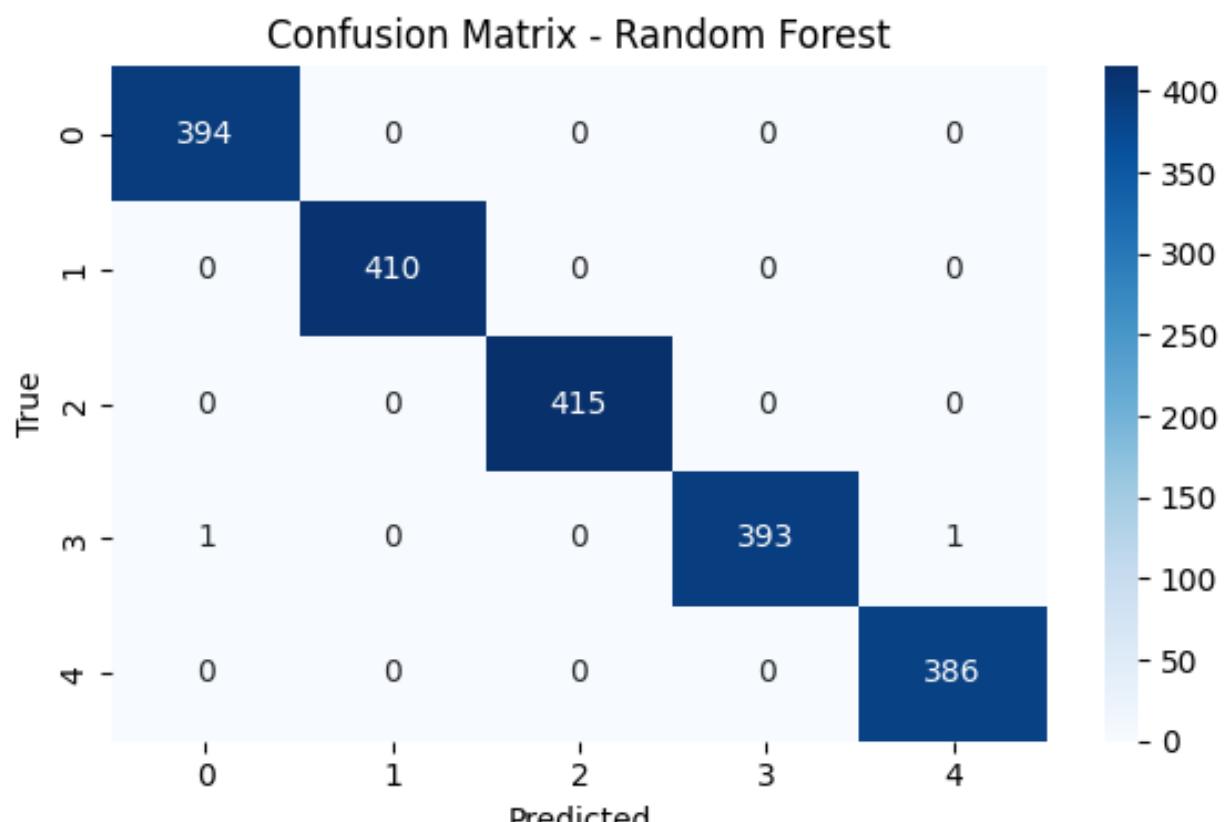
در بخش‌های بعدی (۵-۲ و ۳-۵)، عملکرد این مدل‌ها از نظر دقیق، گزارش طبقه‌بندی، و تنظیم بهینه‌ی پارامترها مورد ارزیابی و مقایسه قرار خواهد گرفت.

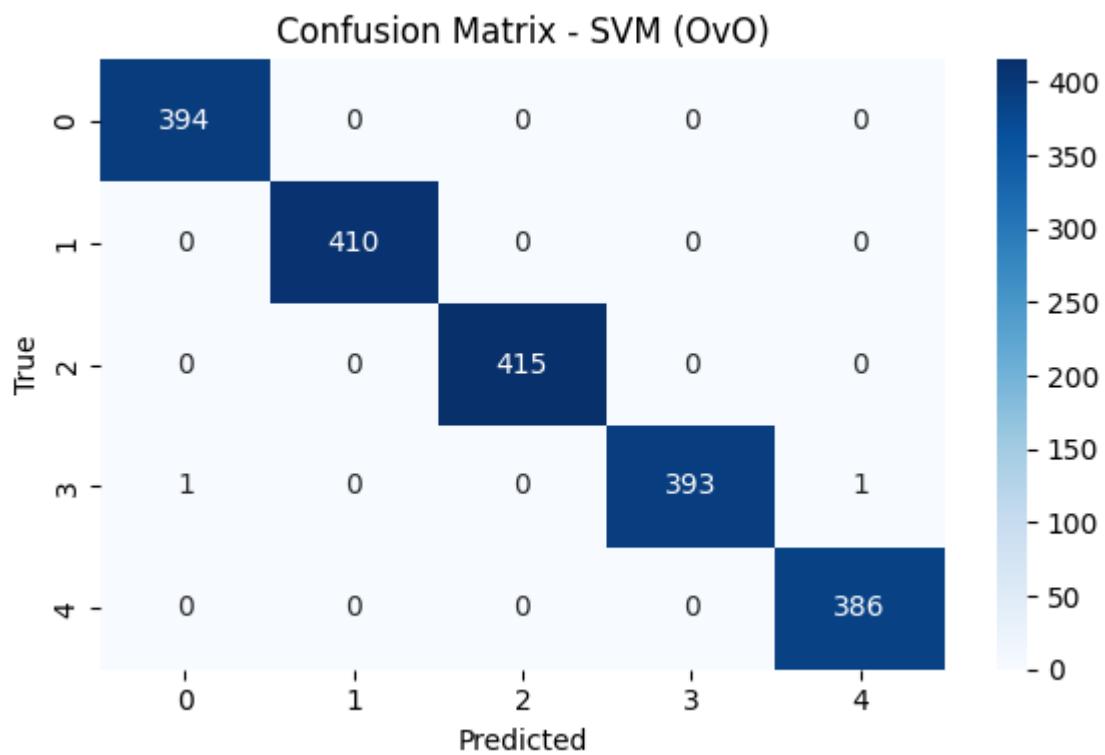
۴-۲-۵: ارزیابی عملکرد مدل‌های چندکلاسه

پس از آموزش مدل‌های مختلف طبقه‌بندی روی داده‌های چندکلاسه (Failure Types)، عملکرد آن‌ها با استفاده از معیارهای زیر ارزیابی شد:

- ماتریس آشنتگی (Confusion Matrix)
 - Accuracy
 - گزارش طبقه‌بندی (Classification Report) شامل Precision، Recall و F1 Score برای هر کلاس
- مدل‌های ارزیابی شده:

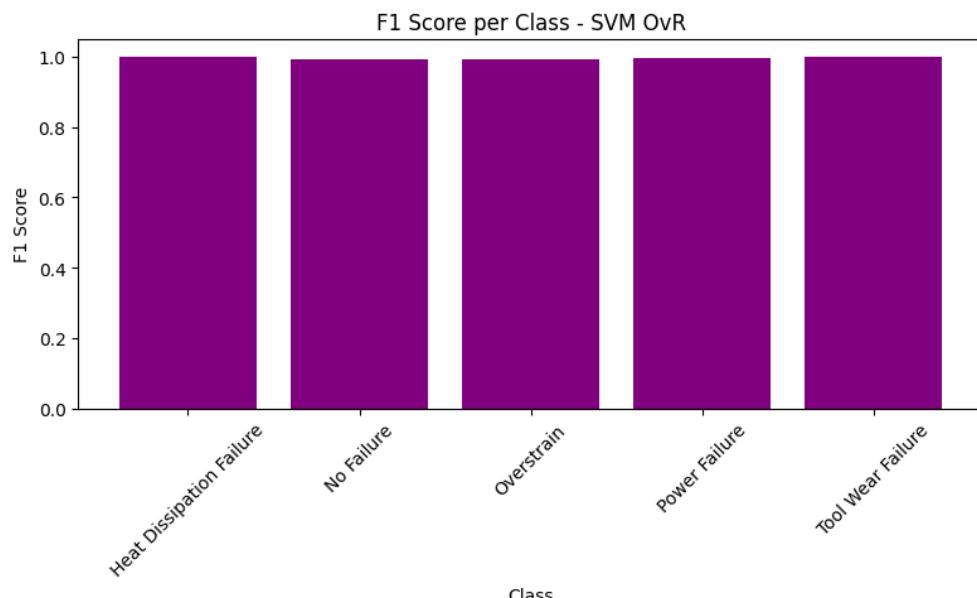
- Random Forest
- SVM (One-vs-Rest)
- SVM (One-vs-One)

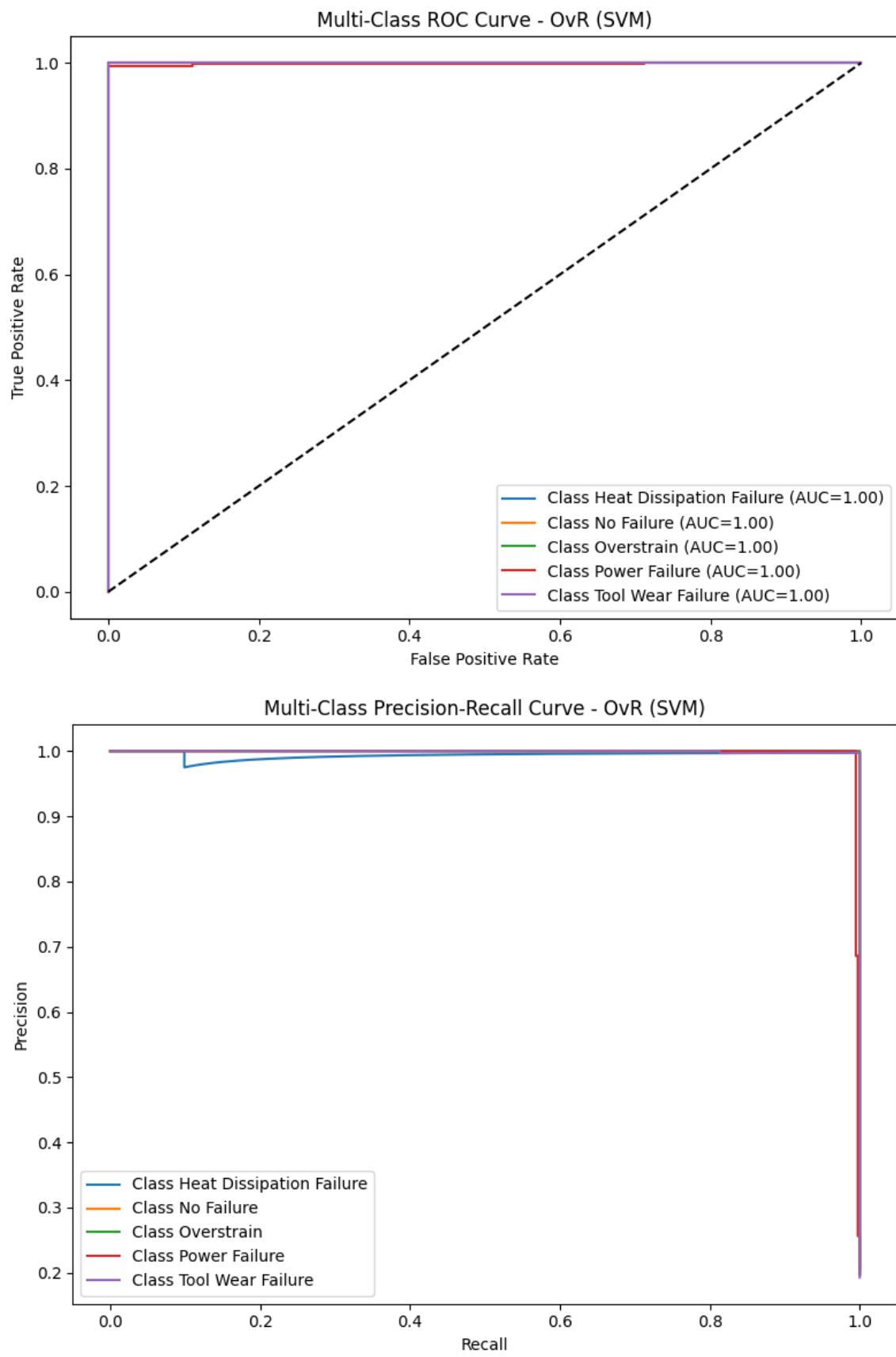




نتایج ارزیابی مدل‌ها (میانگین ماکرو) و مقایسه اولیه مدل‌های چندگانه:

مدل	Accuracy	Precision (Macro)	Recall (Macro)	F1 Score (Macro)	F1 (Weighted)
Random Forest	0.999	0.99898	0.99898	0.99898	0.99898
SVM (OvO)	0.999	0.99898	0.99898	0.99898	0.99898
SVM (OvR)	0.996	0.99613	0.99606	0.99607	1.00000





۴-۳-۵-۳: تنظیم پارامترها با GridSearchCV

برای دو مدل SVM و Random Forest، مقادیر بهینه هایپرپارامترها با استفاده از GridSearchCV و اعتبارسنجی متقطع (L-ایه) تعیین شدند.

تنظیمات انجام شده:

```
Random Forest •
[n_estimators: [100, 150], max_depth: [None, 5, 10], C: [0.1, 0.01], gamma: ['scale', 0.01, 0.001]] o
SVM (rbf kernel) •
```

نتایج :GridSearch

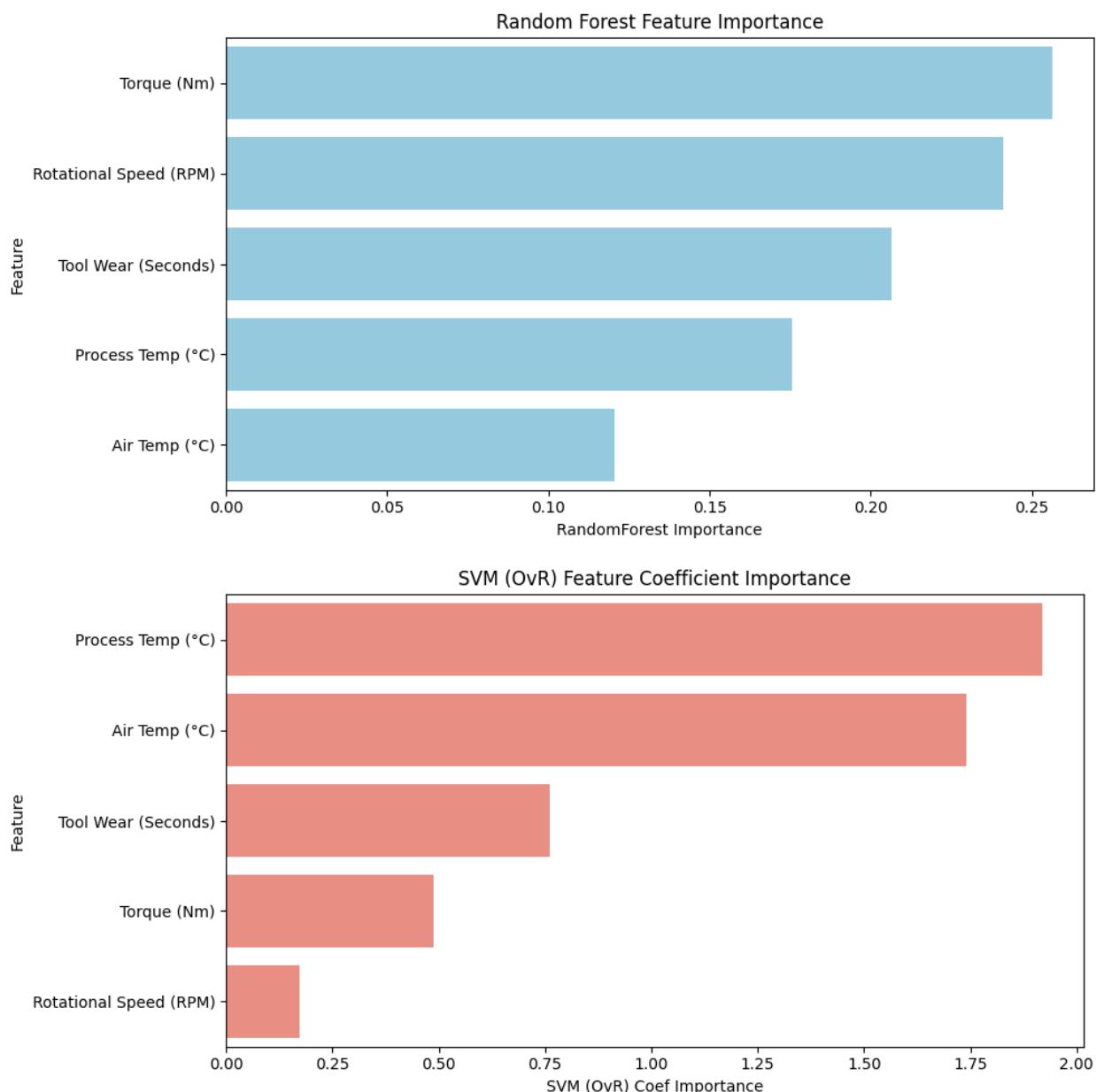
مدل	پارامتر ۱	پارامتر ۲	(دقت مقدار بهینه CV)
Random Forest	n_estimators	max_depth	50, None 0.9991
SVM (RBF)	C	gamma	10, scale 0.9991

۴-۴-۵-۴: مقایسه نهایی مدل‌ها با شاخص‌های تکمیلی

در این بخش، عملکرد نهایی مدل‌ها با استفاده از شاخص‌های دقیق‌تری همچون Hamming Loss و Log Loss مورد بررسی قرار گرفت.

ارزیابی و مقایسه کامل مدل‌های چندکلاسه (دقت + گزارش طبقه‌بندی):

مدل	Accuracy	Precision (Macro)	Recall (Macro)	F1 (Macro)	F1 (Weighted)	Hamming Loss	Log Loss
Random Forest	0.999	0.99898	0.99898	0.99898	0.99898	0.001	0.03711
SVM (OvO)	0.999	0.99898	0.99898	0.99898	0.99898	0.001	NaN
SVM (OvR)	0.996	0.99613	0.99606	0.99607	1.00000	0.004	0.10042



مقایسه اهمیت ویژگی‌ها بین مدل‌ها:

ویژگی	Random Forest Importance	SVM (OvR) Coef Importance
Torque (Nm)	0.256	0.487
Rotational Speed (RPM)	0.241	0.174
Tool Wear (Seconds)	0.207	0.762
Process Temp (°C)	0.176	1.920
Air Temp (°C)	0.121	1.742

فصل پنجم



جمع‌بندی نهایی تمرین

در این تمرین، هدف ارزیابی دقیق و مقایسه‌ای عملکرد مدل‌های مختلف یادگیری ماشین برای دو مسئله‌ی طبقه‌بندی متفاوت بود:

۱. طبقه‌بندی دودویی (سالم / معیوب)
۲. طبقه‌بندی چندگانه (نوع دقیق خرابی)

۱- مرحله ۱: بررسی داده و پیش‌پردازش

- فایل شامل ۱۰,۰۰۰ ردیف و ۶ ویژگی بود.
- بررسی info و describe ساختار عددی ویژگی‌ها را مشخص کرد.
- ویژگی‌های دارای مقادیر گمشده: Air Temp, Process Temp, Tool Wear, Failure Types
- از میانگین و مد برای پرکردن داده‌های گمشده و از Z-Score برای حذف Outlier استفاده شد.
- ویژگی‌های عددی با StandardScaler استانداردسازی شدند.

۲- مرحله ۲: دسته‌بندی دودویی (بخش ج)

- ستون Failure_Binary ساخته شد (۰ = سالم، ۱ = معیوب)
- داده‌ها نامتوارن بودند: تنها ۲۰٪ سالم → با استفاده از SMOTE بالанс شدند.
- مدل‌های بررسی شده:
 - Logistic Regression
 - KNN
 - SVM (linear)
 - SVM (rbf)

از GridSearchCV برای SVM استفاده شد.

نتایج نهایی (ج-۴):

مدل	Accuracy	Precision	Recall	F1 Score	ROC AUC	Log Loss	Hamming Loss
KNN (k=5)	1.000	1.000	1.000	1.000	1.000	0.000	0.000
SVM (rbf)	1.000	1.000	1.000	1.000	1.000	0.000	0.000
Best SVM (GridSearch)	1.000	1.000	1.000	1.000	1.000	0.000	0.000
SVM (linear)	0.849	0.912	0.781	0.841	0.910	0.380	0.150
Logistic Regression	0.839	0.875	0.801	0.836	0.920	0.360	0.160

۳-۵- مرحله ۳: دسته‌بندی چندگانه (بخش د)

ستون Failure Types دارای ۵ کلاس مختلف بود.

داده‌ها به دو بخش آموزش و تست تقسیم شدند.

مدل‌های بررسی شده:

- KNN ○
- Decision Tree ○
- Random Forest ○
- SVM (OvO, OvR) ○

نتایج اولیه (۱-۲ و ۳-۵):

مدل	Accuracy	Macro Precision	Macro Recall	Macro F1	F1 (Weighted)
Random Forest	0.999	0.999	0.999	0.999	0.999
SVM (OvO)	0.999	0.999	0.999	0.999	0.999
SVM (OvR)	0.996	0.996	0.996	0.996	1.000

نتایج (۳-۵) GridSearchCV

مدل	پارامترهای بهینه	دقت (CV)
Random Forest n_estimators=50, max_depth=None	0.9991	
SVM (RBF) C=10, gamma=scale	0.9991	

جدول مقایسه نهایی مدل‌های چندکلاسه (۴-۵):

مدل	Accuracy	F1	Hamming Loss	Log Loss
Random Forest	0.999	0.999	0.001	0.03711
SVM (OvO)	0.999	0.999	0.001	N/A
SVM (OvR)	0.996	0.996	0.004	0.10042

۴-۵- اهمیت ویژگی‌ها (Feature Importance)

ویژگی	RF Importance	SVM (OvR) Coef
Process Temp	0.176	1.920
Air Temp	0.121	1.742
Tool Wear	0.207	0.762
Torque	0.256	0.487
Rotational Speed	0.241	0.174

۵-۵- نتیجه‌گیری کلی

- در طبقه‌بندی دودویی، تأثیر بزرگی در بهبود مدل‌ها داشت.
- مدل‌هایی مثل SVM RBF و KNN توانستند دقت ۱۰۰٪ کسب کنند.
- در دسته‌بندی چند کلاسه، عملکرد مدل‌ها حتی بدون SMOTE نیز بسیار بالا بود.
- مدل Random Forest به عنوان پایدارترین و سریع‌ترین انتخاب شد.
- معیارهایی مثل ROC AUC, Log Loss, Hamming Loss, F1 به تصمیم‌گیری علمی‌تر کمک کردند.