KNN

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import NearestNeighbors
# Scale the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df)
# Fit KNN
k = 3
nn = NearestNeighbors(n_neighbors=k)
nn.fit(X_scaled)
distances, indices = nn.kneighbors(X_scaled)

# Outlier score = distance to k-th neighbor
outlier_scores = distances[:, -1]
threshold = np.percentile(outlier_scores, 90)  # top 10% as outliers
outlier_labels = outlier_scores > threshold

# See actual outlier rows
df_knn = df.copy()
df_knn['knn_score'] = outlier_scores
df_knn['knn_outlier'] = outlier_labels.astype(int)
print("KNN Outliers:")
print(df_knn[df_knn[‘knn_outlier']==1])
```

DBSCAN

```
from sklearn.cluster import DBSCAN

# Scale the data (already done)
# X_scaled

# Fit DBSCAN
dbscan = DBSCAN(eps=1.5, min_samples=2)
dbscan.fit(X_scaled)

# Cluster labels: -1 = outlier
df_dbscan = df.copy()
df_dbscan['dbscan_label'] = dbscan.labels_
print("\nDBSCAN Outliers:")
print(df_dbscan[df_dbscan['dbscan_label']==-1])
```

LOF

```
from sklearn.neighbors import LocalOutlierFactor


lof = LocalOutlierFactor(n_neighbors=3,
contamination=0.2)
# fit_predict returns -1 for outlier, 1 for inlier
lof_labels = lof.fit_predict(X_scaled)
lof_scores = lof.negative_outlier_factor_

df_lof = df.copy()
df_lof['lof_label'] = lof_labels
df_lof['lof_score'] = lof_scores
print("\nLOF Outliers:")
print(df_lof[df_lof['lof_label']==-1])
```

Isolation Forest

```
from sklearn.ensemble import IsolationForest

# Fit Isolation Forest
iso = IsolationForest(n_estimators=200,
max_samples='auto', contamination=0.2,
random_state=42, bootstrap=True)
iso.fit(X_scaled)

iso_labels = iso.predict(X_scaled)  # -1 = outlier, 1 =
normal
iso_scores = iso.decision_function(X_scaled)

df_iso = df.copy()
df_iso['iso_label'] = iso_labels
df_iso['iso_score'] = iso_scores
print("\nIsolation Forest Outliers:")
print(df_iso[df_iso['iso_label']==-1])
```