# Part 6 – Anomaly Detection

The

## 1. Isolation Forests

# 1a. Open the tripadvisor\_reviews.csv file

import pandas as pd

df = pd.read\_csv('../Data/tripadvisor\_reviews.csv')

df.head()

# 1b. Remove the user\_id column

X = df.iloc[:, 1:]

X.head()

# 1c. View the min and max of each rating

X.describe()

# 2. Visualize the data using a seaborn pair plot

import seaborn as sns

sns.pairplot(X, height=2);

# 3. Fit an Isolation Forest model using a contamination of 1%

from sklearn.ensemble import IsolationForest

model = IsolationForest(contamination=0.01)

model.fit(X)

# view the labels

model.predict(X)[:255]

array([ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, -1, 1, 1, 1, 1, 1, 1, 1])

# 4. Visualize the anomalies on the seaborn pair plot

df['anomaly'] = model.predict(X)

df.head()

# add the anomalies as the hue on the pair plot

sns.pairplot(df, hue='anomaly', palette='bright', height=2);

# 5. Where do you notice anomalies in the pair plot?

## Some people really like nightlifre in our town, but rate other attractions lower

## Some people really like our museums

# 6. Modify the contamination to 0.5% to capture fewer anomalies and visualize the differences

model2 = IsolationForest(contamination=0.005)

model2.fit(X)

df['anomaly2'] = model2.predict(X)

df.sort\_values('anomaly2').head()

# add the anomalies as the hue on the pair plot

sns.pairplot(df.drop(columns='anomaly'), hue='anomaly2', palette='bright', height=2);

## 2. DBSCAN

# 1. Paste the DBSCAN function from the anomaly detection demo notebook that loops through multiple eps and min\_samples values to fit multiple DBSCAN models

import numpy as np

from sklearn.cluster import DBSCAN

from sklearn.metrics import silhouette\_score

def tune\_dbscan(data):

results = []

# define a range of eps and min\_samples values to loop through

eps\_values = np.arange(.1, 2, .1)

min\_samples\_values = np.arange(2, 10, 1)

# loop through the combinations of eps and min\_samples

for eps in eps\_values:

for min\_samples in min\_samples\_values:

dbscan = DBSCAN(eps=eps, min\_samples=min\_samples)

dbscan.fit(data)

labels = dbscan.labels\_

# count the number of clusters (excluding noise points labeled as -1)

n\_clusters = len(set(labels)) - (1 if -1 in labels else 0)

# count the number of noise points (labeled as -1)

n\_noise = list(labels).count(-1)

# calculate the silhouette score (excluding noise points)

if n\_clusters > 1: # silhouette score requires at least 2 clusters

silhouette = silhouette\_score(data, labels, metric='euclidean', sample\_size=None)

else:

silhouette = None

results.append([eps, min\_samples, n\_clusters, n\_noise, silhouette])

# put the results in a dataframe

dbscan\_results = pd.DataFrame(results, columns=["Eps", "Min Samples", "Number of Clusters",

"Number of Noise Points", "Silhouette Score"])

return dbscan\_results

# 2. Apply the function on the tourist rating data set

dbscan\_results = tune\_dbscan(X)

dbscan\_results.head()

# 3. Find the highest silhouette score and note down the eps and min\_samples values

dbscan\_results.sort\_values('Silhouette Score', ascending=False).head()

# 4. Fit a single DBSCAN model using those eps and min\_sample values

dbscan = DBSCAN(eps=0.8, min\_samples=2)

dbscan.fit(X)

# 5. Note the anomalies (-1) and visualize them on a pair plot

df['anomaly\_dbscan'] = dbscan.labels\_

df.head()

# view the anomalies

df[df['anomaly\_dbscan'] == -1]

# visualize the anomalies

sns.pairplot(df.drop(['anomaly', 'anomaly2'], axis=1), hue='anomaly\_dbscan', palette='bright', height=2);

# view the number of data points in each cluster

df.anomaly\_dbscan.value\_counts()

anomaly\_dbscan

0 971

-1 7

1 2

Name: count, dtype: int64