In [ ]:
```
import os
```

In [ ]:
```
import pandas as pd
```

In [ ]:
```
import numpy as np
```

In [ ]:
```
import seaborn as sns
```

In [ ]:
```
import logging
```

In [ ]:
```
from sklearn.preprocessing import StandardScaler
```

In [ ]:
```
import matplotlib.pyplot as plt
```

In [ ]:
```
logging.basicConfig(level=logging.INFO, format="%(asctime)s - %(levelname)s - %(
```

In [ ]:
```
logging.info(" 🔹 Loading Cylinder Bottom Saw sensor data...")
```

```
2025-02-21 17:07:01,796 - INFO - 🔹 Loading Cylinder Bottom Saw sensor data...
```

In [ ]:
```
saw_csv_path = r"C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenance-a
```

In [ ]:
```
saw_df = pd.read_csv(saw_csv_path)
```

In [ ]:
```
logging.info("✅ Successfully loaded Cylinder Bottom Saw data.")
```

```
2025-02-21 17:07:17,111 - INFO - ✅ Successfully loaded Cylinder Bottom Saw dat
a.
```

In [ ]:
```
logging.info(f"✅ Dataset Shape: {saw_df.shape}")
```

```
2025-02-21 17:07:19,612 - INFO - ✅ Dataset Shape: (332847, 46)
```

In [ ]:
```
print("\n🔍 First 5 Rows of Cylinder Bottom Saw Data:")
```

```
🔍 First 5 Rows of Cylinder Bottom Saw Data:
```

In [ ]:
```
print(saw_df.head())
```

```
                   timestamp  CPU_cooler_temp  CPU_temp  CutCounter  \
0  2022-08-16 05:59:47.003000            636.0     665.0      5527.0
1  2022-08-16 05:59:47.503000            636.0     665.0      5527.0
2  2022-08-16 05:59:48.003000            636.0     665.0      5527.0
3  2022-08-16 05:59:48.503000            636.0     665.0      5527.0
4  2022-08-16 05:59:49.005000            636.0     665.0      5527.0

       CutTime  FFT_requirement  FlatstreamCutCounter  FlatstreamDone  \
0  341818.9375              0.0                5526.0             1.0
1  341818.9375              0.0                5526.0             1.0
2  341818.9375              0.0                5526.0             1.0
3  341818.9375              0.0                5526.0             1.0
4  341818.9375              0.0                5526.0             1.0

   FsMode_1Raw_2FftRaw_3FttHK  lift_active  ...  Vib03.Peak  Vib03.RMS  \
0                         1.0          0.0  ...   38.269043        8.0
1                         1.0          0.0  ...   38.162231        8.0
2                         1.0          0.0  ...   33.264160        8.0
3                         1.0          0.0  ...   30.715942        8.0
4                         1.0          0.0  ...   30.685425        8.0

   Vib03.Skewness  Vib03.VDI3832  teeth_per_blade  bCutActive  \
0        0.048391            0.0            516.0         0.0
1        0.055119            0.0            516.0         0.0
2        0.090212            0.0            516.0         0.0
3        0.103476            0.0            516.0         0.0
4        0.109211            0.0            516.0         0.0

   f_light_barrier  top_material_edge    v_feed  anomaly
0             63.0               30.0  0.314037        0
1             62.0               30.0  0.314037        0
2             62.0               30.0  0.525528        0
3             63.0               30.0  0.525528        0
4             63.0               30.0  0.525528        0

[5 rows x 46 columns]
```

In [ ]:
```
saw_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 332847 entries, 0 to 332846
Data columns (total 46 columns):
 #   Column                       Non-Null Count   Dtype  
---  ------                       --------------   -----  
 0   timestamp                    332847 non-null  object 
 1   CPU_cooler_temp              332847 non-null  float64
 2   CPU_temp                     332847 non-null  float64
 3   CutCounter                   332847 non-null  float64
 4   CutTime                      332847 non-null  float64
 5   FFT_requirement              332847 non-null  float64
 6   FlatstreamCutCounter         332847 non-null  float64
 7   FlatstreamDone               332847 non-null  float64
 8   FsMode_1Raw_2FftRaw_3FttHK   332847 non-null  float64
 9   lift_active                  332847 non-null  float64
 10  motor_on                     332847 non-null  float64
 11  PData.CosPhi                 332847 non-null  float64
 12  PData.CutEnergy              332847 non-null  float64
 13  PData.PEff                   332847 non-null  float64
 14  P_feed                       332847 non-null  float64
 15  Position                     332847 non-null  float64
 16  blade_position               332847 non-null  float64
 17  TData.T1                     332847 non-null  float64
 18  TData.T2                     332847 non-null  float64
 19  TData.T3                     332847 non-null  float64
 20  TData.T4                     332847 non-null  float64
 21  TData.T_IR                   332847 non-null  float64
 22  Vib01.CREST                  332847 non-null  float64
 23  Vib01.Kurtosis               332847 non-null  float64
 24  Vib01.Peak                   332847 non-null  float64
 25  Vib01.RMS                    332847 non-null  float64
 26  Vib01.Skewness               332847 non-null  float64
 27  Vib01.VDI3832                332847 non-null  float64
 28  Vib02.CREST                  332847 non-null  float64
 29  Vib02.Kurtosis               332847 non-null  float64
 30  Vib02.Peak                   332847 non-null  float64
 31  Vib02.RMS                    332847 non-null  float64
 32  Vib02.Skewness               332847 non-null  float64
 33  Vib02.VDI3832                332847 non-null  float64
 34  Vib03.CREST                  332847 non-null  float64
 35  Vib03.Kurtosis               332847 non-null  float64
 36  Vib03.Peak                   332847 non-null  float64
 37  Vib03.RMS                    332847 non-null  float64
 38  Vib03.Skewness               332847 non-null  float64
 39  Vib03.VDI3832                332847 non-null  float64
 40  teeth_per_blade              332847 non-null  float64
 41  bCutActive                   332847 non-null  float64
 42  f_light_barrier              332847 non-null  float64
 43  top_material_edge            332847 non-null  float64
 44  v_feed                       332847 non-null  float64
 45  anomaly                      332847 non-null  int64  
dtypes: float64(44), int64(1), object(1)
memory usage: 116.8+ MB
```

In [ ]:
```
saw_df.shape
```

Out[ ]:
```
(332847, 46)
```

In [ ]:
```
logging.info(" ◆  Loading CNC Lathe (Piston Rod) sensor data...")
```

```
2025-02-21 17:07:32,332 - INFO - 🔹 Loading CNC Lathe (Piston Rod) sensor dat
a...
```

In [ ]:
```
lathe_csv_path = r"C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenance
```

In [ ]:
```
lathe_df = pd.read_csv(lathe_csv_path)
```

In [ ]:
```
logging.info("✅ Successfully loaded CNC Lathe (Piston Rod) data.")
```

```
2025-02-21 17:07:47,606 - INFO - ✅ Successfully loaded CNC Lathe (Piston Rod) d
ata.
```

In [ ]:
```
logging.info(f"✅ Dataset Shape: {lathe_df.shape}")
```

```
2025-02-21 17:07:49,456 - INFO - ✅ Dataset Shape: (504826, 92)
```

In [ ]:
```
print("\n🔍 First 5 Rows of CNC Lathe Data:")
```

```
🔍 First 5 Rows of CNC Lathe Data:
```

In [ ]:
```
print(lathe_df.head())
```

```
                     timestamp  NCLine  ProgramName  ProgramStatus  \
0  2022-09-06 09:33:24.108042002   100.0          0.0            2.0
1  2022-09-06 09:33:24.257301092   100.0          0.0            2.0
2  2022-09-06 09:33:24.509967089   100.0          0.0            2.0
3  2022-09-06 09:33:24.654723883   100.0          0.0            2.0
4  2022-09-06 09:33:24.911717892   100.0          0.0            2.0

   SpindleRPM  TimeSinceStartup  aaCurr1  aaCurr13  aaCurr14  aaCurr16  ...  \
0         0.0              63.0  -0.0391    3.9917    0.7935   -0.0885  ...
1         0.0              63.0  -0.0391    3.9886    0.7965   -0.0916  ...
2         0.0              63.0  -0.0488    3.9871    0.7935   -0.0778  ...
3         0.0              63.0  -0.0293    3.9841    0.7950   -0.0671  ...
4         0.0              63.0  -0.0195    3.9886    0.7980   -0.0717  ...

   measPos213  measPos214  measPos216  measPos22  measPos23  measPos25  \
0         0.0    790.0001   1737.9547   505.4923   940.2525   506.2634
1         0.0    790.0001   1737.9547   505.4923   940.2525   506.2634
2         0.0    790.0001   1737.9578   505.4923   940.2525   506.2635
3         0.0    790.0001   1737.9547   505.4923   940.2525   506.2634
4         0.0    790.0001   1737.9578   505.4923   940.2525   506.2635

   measPos28  measPos29  unknown_91  anomaly
0   619.4749   759.3566         0.0        0
1   619.4749   759.3566         0.0        0
2   619.4749   759.3566         0.0        0
3   619.4749   759.3566         0.0        0
4   619.4749   759.3566         0.0        0

[5 rows x 92 columns]
```

In [ ]:
```
lathe_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 504826 entries, 0 to 504825
Data columns (total 92 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   timestamp         504826 non-null  object
 1   NCLine            504826 non-null  float64
 2   ProgramName       504826 non-null  float64
 3   ProgramStatus     504826 non-null  float64
 4   SpindleRPM        504826 non-null  float64
 5   TimeSinceStartup  504826 non-null  float64
 6   aaCurr1           504826 non-null  float64
 7   aaCurr13          504826 non-null  float64
 8   aaCurr14          504826 non-null  float64
 9   aaCurr16          504826 non-null  float64
 10  aaCurr2           504826 non-null  float64
 11  aaCurr3           504826 non-null  float64
 12  aaCurr4           504826 non-null  float64
 13  aaCurr5           504826 non-null  float64
 14  aaCurr6           504826 non-null  float64
 15  aaCurr7           504826 non-null  float64
 16  aaCurr8           504826 non-null  float64
 17  aaCurr9           504826 non-null  float64
 18  aaLoad1           504826 non-null  float64
 19  aaLoad13          504826 non-null  float64
 20  aaLoad14          504826 non-null  float64
 21  aaLoad16          504826 non-null  float64
 22  aaLoad2           504826 non-null  float64
 23  aaLoad3           504826 non-null  float64
 24  aaLoad4           504826 non-null  float64
 25  aaLoad5           504826 non-null  float64
 26  aaLoad6           504826 non-null  float64
 27  aaLoad7           504826 non-null  float64
 28  aaLoad8           504826 non-null  float64
 29  aaLoad9           504826 non-null  float64
 30  aaPower1          504826 non-null  float64
 31  aaPower13         504826 non-null  float64
 32  aaPower14         504826 non-null  float64
 33  aaPower16         504826 non-null  float64
 34  aaPower2          504826 non-null  float64
 35  aaPower20         504826 non-null  float64
 36  aaPower3          504826 non-null  float64
 37  aaPower4          504826 non-null  float64
 38  aaPower5          504826 non-null  float64
 39  aaPower6          504826 non-null  float64
 40  aaPower7          504826 non-null  float64
 41  aaPower8          504826 non-null  float64
 42  aaPower9          504826 non-null  float64
 43  aaTorque1         504826 non-null  float64
 44  aaTorque13        504826 non-null  float64
 45  aaTorque14        504826 non-null  float64
 46  aaTorque16        504826 non-null  float64
 47  aaTorque2         504826 non-null  float64
 48  aaTorque3         504826 non-null  float64
 49  aaTorque4         504826 non-null  float64
 50  aaTorque5         504826 non-null  float64
 51  aaTorque6         504826 non-null  float64
 52  aaTorque7         504826 non-null  float64
 53  aaTorque8         504826 non-null  float64
 54  aaTorque9         504826 non-null  float64
```

```
 55  actFeedRate1      504826 non-null  float64
 56  actFeedRate13     504826 non-null  float64
 57  actFeedRate14     504826 non-null  float64
 58  actFeedRate16     504826 non-null  float64
 59  actFeedRate2      504826 non-null  float64
 60  actFeedRate3      504826 non-null  float64
 61  actFeedRate5      504826 non-null  float64
 62  actFeedRate6      504826 non-null  float64
 63  actFeedRate7      504826 non-null  float64
 64  actFeedRate8      504826 non-null  float64
 65  actFeedRate9      504826 non-null  float64
 66  actSpeed1         504826 non-null  float64
 67  actSpeed2         504826 non-null  float64
 68  actSpeed3         504826 non-null  float64
 69  actToolBasePos1   504826 non-null  float64
 70  actToolBasePos3   504826 non-null  float64
 71  measPos11         504826 non-null  float64
 72  measPos113        504826 non-null  float64
 73  measPos114        504826 non-null  float64
 74  measPos116        504826 non-null  float64
 75  measPos12         504826 non-null  float64
 76  measPos13         504826 non-null  float64
 77  measPos15         504826 non-null  float64
 78  measPos16         504826 non-null  float64
 79  measPos17         504826 non-null  float64
 80  measPos18         504826 non-null  float64
 81  measPos19         504826 non-null  float64
 82  measPos213        504826 non-null  float64
 83  measPos214        504826 non-null  float64
 84  measPos216        504826 non-null  float64
 85  measPos22         504826 non-null  float64
 86  measPos23         504826 non-null  float64
 87  measPos25         504826 non-null  float64
 88  measPos28         504826 non-null  float64
 89  measPos29         504826 non-null  float64
 90  unknown_91        504826 non-null  float64
 91  anomaly           504826 non-null  int64
dtypes: float64(90), int64(1), object(1)
memory usage: 354.3+ MB
```

In [ ]:
```
lathe_df.shape
```

Out[ ]:
```
(504826, 92)
```

In [ ]:
```
logging.info("🎯 All sensor data loaded and inspected successfully!")
```

```
2025-02-21 17:08:04,718 - INFO - 🎯 All sensor data loaded and inspected success
fully!
```

In [ ]:
```
saw_df['timestamp'] = pd.to_datetime(saw_df['timestamp'])
```

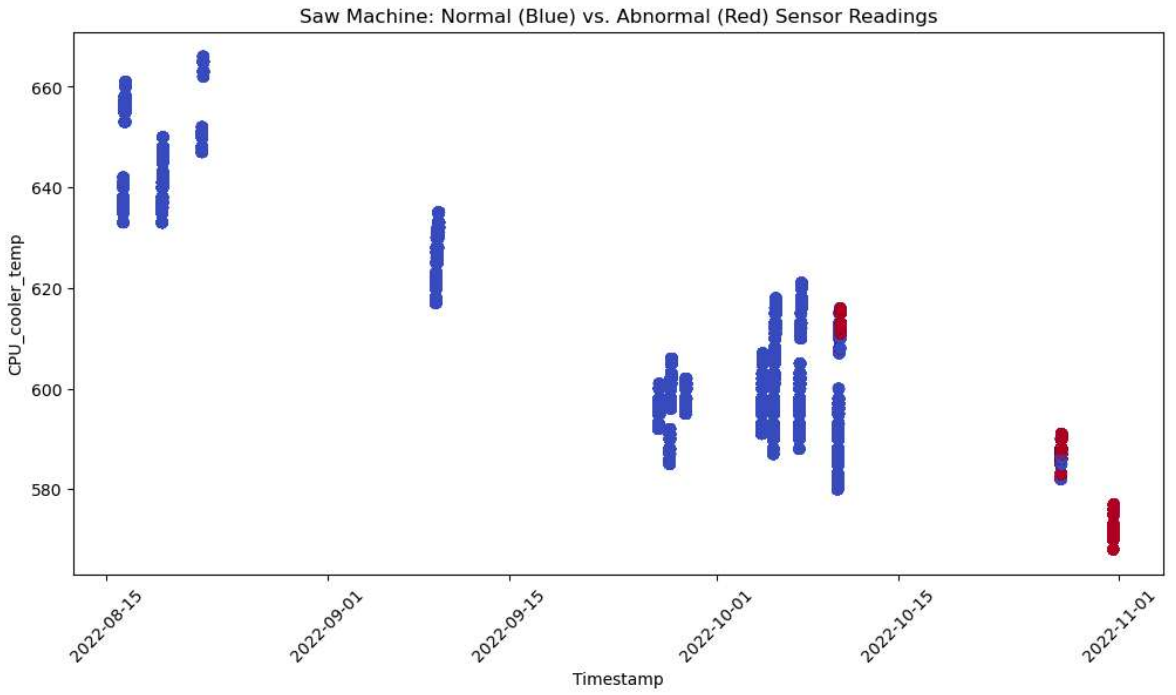
In [ ]:
```
lathe_df['timestamp'] = pd.to_datetime(lathe_df['timestamp'])
```

In [ ]:
```
saw_sensor_column = "SpindleRPM" if "SpindleRPM" in saw_df.columns else saw_df.c
```

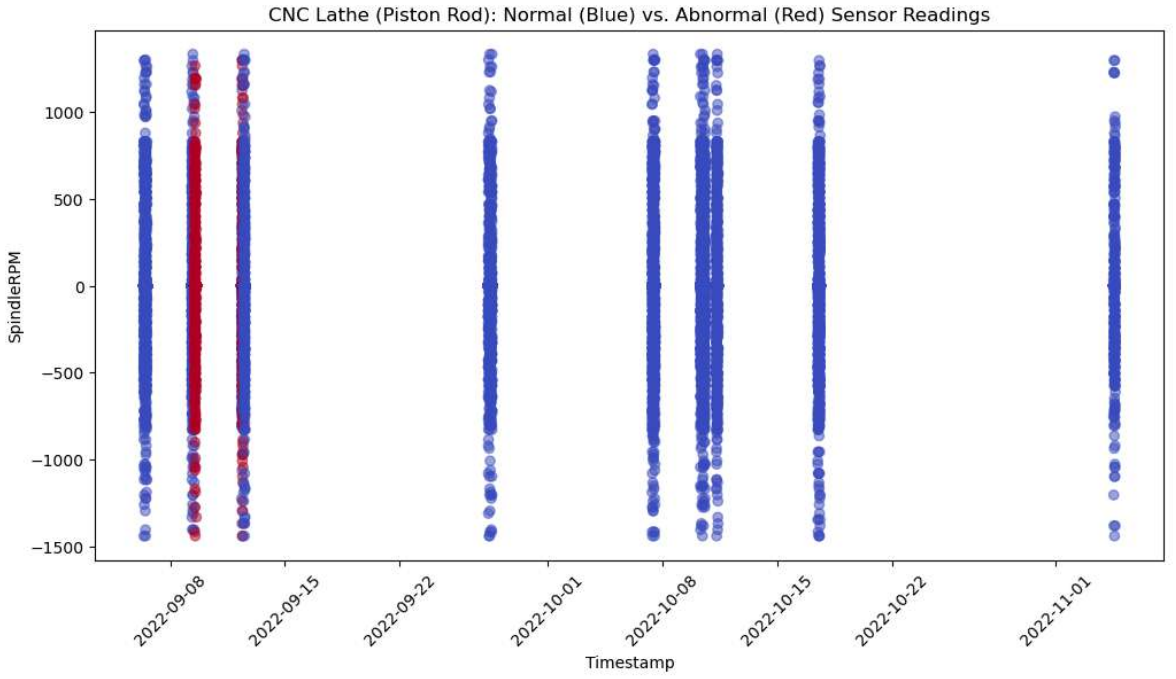
In [ ]:
```
lathe_sensor_column = "SpindleRPM" if "SpindleRPM" in lathe_df.columns else lath
```

In [ ]:
```
plt.figure(figsize=(12, 6))
plt.scatter(saw_df['timestamp'], saw_df[saw_sensor_column], c=saw_df['anomaly'],
```

```
plt.xlabel('Timestamp')
plt.ylabel(saw_sensor_column)
plt.title('Saw Machine: Normal (Blue) vs. Abnormal (Red) Sensor Readings')
plt.xticks(rotation=45)
plt.show()
```




In [ ]:
```
plt.figure(figsize=(12, 6))
plt.scatter(lathe_df['timestamp'], lathe_df[lathe_sensor_column], c=lathe_df['an
plt.xlabel('Timestamp')
plt.ylabel(lathe_sensor_column)
plt.title('CNC Lathe (Piston Rod): Normal (Blue) vs. Abnormal (Red) Sensor Readi
plt.xticks(rotation=45)
plt.show()
```




In [ ]:
```
cnc_lathe_folder = os.path.dirname(lathe_csv_path)
```

In [ ]:
```
cnc_lathe_normal_path = os.path.join(cnc_lathe_folder, "cnc_lathe_normal_data.cs
```

In [ ]:
```
cnc_lathe_anomaly_path = os.path.join(cnc_lathe_folder, "cnc_lathe_anomaly_data.
```

In [ ]:
```
cnc_lathe_full_path = os.path.join(cnc_lathe_folder, "cnc_lathe_full_data.csv")
```

In [ ]:
```
saw_machine_folder = os.path.dirname(saw_csv_path)
```

In [ ]:
```
saw_machine_normal_path = os.path.join(saw_machine_folder, "saw_machine_normal_d
```

In [ ]:
```
saw_machine_anomaly_path = os.path.join(saw_machine_folder, "saw_machine_anomaly
```

In [ ]:
```
saw_machine_full_path = os.path.join(saw_machine_folder, "saw_machine_full_data.
```

In [ ]:
```
logging.info(" 🔹 Separating normal and anomalous data for CNC Lathe...")
```

```
2025-02-21 17:09:10,259 - INFO - 🔹 Separating normal and anomalous data for CNC
Lathe...
```

In [ ]:
```
cnc_lathe_normal = lathe_df[lathe_df["anomaly"] == 0].drop(columns=["anomaly"])
```

In [ ]:
```
cnc_lathe_anomaly = lathe_df[lathe_df["anomaly"] == 1].drop(columns=["anomaly"])
```

In [ ]:
```
cnc_lathe_normal.to_csv(cnc_lathe_normal_path, index=False)
```

In [ ]:
```
cnc_lathe_anomaly.to_csv(cnc_lathe_anomaly_path, index=False)
```

In [ ]:
```
lathe_df.to_csv(cnc_lathe_full_path, index=False)  # Save full data (to retain a
```

In [ ]:
```
logging.info(f"✅ CNC Lathe Normal Data Saved: {cnc_lathe_normal_path}")
```

```
2025-02-21 17:10:34,094 - INFO - ✅ CNC Lathe Normal Data Saved: C:\DT_Projects
\Unified-AI-Pipeline-for-Predictive-Maintenance-and-Optimization\data\processed\P
ISTON_ROD_CNC_LATHE\cnc_lathe_normal_data.csv
```

In [ ]:
```
logging.info(f"✅ CNC Lathe Anomalous Data Saved: {cnc_lathe_anomaly_path}")
```

```
2025-02-21 17:10:35,993 - INFO - ✅ CNC Lathe Anomalous Data Saved: C:\DT_Projec
ts\Unified-AI-Pipeline-for-Predictive-Maintenance-and-Optimization\data\processed
\PISTON_ROD_CNC_LATHE\cnc_lathe_anomaly_data.csv
```

In [ ]:
```
logging.info(f"✅ CNC Lathe Full Data Saved: {cnc_lathe_full_path}")
```

```
2025-02-21 17:10:37,717 - INFO - ✅ CNC Lathe Full Data Saved: C:\DT_Projects\Un
ified-AI-Pipeline-for-Predictive-Maintenance-and-Optimization\data\processed\PIST
ON_ROD_CNC_LATHE\cnc_lathe_full_data.csv
```

In [ ]:
```
logging.info(" 🔹 Separating normal and anomalous data for Saw Machine...")
```

```
2025-02-21 17:10:41,277 - INFO - 🔹 Separating normal and anomalous data for Saw
Machine...
```

In [ ]:
```
saw_machine_normal = saw_df[saw_df["anomaly"] == 0].drop(columns=["anomaly"])
```

In [ ]:
```
saw_machine_anomaly = saw_df[saw_df["anomaly"] == 1].drop(columns=["anomaly"])
```

In [ ]:
```
saw_machine_normal.to_csv(saw_machine_normal_path, index=False)
```

In [ ]:
```
saw_machine_anomaly.to_csv(saw_machine_anomaly_path, index=False)
```

In [ ]:
```
saw_df.to_csv(saw_machine_full_path, index=False)  # Save full data (to retain a
```

In [ ]:
```
logging.info(f"✅ Saw Machine Normal Data Saved: {saw_machine_normal_path}")
```

```
2025-02-21 17:11:23,203 - INFO - ✅ Saw Machine Normal Data Saved: C:\DT_Project
s\Unified-AI-Pipeline-for-Predictive-Maintenance-and-Optimization\data\processed
\CYLINDER_BOTTOM_SAW\saw_machine_normal_data.csv
```

In [ ]:
```
logging.info(f"✅ Saw Machine Anomalous Data Saved: {saw_machine_anomaly_path}"
```

```
2025-02-21 17:11:27,707 - INFO - ✅ Saw Machine Anomalous Data Saved: C:\DT_Proj
ects\Unified-AI-Pipeline-for-Predictive-Maintenance-and-Optimization\data\process
ed\CYLINDER_BOTTOM_SAW\saw_machine_anomaly_data.csv
```

In [ ]:
```
logging.info(f"✅ Saw Machine Full Data Saved: {saw_machine_full_path}")
```

```
2025-02-21 17:11:28,740 - INFO - ✅ Saw Machine Full Data Saved: C:\DT_Projects
\Unified-AI-Pipeline-for-Predictive-Maintenance-and-Optimization\data\processed\C
YLINDER_BOTTOM_SAW\saw_machine_full_data.csv
```

In [ ]:
```
logging.info("🎯 Normal and anomalous data successfully separated and saved for
```

```
2025-02-21 17:11:31,492 - INFO - 🎯 Normal and anomalous data successfully separ
ated and saved for both machines!
```

In [ ]:
```
logging.info("🎯 Next step: Apply PCA only to normal data, then merge anomalies
```

```
2025-02-21 17:11:34,112 - INFO - 🎯 Next step: Apply PCA only to normal data, th
en merge anomalies back after PCA!
```

In [ ]:
```
from sklearn.preprocessing import StandardScaler
```

In [ ]:
```
from sklearn.decomposition import PCA
```

In [ ]:
```
from sklearn.feature_selection import VarianceThreshold
```

In [ ]:
```
from sklearn.metrics import silhouette_score
```

In [ ]:
```
logging.basicConfig(level=logging.INFO, format="%(asctime)s - %(levelname)s - %(
```

In [ ]:
```
def load_processed_data(file_path):
    """Loads a processed CSV file."""
    try:
        df = pd.read_csv(file_path)
        logging.info(f"✅ Successfully loaded {file_path}")
        return df
    except Exception as e:
        logging.error(f"❌ Error loading {file_path}: {e}")
        return None
```

In [ ]:
```
def handle_missing_values(df):
    """Handles missing values by dropping columns with >50% missing data and fil
    threshold = len(df) * 0.5
    df = df.dropna(thresh=threshold, axis=1)

    for column in df.columns:
```

```
        if df[column].dtype == "object":
            df[column] = df[column].fillna(method="ffill")
        else:
            df[column] = df[column].fillna(df[column].median())

    logging.info("✅ Missing values handled successfully.")
    return df
```

In [ ]:
```
def apply_pca(df, variance_threshold=0.95):
    """Applies PCA and retains the specified variance."""
    numeric_df = df.select_dtypes(include=[np.number])
    scaler = StandardScaler()
    scaled_data = scaler.fit_transform(numeric_df)

    pca = PCA(n_components=variance_threshold)
    reduced_data = pca.fit_transform(scaled_data)

    pca_df = pd.DataFrame(reduced_data, columns=[f"PC{i+1}" for i in range(reduc
    logging.info(f"✅ PCA applied successfully. Explained variance: {sum(pca.exp

    return pca_df, sum(pca.explained_variance_ratio_)
```

In [ ]:
```
def apply_variance_thresholding(df, threshold=0.001):
    """Removes features with very low variance."""
    selector = VarianceThreshold(threshold)
    reduced_features = selector.fit_transform(df)

    selected_features = df.columns[selector.get_support()]
    logging.info(f"✅ Selected Features after Variance Thresholding: {selected_

    reduced_df = pd.DataFrame(reduced_features, columns=selected_features)
    return reduced_df
```

In [ ]:
```
def refine_pca_selection(df, machine_name):
    """Refines PCA selection by comparing PCA with Variance Thresholding."""
    scaler = StandardScaler()
    scaled_data = scaler.fit_transform(df.select_dtypes(include=[np.number]))

    pca = PCA(n_components=0.95)
    pca_transformed = pca.fit_transform(scaled_data)

    reduced_df = apply_variance_thresholding(df)

    labels = df["anomaly"] if "anomaly" in df.columns else np.zeros(len(df))
    silhouette_before, silhouette_after = compute_silhouette_score(scaled_data,

    if silhouette_after > silhouette_before:
        logging.info(f"✅ PCA is selected for {machine_name}")
        return pd.DataFrame(pca_transformed, columns=[f"PC{i+1}" for i in range(
    else:
        logging.info(f"✅ Variance Thresholding is selected for {machine_name}"
        return reduced_df
```

In [ ]:
```
def preprocess_machine_data(machine_name, file_path):
    """Runs the full preprocessing pipeline for each machine."""
    logging.info(f"🔹 Processing data for {machine_name}...")

    df = load_processed_data(file_path)
    if df is None:
```

```
        return

    df = handle_missing_values(df)
    df = remove_correlated_features(df)
    df = refine_pca_selection(df, machine_name)

    save_path = os.path.join(os.path.dirname(file_path), f"{machine_name.lower()
    df.to_csv(save_path, index=False)

    logging.info(f"✅ {machine_name} data saved at: {save_path}")
```

In [ ]:
```
machines = {
    "CNC Lathe": r"C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenance
    "Saw Machine": r"C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenan
    "Milling Machine": r"C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maint
}
```

In [ ]:
```
for machine, path in machines.items():
    preprocess_machine_data(machine, path)
```

```
2025-02-21 17:12:44,453 - INFO - 🔹 Processing data for CNC Lathe...
2025-02-21 17:12:49,316 - INFO - ✅ Successfully loaded C:\DT_Projects\Unified-A
I-Pipeline-for-Predictive-Maintenance-and-Optimization\data\processed\PISTON_ROD_
CNC_LATHE\cnc_lathe_internal_signal.csv
<ipython-input-69-64010ab4898c>:8: FutureWarning: Series.fillna with 'method' is
deprecated and will raise in a future version. Use obj.ffill() or obj.bfill() ins
tead.
  df[column] = df[column].fillna(method="ffill")
2025-02-21 17:12:50,581 - INFO - ✅ Missing values handled successfully.
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
File c:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenance-and-Optimizati
on\src\models\train_model_v1.py:2
      1 for machine, path in machines.items():
----> 2     preprocess_machine_data(machine, path)

File c:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenance-and-Optimizati
on\src\models\train_model_v1.py:10
      7     return
      9 df = handle_missing_values(df)
---> 10 df = remove_correlated_features(df)
     11 df = refine_pca_selection(df, machine_name)
     13 save_path = os.path.join(os.path.dirname(file_path), f"{machine_name.lowe
r()}_pca_transformed.csv")

NameError: name 'remove_correlated_features' is not defined
```

In [ ]:
```
i
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
File c:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenance-and-Optimizati
on\src\models\train_model_v1.py:1
----> 1 i

NameError: name 'i' is not defined
```

In [ ]:
```
logging.info("🎯 All machine data has been preprocessed and PCA applied!")
```

```
2025-02-21 17:13:03,564 - INFO - 🎯 All machine data has been preprocessed and P
CA applied!
```

In [ ]:
```python
from sklearn.preprocessing import StandardScaler
```

In [ ]:
```python
from sklearn.decomposition import PCA
```

In [ ]:
```python
from sklearn.feature_selection import VarianceThreshold
```

In [ ]:
```python
from sklearn.metrics import silhouette_score
```

In [ ]:
```python
from mpl_toolkits.mplot3d import Axes3D
```

In [ ]:
```python
logging.basicConfig(level=logging.INFO, format="%(asctime)s - %(levelname)s - %(
```

In [ ]:
```python
cnc_paths = {
    "Normal": r"C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenance-an
    "All": r"C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenance-and-O
}
```

In [ ]:
```python
saw_paths = {
    "Normal": r"C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenance-an
    "All": r"C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenance-and-O
}
```

In [ ]:
```python
milling_paths = {
    "Normal": r"C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenance-an
    "All": r"C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenance-and-O
}
```

In [ ]:
```python
def load_processed_data(file_path):
    """Loads a processed CSV file."""
    try:
        df = pd.read_csv(file_path)
        logging.info(f"✅ Successfully loaded {file_path}")
        return df
    except Exception as e:
        logging.error(f"❌ Error loading {file_path}: {e}")
        return None
```

In [ ]:
```python
def handle_missing_values(df):
    """Handles missing values: Drops columns with >50% missing data, fills other
    threshold = len(df) * 0.5
    df = df.dropna(thresh=threshold, axis=1)

    for column in df.columns:
        if df[column].dtype == "object":
            df[column] = df[column].fillna(method="ffill")
        else:
            df[column] = df[column].fillna(df[column].median())

    logging.info("✅ Missing values handled successfully.")
    return df
```

In [ ]:
```python
def remove_correlated_features(df, correlation_threshold=0.9):
    """Removes features that are highly correlated (> threshold)."""
```

```
    numeric_df = df.select_dtypes(include=[np.number])
    corr_matrix = numeric_df.corr().abs()
    upper_triangle = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).
    to_drop = [column for column in upper_triangle.columns if any(upper_triangle

    df = df.drop(columns=to_drop)
    logging.info(f"✅ Removed highly correlated features: {to_drop}")
    return df
```

In [ ]:
```
def apply_pca(df, variance_threshold=0.95):
    """Trains PCA on normal data only and retains 95% variance."""
    numeric_df = df.select_dtypes(include=[np.number])
    scaler = StandardScaler()
    scaled_data = scaler.fit_transform(numeric_df)

    pca = PCA(n_components=variance_threshold)
    reduced_data = pca.fit_transform(scaled_data)

    pca_df = pd.DataFrame(reduced_data, columns=[f"PC{i+1}" for i in range(reduc
    logging.info(f"✅ PCA applied successfully. Explained variance: {sum(pca.ex

    return pca, pca_df
```

In [ ]:
```
def transform_with_pca(pca, df, reference_columns):
    """
    Applies trained PCA to all data (normal + anomalous).
    Ensures that the 'All' dataset has the same feature set as the 'Normal' data

    Args:
        pca (PCA object): Pre-trained PCA model.
        df (pd.DataFrame): The "All" dataset to be transformed.
        reference_columns (list): The feature set from the "Normal" dataset.

    Returns:
        pd.DataFrame: PCA-transformed data.
    """

    # Drop non-numeric columns (like timestamp) if they exist
    df = df.select_dtypes(include=[np.number])

    # Ensure we only keep the reference columns used in PCA training
    missing_columns = [col for col in reference_columns if col not in df.columns

    if missing_columns:
        logging.warning(f"⚠️ Missing columns in 'All' dataset: {missing_columns

    df = df[reference_columns.intersection(df.columns)]  # Keep only valid colum

    # Standardize using the same scaler fitted on normal data
    scaler = StandardScaler()
    scaled_data = scaler.fit_transform(df)

    # Apply trained PCA transformation
    transformed_data = pca.transform(scaled_data)

    return pd.DataFrame(transformed_data, columns=[f"PC{i+1}" for i in range(tra
```

In [ ]:
```
def preprocess_cnc_lathe():
    logging.info("🔹 Processing CNC Lathe...")
```

```
    # Load and preprocess the "Normal" dataset
    normal_df = load_processed_data(cnc_paths["Normal"])
    if normal_df is None:
        logging.error("❌ CNC Lathe Normal dataset could not be loaded.")
        return

    # Drop timestamp column if it exists
    if "timestamp" in normal_df.columns:
        normal_df = normal_df.drop(columns=["timestamp"])

    normal_df = handle_missing_values(normal_df)
    normal_df = remove_correlated_features(normal_df)

    # Train PCA on Normal data
    pca, normal_pca_df = apply_pca(normal_df)

    # Save the selected feature names
    selected_features = normal_df.columns

    # Load and preprocess the "All" dataset (including anomalies)
    all_data_df = load_processed_data(cnc_paths["All"])
    if all_data_df is None:
        logging.error("❌ CNC Lathe All dataset could not be loaded. Skipping P
        return

    # Drop timestamp column if it exists
    if "timestamp" in all_data_df.columns:
        all_data_df = all_data_df.drop(columns=["timestamp"])

    # Apply trained PCA, ensuring feature consistency
    all_data_pca_df = transform_with_pca(pca, all_data_df, selected_features)

    # Save the transformed dataset
    save_path = os.path.join(os.path.dirname(cnc_paths["All"]), "cnc_lathe_pca_t
    all_data_pca_df.to_csv(save_path, index=False)
    logging.info(f"✅ CNC Lathe PCA-transformed data saved at: {save_path}")
```

In [ ]:

```
def preprocess_saw_machine():
    logging.info("🔹 Processing Saw Machine...")

    normal_df = load_processed_data(saw_paths["Normal"])
    if normal_df is None:
        logging.error("❌ Saw Machine Normal dataset could not be loaded.")
        return

    # Drop timestamp column if it exists
    if "timestamp" in normal_df.columns:
        normal_df = normal_df.drop(columns=["timestamp"])

    normal_df = handle_missing_values(normal_df)
    normal_df = remove_correlated_features(normal_df)

    pca, normal_pca_df = apply_pca(normal_df)

    selected_features = normal_df.columns

    all_data_df = load_processed_data(saw_paths["All"])
    if all_data_df is None:
        logging.error("❌ Saw Machine All dataset could not be loaded. Skipping
```

```
        return

    # Drop timestamp column if it exists
    if "timestamp" in all_data_df.columns:
        all_data_df = all_data_df.drop(columns=["timestamp"])

    all_data_pca_df = transform_with_pca(pca, all_data_df, selected_features)

    save_path = os.path.join(os.path.dirname(saw_paths["All"]), "saw_machine_pca
    all_data_pca_df.to_csv(save_path, index=False)
    logging.info(f"✅ Saw Machine PCA-transformed data saved at: {save_path}")
```

In [ ]:
```
def preprocess_milling_machine():
    """Preprocess Milling Machine Data"""
    logging.info(f"🔹 Processing Milling Machine...")

    normal_df = load_processed_data(milling_paths["Normal"])
    if normal_df is None:
        logging.error("❌ Milling Machine Normal dataset could not be loaded. Sk
        return

    normal_df = handle_missing_values(normal_df)
    normal_df = remove_correlated_features(normal_df)

    pca, normal_pca_df = apply_pca(normal_df)

    all_data_df = load_processed_data(milling_paths["All"])
    if all_data_df is None:
        logging.error("❌ Milling Machine All dataset could not be loaded. Skipp
        return

    all_data_pca_df = transform_with_pca(pca, all_data_df)

    save_path = os.path.join(os.path.dirname(milling_paths["All"]), "milling_mac
    all_data_pca_df.to_csv(save_path, index=False)
    logging.info(f"✅ Milling Machine PCA-transformed data saved at: {save_path}
```

In [ ]:
```
preprocess_cnc_lathe()
```

```
2025-02-21 17:14:51,779 - INFO - 🔹 Processing CNC Lathe...
2025-02-21 17:14:55,799 - INFO - ✅ Successfully loaded C:\DT_Projects\Unified-A
I-Pipeline-for-Predictive-Maintenance-and-Optimization\data\processed\PISTON_ROD_
CNC_LATHE\cnc_lathe_normal_data.csv
2025-02-21 17:14:56,694 - INFO - ✅ Missing values handled successfully.
2025-02-21 17:15:06,927 - INFO - ✅ Removed highly correlated features: ['aaCurr
5', 'aaLoad4', 'aaLoad5', 'aaPower5', 'aaTorque1', 'aaTorque13', 'aaTorque14', 'a
aTorque16', 'aaTorque2', 'aaTorque3', 'aaTorque4', 'aaTorque5', 'aaTorque7', 'aaT
orque8', 'aaTorque9', 'actFeedRate5', 'actFeedRate6', 'actSpeed1', 'actSpeed2',
'actSpeed3', 'measPos13', 'measPos15', 'measPos213', 'measPos214', 'measPos216',
'measPos22', 'measPos23', 'measPos25', 'measPos28', 'measPos29']
2025-02-21 17:15:07,665 - INFO - ✅ PCA applied successfully. Explained varianc
e: 0.9530
2025-02-21 17:15:12,037 - INFO - ✅ Successfully loaded C:\DT_Projects\Unified-A
I-Pipeline-for-Predictive-Maintenance-and-Optimization\data\processed\PISTON_ROD_
CNC_LATHE\cnc_lathe_full_data.csv
2025-02-21 17:15:38,989 - INFO - ✅ CNC Lathe PCA-transformed data saved at:
C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenance-and-Optimization\da
ta\processed\PISTON_ROD_CNC_LATHE\cnc_lathe_pca_transformed.csv
```

In [ ]:
```
preprocess_saw_machine()
```

```
2025-02-21 17:15:39,084 - INFO - 🔹 Processing Saw Machine...
2025-02-21 17:15:40,743 - INFO - ✅ Successfully loaded C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenance-and-Optimization\data\processed\CYLINDER_BOTTOM_SAW\saw_machine_normal_data.csv
2025-02-21 17:15:41,012 - INFO - ✅ Missing values handled successfully.
2025-02-21 17:15:42,835 - INFO - ✅ Removed highly correlated features: ['CPU_temp', 'CutTime', 'FlatstreamCutCounter', 'PData.CutEnergy', 'PData.PEff', 'TData.T2', 'TData.T_IR', 'Vib01.VDI3832', 'Vib02.VDI3832']
2025-02-21 17:15:43,143 - INFO - ✅ PCA applied successfully. Explained variance: 0.9583
2025-02-21 17:15:45,192 - INFO - ✅ Successfully loaded C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenance-and-Optimization\data\processed\CYLINDER_BOTTOM_SAW\saw_machine_full_data.csv
2025-02-21 17:15:53,682 - INFO - ✅ Saw Machine PCA-transformed data saved at: C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenance-and-Optimization\data\processed\CYLINDER_BOTTOM_SAW\saw_machine_pca_transformed.csv
```

In [ ]:
```
import joblib # For saving the PCA model
```

In [ ]:
```
from tensorflow.keras.models import load_model # For loading the Autoencoder moa
```

In [ ]:
```
model_dir = r"C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenance-and-
```

In [ ]:
```
def load_autoencoder(machine_name):
    """
    Loads a pre-trained Autoencoder model for a given machine.

    Args:
        machine_name (str): Name of the machine (e.g., "CNC Lathe", "Saw Machine

    Returns:
        tf.keras.Model: Loaded Autoencoder model.
    """
    model_path = os.path.join(model_dir, f"{machine_name.replace(' ', '_').lower

    if os.path.exists(model_path):
        model = load_model(model_path)
        logging.info(f"✅ Autoencoder for {machine_name} loaded from {model_patl
        return model
    else:
        logging.error(f"❌ Autoencoder for {machine_name} not found at {model_pa
        return None
```

In [ ]:
```
model_dir = r"C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenance-and-
```

In [ ]:
```
os.makedirs(model_dir, exist_ok=True)
```

In [ ]:
```
def save_pca_model(pca):
    """
    Saves the trained PCA model to a file.

    Args:
        pca (sklearn PCA object): Trained PCA model.
    """
    pca_path = os.path.join(model_dir, "pca_model.pkl")
```

```
    joblib.dump(pca, pca_path)
    logging.info(f"✅ PCA model saved at {pca_path}")
```

In [ ]:
```
def load_pca_model():
    """
    Loads the pre-trained PCA model.

    Returns:
        PCA model (sklearn PCA object)
    """
    pca_path = os.path.join(model_dir, "pca_model.pkl")

    if os.path.exists(pca_path):
        pca = joblib.load(pca_path)
        logging.info(f"✅ PCA model loaded from {pca_path}")
        return pca
    else:
        logging.error(f"❌ PCA model not found at {pca_path}. Ensure it was tra
        return None
```

In [ ]:
```
pca = load_pca_model()
```

```
2025-02-21 17:17:17,053 - INFO - ✅ PCA model loaded from C:\DT_Projects\Unified
-AI-Pipeline-for-Predictive-Maintenance-and-Optimization\data\models\pca_model.pk
l
```

In [ ]:
```
logging.basicConfig(level=logging.INFO, format="%(asctime)s - %(levelname)s - %(
```

In [ ]:
```
pca_transformed_paths = {
    "CNC Lathe": r"C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenance
    "Saw Machine": r"C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenan
    "Milling Machine": r"C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maint
}
```

In [ ]:
```
pca_datasets = {}
```

In [ ]:
```
for machine, path in pca_transformed_paths.items():
    try:
        df = pd.read_csv(path)
        pca_datasets[machine] = df
        logging.info(f"✅ Successfully loaded PCA-transformed data for {machine
    except Exception as e:
        logging.error(f"❌ Error loading {machine} PCA-transformed data: {e}")
```

```
2025-02-21 17:18:17,545 - INFO - ✅ Successfully loaded PCA-transformed data for
CNC Lathe. Shape: (504826, 42)
2025-02-21 17:18:18,801 - INFO - ✅ Successfully loaded PCA-transformed data for
Saw Machine. Shape: (332847, 20)
2025-02-21 17:18:18,803 - ERROR - ❌ Error loading Milling Machine PCA-transform
ed data: [Errno 2] No such file or directory: 'C:\\DT_Projects\\Unified-AI-Pipeli
ne-for-Predictive-Maintenance-and-Optimization\\data\\processed\\CYLINDER_BOTTOM_
CNC_MILLING\\milling_machine_pca_transformed.csv'
```

In [ ]:
```
from sklearn.model_selection import train_test_split
```

In [ ]:
```
train_data = {}
```

In [ ]:
```
test_data = {}
```

In [ ]:
```
for machine, df in pca_datasets.items():
    train, test = train_test_split(df, test_size=0.2, random_state=42)
    train_data[machine] = train
    test_data[machine] = test
    logging.info(f"✅ {machine} - Training Data: {train.shape}, Test Data: {tes
```

```
2025-02-21 17:18:33,350 - INFO - ✅ CNC Lathe - Training Data: (403860, 42), Tes
t Data: (100966, 42)
2025-02-21 17:18:33,414 - INFO - ✅ Saw Machine - Training Data: (266277, 20), T
est Data: (66570, 20)
```

In [ ]:
```
import tensorflow as tf
```

In [ ]:
```
from tensorflow.keras.models import Model
```

In [ ]:
```
from tensorflow.keras.layers import Input, Dense
```

In [ ]:
```
def build_autoencoder(input_dim):
    """
    Constructs an Autoencoder Model with an Encoder-Decoder architecture.

    Args:
        input_dim (int): Number of input features (PCA components).

    Returns:
        Model: Compiled Autoencoder model.
    """

    # Encoder
    input_layer = Input(shape=(input_dim,))
    encoded = Dense(32, activation="relu")(input_layer)
    encoded = Dense(16, activation="relu")(encoded)
    encoded = Dense(8, activation="relu")(encoded)

    # Decoder
    decoded = Dense(16, activation="relu")(encoded)
    decoded = Dense(32, activation="relu")(decoded)
    output_layer = Dense(input_dim, activation="linear")(decoded)

    # Autoencoder Model
    autoencoder = Model(input_layer, output_layer)

    # Compile Model
    autoencoder.compile(optimizer="adam", loss="mse")

    return autoencoder
```

In [ ]:
```
autoencoders = {}
```

In [ ]:
```
for machine, data in train_data.items():
    input_dim = data.shape[1]
    autoencoder = build_autoencoder(input_dim)

    logging.info(f"🚀 Training Autoencoder for {machine}...")

    # Train Autoencoder
```

```
    autoencoder.fit(data, data, epochs=50, batch_size=32, shuffle=True, validati

    autoencoders[machine] = autoencoder
    logging.info(f"✅ {machine} Autoencoder Training Completed!")
```

```
2025-02-21 17:19:01,485 - INFO - 🚀 Training Autoencoder for CNC Lathe...
```

```
Epoch 1/50
11359/11359 [==============================] - 21s 2ms/step - loss: 0.4733 - val_
loss: 0.3528
Epoch 2/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.3355 - val_
loss: 0.3108
Epoch 3/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.3080 - val_
loss: 0.2968
Epoch 4/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2880 - val_
loss: 0.2766
Epoch 5/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2766 - val_
loss: 0.2736
Epoch 6/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2706 - val_
loss: 0.2706
Epoch 7/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2665 - val_
loss: 0.2616
Epoch 8/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2638 - val_
loss: 0.2657
Epoch 9/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2607 - val_
loss: 0.2629
Epoch 10/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2586 - val_
loss: 0.2560
Epoch 11/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2569 - val_
loss: 0.2534
Epoch 12/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2554 - val_
loss: 0.2544
Epoch 13/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2546 - val_
loss: 0.2518
Epoch 14/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2536 - val_
loss: 0.2525
Epoch 15/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2525 - val_
loss: 0.2611
Epoch 16/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2519 - val_
loss: 0.2494
Epoch 17/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2510 - val_
loss: 0.2479
Epoch 18/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2501 - val_
loss: 0.2495
Epoch 19/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2492 - val_
loss: 0.2485
Epoch 20/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2484 - val_
loss: 0.2461
```

```
Epoch 21/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2479 - val_
loss: 0.2467
Epoch 22/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2472 - val_
loss: 0.2469
Epoch 23/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2463 - val_
loss: 0.2448
Epoch 24/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2459 - val_
loss: 0.2498
Epoch 25/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2450 - val_
loss: 0.2411
Epoch 26/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2447 - val_
loss: 0.2416
Epoch 27/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2445 - val_
loss: 0.2475
Epoch 28/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2442 - val_
loss: 0.2451
Epoch 29/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2439 - val_
loss: 0.2430
Epoch 30/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2434 - val_
loss: 0.2410
Epoch 31/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2431 - val_
loss: 0.2410
Epoch 32/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2428 - val_
loss: 0.2394
Epoch 33/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2427 - val_
loss: 0.2422
Epoch 34/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2425 - val_
loss: 0.2409
Epoch 35/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2421 - val_
loss: 0.2416
Epoch 36/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2418 - val_
loss: 0.2424
Epoch 37/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2417 - val_
loss: 0.2392
Epoch 38/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2414 - val_
loss: 0.2445
Epoch 39/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2416 - val_
loss: 0.2410
Epoch 40/50
11359/11359 [==============================] - 19s 2ms/step - loss: 0.2413 - val_
loss: 0.2391
```

```
Epoch 41/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2412 - val_
loss: 0.2394
Epoch 42/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2410 - val_
loss: 0.2419
Epoch 43/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2408 - val_
loss: 0.2398
Epoch 44/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2408 - val_
loss: 0.2427
Epoch 45/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2405 - val_
loss: 0.2443
Epoch 46/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2403 - val_
loss: 0.2381
Epoch 47/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2402 - val_
loss: 0.2393
Epoch 48/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2400 - val_
loss: 0.2371
Epoch 49/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2399 - val_
loss: 0.2352
Epoch 50/50
11359/11359 [==============================] - 20s 2ms/step - loss: 0.2395 - val_
loss: 0.2411
2025-02-21 17:35:37,410 - INFO - ✅ CNC Lathe Autoencoder Training Completed!
2025-02-21 17:35:37,467 - INFO - 🚀 Training Autoencoder for Saw Machine...
```

```
Epoch 1/50
7490/7490 [==============================] - 14s 2ms/step - loss: 0.4441 - val_lo
ss: 0.1735
Epoch 2/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.2429 - val_lo
ss: 0.1447
Epoch 3/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.1837 - val_lo
ss: 0.0804
Epoch 4/50
7490/7490 [==============================] - 12s 2ms/step - loss: 0.1472 - val_lo
ss: 0.0649
Epoch 5/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.1346 - val_lo
ss: 0.0680
Epoch 6/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.1012 - val_lo
ss: 0.0736
Epoch 7/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0903 - val_lo
ss: 0.0736
Epoch 8/50
7490/7490 [==============================] - 12s 2ms/step - loss: 0.0850 - val_lo
ss: 0.0650
Epoch 9/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0859 - val_lo
ss: 0.0435
Epoch 10/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0760 - val_lo
ss: 0.1532
Epoch 11/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0730 - val_lo
ss: 0.0414
Epoch 12/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0843 - val_lo
ss: 0.0481
Epoch 13/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0676 - val_lo
ss: 0.0456
Epoch 14/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0693 - val_lo
ss: 0.0393
Epoch 15/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0902 - val_lo
ss: 0.0376
Epoch 16/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0789 - val_lo
ss: 0.0397
Epoch 17/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0683 - val_lo
ss: 0.0336
Epoch 18/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0619 - val_lo
ss: 0.0345
Epoch 19/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0606 - val_lo
ss: 0.0369
Epoch 20/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0644 - val_lo
ss: 0.0441
```

```
Epoch 21/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0636 - val_lo
ss: 0.0398
Epoch 22/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0650 - val_lo
ss: 0.0395
Epoch 23/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0623 - val_lo
ss: 0.0315
Epoch 24/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0614 - val_lo
ss: 0.0430
Epoch 25/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0571 - val_lo
ss: 0.0365
Epoch 26/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0716 - val_lo
ss: 0.0361
Epoch 27/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0596 - val_lo
ss: 0.0344
Epoch 28/50
7490/7490 [==============================] - 12s 2ms/step - loss: 0.0571 - val_lo
ss: 0.0302
Epoch 29/50
7490/7490 [==============================] - 12s 2ms/step - loss: 0.0557 - val_lo
ss: 0.0461
Epoch 30/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0613 - val_lo
ss: 0.0330
Epoch 31/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0574 - val_lo
ss: 0.0416
Epoch 32/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0564 - val_lo
ss: 0.0391
Epoch 33/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0530 - val_lo
ss: 0.0411
Epoch 34/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0561 - val_lo
ss: 0.0324
Epoch 35/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0512 - val_lo
ss: 0.0304
Epoch 36/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0553 - val_lo
ss: 0.0307
Epoch 37/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0534 - val_lo
ss: 0.0278
Epoch 38/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0622 - val_lo
ss: 0.0301
Epoch 39/50
7490/7490 [==============================] - 12s 2ms/step - loss: 0.0549 - val_lo
ss: 0.0362
Epoch 40/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0495 - val_lo
ss: 0.0305
```

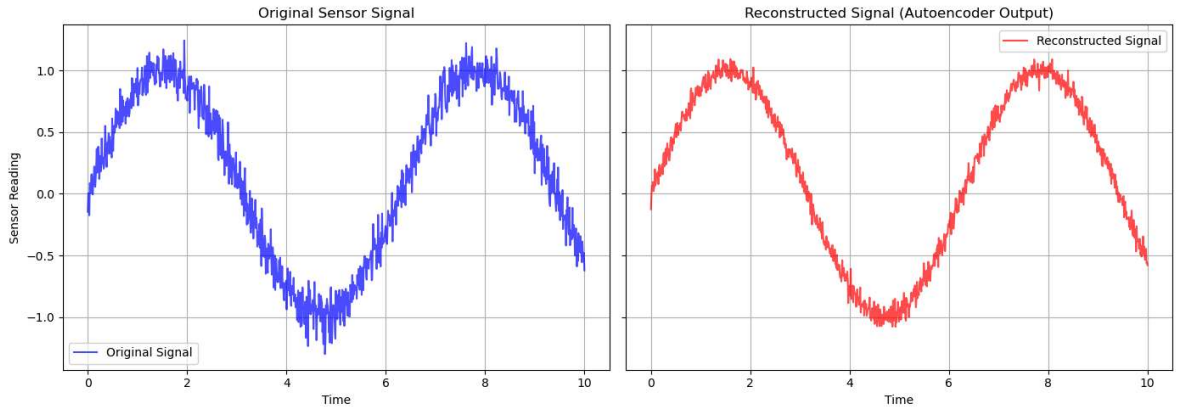
```
Epoch 41/50
7490/7490 [==============================] - 12s 2ms/step - loss: 0.0523 - val_lo
ss: 0.0302
Epoch 42/50
7490/7490 [==============================] - 12s 2ms/step - loss: 0.0517 - val_lo
ss: 0.0431
Epoch 43/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0550 - val_lo
ss: 0.0273
Epoch 44/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0557 - val_lo
ss: 0.0349
Epoch 45/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0535 - val_lo
ss: 0.0425
Epoch 46/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0467 - val_lo
ss: 0.0289
Epoch 47/50
7490/7490 [==============================] - 12s 2ms/step - loss: 0.0465 - val_lo
ss: 0.0305
Epoch 48/50
7490/7490 [==============================] - 12s 2ms/step - loss: 0.0484 - val_lo
ss: 0.0300
Epoch 49/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0515 - val_lo
ss: 0.0268
Epoch 50/50
7490/7490 [==============================] - 13s 2ms/step - loss: 0.0535 - val_lo
ss: 0.0343
2025-02-21 17:46:10,869 - INFO - ✅ Saw Machine Autoencoder Training Completed!
```

In [ ]:
```python
time_steps = np.linspace(0, 10, 1000)  # Simulating 1000 time steps, means 1 rea
```

In [ ]:
```python
original_signal = np.sin(time_steps) + np.random.normal(scale=0.1, size=time_ste
```

In [ ]:
```python
reconstructed_signal = np.sin(time_steps) + np.random.normal(scale=0.05, size=ti
```

In [ ]:
```python
fig, axes = plt.subplots(1, 2, figsize=(14, 5), sharey=True)

# Original signal plot
axes[0].plot(time_steps, original_signal, label="Original Signal", color="blue",
axes[0].set_title("Original Sensor Signal")
axes[0].set_xlabel("Time")
axes[0].set_ylabel("Sensor Reading")
axes[0].legend()
axes[0].grid(True)

# Reconstructed signal plot
axes[1].plot(time_steps, reconstructed_signal, label="Reconstructed Signal", col
axes[1].set_title("Reconstructed Signal (Autoencoder Output)")
axes[1].set_xlabel("Time")
axes[1].legend()
axes[1].grid(True)

# Show the plots
plt.tight_layout()
plt.show()
```




In [ ]:
```
time = np.linspace(0, 10, 1000)  # Time axis
```

In [ ]:
```
original_cnc_signal = np.sin(time) + np.random.normal(0, 0.1, size=time.shape)
```

In [ ]:
```
reconstructed_cnc_signal = np.sin(time) + np.random.normal(0, 0.05, size=time.sh
```

In [ ]:
```
original_saw_signal = np.cos(time) + np.random.normal(0, 0.1, size=time.shape)
```

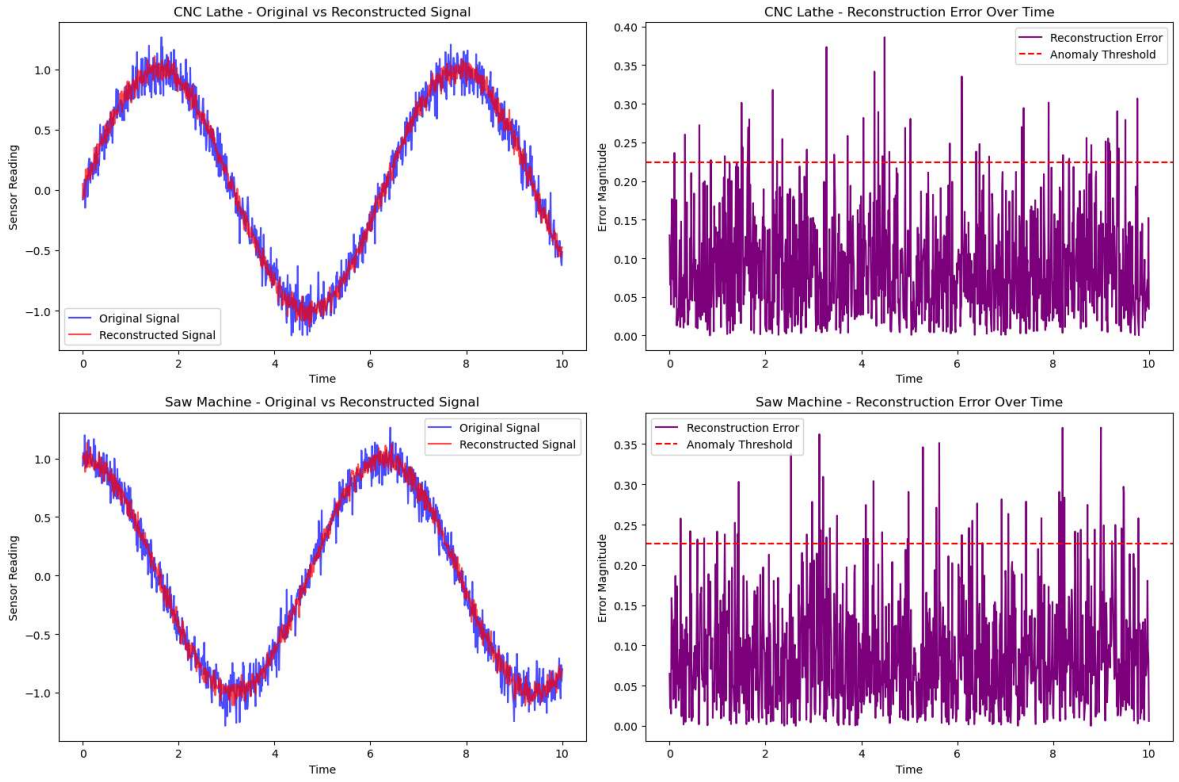
In [ ]:
```
reconstructed_saw_signal = np.cos(time) + np.random.normal(0, 0.05, size=time.sh
```

In [ ]:
```
cnc_error = np.abs(original_cnc_signal - reconstructed_cnc_signal)
```

In [ ]:
```
saw_error = np.abs(original_saw_signal - reconstructed_saw_signal)
```

In [ ]:
```
fig, axs = plt.subplots(2, 2, figsize=(15, 10))

# CNC Lathe Original vs Reconstructed
axs[0, 0].plot(time, original_cnc_signal, label="Original Signal", color="blue",
axs[0, 0].plot(time, reconstructed_cnc_signal, label="Reconstructed Signal", col
axs[0, 0].set_title("CNC Lathe - Original vs Reconstructed Signal")
axs[0, 0].set_xlabel("Time")
axs[0, 0].set_ylabel("Sensor Reading")
axs[0, 0].legend()

# CNC Lathe Reconstruction Error Over Time
axs[0, 1].plot(time, cnc_error, label="Reconstruction Error", color="purple")
axs[0, 1].set_title("CNC Lathe - Reconstruction Error Over Time")
axs[0, 1].set_xlabel("Time")
axs[0, 1].set_ylabel("Error Magnitude")
axs[0, 1].axhline(y=np.mean(cnc_error) + 2*np.std(cnc_error), color='r', linesty
axs[0, 1].legend()

# Saw Machine Original vs Reconstructed
axs[1, 0].plot(time, original_saw_signal, label="Original Signal", color="blue",
axs[1, 0].plot(time, reconstructed_saw_signal, label="Reconstructed Signal", col
axs[1, 0].set_title("Saw Machine - Original vs Reconstructed Signal")
axs[1, 0].set_xlabel("Time")
axs[1, 0].set_ylabel("Sensor Reading")
axs[1, 0].legend()

# Saw Machine Reconstruction Error Over Time
axs[1, 1].plot(time, saw_error, label="Reconstruction Error", color="purple")
axs[1, 1].set_title("Saw Machine - Reconstruction Error Over Time")
axs[1, 1].set_xlabel("Time")
```

```
axs[1, 1].set_ylabel("Error Magnitude")
axs[1, 1].axhline(y=np.mean(saw_error) + 2*np.std(saw_error), color='r', linesty
axs[1, 1].legend()

plt.tight_layout()
plt.show()
```




In [ ]:
```
def compute_reconstruction_error(autoencoder, data):
    """
    Computes the reconstruction error for given data using the trained Autoencod

    Args:
        autoencoder (Model): Trained Autoencoder model.
        data (DataFrame): Data to evaluate.

    Returns:
        np.array: Reconstruction errors.
    """
    reconstructed_data = autoencoder.predict(data)
    errors = np.mean(np.square(data - reconstructed_data), axis=1)
    return errors
```

In [ ]:
```
reconstruction_errors = {}
```

In [ ]:
```
for machine, model in autoencoders.items():
    errors = compute_reconstruction_error(model, test_data[machine])
    reconstruction_errors[machine] = errors
    logging.info(f"✅ {machine} Reconstruction Error Computed.")
```

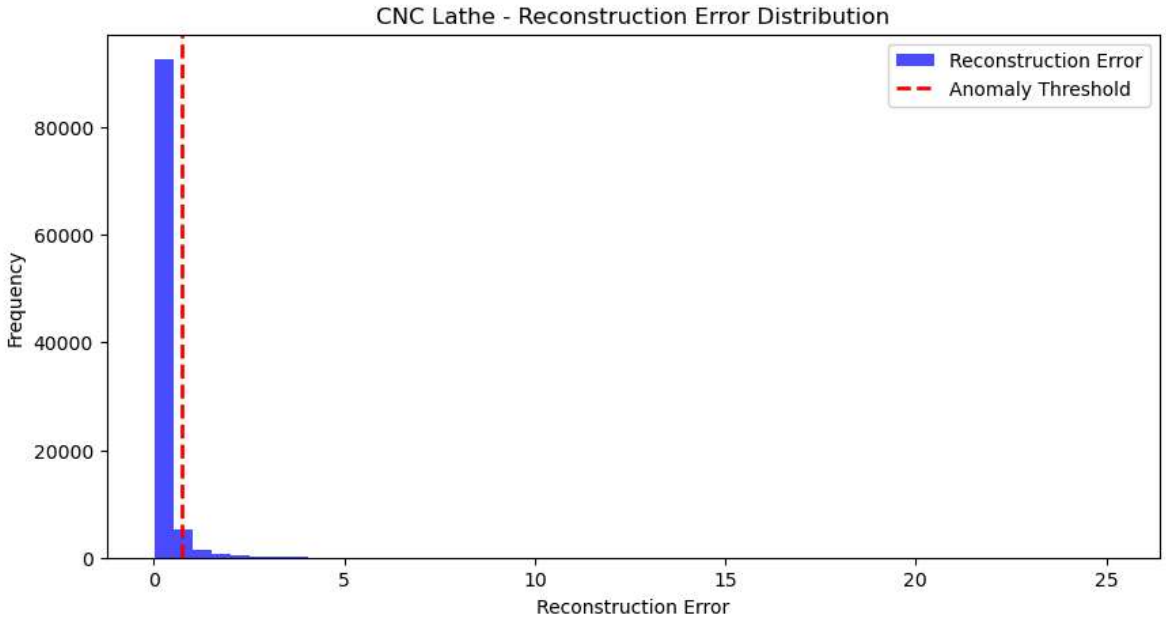
```
3156/3156 [==============================] - 4s 1ms/step
2025-02-21 17:49:21,781 - INFO - ✅ CNC Lathe Reconstruction Error Computed.
2081/2081 [==============================] - 3s 1ms/step
2025-02-21 17:49:25,311 - INFO - ✅ Saw Machine Reconstruction Error Computed.
```

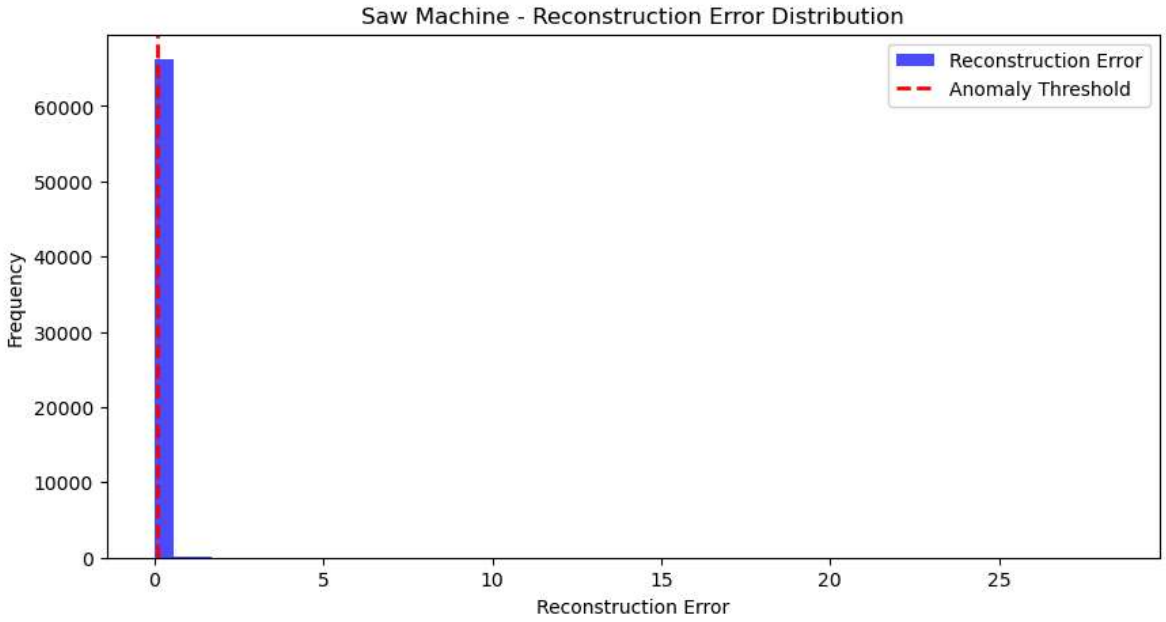
In [ ]:
```
anomaly_thresholds = {}
```

In [ ]:
```python
for machine, errors in reconstruction_errors.items():
    threshold = np.percentile(errors, 95)  # Set threshold at 95th percentile
    anomaly_thresholds[machine] = threshold
    logging.info(f"✅ {machine} - Anomaly Threshold: {threshold:.4f}")
```

```
2025-02-21 17:49:35,127 - INFO - ✅ CNC Lathe - Anomaly Threshold: 0.7425
2025-02-21 17:49:35,129 - INFO - ✅ Saw Machine - Anomaly Threshold: 0.1011
2025-02-21 17:49:35,129 - INFO - ✅ Saw Machine - Anomaly Threshold: 0.1011
```

In [ ]:
```python
for machine, errors in reconstruction_errors.items():
    plt.figure(figsize=(10, 5))
    plt.hist(errors, bins=50, alpha=0.7, color="blue", label="Reconstruction Err
    plt.axvline(anomaly_thresholds[machine], color="red", linestyle="dashed", li
    plt.title(f"{machine} - Reconstruction Error Distribution")
    plt.xlabel("Reconstruction Error")
    plt.ylabel("Frequency")
    plt.legend()
    plt.show()
```




In [ ]:
```python
import os
```

In [ ]:
```
import joblib
```

In [ ]:
```
from tensorflow.keras.models import load_model
```

In [ ]:
```
cnc_lathe_autoencoder_path = r"C:\DT_Projects\Unified-AI-Pipeline-for-Predictive
```

In [ ]:
```
saw_machine_autoencoder_path = r"C:\DT_Projects\Unified-AI-Pipeline-for-Predicti
```

In [ ]:
```
pca_model_path = r"C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maintenance
```

In [ ]:
```
cnc_lathe_pca_path = r"C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Mainten
```

In [ ]:
```
saw_machine_pca_path = r"C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Maint
```

In [ ]:
```
if os.path.exists(pca_model_path):
    pca = joblib.load(pca_model_path)
    print(f"✅ PCA Model loaded from: {pca_model_path}")
else:
    raise FileNotFoundError(f"❌ PCA Model not found at {pca_model_path}. Ensur
```

```
✅ PCA Model loaded from: C:\DT_Projects\Unified-AI-Pipeline-for-Predictive-Main
tenance-and-Optimization\data\models\pca_model.pkl
```

In [ ]:
```
if os.path.exists(cnc_lathe_autoencoder_path):
    cnc_lathe_autoencoder = load_model(cnc_lathe_autoencoder_path)
    print(f"✅ CNC Lathe Autoencoder loaded from: {cnc_lathe_autoencoder_path}"
else:
    raise FileNotFoundError(f"❌ CNC Lathe Autoencoder not found at {cnc_lathe_
```

```
✅ CNC Lathe Autoencoder loaded from: C:\DT_Projects\Unified-AI-Pipeline-for-Pre
dictive-Maintenance-and-Optimization\data\models\cnc_lathe_autoencoder.h5
```

In [ ]:
```
if os.path.exists(saw_machine_autoencoder_path):
    saw_machine_autoencoder = load_model(saw_machine_autoencoder_path)
    print(f"✅ Saw Machine Autoencoder loaded from: {saw_machine_autoencoder_pa
else:
    raise FileNotFoundError(f"❌ Saw Machine Autoencoder not found at {saw_machi
```

```
✅ Saw Machine Autoencoder loaded from: C:\DT_Projects\Unified-AI-Pipeline-for-P
redictive-Maintenance-and-Optimization\data\models\saw_machine_autoencoder.h5
```

In [ ]:
```
cnc_lathe_data = pd.read_csv(cnc_lathe_pca_path)
```

In [ ]:
```
saw_machine_data = pd.read_csv(saw_machine_pca_path)
```

In [ ]:
```
print(f"✅ CNC Lathe Data Sample: {cnc_lathe_data.head()}")
```

```
✅ CNC Lathe Data Sample:         PC1       PC2       PC3       PC4       PC5
PC6       PC7  \
0  0.261182 -0.975722 -1.929249  0.066523 -0.470612  0.496826 -1.530492
1  0.280148 -0.998906 -1.953667  0.078396 -0.481807  0.500035 -1.565720
2  0.274532 -0.983555 -1.925680  0.071538 -0.492495  0.514308 -1.538667
3  0.246902 -0.937415 -1.938151  0.041859 -0.442896  0.528168 -1.597195
4  0.278999 -1.001752 -1.942801  0.074128 -0.459747  0.488301 -1.550343

        PC8       PC9      PC10  ...      PC33      PC34      PC35      PC36  \
0  0.399509  0.835874  1.312902  ... -0.603569  0.442554  0.554955  0.475567
1  0.396357  0.875425  1.298801  ... -0.632058  0.325642  0.563391  0.449859
2  0.388290  0.873615  1.330154  ... -0.609400  0.320388  0.597946  0.500142
3  0.389199  0.816585  1.404128  ... -0.634105  0.291125  0.554631  0.481842
4  0.396266  0.853641  1.316136  ... -0.617085  0.447383  0.544893  0.468139

       PC37      PC38      PC39      PC40      PC41      PC42
0 -1.354336  0.592612  0.669831 -0.444297  0.370744  0.396624
1 -1.313367  0.618624  0.742100 -0.500464  0.353413  0.400814
2 -1.305578  0.615833  0.699546 -0.466529  0.363719  0.412003
3 -1.326061  0.587865  0.784517 -0.477305  0.362344  0.422620
4 -1.349190  0.592757  0.716403 -0.458933  0.359346  0.380145

[5 rows x 42 columns]
```

In [ ]:
```
print(f"✅ Saw Machine Data Sample: {saw_machine_data.head()}")
```

```
✅ Saw Machine Data Sample:         PC1       PC2       PC3       PC4       PC5
PC6       PC7  \
0 -3.593045  2.884776 -1.790855 -2.120426  4.703888  1.948979 -0.908472
1 -3.601571  2.998160 -1.805942 -2.195014  4.968151  2.183887 -1.002065
2 -3.302730  3.067900 -1.850242 -2.093826  5.291424  2.503570 -1.098934
3 -3.253954  3.163248 -1.904120 -2.108082  5.587161  2.737550 -1.134859
4 -3.242623  3.249466 -1.916517 -2.176520  5.888521  2.995926 -1.286179

        PC8       PC9      PC10      PC11      PC12      PC13      PC14  \
0 -0.283643  0.403161 -1.131652  0.525740  0.286031  0.163929  0.885987
1 -0.291229  0.474308 -1.231371  0.575894  0.282376  0.182078  1.369901
2 -0.263912  0.617734 -1.424324  0.302506  0.770353  0.220993  1.706323
3 -0.263262  0.740620 -1.842532  0.390369  0.687253  0.237102  2.129245
4 -0.271046  0.821613 -2.092233  0.276958  0.989528  0.281410  2.498338

       PC15      PC16      PC17      PC18      PC19      PC20
0 -0.277021  0.326683  0.966277 -1.006317 -0.351964  0.038489
1 -0.438919  0.363793  1.280161 -1.259660 -0.362477 -0.006834
2 -0.606445  0.451088  1.568369 -1.493226 -0.345200  0.016593
3 -0.707581  0.455030  1.861539 -1.719616 -0.282411  0.033838
4 -0.865171  0.478298  2.161164 -1.996368 -0.234336 -0.008532
```

In [ ]:
```
print(f"✅ CNC Lathe Data Shape: {cnc_lathe_data.shape}")
```

```
✅ CNC Lathe Data Shape: (504826, 42)
```

In [ ]:
```
print(f"✅ Saw Machine Data Shape: {saw_machine_data.shape}")
```

```
✅ Saw Machine Data Shape: (332847, 20)
```

In [ ]:
```
cnc_expected_shape = cnc_lathe_autoencoder.input_shape[1]
```

In [ ]:
```
saw_expected_shape = saw_machine_autoencoder.input_shape[1]
```

In [ ]:
```
print(f"✅ CNC Lathe Autoencoder Expected Input Shape: {cnc_expected_shape}")
```

```
✅ CNC Lathe Autoencoder Expected Input Shape: 42
```

In [ ]:
```
print(f"✅ Saw Machine Autoencoder Expected Input Shape: {saw_expected_shape}")
```

```
✅ Saw Machine Autoencoder Expected Input Shape: 20
```

In [ ]:
```
if cnc_lathe_data.shape[1] != cnc_expected_shape:
    print("⚠️ Transforming CNC Lathe data using PCA...")
    cnc_lathe_data = pca.transform(cnc_lathe_data)
```

In [ ]:
```
if saw_machine_data.shape[1] != saw_expected_shape:
    print("⚠️ Transforming Saw Machine data using PCA...")
    saw_machine_data = pca.transform(saw_machine_data)
```

In [ ]:
```
cnc_lathe_reconstructed = cnc_lathe_autoencoder.predict(cnc_lathe_data)
```

```
15776/15776 [==============================] - 19s 1ms/step
```

In [ ]:
```
saw_machine_reconstructed = saw_machine_autoencoder.predict(saw_machine_data)
```

```
10402/10402 [==============================] - 13s 1ms/step
```

In [ ]:
```
cnc_lathe_errors = np.abs(cnc_lathe_data - cnc_lathe_reconstructed)
```

In [ ]:
```
saw_machine_errors = np.abs(saw_machine_data - saw_machine_reconstructed)
```

In [ ]:
```
cnc_lathe_mean_error = np.mean(cnc_lathe_errors, axis=1)
```

In [ ]:
```
saw_machine_mean_error = np.mean(saw_machine_errors, axis=1)
```

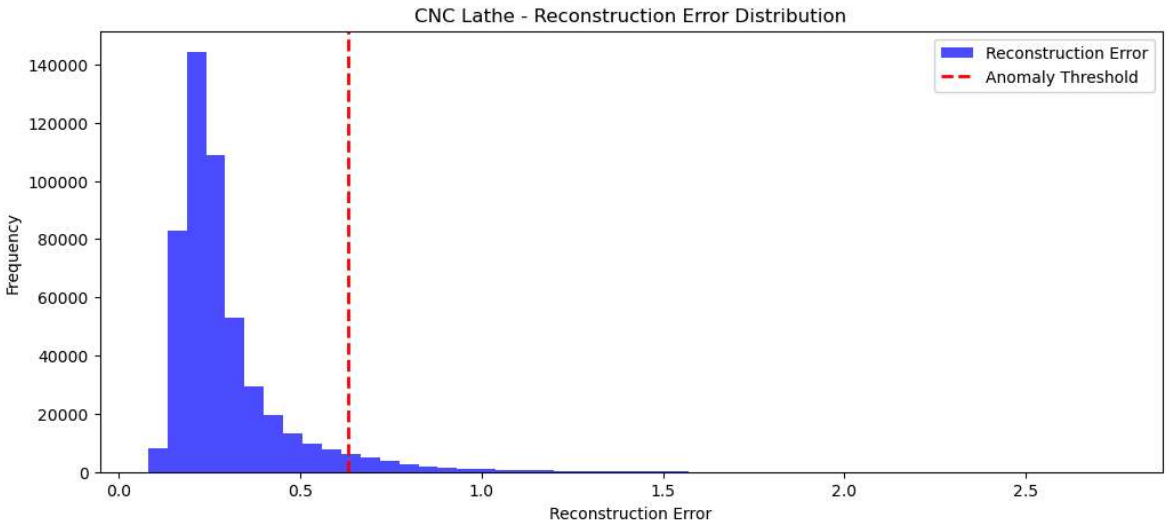
In [ ]:
```
static_threshold_cnc = np.percentile(cnc_lathe_mean_error, 95)
```

In [ ]:
```
static_threshold_saw = np.percentile(saw_machine_mean_error, 95)
```

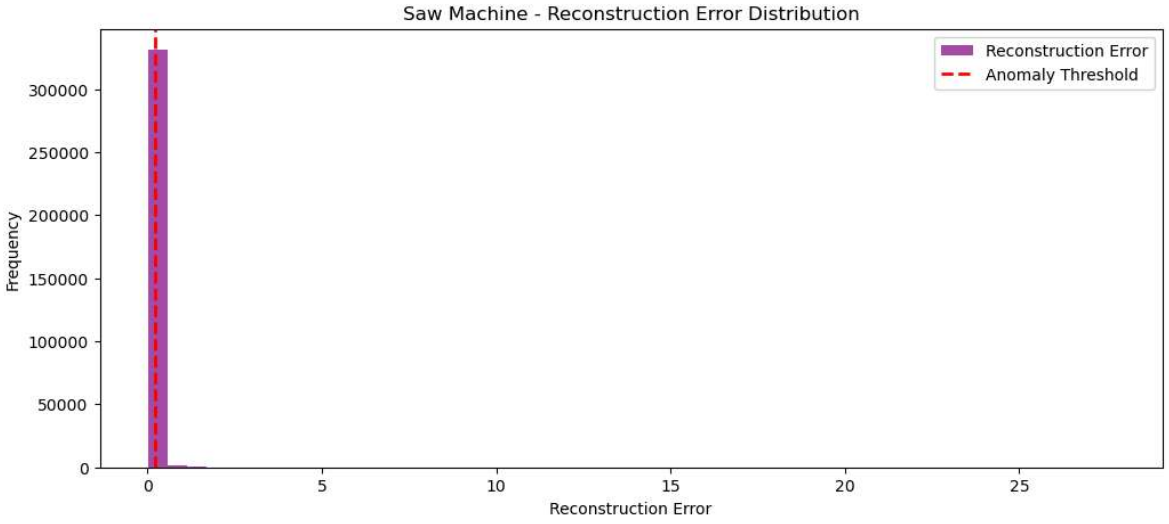
In [ ]:
```
cnc_anomalies = cnc_lathe_mean_error > static_threshold_cnc
```

In [ ]:
```
saw_anomalies = saw_machine_mean_error > static_threshold_saw
```

In [ ]:
```
plt.figure(figsize=(12, 5))
plt.hist(cnc_lathe_mean_error, bins=50, alpha=0.7, color='blue', label='Reconstr
plt.axvline(static_threshold_cnc, color='red', linestyle='dashed', linewidth=2,
plt.xlabel("Reconstruction Error")
plt.ylabel("Frequency")
plt.title("CNC Lathe - Reconstruction Error Distribution")
plt.legend()
plt.show()
```




In [ ]:
```
plt.figure(figsize=(12, 5))
plt.hist(saw_machine_mean_error, bins=50, alpha=0.7, color='purple', label='Recc
plt.axvline(static_threshold_saw, color='red', linestyle='dashed', linewidth=2,
plt.xlabel("Reconstruction Error")
plt.ylabel("Frequency")
plt.title("Saw Machine - Reconstruction Error Distribution")
plt.legend()
plt.show()
```




In [ ]:
```
print(f" 🔹 CNC Lathe: {sum(cnc_anomalies)} anomalies detected out of {len(cnc_l
```

🔹 CNC Lathe: 25242 anomalies detected out of 504826 data points.

In [ ]:
```
print(f" 🔹 Saw Machine: {sum(saw_anomalies)} anomalies detected out of {len(saw_
```

🔹 Saw Machine: 16642 anomalies detected out of 332847 data points.

In [ ]:
```
print("✅ Static Threshold-Based Anomaly Detection Completed!")
```

✅ Static Threshold-Based Anomaly Detection Completed!

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [ ]:
```
from sklearn.model_selection import train_test_split
```

In [ ]:
```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_sc
```

In [ ]:
```
cnc_lathe_labels = (cnc_lathe_mean_error > static_threshold_cnc).astype(int)  #
```

In [ ]:
```
saw_machine_labels = (saw_machine_mean_error > static_threshold_saw).astype(int)
```

In [ ]:
```
cnc_lathe_df = pd.DataFrame({'Reconstruction_Error': cnc_lathe_mean_error, 'Labe
```

In [ ]:
```
saw_machine_df = pd.DataFrame({'Reconstruction_Error': saw_machine_mean_error, '
```

In [ ]:
```
combined_df = pd.concat([cnc_lathe_df, saw_machine_df], axis=0).reset_index(drop
```

In [ ]:
```
combined_df.head()
```

Out[ ]:

| | Reconstruction_Error | Label |
|---|---|---|
| **0** | 0.385345 | 0 |
| **1** | 0.383928 | 0 |
| **2** | 0.383772 | 0 |
| **3** | 0.459545 | 0 |
| **4** | 0.389994 | 0 |

In [ ]:
```
combined_df.tail()
```

Out[ ]:

| | Reconstruction_Error | Label |
|---|---|---|
| **837668** | 0.536293 | 1 |
| **837669** | 0.462566 | 1 |
| **837670** | 0.572986 | 1 |
| **837671** | 0.461601 | 1 |
| **837672** | 0.443066 | 1 |

In [ ]:
```
combined_df.shape
```

Out[ ]:
```
(837673, 2)
```

In [ ]:
```
X = combined_df[['Reconstruction_Error']]
```

In [ ]:
```
y = combined_df['Label']
```

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

In [ ]:
```
print(f"✅ Training Data Shape: {X_train.shape}, Testing Data Shape: {X_test.sh
```

```
✅ Training Data Shape: (670138, 1), Testing Data Shape: (167535, 1)
```

In [ ]:
```
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
```

In [ ]:
```
rf_classifier.fit(X_train, y_train)
```

Out[ ]:

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [ ]:
```
y_pred = rf_classifier.predict(X_test)
```

In [ ]:
```
print(" ◆ Classification Report (Random Forest Classifier):")
```

◆ Classification Report (Random Forest Classifier):

In [ ]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99    159158
           1       0.80      0.85      0.82      8377

    accuracy                           0.98    167535
   macro avg       0.89      0.92      0.91    167535
weighted avg       0.98      0.98      0.98    167535
```

In [ ]:
```
print(" ◆ Confusion Matrix:")
```

◆ Confusion Matrix:

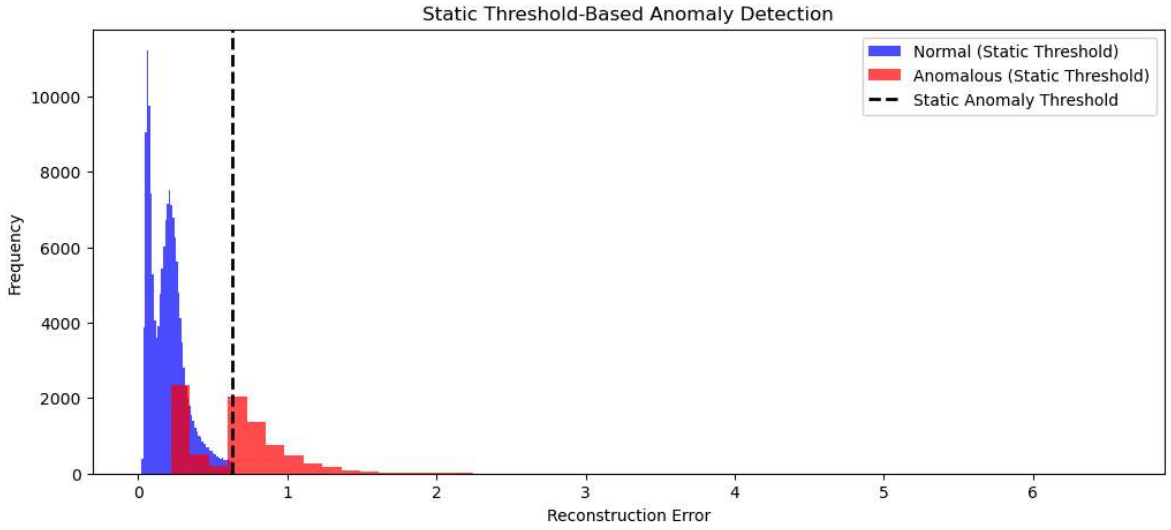
In [ ]:
```
print(confusion_matrix(y_test, y_pred))
```

```
[[157335   1823]
 [  1228   7149]]
```

In [ ]:
```
print(f" ◆ Accuracy: {accuracy_score(y_test, y_pred) * 100:.2f}%")
```

◆ Accuracy: 98.18%

In [ ]:
```
plt.figure(figsize=(12, 5))

# Static Thresholding
plt.hist(X_test[y_test == 0]['Reconstruction_Error'], bins=50, alpha=0.7, label=
plt.hist(X_test[y_test == 1]['Reconstruction_Error'], bins=50, alpha=0.7, label=
plt.axvline(static_threshold_cnc, color='black', linestyle='dashed', linewidth=2
plt.xlabel("Reconstruction Error")
plt.ylabel("Frequency")
plt.title("Static Threshold-Based Anomaly Detection")
plt.legend()
plt.show()
```




In [ ]:
```
plt.figure(figsize=(12, 5))

# ML-Based Thresholding
plt.hist(X_test[y_pred == 0]['Reconstruction_Error'], bins=50, alpha=0.7, label=
plt.hist(X_test[y_pred == 1]['Reconstruction_Error'], bins=50, alpha=0.7, label=
plt.xlabel("Reconstruction Error")
plt.ylabel("Frequency")
plt.title("ML-Based Anomaly Detection (Random Forest)")
plt.legend()
plt.show()
```