Code for Hierarchical Clustering to obtain labels (suitability category)

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
import scipy.cluster.hierarchy as sch

#Loading locational dataset
df = pd.read_csv('Normalised DACCS data.csv', index_col ='Location')

# Using the elbow method to determine the k value to be applied
k_rng = range(1, 10)
sse = []
for k in k_rng:
    km = KMeans(n_clusters=k)
    km.fit(df[['WA','GSP','LCEA','GNI']])
    sse.append(km.inertia_)

# Plot to obtain elbow
plt.xlabel('K')
plt.ylabel('Sum of squared error')
plt.plot(k_rng, sse)

# create clusters using obtained k value
hc = AgglomerativeClustering(n_clusters=5, affinity = 'euclidean', linkage = 'ward')



# Determining mean cluster characterisitics
```

```
y_hc = hc.fit_predict(df[['WA','GSP','LCEA','GNI']])
df['cluster'] = y_hc
df.sort_values("cluster", inplace = True, ascending=True)
df_cluster = df.groupby('cluster').mean()

#Heatmap of cluster characteristics
plt.figure(figsize=(8,6))
sns.heatmap(df_cluster, annot=True, cmap="Blues", linewidths=.5)

#To obtain the entire cluster labels
print(hc.labels_)
```

Supervised ML for suitability prediction (Random Forest)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (9, 6)
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

# Reading data
df = pd.read_csv("Labelled Normalised DACCS data.csv")

# creating input identity
sub_df = df[['WA','GSP','LCEA','GNI']]

X = sub_df
y = df.Category
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, random_state=1)
```

```
# creating Model (with 100 trees)
rf = RandomForestClassifier(n_estimators=100)

# Fitting training data
rf.fit(X_train, y_train)

# y prediction for set of data
y_pred = rf.predict(X_test)
print(y_pred)

# To get model performance
from sklearn.metrics import classification_report
cr = classification_report(y_test, y_pred)
print(cr)

# Determining feature importance (Model Interpretation)
feature_imp = pd.DataFrame(rf.feature_importances_,index= X_train.columns,
                    columns=['importance']).sort_values('importance',ascending=False)
print(feature_imp)

#Feature importance visualisation
feature_imp.plot(kind='barh')
```