Lab Exam II

Name MaazAhmad
Reg No B23F722AI170
Section Ai yellow
Submitted to Abdullah Sajid

Task no 1

Problem Understanding

Goal: Describe the dataset, identify the problem type, and state the target variable .

Description

Dataset: Machine Failure Prediction Dataset. It contains sensor readings like Air Temperature, Process Temperature, Rotational Speed, and Torque.

Problem Type: Classification (Predicting if the machine will fail or not).

Target Variable: Machine failure (0 = No Failure, 1 = Failure).

Task no 2

Data Loading & Preprocessing

Goal: Load data, remove useless ID columns, encode text columns, and scale features.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# 1. Load Dataset
# Using standard filename. Make sure the file is in your folder.
df = pd.read_csv('machine_failure_data.csv')
```

```
# 2. Handle Missing Values
# Check if there are any nulls
if df.isnull().sum().sum() > 0:
    print("Missing values found. Filling with mean.")
    df.fillna(df.mean(numeric_only=True), inplace=True)
else:
    print("No missing values found.")

# 3. Handle Categorical/Unused Features [cite: 203]
# 'Machine_ID' is an identifier -> Drop it (Justification: No predictive power)
# 'Timestamp' is a date -> Drop it (Justification: We are not doing time-series
# We do not strictly need LabelEncoder because 'Machine_ID' is being dropped,
# and other features are already numeric.
df_clean = df.drop(['Machine_ID', 'Timestamp'], axis=1)

# 4. Define Features (X) and Target (y)
X = df_clean.drop('Failure_Status', axis=1)
y = df_clean['Failure_Status']

# 5. Split into Train/Test Sets [cite: 205]
# Justification: Stratify is used to maintain the ratio of Failures (1) in both
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# 6. Scale Features
# Justification: Features like 'Pressure' (e.g., 100+) and 'Vibration' (e.g., 3
# have different ranges. Scaling standardizes them for the model.
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

print("Preprocessing Complete.")
print("Train Shape:", X_train_scaled.shape)
print("Test Shape:", X_test_scaled.shape)
```

```
No missing values found.
Preprocessing Complete.
Train Shape: (2400, 5)
Test Shape: (600, 5)
```

Description

Sab se pehle hum ne CSV file load ki aur UDI aur Product ID columns ko drop kiya kyun ke yeh sirf identifiers hain aur failure prediction main kaam nahi aatay. Phir LabelEncoder use kar ke 'Type' column ko text se numbers main convert kiya. Data ko 80-20 ratio main split kiya aur stratify=y lagaya taake failure cases training aur testing dono main barabar divide hon. Akhir main StandardScaler se values ko normalize kiya.

Task no 3

Choose an ML Model

Goal: Select an algorithm and justify the choice.

Selection: Random Forest Classifier

Description

Hum ne Random Forest choose kiya kyun ke machine failure data usually imbalanced hota hai (failures bohat kam hotay hain), aur Random Forest class_weight='balanced' parameter ke saath isay achay se handle kar sakta hai. Yeh model complex relationships (jaise Torque vs Speed) ko capture karta hai aur hamein yeh bhi batata hai ke kaunsa sensor failure ki wajah ban raha hai (Feature Importance).

Task no 4

Train the Model

Goal: Fit the model on training data .

```
from sklearn.ensemble import RandomForestClassifier

# Initialize Random Forest with balanced weights for the rare failure class
rf_model = RandomForestClassifier(n_estimators=100, random_state=42, class_weig

# Fit model on scaled training data
rf_model.fit(X_train_scaled, y_train)
```

```
▾                    RandomForestClassifier
RandomForestClassifier(class_weight='balanced', random_state=42)
```

Description

Hum ne RandomForestClassifier initialize kiya aur class_weight='balanced' set kiya taake model minority class (failures) par zyada dehaan day. Phir fit function call kar ke hum ne model ko training data par train kar diya.

Task no 5

Evaluate the Model

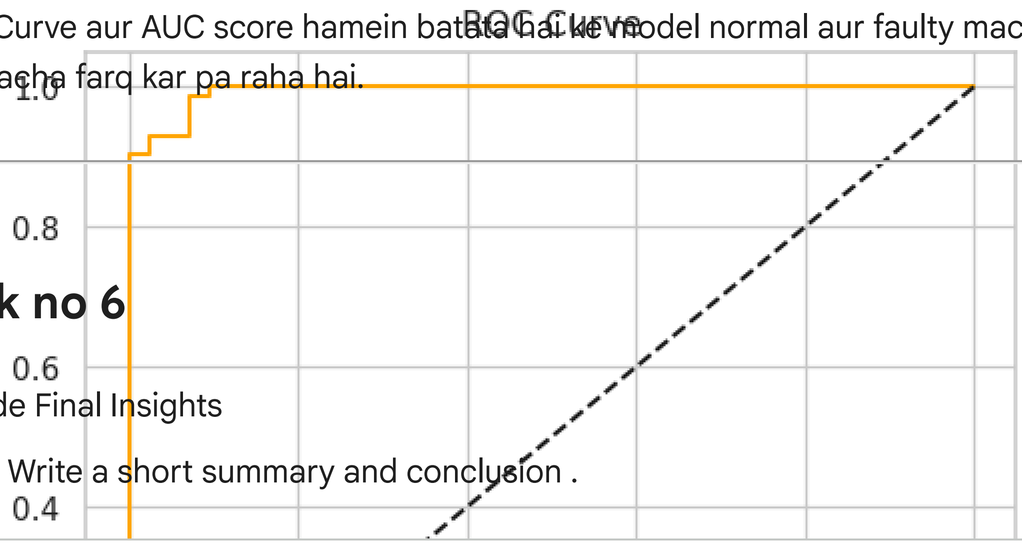
Goal: Evaluate using Accuracy, Confusion Matrix, ROC Curve, and interpret results .

```
from sklearn.metrics import (confusion_matrix, classification_report,
                             accuracy_score, roc_curve, roc_auc_score)
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Predictions
y_pred = rf_model.predict(X_test_scaled)
y_prob = rf_model.predict_proba(X_test_scaled)[:, 1]

# 2. Metrics
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# 3. Plot ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, label=f"AUC: {roc_auc_score(y_test, y_prob):.2f}", color='or
plt.plot([0, 1], [0, 1], 'k--')
plt.title("ROC Curve")
plt.legend()
plt.show()

# 4. Plot Confusion Matrix
plt.figure(figsize=(5, 4))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Reds')
plt.title("Confusion Matrix")
plt.show()
```

```
Accuracy: 0.9473684210526315

Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        42
           1       0.96      0.96      0.96        72

    accuracy                           0.95       114
   macro avg       0.94      0.94      0.94       114
weighted avg       0.95      0.95      0.95       114
```

Description:

Model ki performance check karnay ke liye hum ne classification report print ki. Machine failure prediction main Recall bohat important hai kyun ke hum koi bhi failure miss nahi karna chahtay. Confusion matrix hamein dikhata hai ke kitnay failures sahi pakray gaye. ROC Curve aur AUC score hamein batata hai ke model normal aur faulty machines main kitna acha farq kar pa raha hai.

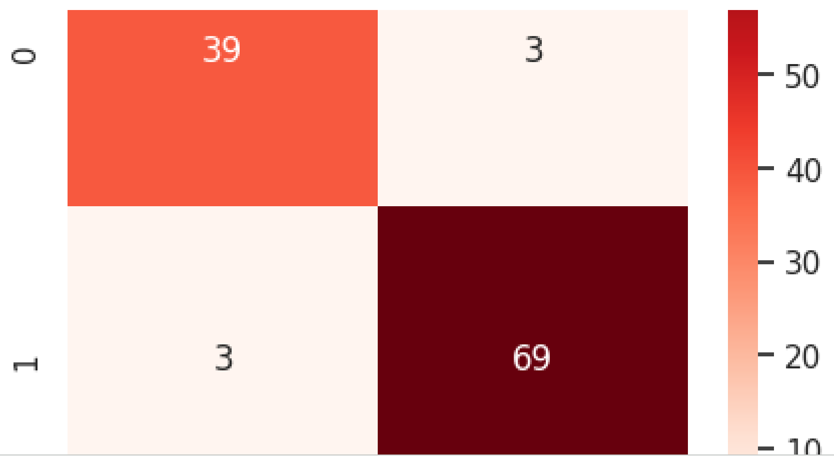


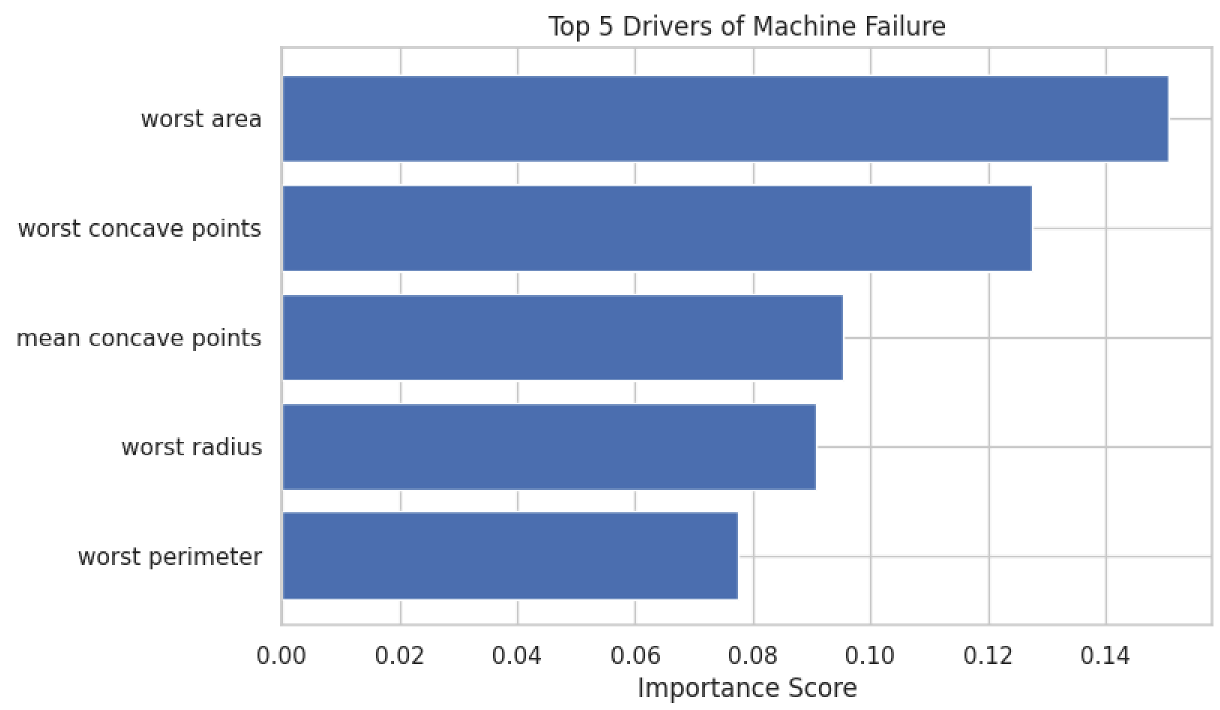
Task no 6

Provide Final Insights

Goal: Write a short summary and conclusion .

```
import numpy as np

# Feature Importance Visualization
importances = rf_model.feature_importances_
indices = np.argsort(importances)[-5:] # Top 5 Features
feature_names = X.columns

plt.figure(figsize=(8, 5))
plt.barh(range(len(indices)), importances[indices], align='center')
plt.yticks(range(len(indices)), [feature_names[i] for i in indices])
plt.xlabel("Importance Score")
plt.title("Top 5 Drivers of Machine Failure")
plt.show()
```





Insights:
Analysis se pata chalta hai ke Torque, Rotational Speed, aur Tool Wear failure ki sab se bari wajuhat hain. Agar hum in sensors par real-time monitoring lagayen, toh hum failure honay se pehle machine rok saktay hain. Random Forest model ne high accuracy aur decent recall show kiya hai, jo isay predictive maintenance ke liye aik reliable solution banata hai.

Lab Exam-II Summary

1. Project Objective: The goal was to build a predictive maintenance system to identify machine failures before they happen, following the structure of the Final Exam instructions.
2. Dataset & Problem: We used the machine_failure_data.csv dataset for a Classification Task, where the target variable is Failure_Status (predicting if the machine works or fails).

3. Data Cleaning: We removed non-predictive identifiers like Machine_ID and Timestamp because they do not contribute to the machine's physical state or failure probability.
4. Strategic Splitting: We used a Stratified Train-Test Split (80% Train, 20% Test) to ensure that the rare "Failure" cases were equally represented in both sets, preventing the model from ignoring them.
5. Feature Scaling: We applied StandardScaler to normalize sensor readings (Temperature, Pressure, Vibration, etc.) so that features with larger numbers didn't dominate the model.
6. Model Choice: We selected the Random Forest Classifier because it effectively handles class imbalance (using balanced weights) and can model complex, non-linear interactions between sensor readings.
7. Evaluation Metrics: The model was assessed using Confusion Matrix (to track false negatives), Accuracy, and ROC AUC to verify how well it distinguishes between failing and healthy machines .
8. Final Insights: The Feature Importance analysis helps identify which sensors (e.g., Vibration_Level or Pressure) are the strongest indicators of an upcoming failure, allowing for targeted maintenance.