```
import os

os.chdir("/Users/hunter/Desktop/QBio_490_HunterN/analysis_data")

import cptac

datasets = cptac.list_datasets()
print(datasets)
```

```
                                                Description     Data reuse
status  \
Dataset name

Brca                                        breast cancer       no
restrictions
Ccrcc         clear cell renal cell carcinoma (kidney)       no
restrictions
Colon                                  colorectal cancer       no
restrictions
Endometrial            endometrial carcinoma (uterine)       no
restrictions
Gbm                                         glioblastoma       no
restrictions
Hnscc           head and neck squamous cell carcinoma       no
restrictions
Lscc                       lung squamous cell carcinoma       no
restrictions
Luad                                lung adenocarcinoma       no
restrictions
Ovarian               high grade serous ovarian cancer       no
restrictions
Pdac                    pancreatic ductal adenocarcinoma       no
restrictions
UcecConf            endometrial confirmatory carcinoma  password
access only
GbmConf                        glioblastoma confirmatory  password
access only

                                          Publication link
Dataset name
Brca          https://pubmed.ncbi.nlm.nih.gov/33212010/
Ccrcc         https://pubmed.ncbi.nlm.nih.gov/31675502/
Colon         https://pubmed.ncbi.nlm.nih.gov/31031003/
Endometrial   https://pubmed.ncbi.nlm.nih.gov/32059776/
Gbm           https://pubmed.ncbi.nlm.nih.gov/33577785/
Hnscc         https://pubmed.ncbi.nlm.nih.gov/33417831/
Lscc          https://pubmed.ncbi.nlm.nih.gov/34358469/
Luad          https://pubmed.ncbi.nlm.nih.gov/32649874/
Ovarian       https://pubmed.ncbi.nlm.nih.gov/27372738/
Pdac          https://pubmed.ncbi.nlm.nih.gov/34534465/
```

```
UcecConf                                        unpublished
GbmConf                                         unpublished
```

```
cptac.download(dataset='ccrcc')
ccrcc = cptac.Ccrcc()
rna = ccrcc.get_transcriptomics()
clinical = ccrcc.get_clinical()
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import GaussianNB
protein = ccrcc.get_proteomics()

protein.columns = protein.columns.get_level_values(0)
protein = protein.dropna(axis=1)
rna = rna.dropna(axis=1)
```

```
stage_1 = np.where(clinical.loc[:,
'tumor_stage_pathological'].isin(['Stage I']), True, False)
stage_3 = np.where(clinical.loc[:,
'tumor_stage_pathological'].isin(['Stage III']), True, False)

difference = np.abs(protein.loc[stage_1, :].dropna(axis=1).mean() -
protein.loc[stage_3, :].dropna(axis=1).mean())

top_5_proteins = difference.sort_values(ascending=False)
[:5].index.unique()

stage_1_IDs = clinical.loc[stage_1, :].index
stage_3_IDs = clinical.loc[stage_3, :].index

rna = np.log2(rna)
```

```
/Users/hunter/anaconda3/lib/python3.11/site-packages/pandas/core/
internals/blocks.py:366: RuntimeWarning: divide by zero encountered in
log2
  result = func(self.values, **kwargs)
```

```
rna = rna.replace(-np.inf, np.nan).dropna(axis=1)

rna_id_1 = np.intersect1d(stage_1_IDs, rna.index)
rna_id_3 = np.intersect1d(stage_3_IDs, rna.index)
```

```
difference_rna = rna.loc[rna_id_1, :].dropna(axis=1).mean() -
rna.loc[rna_id_3, :].dropna(axis=1).mean()

top_5_rna = difference_rna.sort_values(ascending=False)
[:5].index.unique()

features = pd.DataFrame()

clinical_ID = clinical.loc[clinical.loc[:, 'Sample_Tumor_Normal'] ==
'Tumor', :].index

features = pd.concat([rna.loc[clinical_ID, top_5_rna],
protein.loc[clinical_ID, top_5_proteins]], axis=1)
features = features.dropna(axis=0)

y_data = clinical.loc[features.index,
'tumor_stage_pathological'].dropna()


scaler = StandardScaler()
scaled = scaler.fit_transform(features)
label_encoder = LabelEncoder()
target = label_encoder.fit_transform(y_data)
X_train, X_test, y_train, y_test = train_test_split(scaled, y_data,
train_size=0.7)

num_runs = 10
accuracies = {
    'KNeighborsClassifier': [],
    'DecisionTreeClassifier': [],
    'MLPClassifier': [],
    'GaussianNB': []
}

for run in range(num_runs):

    knn_model = KNeighborsClassifier()
    dt_model = DecisionTreeClassifier()
    mlp_model = MLPClassifier()
    nb_model = GaussianNB()

    knn_model.fit(X_train, y_train)
    dt_model.fit(X_train, y_train)
    mlp_model.fit(X_train, y_train)
    nb_model.fit(X_train, y_train)


    knn_accuracy = knn_model.score(X_test, y_test)
    dt_accuracy = dt_model.score(X_test, y_test)
    mlp_accuracy = mlp_model.score(X_test, y_test)
    nb_accuracy = nb_model.score(X_test, y_test)
```

```
    accuracies['KNeighborsClassifier'].append(knn_accuracy)
    accuracies['DecisionTreeClassifier'].append(dt_accuracy)
    accuracies['MLPClassifier'].append(mlp_accuracy)
    accuracies['GaussianNB'].append(nb_accuracy)

mean_accuracies = {model: np.mean(accuracies[model]) for model in
accuracies}
print("Mean Accuracies:")
for model, accuracy in mean_accuracies.items():
    print(f"{model}: {accuracy}")

best_model = max(mean_accuracies, key=mean_accuracies.get)
print(f"The best model is: {best_model}")
```

```
/Users/hunter/anaconda3/lib/python3.11/site-packages/sklearn/
neural_network/_multilayer_perceptron.py:691: ConvergenceWarning:
Stochastic Optimizer: Maximum iterations (200) reached and the
optimization hasn't converged yet.
  warnings.warn(
/Users/hunter/anaconda3/lib/python3.11/site-packages/sklearn/neural_ne
twork/_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (200) reached and the optimization
hasn't converged yet.
  warnings.warn(
/Users/hunter/anaconda3/lib/python3.11/site-packages/sklearn/neural_ne
twork/_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (200) reached and the optimization
hasn't converged yet.
  warnings.warn(
/Users/hunter/anaconda3/lib/python3.11/site-packages/sklearn/neural_ne
twork/_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (200) reached and the optimization
hasn't converged yet.
  warnings.warn(
/Users/hunter/anaconda3/lib/python3.11/site-packages/sklearn/neural_ne
twork/_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (200) reached and the optimization
hasn't converged yet.
  warnings.warn(
/Users/hunter/anaconda3/lib/python3.11/site-packages/sklearn/neural_ne
twork/_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (200) reached and the optimization
hasn't converged yet.
  warnings.warn(
/Users/hunter/anaconda3/lib/python3.11/site-packages/sklearn/neural_ne
twork/_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (200) reached and the optimization
hasn't converged yet.
  warnings.warn(
```

```
/Users/hunter/anaconda3/lib/python3.11/site-packages/sklearn/neural_network/_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
/Users/hunter/anaconda3/lib/python3.11/site-packages/sklearn/neural_network/_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

Mean Accuracies:
KNeighborsClassifier: 0.5757575757575759
DecisionTreeClassifier: 0.35454545454545455
MLPClassifier: 0.5515151515151515
GaussianNB: 0.6363636363636365
The best model is: GaussianNB

/Users/hunter/anaconda3/lib/python3.11/site-packages/sklearn/neural_network/_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
```