method

April 11, 2024

1 Method

In this part you can run my Dimension_Reduction step by step, or you can just run the **reduction.py**. Results are in `mds_data.csv`, `pca_data.csv`, `tsne_data.csv`, `umap_data.csv` .

[ ]:
```
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE, MDS
import umap as umap_module
import pandas as pd
import numpy as np
```

```
/Users/liruifeng/miniconda3/lib/python3.11/site-packages/tqdm/auto.py:21:
TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

1.1 Data Preparation

[ ]:
```
def data_prep(datapath, num_features, cat_features, date_feature,␣
 ↪impute_strategy='mean', scale=True):
    # Load the data
    data = pd.read_csv(datapath)

    # Convert date feature to numerical
    data[date_feature] = pd.to_datetime(data[date_feature])
    data['Year'] = data[date_feature].dt.year
    data['Month'] = data[date_feature].dt.month
    data['Day'] = data[date_feature].dt.day
    num_features += ['Year', 'Month', 'Day']

    # Define the transformers
    num_transformer = Pipeline(steps=[
        ('imputer', SimpleImputer(strategy=impute_strategy)),
```

```
        ('scaler', StandardScaler())]) if scale else␣
↪SimpleImputer(strategy=impute_strategy)

    cat_transformer = Pipeline(steps=[
        ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
        ('onehot', OneHotEncoder(handle_unknown='ignore'))])

    # Combine the transformers
    preprocessor = ColumnTransformer(
        transformers=[
            ('num', num_transformer, num_features),
            ('cat', cat_transformer, cat_features)])

    # Fit and transform the data
    data_prepared = preprocessor.fit_transform(data)

    return data_prepared
```

1.2 Reduction

[ ]:
```
def mds(data, n_components=2, metric=True):
    mds = MDS(n_components=n_components, metric=metric)
    return mds.fit_transform(data)

def pca(data, n_components=2):
    pca = PCA(n_components=n_components)
    return pca.fit_transform(data)

def tsne(data, n_components=2, perplexity=30, learning_rate=200):
    tsne = TSNE(n_components=n_components, perplexity=perplexity,␣
↪learning_rate=learning_rate)
    return tsne.fit_transform(data)

def umap_func(data, n_components=2, n_neighbors=15, min_dist=0.1):
    umap = umap_module.UMAP(n_components=n_components, n_neighbors=n_neighbors,␣
↪min_dist=min_dist)
    return umap.fit_transform(data)
```

1.3 Data

[ ]:
```
# Load the data
datapath = "/Users/liruifeng/Desktop/DataVisualization/homework/hw2/
↪Dimension_Reduction/Data/EssayAnalysis.csv"

# Define the numerical and categorical features
num_features = ['Number of Essays', 'True', 'False', 'Blank', 'Net',␣
↪'ExamDuration']
```

```
cat_features = ['EssayPublication']
date_feature = 'Date'
data_prepared = data_prep(datapath, num_features, cat_features, date_feature)

# Prepare the data
data = data_prep(datapath, num_features, cat_features, date_feature)
```

```
/var/folders/68/y816p_6x2018gj5z_9g2nwqr0000gn/T/ipykernel_4965/1499748032.py:6:
UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the
default) was specified. This may lead to inconsistently parsed dates! Specify a
format to ensure consistent parsing.
  data[date_feature] = pd.to_datetime(data[date_feature])
/var/folders/68/y816p_6x2018gj5z_9g2nwqr0000gn/T/ipykernel_4965/1499748032.py:6:
UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the
default) was specified. This may lead to inconsistently parsed dates! Specify a
format to ensure consistent parsing.
  data[date_feature] = pd.to_datetime(data[date_feature])
```

1.4 pca

[ ]:
```
# Perform PCA
pca_data = pca(data, n_components=2)
pca_data_df = pd.DataFrame(pca_data, columns=['PC1', 'PC2'])
```

1.5 t-sne

[ ]:
```
# Perform t-SNE
tsne_data = tsne(data, n_components=2, perplexity=5, learning_rate=200)
tsne_data_df = pd.DataFrame(tsne_data, columns=['TSNE1', 'TSNE2'])
```

1.6 mds

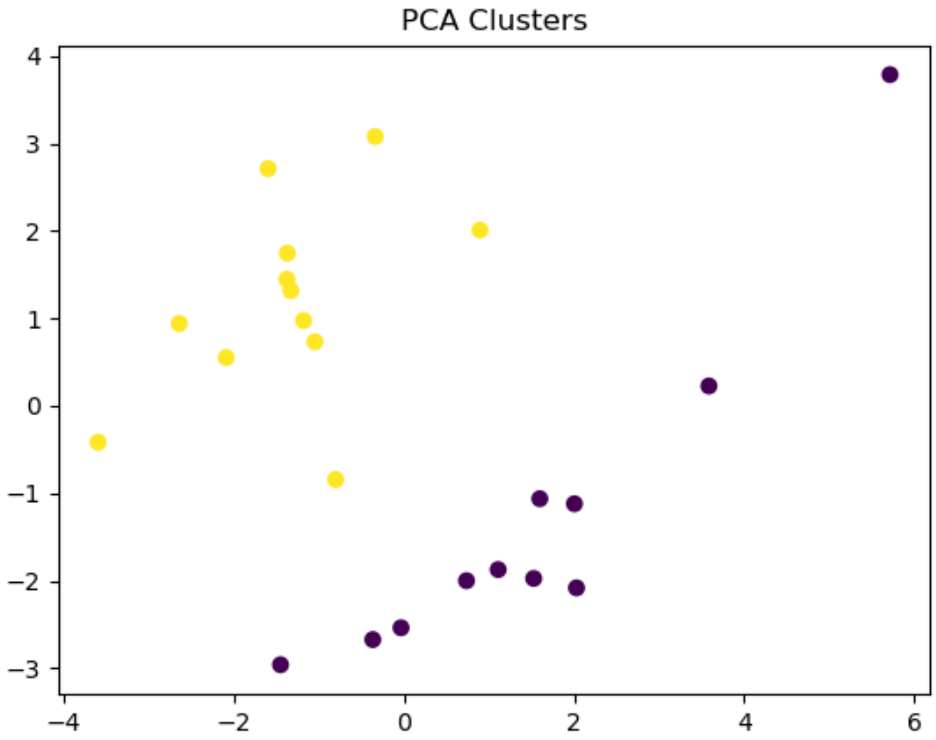
[ ]:
```
# Perform MDS
mds_data = mds(data, n_components=2, metric=True)
mds_data_df = pd.DataFrame(mds_data, columns=['MDS1', 'MDS2'])
```

```
/Users/liruifeng/miniconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:299: FutureWarning: The default value of
`normalized_stress` will change to `'auto'` in version 1.4. To suppress this
warning, manually set the value of `normalized_stress`.
  warnings.warn(
```

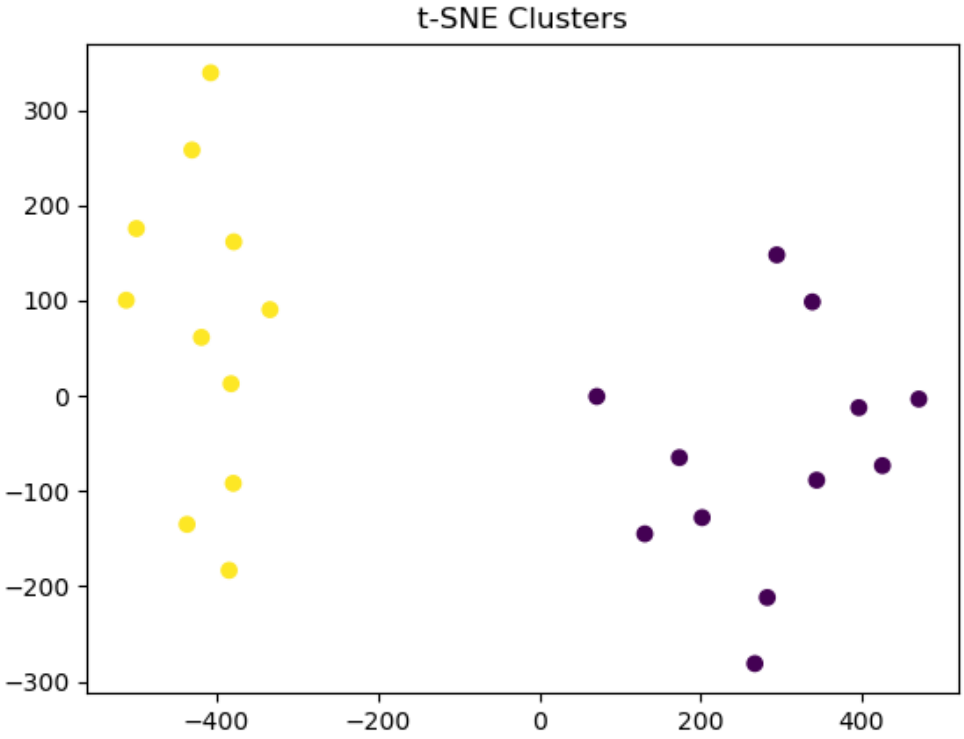
1.7 umap

[ ]:
```
# Perform UMAP
umap_data = umap_func(data, n_components=2, n_neighbors=15, min_dist=0.1)
umap_data_df = pd.DataFrame(umap_data, columns=['UMAP1', 'UMAP2'])
```

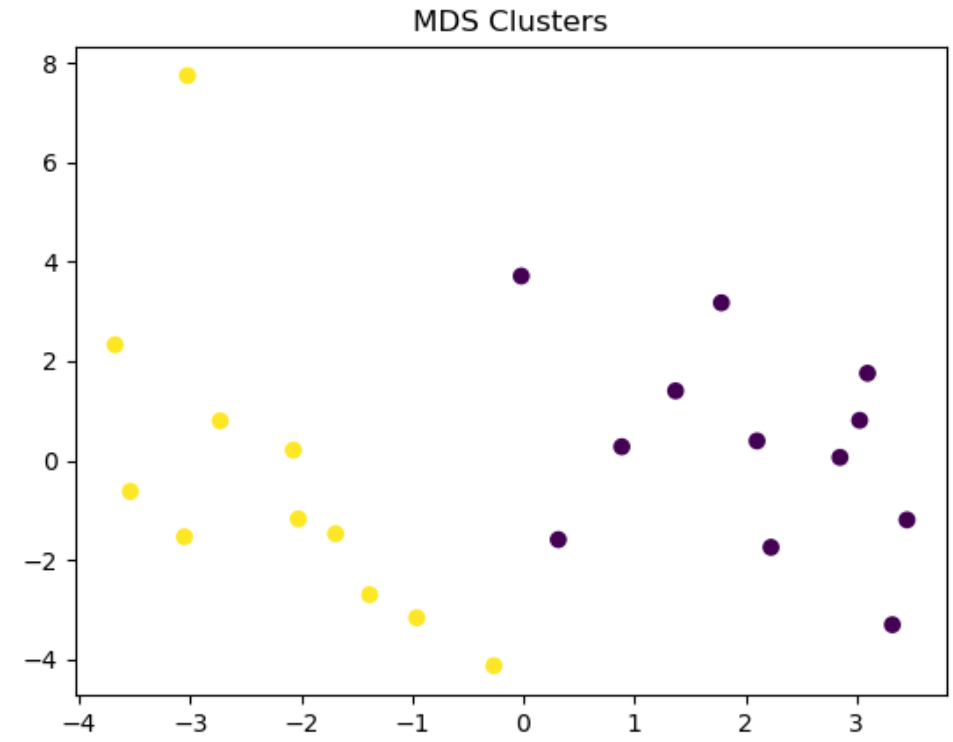
1.8 Data after cluster



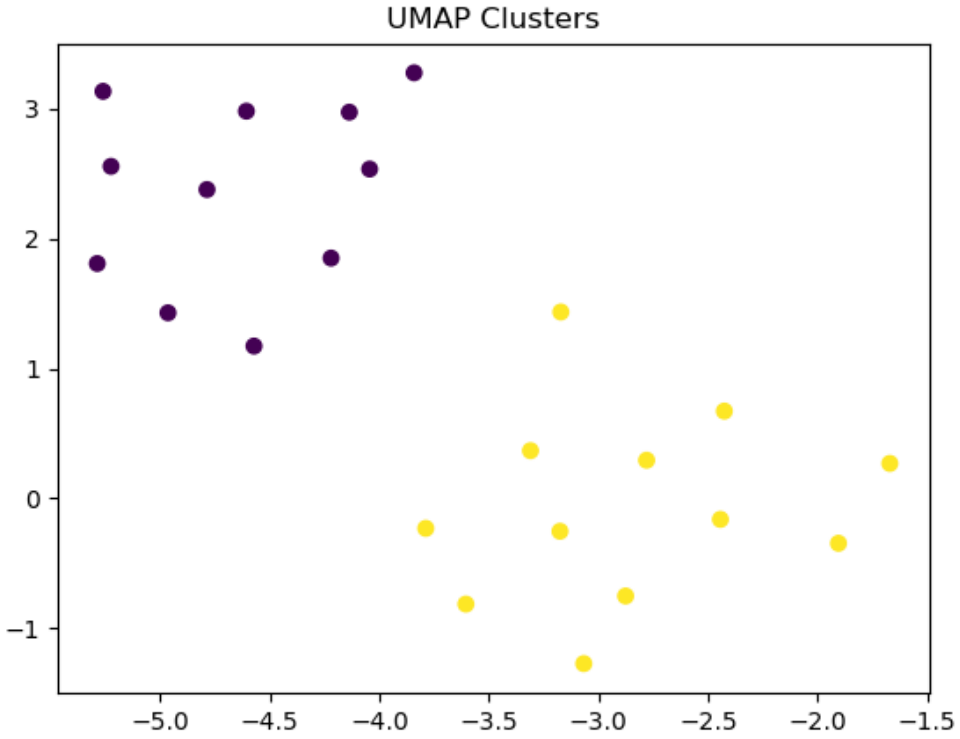
pca


t-SNE Clusters
300
200
100
0
−100
−200
−300
−400
−200
0
200
400

t-sne


MDS Clusters
8
6
4
2
0
−2
−4
−4
−3
−2
−1
0
1
2
3

mds



umap