

```
# Install required libraries
!pip install xgboost

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, roc_curve
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from xgboost import XGBClassifier
```

Requirement already satisfied: xgboost in /usr/local/lib/python3.12/dist-packages (3.2.0)
 Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (from xgboost) (2.0.2)
 Requirement already satisfied: nvidia-nccl-cu12 in /usr/local/lib/python3.12/dist-packages (from xgboost) (2.29.3)
 Requirement already satisfied: scipy in /usr/local/lib/python3.12/dist-packages (from xgboost) (1.16.3)

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

```
cp: cannot stat 'kaggle.json': No such file or directory
chmod: cannot access '/root/.kaggle/kaggle.json': No such file or directory
```

```
!kaggle datasets list -s smart
```

```
Traceback (most recent call last):
  File "/usr/local/bin/kaggle", line 4, in <module>
    from kaggle.cli import main
  File "/usr/local/lib/python3.12/dist-packages/kaggle/__init__.py", line 6, in <module>
    api.authenticate()
  File "/usr/local/lib/python3.12/dist-packages/kaggle/api/kaggle_api_extended.py", line 434, in authenticate
    raise IOError('Could not find {}. Make sure it\'s located in'
OSError: Could not find kaggle.json. Make sure it's located in /root/.kaggle. Or use the environment method. See setup instr
```

```
!kaggle datasets download -d backblaze/hard-drive-test-data
```

```
Traceback (most recent call last):
  File "/usr/local/bin/kaggle", line 10, in <module>
    sys.exit(main())
    ^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/kaggle/cli.py", line 68, in main
    out = args.func(**command_args)
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/kaggle/api/kaggle_api_extended.py", line 1741, in dataset_download_cli
    with self.build_kaggle_client() as kaggle:
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/kaggle/api/kaggle_api_extended.py", line 688, in build_kaggle_client
    username=self.config_values['username'],
    ~~~~~~^~~~~~
KeyError: 'username'
```

```
!unzip hard-drive-test-data.zip
```

```
unzip: cannot find or open hard-drive-test-data.zip, hard-drive-test-data.zip.zip or hard-drive-test-data.zip.ZIP.
```

```
import os
os.listdir()
```

```
['.config', 'sample_data']
```

```
import os
os.listdir()
```

```
['.config', 'sample_data']
```

```
!unzip hard-drive-test-data.zip
```

```
unzip: cannot find or open hard-drive-test-data.zip, hard-drive-test-data.zip.zip or hard-drive-test-data.zip.ZIP.
```

```
import os
os.listdir()
```

```
['.config', 'sample_data']
```

```
import pandas as pd
```

```
df = pd.read_csv('harddrive.csv')
df.head()
```

```
-----
FileNotFoundError                                Traceback (most recent call last)
/tmp/ipython-input-30530380.py in <cell line: 0>()
      1 import pandas as pd
      2
----> 3 df = pd.read_csv('harddrive.csv')
      4 df.head()
```

```
----- 4 frames -----
/usr/local/lib/python3.12/dist-packages/pandas/io/common.py in get_handle(path_or_buf, mode, encoding, compression,
memory_map, is_text, errors, storage_options)
      871         if ioargs.encoding and "b" not in ioargs.mode:
      872             # Encoding
--> 873             handle = open(
      874                 handle,
      875                 ioargs.mode,
```

```
FileNotFoundError: [Errno 2] No such file or directory: 'harddrive.csv'
```

Next steps: [Explain error](#)

```
df = df.drop(['date', 'serial_number', 'model'], axis=1)
```

```
df.isnull().sum().sort_values(ascending=False).head(10)
```

```
threshold = 0.8

missing_ratio = df.isnull().mean()

cols_to_drop = missing_ratio[missing_ratio > threshold].index

df = df.drop(columns=cols_to_drop)

print("Dropped columns:", len(cols_to_drop))
print("Remaining columns:", df.shape[1])
```

```
df = df.fillna(0)
```

```
df.isnull().sum().sum()
```

```
df['failure'].value_counts(normalize=True)
```

```
df_sample = df.sample(n=200000, random_state=42)
```

```
df_sample['failure'].value_counts()
```

```
# فصل الفئات
df_majority = df[df['failure'] == 0]
df_minority = df[df['failure'] == 1]
```

```
print(len(df_minority))
```

```
df_majority_sample = df_majority.sample(n=50000, random_state=42)

df_balanced = pd.concat([df_majority_sample, df_minority])

df_balanced = df_balanced.sample(frac=1, random_state=42)

df_balanced['failure'].value_counts()
```

```
from sklearn.model_selection import train_test_split

X = df_balanced.drop('failure', axis=1)
y = df_balanced['failure']

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

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(
    n_estimators=100,
    random_state=42,
    class_weight='balanced' # مهم جدًا هنا
)

rf.fit(X_train, y_train)

y_pred = rf.predict(X_test)
```

```
from sklearn.metrics import classification_report, confusion_matrix

print("Classification Report:")
print(classification_report(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

y_prob = rf.predict_proba(X_test)[:,1]

fpr, tpr, thresholds = roc_curve(y_test, y_prob)

plt.plot(fpr, tpr)
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve - Random Forest")
plt.show()

print("AUC Score:", roc_auc_score(y_test, y_prob))
```

```
import numpy as np
from sklearn.metrics import classification_report

# خذ الاحتمالات
y_prob = rf.predict_proba(X_test)[:,1]

# إلى 0.2 مثلاً threshold غير
threshold = 0.2
y_pred_new = (y_prob > threshold).astype(int)

print(classification_report(y_test, y_pred_new))
```

```
threshold = 0.1
y_pred_new = (y_prob > threshold).astype(int)

print(classification_report(y_test, y_pred_new))
```

```
from xgboost import XGBClassifier

xgb = XGBClassifier(
    n_estimators=200,
    max_depth=6,
    learning_rate=0.1,
    scale_pos_weight = len(df_majority_sample) / len(df_minority),
    random_state=42
)

xgb.fit(X_train, y_train)

y_prob_xgb = xgb.predict_proba(X_test)[:,-1]

threshold = 0.2
y_pred_xgb = (y_prob_xgb > threshold).astype(int)

from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred_xgb))
```

```
from sklearn.metrics import roc_auc_score, roc_curve
import matplotlib.pyplot as plt

auc_xgb = roc_auc_score(y_test, y_prob_xgb)
print("XGBoost AUC:", auc_xgb)
```

```
fpr, tpr, thresholds = roc_curve(y_test, y_prob_xgb)

optimal_idx = (tpr - fpr).argmax()
optimal_threshold = thresholds[optimal_idx]

print("Best Threshold:", optimal_threshold)
```

```
optimal_threshold = 0.0149

y_pred_opt = (y_prob_xgb > optimal_threshold).astype(int)

from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred_opt))
```

```
import pandas as pd
import matplotlib.pyplot as plt

feature_importance = pd.Series(
    xgb.feature_importances_,
    index=X_train.columns
).sort_values(ascending=False)

feature_importance.head(15)
```

