بسمه تعالی

دانشگاه تهران

دانشکده مهندسی مکانیک

تمرین سری سوم درس هوش مصنوعی

نام و نام خانوادگی:

محمدمهدی تویسرکانی

شماره دانشجویی:

۸۱۰۶۰۳۰۰۵

تاریخ تحویل:

۱۴۰۴/۰۲/۲۴

نیم‌سال دوم سال تحصیلی ۱۴۰۳-۱۴۰۴

فهرست مطالب

عنوان **صفحه**

هوش مصنوعی و یادگیری ماشین (دسته‌بندی)

❖ اطلاعات مسئله:

هدف این تمرین ارزیابی عملکرد روش‌های دسته‌بندی دوگانه و چندگانه است. برای این کار مسالهٔ پایش سلامت و تشخیص عیب ابزار برش یک دستگاه فرز در نظر گرفته شده است.

دادگان مورد نظر (فایل milling_machine.csv) شامل ۱۰۰۰ داده است که هر داده، وضعیت ابزار فرز (ستون ششم) را بر اساس پنج ویژگی ("دمای هوا"، "دمای فرآیند"، "سرعت چرخشی ابزار فرز"، "گشتاور وارد به محور ابزار" و "مدت زمان قرارگیری ابزار در معرض سایش") (ستون‌های اول تا پنجم) نشان می‌دهد.

الف) بررسی داده‌های خام

برای آشنایی بهتر با دادگان مورد نظر:

الف-۱) ساختار کلی داده‌ها را با روش‌های info و describe بدست آورید.

الف-۲) برای هر ویژگی، تعداد و نسبت مقادیر ناموجود (missing values) را بدست آورید.

الف-۳) correlation دو به دو ویژگی‌ها را بررسی کنید و با تشکیل ماتریس همبستگی تعیین کنید که وضعیت ابزار فرز به کدام ویژگی‌ها وابستگی بیشتری دارد.

الف-۴) برای سه ویژگی دارای بیشترین تاثیر بر خروجی (بر پایهٔ بند الف-۳) نمودار تعداد مشاهدات هر مقدار منحصر به فرد را رسم کنید.

ب) پیش پردازش داده‌ها

در این بخش لازم است کاستی‌های احتمالی موجود در دادگان (مانند مقادیر خارج از بازهٔ مجاز یا مقادیر ناموجود یا داده‌های پرت) که می‌تواند بر همگرایی و تعمیم‌پذیری مدل تاثیر منفی بگذارد برطرف شود.

ب-۱) ابتدا بررسی کنید که کدام داده‌ها بیشترین میزان مقادیر ناموجود (missing values) را دارند و سپس با توجه به توضیحاتی که در ویدیوی تکمیلی در این باره داده شده است مشکل مقادیر ناموجود را برای همهٔ ویژگی‌ها حل کنید (با ذکر روش بکار رفته و دلیل انتخاب آن).

ب-۲) فرآیندهای standardizing و normalizing را (برای ویژگی‌های کمی) توضیح دهید. آیا در این تمرین نیاز به این فرآیندها هست؟ اگر نیاز به این کار هست آن را اعمال کنید.

ج) دسته‌بندی دوگانه

هدف این بخش دسته‌بندی دوگانهٔ داده‌هاست به گونه‌ای که بتوان سالم یا معیوب بودن ابزار را تشخیص داد.

ج-۱) ابتدا در محیط پایتون یک ستون به ستون‌های فایل دادگان اضافه کنید که محتوای آن برای ابزارهای سالم برچسب No Failure و برای ابزارهای آسیب دیده برچسب Failure باشد. به این ترتیب داده‌ها به دو دستهٔ سالم و معیوب تقسیم می‌شوند که اگر هدف خود را این ستون جدید قرار دهید یک مسئلهٔ دسته‌بندی دوگانه خواهید داشت.

ج-۲) با رسم نمودار میله‌ای (Chart Bar) برای توزیع دسته‌های دوگانهٔ جدید، عدم توازن احتمالی داده‌ها رانشان دهید.

ج-۳) توضیح دهید که عدم توازن داده‌ها چه مشکلی برای مدل ایجاد می‌کند.

ج-۴) با استفاده از روش‌های متوازن‌سازی داده‌ها (مانند smote) مشکل یاد شده را برطرف کنید.

ج-۵) دادگان پالایش شده را به صورت تصادفی به دو بخش آموزش (۸۰٪) و آزمون (۲۰٪) تقسیم کنید(Random_State = 42) و با استفاده از کتابخانه scikit-learn مدل‌های زیر را آموزش دهید:

Logistic Regression
K-Nearest-Neighbors
Support Vector Machine

* برای مدل SVM هسته‌های خطی و غیرخطی را بکار ببرید و تفاوت احتمالی نتایج را تفسیر کنید.

ج-۶) با تشکیل ماتریس آشفتگی (confusion matrix) ، دقت (accuracy) و Classification Report هر مدل را بدست آورید و نتایج را در جدولی ارایه کنید.

ج-۷) برای مدل‌های یاد شده دو پارامتر را از میان هایپرپارامترها انتخاب کرده و آن‌ها را تغییر دهید (در مورد مدل KNN تنها مقدار K را تغییر دهید) و به کمک تابع GridSearchCV مقادیر بهینهٔ پارامترها را (که دقت مدل را بیشینه می‌کند) بدست آورید.

ج-۸) به کمک شاخص‌های معرفی شده در درس، عملکرد مدل‌های مختلف را با یکدیگر مقایسه کنید.

د) دسته‌بندی چندگانه

هدف این بخش دسته‌بندی چندگانهٔ داده‌هاست به گونه‌ای که بتوان سالم بودن یا نوع آسیب ابزار را تشخیص داد. در این بخش هدف شما همان ستون Failure Types خواهد بود.

د-۱) دادگان پالایش شده را به صورت تصادفی به دو بخش آموزش (۸۰٪) و آزمون (۲۰٪) تقسیم کنید(Random_State = 42) و با استفاده از کتابخانه scikit-learn مدل‌های زیر را آموزش دهید:

K-Nearest-Neighbors
Decision Tree

Random Forest
Support Vector Machine

* برای مدل SVM، با استفاده از روش‌های "یکی در برابر یکی" یا "یکی در برابر همه" دسته‌بندی چندگانه را انجام دهید.

د-۲) با تشکیل ماتریس آشفتگی (confusion matrix) ، دقت (accuracy) و Classification Report هر مدل را بدست آورید و نتایج را در جدولی ارایه کنید.

د-۳) برای مدل‌های ذکر شده دو پارامتر را از میان هایپرپارامترها انتخاب کرده و آن‌ها را تغییر دهید (در مورد مدل KNN تنها مقدار K را تغییر دهید) و به کمک تابعGridSearchCV مقادیر بهینهٔ پارامترها را (که دقت مدل را بیشینه می‌کند) بدست آورید..

د-۴) به کمک شاخص‌های معرفی شده در درس، عملکرد مدل‌های مختلف را با یکدیگر مقایسه کنید.

❖ حل مسئله

➢ معرفی کتابخانه‌های استفاده شده

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from imblearn.over_sampling import SMOTE
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, ConfusionMatrixDisplay
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

Pandas: برای کار با داده ها و تحلیل داده‌ها.

Seaborn: یک کتابخانه نمایش داده‌ها بر روی matplotlib.

Matplotlib: یک کتابخانه رسم نمودار اصلی برای ایجاد نمودارها و شکل‌ها.

StandardScaler (از sklearn.preprocessing): ویژگی‌ها را با حذف میانگین و مقیاس‌بندی به واریانس واحد (استانداردسازی) مقیاس‌بندی می‌کند.

train_test_split: مجموعه داده‌ها را به مجموعه‌های آموزشی و آزمایشی تقسیم می‌کند.

LogisticRegression: یک مدل خطی برای دسته‌بندی دوگانه یا چندگانه.

KneighborsClassifier: یک مدل که یک نمونه را بر اساس برچسب اکثریت نزدیکترین همسایه‌هایش دسته‌بندی می‌کند.

SVC: یک الگوریتم دسته‌بندی که بهترینhyperplane را برای جداسازی هدف‌ها پیدا می‌کند.

DecisionTreeClassifier: یک مدل مبتنی بر درخت که داده‌ها را بر اساس آستانه‌های ویژگی برای تصمیم‌گیری تقسیم می‌کند.

RandomForestClassifier: مجموعه‌ای از چندین درخت تصمیم‌گیری برای بهبود دقت و کنترل بیش‌برازش.

SMOTE: تکنیکی برای تولید نمونه‌های مصنوعی برای هر هدف.

classification_report: دقت، recall، F1-Score و Precision را برای هر مدل چاپ می‌کند.

accuracy_score: نسبت پیش‌بینی‌های صحیح.

confusion_matrix: ماتریسی که برچسب‌های درست را در مقابل برچسب‌های پیش‌بینی‌شده نشان می‌دهد.

ConfusionMatrixDisplay: یک ابزار کمکی برای نمایش مرتب ماتریس درهم‌ریختگی.

GridSearchCV: با استفاده از اعتبارسنجی متقابل، جستجوی کاملی را روی مقادیر hyperparameter مشخص‌شده انجام می‌دهد.

الف) بررسی داده‌های خام

الف-۱) به‌دست آوردن ساختار کلی داده‌ها با روش‌های info و describe

ابتدا مجموعه داده‌های ماشین فرز از یک فایل CSV در یک DataFrame از کتابخانه Pandas بارگذاری می‌شود.

```
df = pd.read_csv("milling_machine.csv")
```

سپس ساختار کلی داده‌ها با استفاده از روش‌های info و describe تعیین می‌شود.

```
print("Data Info:")
print(df.info())
print("\nData Description:")
print(df.describe())
```

df.info(): انواع ستون‌ها، تعداد مقادیر غیر تهی و میزان استفاده از حافظه را نمایش می‌دهد.

df.describe(): آمار توصیفی (میانگین، انحراف معیار، حداقل و ...) را برای ستون‌های عددی تولید می‌کند.

پاسخ این بخش به‌صورت زیر است:

```
Data Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 6 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Air Temp (°C)           9965 non-null   float64
 1   Process Temp (°C)       9990 non-null   float64
 2   Rotational Speed (RPM)  10000 non-null  float64
 3   Torque (Nm)             10000 non-null  float64
 4   Tool Wear (Seconds)     9993 non-null   float64
 5   Failure Types           9991 non-null   object
dtypes: float64(5), object(1)
memory usage: 468.9+ KB
None

Data Description:
       Air Temp (°C)  Process Temp (°C)  Rotational Speed (RPM)   Torque (Nm)  \
count    9965.000000        9990.000000            10000.000000  10000.000000
mean       28.516926          80.812186             1401.909988     46.998845
std         7.719340          15.548350              968.446183     26.747646
min        20.001366          60.001876                0.047731      0.015920
25%        23.176455          68.090324              423.672240     18.091381
50%        26.212082          76.553203             1377.047835     54.983239
75%        29.377536          92.825894             2307.969925     67.258375
max        49.998008         119.971025             2999.953724     89.993221

       Tool Wear (Seconds)
count          9993.000000
mean          11393.143344
std            9023.336380
min               3.469877
25%            5023.027818
50%            8995.172952
75%           15024.825673
max           35999.566519
```

الف-۲) تعداد و نسبت مقادیر ناموجود (Missing values) برای هر ویژگی

ابتدا با استفاده از دستورات missing_count و missing_proportion به ترتیب تعداد کل و نسبت مقادیر گمشده در هر ستون محاسبه می‌شود. سپس، اطلاعات مقدار از دست رفته در یک DataFrame جدید قرار داده شده و چاپ می‌شود.

```
missing_count = df.isnull().sum()
missing_proportion = df.isnull().mean()

missing_df = pd.DataFrame({
    "Missing Count": missing_count,
    "Missing Proportion": missing_proportion
})
print("\nMissing Values Info:")
print(missing_df)
```

پاسخ این بخش به‌صورت زیر است:

```
Missing Values Info:
                      Missing Count  Missing Proportion
Air Temp (°C)                    35              0.0035
Process Temp (°C)                10              0.0010
Rotational Speed (RPM)            0              0.0000
Torque (Nm)                       0              0.0000
Tool Wear (Seconds)               7              0.0007
Failure Types                     9              0.0009
```

الف-۳) بررسی correlation دو به دو ویژگی‌ها و تشکیل ماتریس همبستگی برای تعیین وضعیت ابزار فرز نسبت به هر ویژگی

برای انجام این بخش، ابتدا یک کپی از مجموعه داده ایجاد می‌شود و هدف به دودویی تبدیل می‌شود (۰ = سالم، ۱ = خرابی).

```
df_encoded = df.copy()
df_encoded['Failure Types'] = df_encoded['Failure Types'].apply(lambda x: 0 if x == 'No Failure' else 1)
```

سپس، ماتریس همبستگی محاسبه و چاپ می‌شود.

```
correlation_matrix = df_encoded.corr()
print("\nCorrelation Matrix:")
print(correlation_matrix)
```

در ادامه، سه ویژگی برتر که بیشترین همبستگی را با خرابی ابزار دارند (با همبستگی مطلق) پیدا می‌شود.

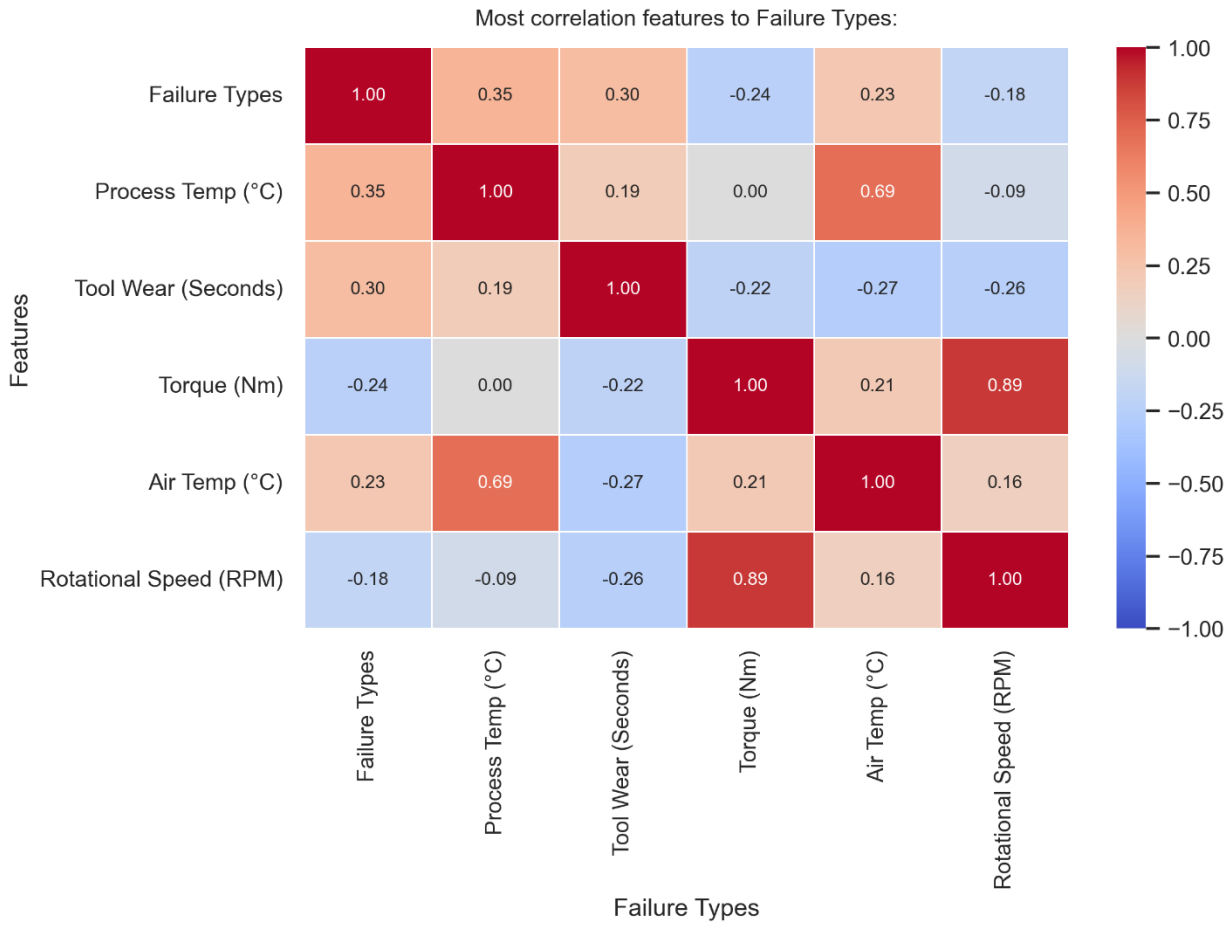
```
target_corr = correlation_matrix["Failure Types"].drop("Failure Types").abs().sort_values(ascending=False)
top_3_features = target_corr.head(3).index.tolist()
print("\nTop 3 Features Most Correlated with 'Failure Types':")
print(top_3_features)
```

و در آخر، برای نمایش بهتر میزان وابستگی وضعیت ابزار فرز به هر کدام از ویژگی‌ها، نمودار HeatMap مربوط به ماتریس همبستگی رسم می‌شود.

```
top_features = correlation_matrix['Failure Types'].abs().sort_values(ascending=False).index[0:39]
top_corr_with_Failure_Types = pd.DataFrame(correlation_matrix.loc[top_features, top_features])
plt.figure(figsize=(14, 10))
sns.set(font_scale=0.9)
ax = sns.heatmap(top_corr_with_Failure_Types,
                 annot=True,
                 cmap='coolwarm',
                 fmt=".2f",
                 cbar=True,
                 vmin=-1, vmax=1,
                 linewidths=0.4,
                 annot_kws={"size": 10})

plt.title("Most correlation features to Failure Types:", pad=10, fontsize=10)
plt.xlabel("Failure Types", labelpad=5)
plt.ylabel("Features", labelpad=5)
ax.set_yticklabels(ax.get_yticklabels(), rotation=0)
plt.tight_layout()
plt.show()
```

پاسخ این بخش به‌صورت زیر است:

```
Correlation Matrix:
                        Air Temp (°C)  Process Temp (°C)  \
Air Temp (°C)                1.000000           0.693634
Process Temp (°C)            0.693634           1.000000
Rotational Speed (RPM)       0.156958          -0.086616
Torque (Nm)                  0.213051           0.002808
Tool Wear (Seconds)         -0.266371           0.188924
Failure Types                0.229993           0.352946

                        Rotational Speed (RPM)  Torque (Nm)  \
Air Temp (°C)                         0.156958     0.213051
Process Temp (°C)                    -0.086616     0.002808
Rotational Speed (RPM)                1.000000     0.888487
Torque (Nm)                           0.888487     1.000000
Tool Wear (Seconds)                  -0.256103    -0.216528
Failure Types                        -0.183382    -0.241391

                        Tool Wear (Seconds)  Failure Types
Air Temp (°C)                     -0.266371       0.229993
Process Temp (°C)                  0.188924       0.352946
Rotational Speed (RPM)            -0.256103      -0.183382
Torque (Nm)                       -0.216528      -0.241391
Tool Wear (Seconds)                1.000000       0.304598
Failure Types                      0.304598       1.000000

Top 3 Features Most Correlated with 'Failure Types':
['Process Temp (°C)', 'Tool Wear (Seconds)', 'Torque (Nm)']
```

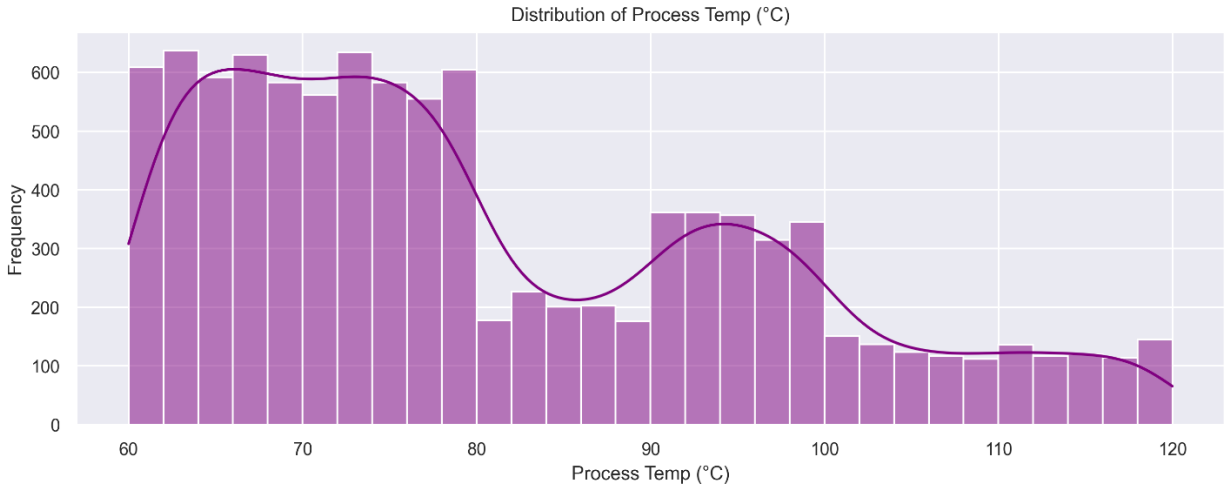


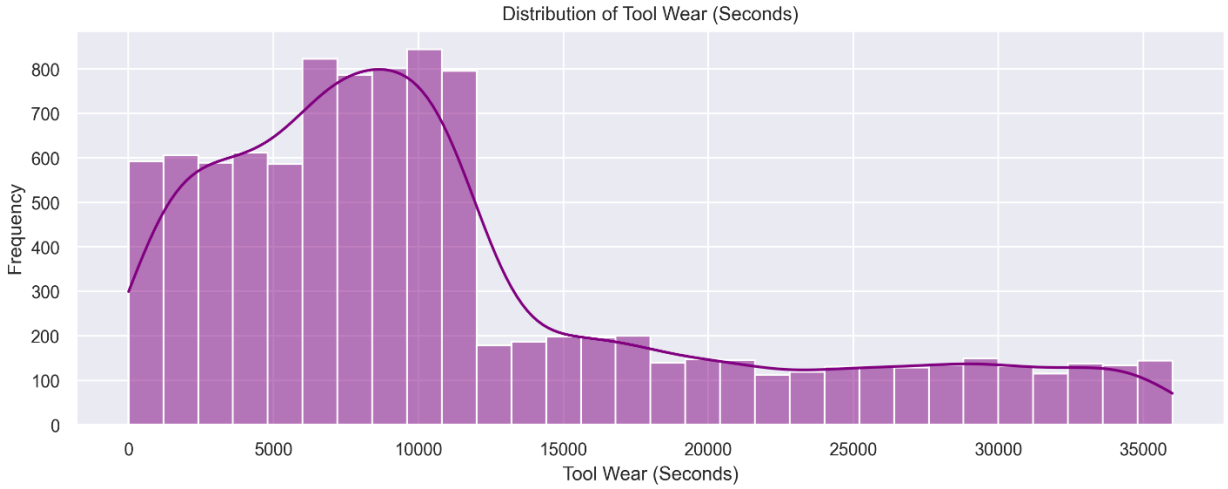
الف-۴) رسم نمودار تعداد مشاهدات برای سه ویژگی دارای بیشترین تأثیر بر خروجی

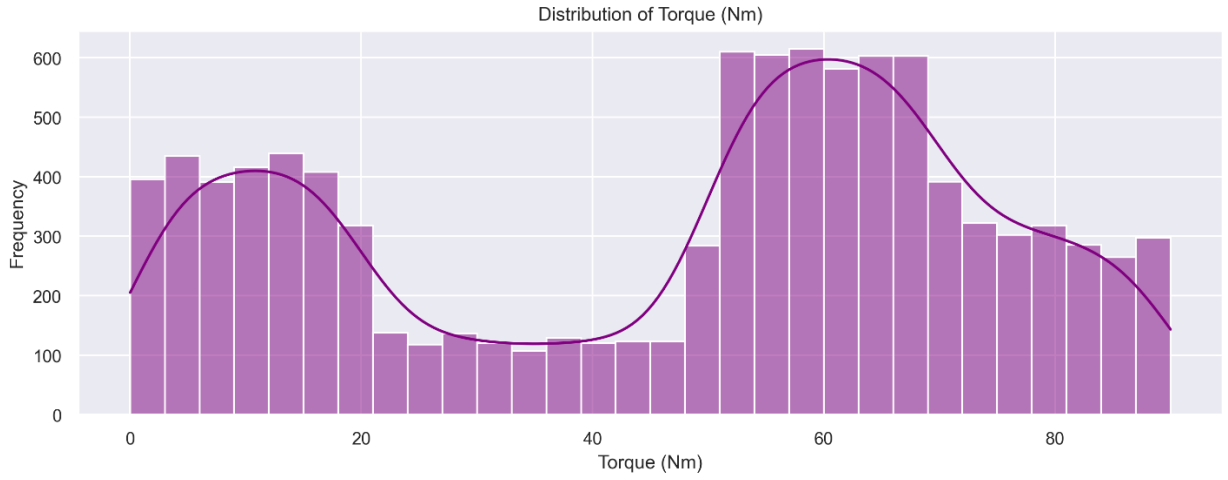
نمودارهای هیستوگرام برای ۳ ویژگی برتری که طبق بخش قبلی تعیین شد و بیشترین همبستگی را با خروجی دارند، رسم می‌شود.

```
for feature in top_3_features:
    plt.figure(figsize=(10, 4))
    sns.histplot(df[feature].dropna(), bins=30, kde=True, color='purple')
    plt.title(f'Distribution of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Frequency')
    plt.tight_layout()
    plt.show()
```

نمودار تعداد مشاهدات ۳ ویژگی برتر یعنی Process Temp، Tool Wear و Torque در زیر آورده شده است:


Distribution of Process Temp (°C)
Frequency
Process Temp (°C)
0
100
200
300
400
500
600
60
70
80
90
100
110
120


Distribution of Tool Wear (Seconds)
Frequency
Tool Wear (Seconds)
0
100
200
300
400
500
600
700
800
0
5000
10000
15000
20000
25000
30000
35000


Distribution of Torque (Nm)
Frequency
Torque (Nm)
0
100
200
300
400
500
600
0
20
40
60
80

ب) پیش‌پردازش داده‌ها

ب-۱) بررسی بیشترین میزان مقادیر ناموجود در داده‌ها و حل آن

برای ویژگی‌های عددی، از میانه استفاده می‌کنیم که توزیع‌های ناهموار (skewed distributions) را مدیریت می‌کند و در برابر داده‌های پرت مقاوم است. برای دسته‌بندی (Failure Types) از مد استفاده می‌کنیم. حالت دسته‌بندی تضمین می‌کند که هیچ دسته‌بندی جدیدی معرفی نمی‌شود و ثبات برچسب‌ها حفظ می‌شود.

```
df['Air Temp (°C)'] = df['Air Temp (°C)'].fillna(df['Air Temp (°C)'].median())
df['Process Temp (°C)'] = df['Process Temp (°C)'].fillna(df['Process Temp (°C)'].median())
df['Tool Wear (Seconds)'] = df['Tool Wear (Seconds)'].fillna(df['Tool Wear (Seconds)'].median())
df['Failure Types'] = df['Failure Types'].fillna(df['Failure Types'].mode()[0])
```

میزان و درصد مقادیر ناموجود هر یک از داده‌ها در جدول زیر آورده شده است:

ویژگی	تعداد مقادیر ناموجود	Proportion از دست رفته
Air Temp (°C)	35	0.0035
Process Temp (°C)	10	0.0010
Tool Wear (Seconds)	7	0.0007
Failure Types	9	0.0009
بقیه ویژگی ها	0	0.0000

ب-۲) توضیح فرآیندهای Standardizing و Normalizing و نحوه استفاده از آنها

استانداردسازی (Standardizing):

ویژگی‌ها را به گونه‌ای تبدیل می‌کند که میانگین = ۰ و انحراف معیار = ۱ داشته باشند. وقتی داده‌ها دارای داده‌های پرت یا فرضیات توزیع نرمال هستند، استفاده از آن بهتر است.

نرمال‌سازی (Normalizing):

ویژگی‌ها را در محدوده [۰،۱] مقیاس‌بندی می‌کند و به داده‌های پرت حساس است.

در این تمرین با توجه به اینکه از مدل‌های یادگیری ماشین نظیر Logistic Regression، K-Nearest-Neighbors و Support Vector Machine استفاده می‌شود، استفاده از این فرآیندها لازم است. در این تمرین، برای نرمال سازی داده‌ها از روش Min-Max Scaling استفاده می‌شود. زیرا این روش، داده‌ها را در محدوده [۰،۱] مقیاس‌بندی می کند و مقایسه مجموعه داده‌های مختلف را آسان‌تر می‌کند. علاوه‌بر این، بسیاری از مدل‌های یادگیری ماشین، زمانی که ویژگی‌ها به‌طور یکنواخت مقیاس‌بندی شوند، عملکرد بهتری دارند.

در کد این بخش، ابتدا نام ستون‌ها با حذف فاصله‌های اضافی پاک می‌شود. سپس، مقادیر ویژگی‌ها (میانگین صفر، واریانس واحد) برای عملکرد مدل، استاندارد می‌شود. برای اعمال روش Min-Max Scaling، در کد از عبارت StandardScalar استفاده شده است.

```
df.columns = df.columns.str.replace(r'\s+', ' ', regex=True).str.strip()
features = ['Air Temp (°C)', 'Process Temp (°C)', 'Rotational Speed (RPM)',
            'Torque (Nm)', 'Tool Wear (Seconds)']

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

ج) دسته‌بندی دوگانه

ج-۱) اضافه کردن یک ستون به ستون فایل دادگان و افزودن برچسب No Failure برای ابزارهای سالم و برچسب Failure برای ابزارهای آسیب دیده

در کد این بخش، یک متغیر هدف دوگانه جدید برای ابزارهای سالم در مقابل ابزارهای آسیب دیده ایجاد شده و سپس، برچسب No Failure برای ابزارهای سالم و برچسب Failure برای ابزارهای آسیب دیده استفاده می‌شود.

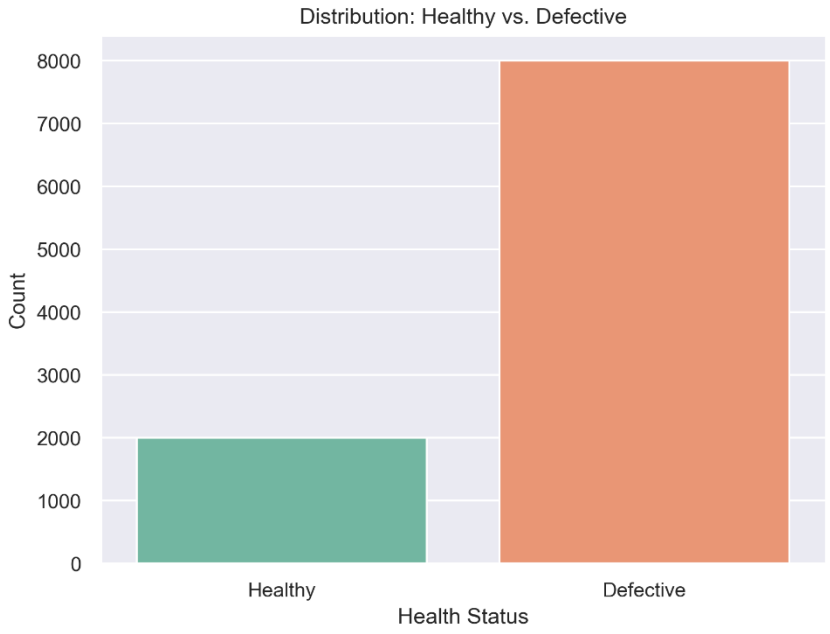
```
df['Health Status'] = df['Failure Types'].apply(lambda x: 'Healthy' if x == 'No Failure' else 'Defective')
```

ج-۲) رسم نمودار میله‌ای (Chart Bar) برای توزیع دسته‌های دوگانه جدید و نشان دادن عدم توازن داده‌ها

برای رسم نمودار میله‌ای مربوط به توزیع دسته‌های دوگانه (Healthy و Defective) که در بخش قبل مشخص شد، از کد زیر استفاده شده است:

```
sns.countplot(data=df, x='Health Status', hue='Health Status', palette='Set2', legend=False)
plt.title('Distribution: Healthy vs. Defective')
plt.xlabel('Health Status')
plt.ylabel('Count')
plt.show()
```

و پاسخ کد به‌صورت زیر است:



ج-۳) مشکلات ایجاد شده در مدل به‌دلیل عدم توازن داده‌ها

۱- گرایش مدل به سمت داده‌های غالب

مدل یادگیری ماشین تمایل دارد به سمت یادگیری الگوهای داده‌های غالب متمایل شود، چرا که نمونه‌های بیشتری از آن می‌بینند. در نتیجه، دقت کلی (Accuracy) ممکن است بالا به نظر برسد، اما دقت در پیش‌بینی داده‌های اقلیت بسیار پایین خواهد بود. مثلاً اگر ۹۵ درصد داده‌ها متعلق به دسته A باشند و فقط ۵ درصد به دسته B، مدلی که همیشه A را پیش‌بینی کند، ۹۵ درصد دقت دارد ولی عملاً بی‌فایده است.

۲- ناتوانی در شناسایی موارد بحرانی

در بسیاری از کاربردها، داده‌های اقلیت، اهمیت بیشتری دارد (مثلاً تشخیص بیماری، تشخیص چهره، شناسایی نقص فنی و ...) و اگر مدل آن را نادیده بگیرد، نتایج ممکن است در عمل بی‌استفاده یا حتی خطرناک باشند.

۳- نارسا بودن معیار دقت (Accuracy)

در داده‌های نامتوازن، معیارهایی مثل دقت (Accuracy) تصویر نادرستی از عملکرد مدل ارائه می‌دهند. باید از معیارهای دقیق‌تری مانند Recall، Precision و F1-Score استفاده کرد.

۴- اختلال در فرآیند یادگیری

الگوریتم‌های یادگیری ممکن است در به‌روزرسانی وزن‌ها یا پیدا کردن مرزهای تصمیم‌گیری صحیح برای داده‌های اقلیت دچار مشکل شوند، زیرا نمونه‌های کافی برای یادگیری الگوهای آن وجود ندارد و هم‌چنین، الگوریتم دچار bias می‌شود.

۵- عدم تعمیم‌پذیری (Poor Generalization)

مدل ممکن است فقط برای داده‌های غالب در بخش آموزشی عملکرد خوبی داشته باشد، اما در داده‌های واقعی که توزیع متفاوتی دارند، شکست بخورد.

ج-۴) حل مشکل عدم توازن داده‌ها با استفاده از روش SMOTE

در کد این بخش، با تولید نمونه‌های اقلیت مصنوعی، روش SMOTE برای متعادل کردن توزیع دسته‌های دوگانه اعمال می‌شود.

```
X = df[features]
y = df['Health Status']

smote = SMOTE(random_state=42)
X_bal, y_bal = smote.fit_resample(X, y)
```

ج-۵) تقسیم دادگان پالایش به دو بخش آموزش و آزمون و آموزش مدل‌های Logistic Regression، K-Nearest-Neighbors و Support Vector Machine (SVM) با هسته‌های خطی و غیرخطی با استفاده از کتابخانه scikit-learn

در کد این بخش، ابتدا دادگان پالایش به دو بخش آموزش (۸۰٪) و آزمون (۲۰٪) با Random-State=42 تقسیم شده و سپس، مدل‌های Logistic Regression (log-reg)، K-Nearest-Neighbors (knn)، SVM با هسته خطی (svm-linear) و SVM با هسته غیرخطی (svm-rbf) با استفاده از کتابخانه scikit-learn آموزش داده می‌شوند.

```
X_train, X_test, y_train, y_test = train_test_split(X_bal, y_bal, test_size=0.2, random_state=42)

log_reg = LogisticRegression()
knn = KNeighborsClassifier()
svm_linear = SVC(kernel='linear')
svm_rbf = SVC(kernel='rbf')

log_reg.fit(X_train, y_train)
knn.fit(X_train, y_train)
svm_linear.fit(X_train, y_train)
svm_rbf.fit(X_train, y_train)
```

ج-۶) تعیین دقت (Accuracy) و Classification Report هر مدل با تشکیل ماتریس آشفتگی

در کد این بخش، معیارهای عملکرد هر یک از چهار مدل آموزش داده شده شامل دقت (Accuracy)، Precision، Recall و F1-Score برای دو دسته Healthy و Defective با تشکیل ماتریس آشفتگی مورد بررسی قرار گرفته و با یکدیگر مقایسه می‌شود.

```python
models = {
    'Logistic Regression': log_reg,
    'KNN': knn,
    'SVM Linear': svm_linear,
    'SVM RBF': svm_rbf
}

results = []

for name, model in models.items():
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred, output_dict=True)
    results.append((name, acc, report))

comparison_table = []

for name, acc, report in results:
    comparison_table.append({
        'Model': name,
        'Accuracy': round(acc, 4),
        'Precision (Healthy)': round(report['Healthy']['precision'], 4),
        'Recall (Healthy)': round(report['Healthy']['recall'], 4),
        'F1-score (Healthy)': round(report['Healthy']['f1-score'], 4),
        'Precision (Defective)': round(report['Defective']['precision'], 4),
        'Recall (Defective)': round(report['Defective']['recall'], 4),
        'F1-score (Defective)': round(report['Defective']['f1-score'], 4),
        'Macro F1-score': round(report['macro avg']['f1-score'], 4)
    })
```

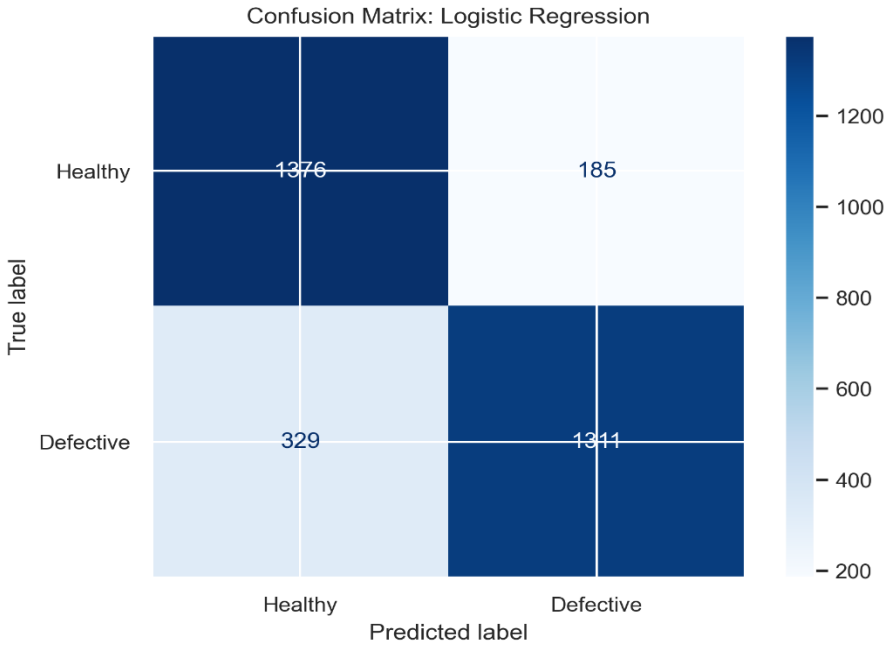
و پاسخ کد فوق به‌صورت زیر است:

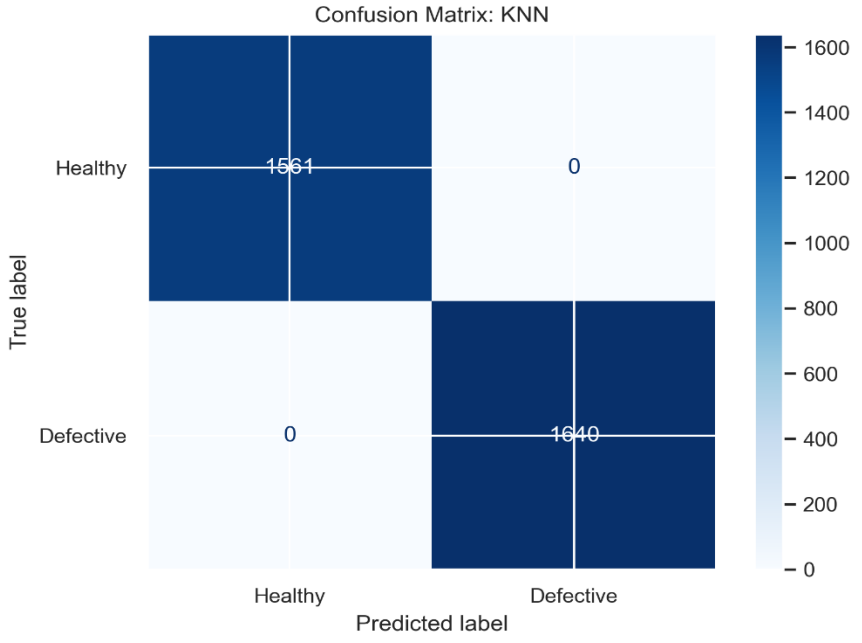
```
                Model  Accuracy  Precision (Healthy)  Recall (Healthy)  \
0  Logistic Regression    0.8394               0.8070            0.8815
1                  KNN    1.0000               1.0000            1.0000
2           SVM Linear    0.8497               0.8007            0.9212
3              SVM RBF    1.0000               1.0000            1.0000

   F1-score (Healthy)  Precision (Defective)  Recall (Defective)  \
0              0.8426                 0.8763              0.7994
1              1.0000                 1.0000              1.0000
2              0.8567                 0.9125              0.7817
3              1.0000                 1.0000              1.0000

   F1-score (Defective)  Macro F1-score
0                0.8361          0.8394
1                1.0000          1.0000
2                0.8420          0.8494
3                1.0000          1.0000
```

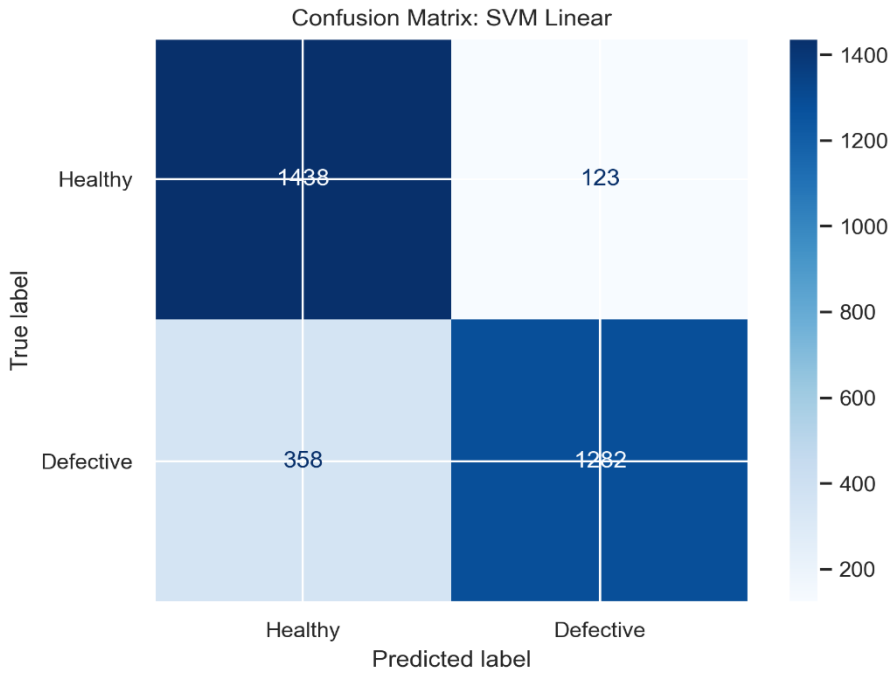
برای رسم ماتریس آشفتگی هر مدل نیز از کد زیر استفاده شده است:

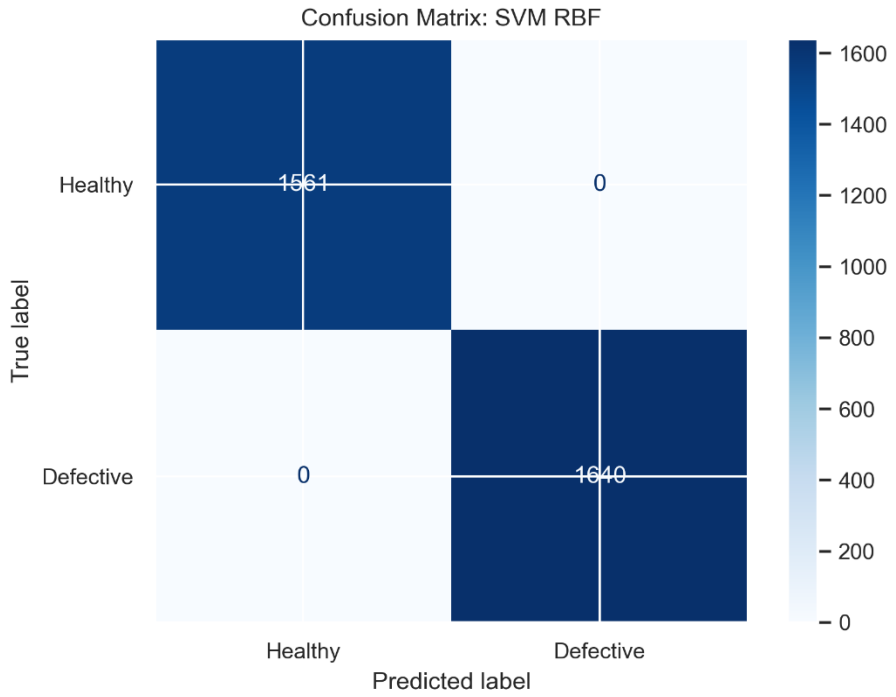
```python
for name, model in models.items():
    y_pred = model.predict(X_test)
    cm = confusion_matrix(y_test, y_pred, labels=['Healthy', 'Defective'])
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Healthy', 'Defective'])
    disp.plot(cmap='Blues')
    plt.title(f'Confusion Matrix: {name}')
    plt.show()
```

ماتریس‌های آشفتگی حاصل از چهار مدل در ادامه ارائه شده اند:






Confusion Matrix: SVM Linear
Healthy
Defective
True label
1438
123
358
1282
Healthy
Defective
Predicted label
1400
1200
1000
800
600
400
200


Confusion Matrix: SVM RBF
Healthy
Defective
True label
1561
0
0
1640
Healthy
Defective
Predicted label
1600
1400
1200
1000
800
600
400
200
0

هم‌چنین، خلاصه نتایج عملکرد هر مدل در جدول زیر ارائه شده است:

F1-score (Defective)	Recall (Defective)	Precision (Defective)	F1-score (Healthy)	Recall (Healthy)	Precision (Healthy)	Accuracy	Model
0.8361	0.7994	0.8763	0.8426	0.8815	0.8070	0.8394	**Logistic Regression**
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	**KNN**
0.8420	0.7817	0.9125	0.8567	0.9212	0.8007	0.8497	**SVM Linear**
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	**SVM RBF**

- KNN و SVM با هسته RBF در مجموعه تست به عملکرد کاملی دست یافتند که نشان‌دهنده مدل‌های بسیار مناسب است.
- Logistic Regression و SVM خطی با دقت حدود ۸۴ تا ۸۵ درصد، عملکرد نسبتاً خوبی دارند.
- امتیاز Macro F1 (میانگین F1 برای هر دو کلاس) تعادل بین دقت و recall را تأیید می‌کند.
- Logistic Regression برای ابزارهای آسیب دیده دقت کمی بهتری نسبت به SVM خطی دارد، در حالی که SVM خطی برای ابزارهای سالم recall بالاتری دارد.

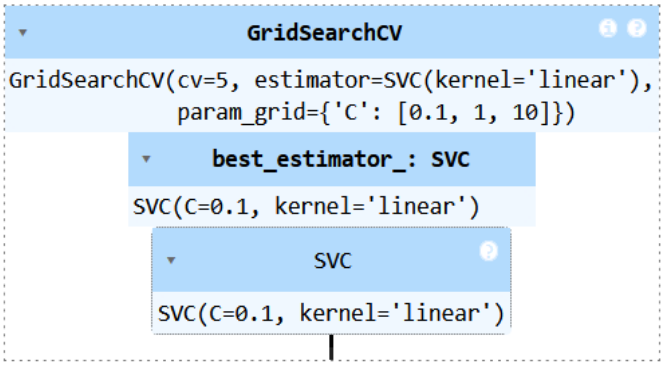
ج-۷) تغییر دو پارامتر از میان هایپرپارامترها برای هر مدل و تعیین مقادیر بهینه پارامترها به کمک تابع GridSearchCV

در کد این بخش، مدل‌ها روی x-train و y-train آموزش می‌بینند و از GridSearchCV برای پیدا کردن بهترین پارامترها و هم‌چنین از ۵ کراس ولیدیشن (Cross-Validation) برای ارزیابی عملکرد هر مدل استفاده شده است که در کد به‌صورت cv نمایش داده شده است. علاوه‌بر این، برای ارزیابی بهتر هر مدل، برای پارامتر تنظیم (C)، سه مقدار مختلف ۰.۱، ۱ و ۱۰ در نظر گرفته شده است.

در مدل KNN، تعداد همسایه‌ها برای ارزیابی مدل در محدوده ۳ تا ۱۱ تنظیم شده است و برای ارزیابی دقیق‌تر مدل SVM با هسته غیرخطی، تأثیر فاصله داده‌ها در تصمیم مدل (gamma) نیز در نظر گرفته شده است.

```
param_log = {'C': [0.1, 1, 10], 'penalty': ['l2']}
grid_log = GridSearchCV(LogisticRegression(), param_log, cv=5)
grid_log.fit(X_train, y_train)

param_knn = {'n_neighbors': list(range(3, 11))}
grid_knn = GridSearchCV(KNeighborsClassifier(), param_knn, cv=5)
grid_knn.fit(X_train, y_train)

param_svm = {'C': [0.1, 1, 10], 'gamma': ['scale', 0.1, 1]}
grid_svm = GridSearchCV(SVC(kernel='rbf'), param_svm, cv=5)
grid_svm.fit(X_train, y_train)

param_svm_linear = {'C': [0.1, 1, 10]}
grid_svm_linear = GridSearchCV(SVC(kernel='linear'), param_svm_linear, cv=5)
grid_svm_linear.fit(X_train, y_train)
```

که پاسخ کد فوق به این صورت نمایش داده می‌شود:



در ادامه، بهترین هایپرپارامترهای پیدا شده توسط GridSearchCV برای هر مدل با استفاده از کد زیر به‌دست می‌آید:

```
print("Best Hyperparameters Found by GridSearchCV:")
print("Logistic Regression:", grid_log.best_params_)
print("KNN:", grid_knn.best_params_)
print("SVM RBF:", grid_svm.best_params_)
print("SVM Linear:", grid_svm_linear.best_params_)
```

پاسخ کد فوق در زیر ارائه شده است:

```
Best Hyperparameters Found by GridSearchCV:
Logistic Regression: {'C': 10, 'penalty': 'l2'}
KNN: {'n_neighbors': 6}
SVM RBF: {'C': 10, 'gamma': 'scale'}
SVM Linear: {'C': 0.1}
```

ج-۸) مقایسه عملکرد چهار مدل استفاده شده

در کد این بخش، از بهترین تخمین‌گرهای یافت‌شده توسط GridSearchCV استفاده شده و مقادیر بهینه هر مدل شامل Accuracy، Precision، Recall و F1-Score برای دو دسته‌ی ابزارهای سالم و آسیب دیده تعیین می‌شود.

```
optimized_models = {
    'Logistic Regression (Tuned)': grid_log.best_estimator_,
    'KNN (Tuned)': grid_knn.best_estimator_,
    'SVM RBF (Tuned)': grid_svm.best_estimator_,
    'SVM Linear (Tuned)': grid_svm_linear.best_estimator_
}

optimized_results = []

for name, model in optimized_models.items():
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred, output_dict=True)
    optimized_results.append((name, acc, report))

tuned_comparison = []

for name, acc, report in optimized_results:
    tuned_comparison.append({
        'Model': name,
        'Accuracy': round(acc, 4),
        'Precision (Healthy)': round(report['Healthy']['precision'], 4),
        'Recall (Healthy)': round(report['Healthy']['recall'], 4),
        'F1-score (Healthy)': round(report['Healthy']['f1-score'], 4),
        'Precision (Defective)': round(report['Defective']['precision'], 4),
        'Recall (Defective)': round(report['Defective']['recall'], 4),
        'F1-score (Defective)': round(report['Defective']['f1-score'], 4),
        'Macro F1-score': round(report['macro avg']['f1-score'], 4)
    })
tuned_df = pd.DataFrame(tuned_comparison)
print(tuned_df)
```

که پاسخ کد فوق در ادامه ارائه شده است:

```
                      Model  Accuracy  Precision (Healthy)  \
0  Logistic Regression (Tuned)    0.8391               0.8069
1                KNN (Tuned)    1.0000               1.0000
2            SVM RBF (Tuned)    1.0000               1.0000
3         SVM Linear (Tuned)    0.8485               0.7976

   Recall (Healthy)  F1-score (Healthy)  Precision (Defective)  \
0            0.8808              0.8423                 0.8758
1            1.0000              1.0000                 1.0000
2            1.0000              1.0000                 1.0000
3            0.9238              0.8560                 0.9146

   Recall (Defective)  F1-score (Defective)  Macro F1-score
0              0.7994                0.8358          0.8390
1              1.0000                1.0000          1.0000
2              1.0000                1.0000          1.0000
3              0.7768                0.8401          0.8481
```

به‌منظور مقایسه بهتر مدل‌ها، خلاصه نتایج حاصل از چهار مدل در جدول زیر آورده شده است:

F1-score (Defective)	Recall (Defective)	Precision (Defective)	F1-score (Healthy)	Recall (Healthy)	Precision (Healthy)	Accuracy	Model
0.8358	0.7994	0.8758	0.8423	0.8808	0.8069	0.8391	**Logistic Regression**
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	**KNN**
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	**SVM RBF**
0.8401	0.7768	0.9146	0.8560	0.9238	0.7976	0.8485	**SVM Linear**

- KNN و SVM RBF حتی پس از تنظیم، امتیازهای خود را حفظ کردند که عملکرد قوی آنها را در این مجموعه داده تأیید می‌کند.
- Logistic Regression در حالت تنظیم شده در مقایسه با حالت قبل از تنظیم، عملکرد تقریباً یکسانی را نشان داد که نشان می‌دهد مدل با پارامترهای پیش‌فرض، از قبل نزدیک به بهینه بوده است.
- SVM Linear در حالت تنظیم‌شده نسبت به حالت قبل از تنظیم، به ویژه در دقت (Accuracy) و F1-Score برای دسته‌ی ابزارهای آسیب دیده، بهبودهای جزئی را نشان داده است.
- Macro F1-Score که نشان‌دهنده‌ی عملکرد متعادل در بین داده‌ها است، برای مدل‌های KNN و SVM RBF بالاترین مقدار (۱.۰) و برای مدل Logistic Regression پایین‌ترین مقدار (۰.۸۳۹۰) را دارد.

د) دسته‌بندی چندگانه

د-۱) تقسیم دادگان پالایش به دو بخش آموزش و آزمون و آموزش مدل‌های K-Nearest-Neighbors، Decision Tree، Random Forest و Support Vector Machine (SVM) با روش‌های "یکی در برابر یکی" یا "یکی در برابر همه" با استفاده از کتابخانه scikit-learn

در کد این بخش، ابتدا دادگان پالایش به دو بخش آموزش (۸۰٪) و آزمون (۲۰٪) با Random-State=42 تقسیم شده و سپس، مدل‌های K-Nearest-Neighbors (knn)، Decision Tree، Random Forest و SVM با روش "یکی در برابر همه" با استفاده از کتابخانه scikit-learn آموزش داده می‌شوند.

```
X = df[features]
y = df['Failure Types']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

```
models = {
    'KNN': KNeighborsClassifier(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
    'SVM (OvR)': SVC(decision_function_shape='ovr')
}

results = []
```

د-۲) تعیین دقت (Accuracy) و Classification Report هر مدل با تشکیل ماتریس آشفتگی

در کد این بخش، معیارهای عملکرد هر یک از چهار مدل آموزش داده شده شامل دقت (Accuracy)، Precision، Recall و F1-Score برای دو دسته Healthy و Defective با تشکیل ماتریس آشفتگی مورد بررسی قرار گرفته و با یکدیگر مقایسه می‌شود.

```python
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred, output_dict=True)
    results.append((name, acc, report))

summary = []

for name, acc, report in results:
    summary.append({
        'Model': name,
        'Accuracy': round(acc, 4),
        'Macro F1-score': round(report['macro avg']['f1-score'], 4),
        'Weighted F1-score': round(report['weighted avg']['f1-score'], 4)
    })

import pandas as pd
results_df = pd.DataFrame(summary)
print(results_df)
```

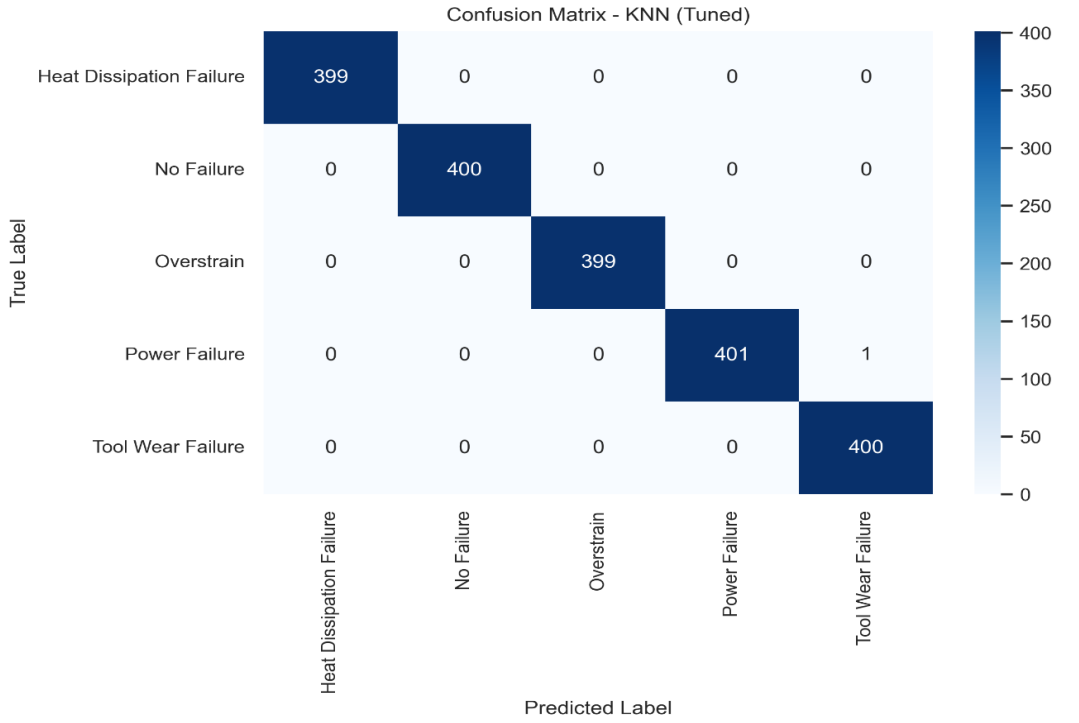
و پاسخ کد فوق به‌صورت زیر است:

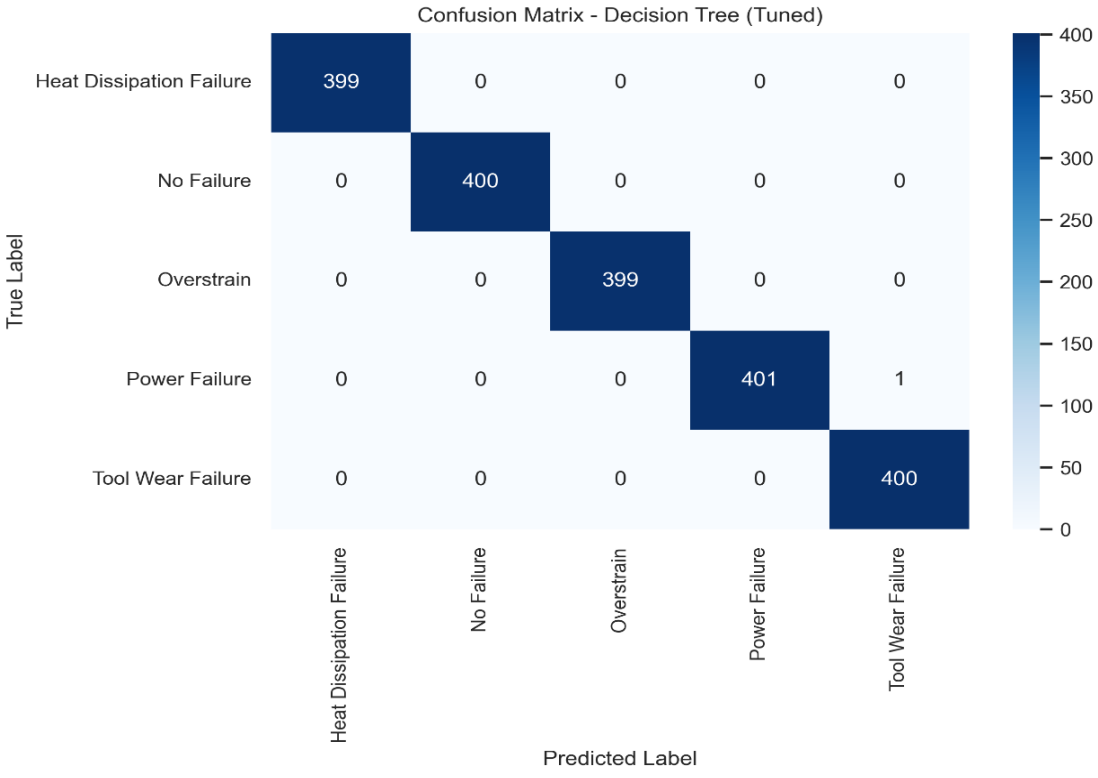
```
           Model  Accuracy  Macro F1-score  Weighted F1-score
0            KNN    0.9995          0.9995             0.9995
1  Decision Tree    0.9980          0.9980             0.9980
2  Random Forest    0.9995          0.9995             0.9995
3      SVM (OvR)    0.9995          0.9995             0.9995
```

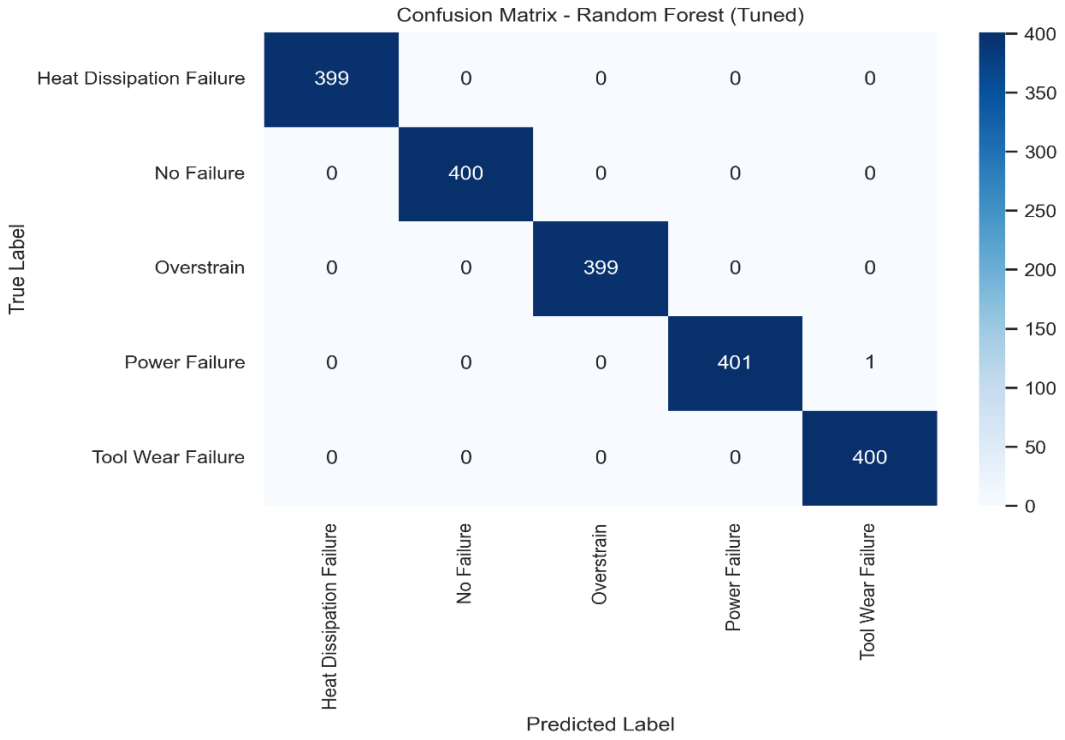
برای رسم ماتریس آشفتگی هر مدل نیز از کد زیر استفاده شده است:

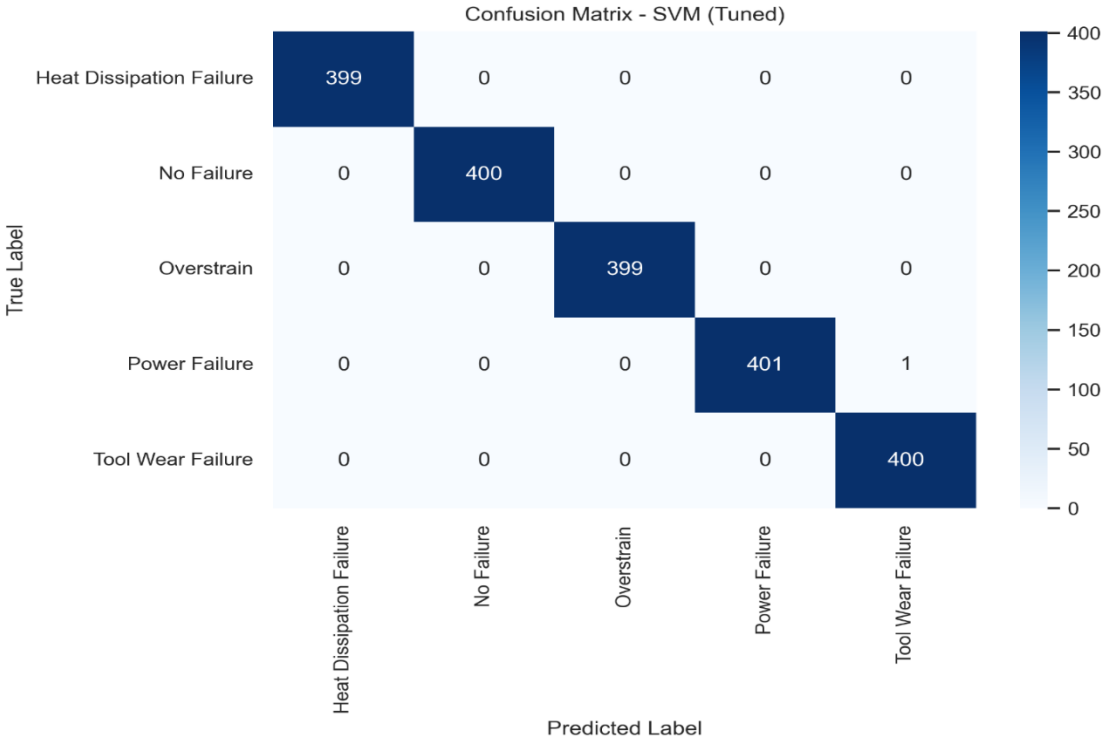
```python
labels = sorted(y.unique())

for name, model in models.items():
    y_pred = model.predict(X_test)
    cm = confusion_matrix(y_test, y_pred, labels=labels)

    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=labels, yticklabels=labels)
    plt.title(f'Confusion Matrix - {name}')
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')
    plt.tight_layout()
    plt.show()
```

ماتریس‌های آشفتگی حاصل از چهار مدل در ادامه ارائه شده اند:


Confusion Matrix - KNN (Tuned)
Heat Dissipation Failure
No Failure
Overstrain
Power Failure
Tool Wear Failure
True Label
399 0 0 0 0
0 400 0 0 0
0 0 399 0 0
0 0 0 401 1
0 0 0 0 400
Heat Dissipation Failure
No Failure
Overstrain
Power Failure
Tool Wear Failure
Predicted Label
400
350
300
250
200
150
100
50
0


Confusion Matrix - Decision Tree (Tuned)
Heat Dissipation Failure
No Failure
Overstrain
Power Failure
Tool Wear Failure
True Label
399 0 0 0 0
0 400 0 0 0
0 0 399 0 0
0 0 0 401 1
0 0 0 0 400
Heat Dissipation Failure
No Failure
Overstrain
Power Failure
Tool Wear Failure
Predicted Label
400
350
300
250
200
150
100
50
0


Confusion Matrix - Random Forest (Tuned)
Heat Dissipation Failure
No Failure
Overstrain
Power Failure
Tool Wear Failure
True Label
399 0 0 0 0
0 400 0 0 0
0 0 399 0 0
0 0 0 401 1
0 0 0 0 400
Heat Dissipation Failure
No Failure
Overstrain
Power Failure
Tool Wear Failure
Predicted Label
400
350
300
250
200
150
100
50
0


Confusion Matrix - SVM (Tuned)
Heat Dissipation Failure
No Failure
Overstrain
Power Failure
Tool Wear Failure
True Label
399 0 0 0 0
0 400 0 0 0
0 0 399 0 0
0 0 0 401 1
0 0 0 0 400
Heat Dissipation Failure
No Failure
Overstrain
Power Failure
Tool Wear Failure
Predicted Label
400
350
300
250
200
150
100
50
0

هم‌چنین، خلاصه نتایج عملکرد هر مدل در جدول زیر ارائه شده است:

Model	Accuracy	Macro F1-score	Weighted F1-score
KNN	0.9995	0.9995	0.9995
Decision Tree	0.9980	0.9980	0.9980
Random Forest	0.9995	0.9995	0.9995
SVM (OvR)	0.9995	0.9995	0.9995

- Accuracy: درصد برچسب‌های درست پیش‌بینی‌شده را اندازه‌گیری می‌کند. همه مدل‌ها به دقت بسیار بالایی (≥ 99.8%) دست یافتند که نشان‌دهنده عملکرد عالی است.
- Macro F1-score: میانگین امتیاز F1 صرف نظر از اندازه دسته‌ها، در تمام داده‌ها به طور مساوی است و مقادیر بالا (۰.۹۹۹۵) را نشان می‌دهد.
- Weighted F1-score: عدم تعادل داده‌ها را با وزن‌دهی به F1 بر اساس پشتیبانی دسته‌ها در نظر می‌گیرد. مقادیر مشابه در اینجا نشان می‌دهد که مدل‌ها نه تنها داده‌های اکثریت را به خوبی پیش‌بینی کرده‌اند، بلکه داده‌های اقلیت را نیز به طور مؤثر مدیریت کرده‌اند.
- KNN، Random Forest و SVM (OvR) همگی به دسته‌بندی تقریباً کامل دست یافتند.

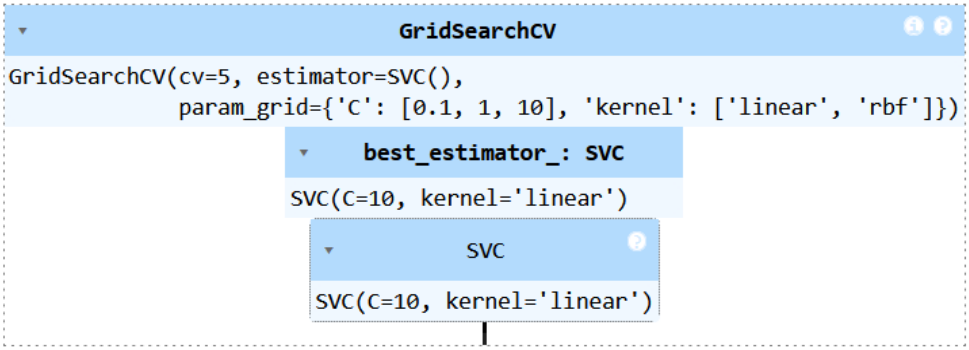
د-۳) تغییر دو پارامتر از میان هایپرپارامترها برای هر مدل و تعیین مقادیر بهینه پارامترها به کمک تابع GridSearchCV

در کد این بخش، مدل‌ها روی x-train و y-train آموزش می‌بینند و از GridSearchCV برای پیدا کردن بهترین پارامترها و هم‌چنین از ۵ کراس ولیدیشن (Cross-Validation) برای ارزیابی عملکرد هر مدل استفاده شده است که در کد به‌صورت cv نمایش داده شده است. علاوه‌بر این، برای ارزیابی بهتر هر مدل، برای پارامتر تنظیم (C)، سه مقدار مختلف ۰.۱، ۱ و ۱۰ در نظر گرفته شده است.

لازم به ذکر است، در مدل KNN، تعداد همسایه‌ها برای ارزیابی مدل در محدوده ۳ تا ۱۱ تنظیم شده است. برای ارزیابی دقیق‌تر مدل Decision Tree سه مقدار مختلف برای عمق درخت (max-depth) و دو معیار مختلف برای سنجش تقسیم داده‌ها (یعنی gini و entropy) در نظر گرفته شده است. بعلاوه، در مدل Random Forest، سومین مقدار عمق درخت None نوشته شده که به معنی این است که عمق درخت محدود نیست.

```
param_knn = {'n_neighbors': list(range(3, 11))}
grid_knn = GridSearchCV(KNeighborsClassifier(), param_knn, cv=5)
grid_knn.fit(X_train, y_train)

param_dt = {'max_depth': [5, 10, 15], 'criterion': ['gini', 'entropy']}
grid_dt = GridSearchCV(DecisionTreeClassifier(), param_dt, cv=5)
grid_dt.fit(X_train, y_train)

param_rf = {'n_estimators': [50, 100], 'max_depth': [5, 10, None]}
grid_rf = GridSearchCV(RandomForestClassifier(), param_rf, cv=5)
grid_rf.fit(X_train, y_train)

param_svm = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']}
grid_svm = GridSearchCV(SVC(decision_function_shape='ovr'), param_svm, cv=5)
grid_svm.fit(X_train, y_train)
```

که پاسخ کد فوق به این صورت نمایش داده می‌شود:



در ادامه، بهترین هایپرپارامترهای پیدا شده توسط GridSearchCV برای هر مدل با استفاده از کد زیر به‌دست می‌آید:

```
print("Best Hyperparameters Found by GridSearchCV:")
print("KNN:", grid_knn.best_params_)
print("Decision Tree:", grid_dt.best_params_)
print("Random Forest:", grid_rf.best_params_)
print("SVM ovr:", grid_svm.best_params_)
```

پاسخ کد فوق در زیر ارائه شده است:

```
Best Hyperparameters Found by GridSearchCV:
KNN: {'n_neighbors': 4}
Decision Tree: {'criterion': 'gini', 'max_depth': 5}
Random Forest: {'max_depth': 5, 'n_estimators': 50}
SVM ovr: {'C': 10, 'kernel': 'linear'}
```

د-۴) مقایسه عملکرد چهار مدل استفاده شده

در کد این بخش، از بهترین تخمین‌گرهای یافت‌شده توسط GridSearchCV استفاده شده و مقادیر بهینه هر مدل شامل Accuracy، Precision، Recall و F1-Score برای دو دسته‌ی ابزارهای سالم و آسیب دیده تعیین می‌شود.

```python
tuned_models = {
    'KNN (Tuned)': grid_knn.best_estimator_,
    'Decision Tree (Tuned)': grid_dt.best_estimator_,
    'Random Forest (Tuned)': grid_rf.best_estimator_,
    'SVM (Tuned)': grid_svm.best_estimator_
}

tuned_results = []

for name, model in tuned_models.items():
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred, output_dict=True)
    tuned_results.append((name, acc, report))

final_summary = []

for name, acc, report in tuned_results:
    final_summary.append({
        'Model': name,
        'Accuracy': round(acc, 4),
        'Macro F1-score': round(report['macro avg']['f1-score'], 4),
        'Weighted F1-score': round(report['weighted avg']['f1-score'], 4)
    })

tuned_df = pd.DataFrame(final_summary)
print(tuned_df)
```

که پاسخ کد فوق در ادامه ارائه شده است:

	Model	Accuracy	Macro F1-score	Weighted F1-score
0	KNN (Tuned)	0.9995	0.9995	0.9995
1	Decision Tree (Tuned)	0.9995	0.9995	0.9995
2	Random Forest (Tuned)	0.9995	0.9995	0.9995
3	SVM (Tuned)	0.9995	0.9995	0.9995

به‌منظور مقایسه بهتر مدل‌ها، خلاصه نتایج حاصل از چهار مدل در جدول زیر آورده شده است:

Weighted F1-score	Macro F1-score	Accuracy	Model
0.9995	0.9995	0.9995	**KNN (Tuned)**
0.9995	0.9995	0.9995	**Decision Tree (Tuned)**
0.9995	0.9995	0.9995	**Random Forest (Tuned)**
0.9995	0.9995	0.9995	**SVM (Tuned)**

- Accuracy: هر چهار مدل به ۹۹.۹۵٪ رسیدند، به این معنی که تقریباً همه پیش‌بینی‌ها در مجموعه آزمون درست هستند.
- Macro F1-score: میانگین امتیازات F1 در تمام کلاس‌ها، با هر کلاس به طور مساوی رفتار می‌کند. امتیاز بالا نشان می‌دهد که مدل‌ها همه کلاس‌ها (انواع خرابی) را به طور یکسان، از جمله کلاس‌های اقلیت، به خوبی مدیریت کرده‌اند.
- Weighted F1-score: ۰.۹۹۹۵ در کل، عملکرد عالی را در عین در نظر گرفتن عدم تعادل کلاس‌ها تأیید می‌کند.
- تنظیمات، امتیازات را بهبود بخشیده است، به خصوص برای درخت تصمیم، که در مقایسه با نتیجه تنظیم نشده آن (از ۰.۹۹۸۰ به ۰.۹۹۹۵) کمی بهبود یافته است.

❖ گیت هاب

گزارش و کد مربوط به این تمرین در گیت هاب به آدرس

https://github.com/MM-Touiserkani/AI_HW3_Touiserkani

قرار داده شده است.