# If packages are missing, uncomment the pip lines and run once

# %pip install scikit-learn pandas matplotlib

import pandas as pd

import numpy as np

from sklearn.ensemble import IsolationForest

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

# Option A: Generate synthetic dataset (fast, use this if you don't have real data)

def gen\_synthetic(n=2000, anomaly\_frac=0.03, seed=42):

rng = np.random.RandomState(seed)

# normal behavior

face\_present = rng.binomial(1, 0.98, size=n)

multiple\_faces = rng.binomial(1, 0.01, size=n)

gaze\_away\_count = rng.poisson(1, size=n)

tab\_switch\_count = rng.poisson(0.2, size=n)

audio\_event\_count = rng.poisson(0.1, size=n)

motion\_score = rng.normal(0.2, 0.1, size=n) # low motion normally

df = pd.DataFrame({

"face\_present": face\_present,

"multiple\_faces": multiple\_faces,

"gaze\_away\_count": gaze\_away\_count,

"tab\_switch\_count": tab\_switch\_count,

"audio\_event\_count": audio\_event\_count,

"motion\_score": motion\_score

})

# Inject anomalies

k = max(1, int(n \* anomaly\_frac))

idx = rng.choice(n, k, replace=False)

df.loc[idx, "multiple\_faces"] = 1

df.loc[idx, "gaze\_away\_count"] += rng.poisson(10, size=k)

df.loc[idx, "tab\_switch\_count"] += rng.poisson(5, size=k)

df.loc[idx, "audio\_event\_count"] += rng.poisson(3, size=k)

df.loc[idx, "motion\_score"] += rng.normal(1.5, 0.5, size=k)

df["session\_id"] = ["sess\_" + str(i) for i in range(len(df))]

return df

df = gen\_synthetic(2000)

display(df.head())

features = ["face\_present","multiple\_faces","gaze\_away\_count","tab\_switch\_count","audio\_event\_count","motion\_score"]

X = df[features].fillna(0).values

scaler = StandardScaler()

Xs = scaler.fit\_transform(X)

clf = IsolationForest(n\_estimators=200, contamination=0.03, random\_state=42)

clf.fit(Xs)

# anomaly score: lower => more anomalous, sklearn has decision\_function (the higher, the less abnormal)

df["anomaly\_score\_raw"] = clf.decision\_function(Xs) # higher = more normal

df["suspicion\_score"] = -df["anomaly\_score\_raw"] # flip so higher = more suspicious

df["is\_anomaly"] = clf.predict(Xs) # -1 anomaly, 1 normal

df["is\_anomaly"] = df["is\_anomaly"].map({1:0, -1:1})

anoms = df.sort\_values("suspicion\_score", ascending=False).head(30)

print("Top anomalies (session\_id, suspicion\_score):")

display(anoms[["session\_id","suspicion\_score","is\_anomaly"] + features])

# Quick histogram of suspicion scores

plt.figure(figsize=(8,4))

plt.hist(df["suspicion\_score"], bins=50)

plt.title("Suspicion score distribution")

plt.xlabel("suspicion\_score (higher → more suspicious)")

plt.show()

out = anoms.copy()

out\_path = "/dbfs/FileStore/proctoring\_anomalies\_top30.csv"

out.to\_csv(out\_path, index=False)

print("Saved top anomalies to:", out\_path)

# In UI you can go to Data -> DBFS -> FileStore -> get the file, or use workspace file browser.