project2-tangential

November 18, 2024

[1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import copy
import scipy as sp
import scipy.stats as stats
import time
import itertools
from sklearn.model_selection import GridSearchCV
from sklearn.decomposition import NMF, TruncatedSVD
from sklearn.cluster import AgglomerativeClustering, KMeans,␣
 ↪SpectralClustering, OPTICS
from sklearn.mixture import GaussianMixture
from sklearn.metrics import accuracy_score, confusion_matrix, silhouette_score
from scipy.cluster.hierarchy import dendrogram
from sklearn.preprocessing import StandardScaler, MinMaxScaler

df = pd.read_excel("data/titanic3.xls")
RANDOMSTATE = 42
NCLUST = 2

td = df[['pclass',
        'sex',
        'age',
        'sibsp',
        'parch',
        'fare',
        'embarked',
        'survived']].copy()

td.sex = td.sex.map({'male': 0, 'female': 1})
td.age = td.age.fillna(td.groupby('sex')['age'].transform('mean'))
td.fare = td.fare.fillna(td.fare.median())
td.embarked = td.embarked.fillna('S').map({'S': 0, 'C': 1, 'Q': 2})
gtruth = df.survived
```

```
corr_order = list(td.corr().iloc[:,-1].abs().sort_values(ascending=False)[1:].
 ↪index)
td = td[corr_order] # drops the survived column from titanic data

X_std = StandardScaler(with_mean=True, with_std=True).fit_transform(td)
X_zo = MinMaxScaler(feature_range=(0, 1)).fit_transform(td)
```

Picking through data was necessary to catch as many issues as possible. Some values were simple transcription errors, such as a seemingly random 'T' under 'embarked'. Thankfully, the dataset featured no actual duplicates of passengers. However, two pairs of passengers did share an identical name.

```
[2]: df.name[df.name.duplicated()]
```

```
[2]: 726    Connolly, Miss. Kate
     925        Kelly, Mr. James
     Name: name, dtype: object
```

```
[3]: df[df.name == 'Connolly, Miss. Kate']
```

```
[3]:      pclass  survived                   name     sex   age  sibsp  parch  \
     725       3         1  Connolly, Miss. Kate  female  22.0      0      0
     726       3         0  Connolly, Miss. Kate  female  30.0      0      0

          ticket    fare cabin embarked boat  body home.dest
     725  370373  7.7500   NaN        Q   13   NaN   Ireland
     726  330972  7.6292   NaN        Q  NaN   NaN   Ireland
```

```
[4]: df[df.name == 'Kelly, Mr. James']
```

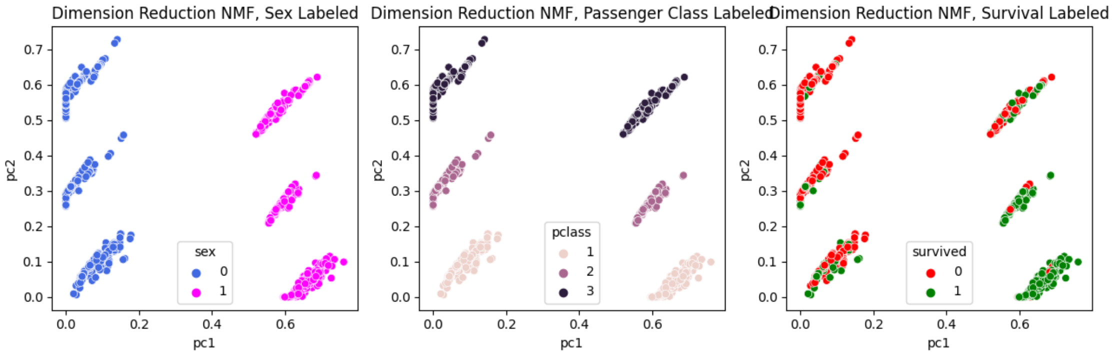
```
[4]:      pclass  survived               name   sex   age  sibsp  parch  ticket  \
     924       3         0  Kelly, Mr. James  male  34.5      0      0  330911
     925       3         0  Kelly, Mr. James  male  44.0      0      0  363592

            fare cabin embarked boat  body home.dest
     924  7.8292   NaN        Q  NaN  70.0       NaN
     925  8.0500   NaN        S  NaN   NaN       NaN
```

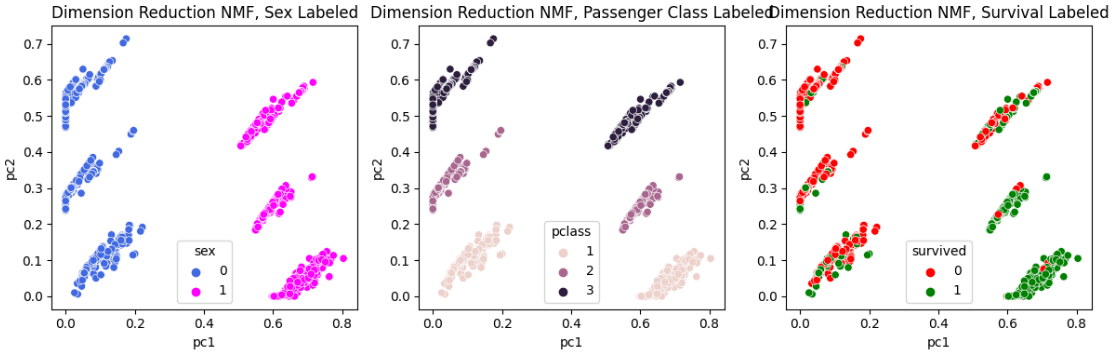
0.0.1 Manipulating The Projection

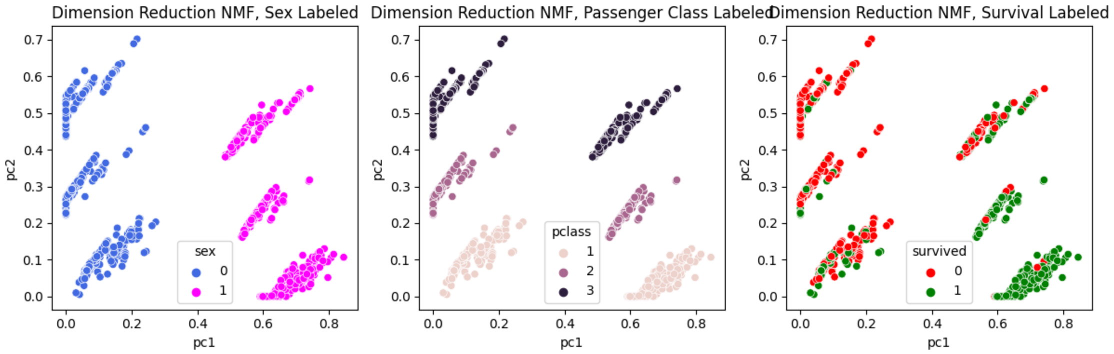
Whether we look to NMF or SVD it is possible to observe how the outcome of a projection can change by preemptively controlling for the scale of some features. This alters the angle and spread among datapoints. Scroll down to obsereve how the preprocessed data transitions between six and three groupings when 'sex' and 'pclass' are scaled between 0-1 and 0-.5.

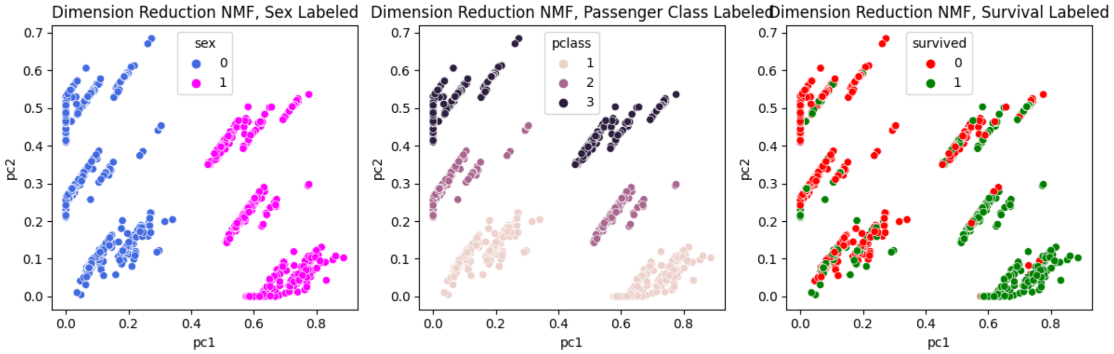
```
[5]: def cycle_scaling(X):
         for i in range(10):
             X = MinMaxScaler(feature_range=(0, 1)).fit_transform(X)
```

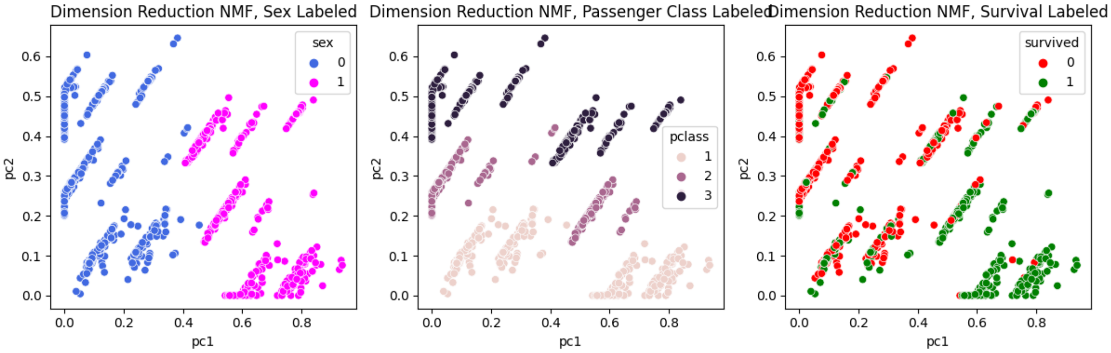
```
        X[:, 1] = np.vectorize(lambda x: x/(i/10+1))(X[:, 1]) # reducing spread/
↪tightness
        X[:, 0] = np.vectorize(lambda x: x/(i/10+1))(X[:, 0]) # scaling␣
↪adjustment

        #X_svd = pd.DataFrame(TruncatedSVD(n_components=2, algorithm='arpack',␣
↪random_state=RANDOMSTATE).fit_transform(X),
        #             columns=['pc1', 'pc2'])
        X_nmf = pd.DataFrame(NMF(n_components=2, init='random', max_iter=600,␣
↪random_state=RANDOMSTATE).fit_transform(X),
                     columns=['pc1', 'pc2'])

        fig, axes = plt.subplots(1, 3, figsize=(12, 4), sharey=False)
        sns.scatterplot(x='pc1', y='pc2', data=X_nmf, hue=td.sex, palette={0:␣
↪'royalblue', 1: 'magenta'}, ax=axes[0])
        axes[0].set_title('Dimension Reduction NMF, Sex Labeled')
        sns.scatterplot(x='pc1', y='pc2', data=X_nmf, hue=td.pclass, ax=axes[1])
        axes[1].set_title('Dimension Reduction NMF, Passenger Class Labeled')
        sns.scatterplot(x='pc1', y='pc2', data=X_nmf, hue=gtruth, palette={0:␣
↪'red', 1: 'green'}, ax=axes[2])
        axes[2].set_title('Dimension Reduction NMF, Survival Labeled')
        plt.tight_layout()
        plt.show()

cycle_scaling(td)
```

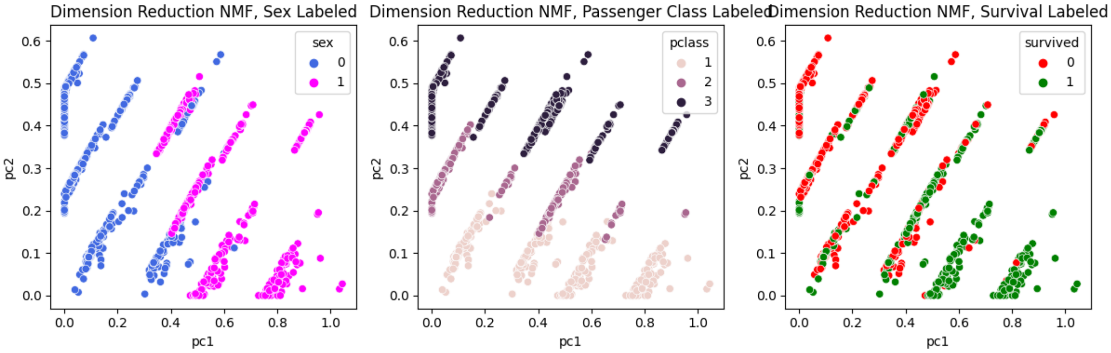


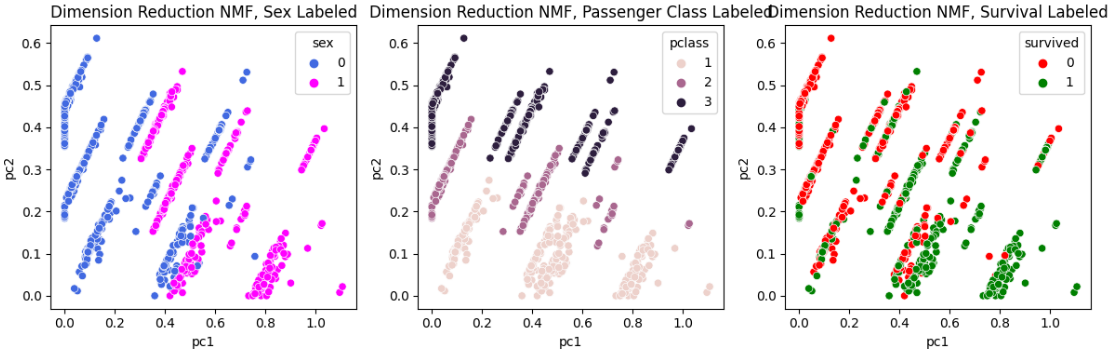

Dimension Reduction NMF, Sex Labeled
Dimension Reduction NMF, Passenger Class Labeled
Dimension Reduction NMF, Survival Labeled
pc1
pc2
0.0
0.2
0.4
0.6
0.8
0.1
0.3
0.5
0.7
sex
● 0
● 1
pclass
● 1
● 2
● 3
survived
● 0
● 1

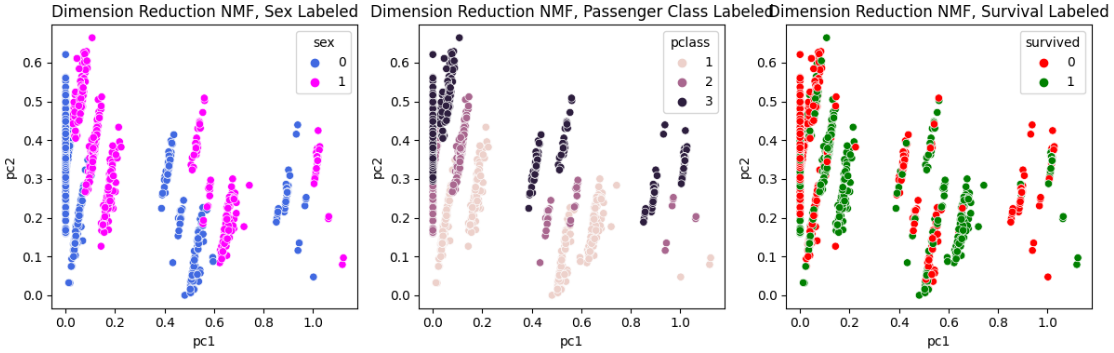

Dimension Reduction NMF, Sex Labeled
Dimension Reduction NMF, Passenger Class Labeled
Dimension Reduction NMF, Survival Labeled
pc1
pc2
0.0
0.2
0.4
0.6
0.8
0.1
0.3
0.5
0.7
sex
● 0
● 1
pclass
● 1
● 2
● 3
survived
● 0
● 1

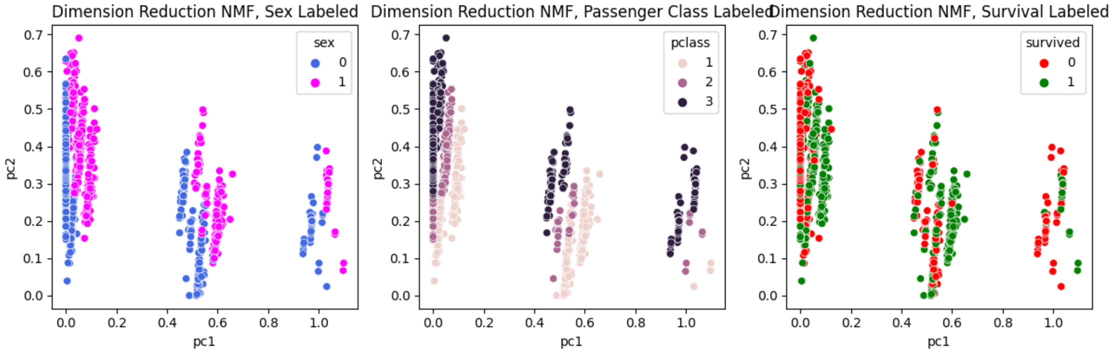

Dimension Reduction NMF, Sex Labeled
Dimension Reduction NMF, Passenger Class Labeled
Dimension Reduction NMF, Survival Labeled
pc1
pc2
0.0
0.2
0.4
0.6
0.8
0.1
0.3
0.5
0.7
sex
● 0
● 1
pclass
● 1
● 2
● 3
survived
● 0
● 1

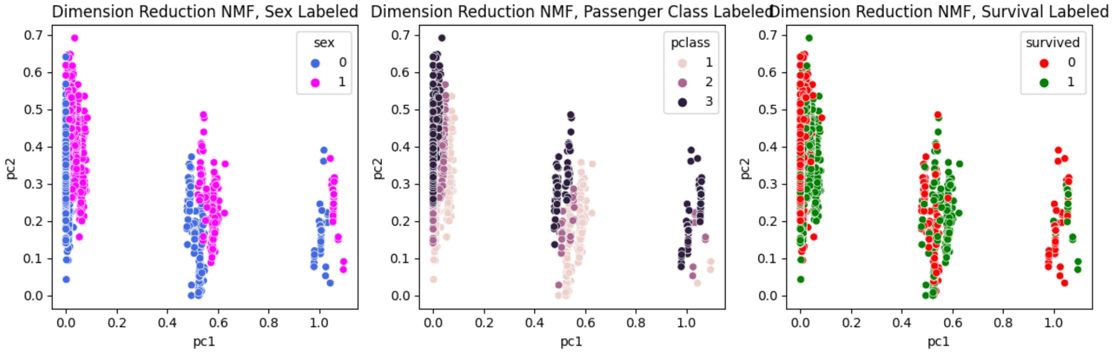

Dimension Reduction NMF, Sex Labeled
Dimension Reduction NMF, Passenger Class Labeled
Dimension Reduction NMF, Survival Labeled


Dimension Reduction NMF, Sex Labeled
Dimension Reduction NMF, Passenger Class Labeled
Dimension Reduction NMF, Survival Labeled


Dimension Reduction NMF, Sex Labeled
Dimension Reduction NMF, Passenger Class Labeled
Dimension Reduction NMF, Survival Labeled





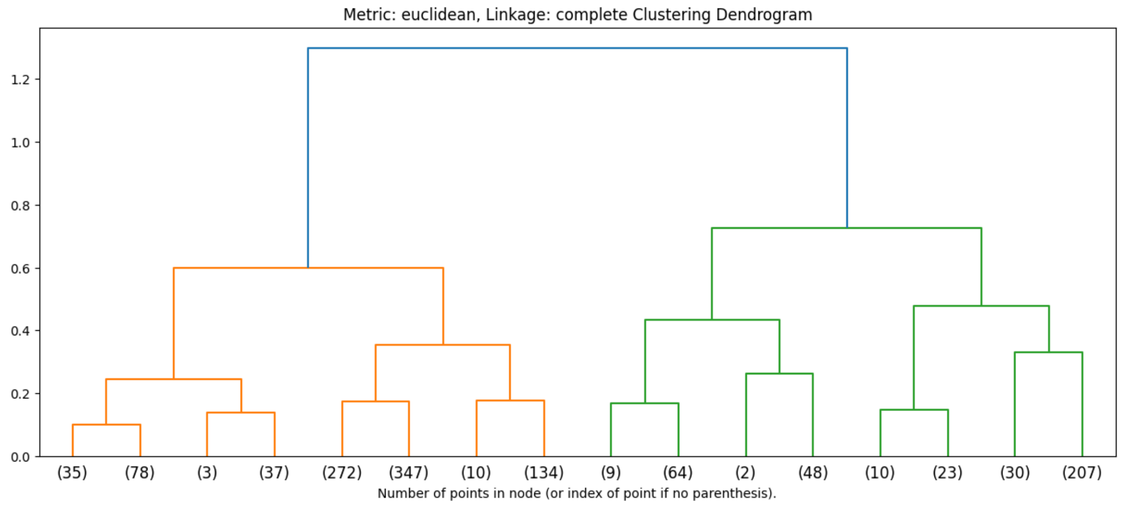


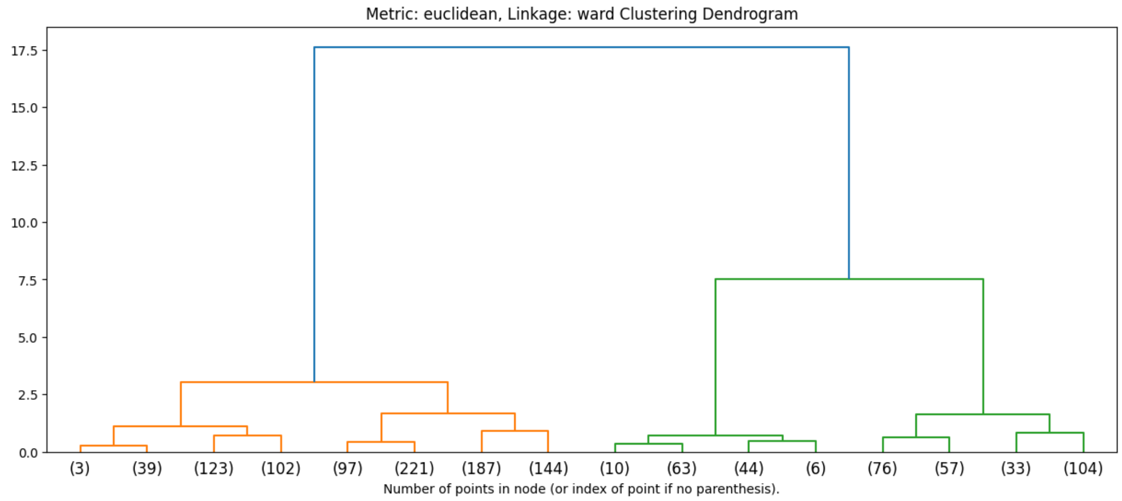
0.0.2 Complete Vs. Ward Dendrogram

In our case ‘complete’ linkage won out over ‘ward’ fairly significantly. From the dendrogram, the complete scheme interprets groupings more evenly, with agglomeration occuring at more consistant distances. The ward scheme is comparitively squashed, and percieves the final two groups as significantly farther apart.

[8]:

```python
X_zo[:, 1] = np.vectorize(lambda x: x/1.4)(X_zo[:, 1]) # 'sex'
X_zo[:, 0] = np.vectorize(lambda x: x/1.4)(X_zo[:, 0]) # 'pclass'

X_svd = pd.DataFrame(TruncatedSVD(n_components=2, algorithm='arpack',␣
 ↪random_state=RANDOMSTATE).fit_transform(X_zo),
                     columns=['pc1', 'pc2'])
X_nmf = pd.DataFrame(NMF(n_components=2, init='random', max_iter=400,
                         beta_loss='frobenius', solver='cd',
                         random_state=RANDOMSTATE).fit_transform(X_zo),
                     columns=['pc1', 'pc2'])


def plot_dendrogram(model, **kwargs):
    # Create linkage matrix and then plot the dendrogram

    # create the counts of samples under each node
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1  # leaf node
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

    linkage_matrix = np.column_stack(
        [model.children_, model.distances_, counts]
    ).astype(float)

    dendrogram(linkage_matrix, **kwargs)


metrics = ['euclidean']
linkages = ['complete', 'ward']
for linkage in linkages:

    model = AgglomerativeClustering(metric=metrics[0],
                                    distance_threshold=0,
                                    n_clusters=None,
                                    linkage=linkage).fit(X_nmf)

    plt.figure(figsize=(15, 6))
    plt.title(f"Metric: {metrics[0]}, Linkage: {linkage} Clustering Dendrogram")

    plot_dendrogram(model, truncate_mode="level", p=3)
    plt.xlabel("Number of points in node (or index of point if no parenthesis).
↪")
    plt.show()
```

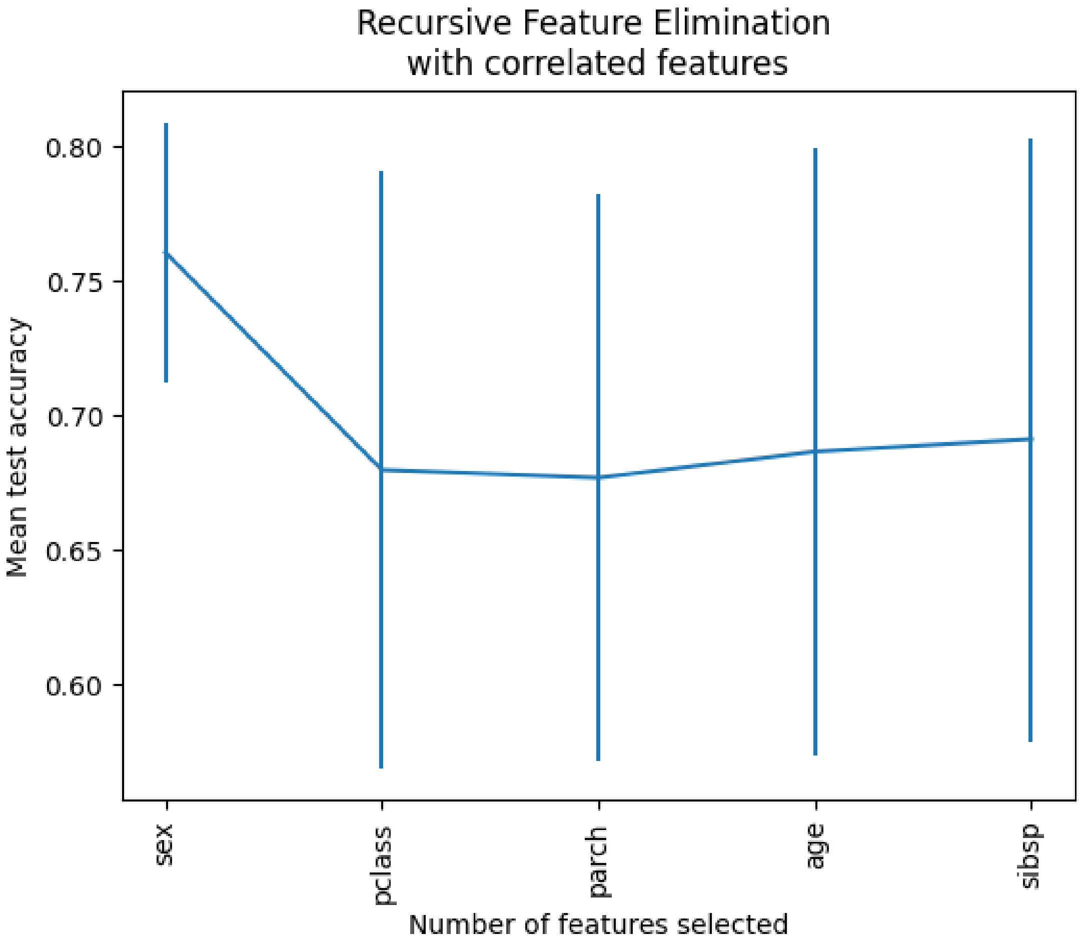




[9]:

```
from sklearn.feature_selection import RFECV
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import StratifiedKFold

min_features_to_select = 1
clf = GradientBoostingClassifier()
cv = StratifiedKFold(4)

rfecv = RFECV(
    estimator=clf,
    step=1,
    cv=cv,
    scoring="balanced_accuracy",
    min_features_to_select=min_features_to_select,
    n_jobs=2,
)
X = pd.DataFrame(X_zo, columns=td.columns).drop(columns=['embarked','fare'])

rfecv.fit(X, gtruth)

print(f"Optimal number of features: {rfecv.n_features_}")

cv_results = pd.DataFrame(rfecv.cv_results_)
plt.figure()
plt.xlabel("Number of features selected")
plt.ylabel("Mean test accuracy")
plt.errorbar(
    x=rfecv.feature_names_in_,
    y=cv_results["mean_test_score"],
    yerr=cv_results["std_test_score"],
)
plt.xticks(rotation=90)
plt.title("Recursive Feature Elimination \nwith correlated features")
plt.show()
```

```
Optimal number of features: 1
```



0.0.3 Cabin Subset

[10]:
```
NCLUST = 7

td = df[['pclass',
         'sex',
         'fare',
         'embarked',
         'age',
         'parch',
         'sibsp']].copy()

td.sex = td.sex.map({'male': 0, 'female': 1})
td.age = td.age.fillna(td.groupby('sex')['age'].transform('mean'))
td.fare = td.fare.fillna(td.fare.median())
td.embarked = td.embarked.fillna('S').map({'S': 0, 'C': 1, 'Q': 2})
```

```
cabins = df[df.cabin.notna()].cabin.str[0].map({'A': 0, 'B': 1, 'C': 2, 'D': 3,␣
 ↪'E': 4, 'F': 5, 'G': 6, 'T': 5})

tdc = td[df.cabin.notna()]
```

[11]:
```
tdc = tdc.reset_index(drop=True)
tdc
```

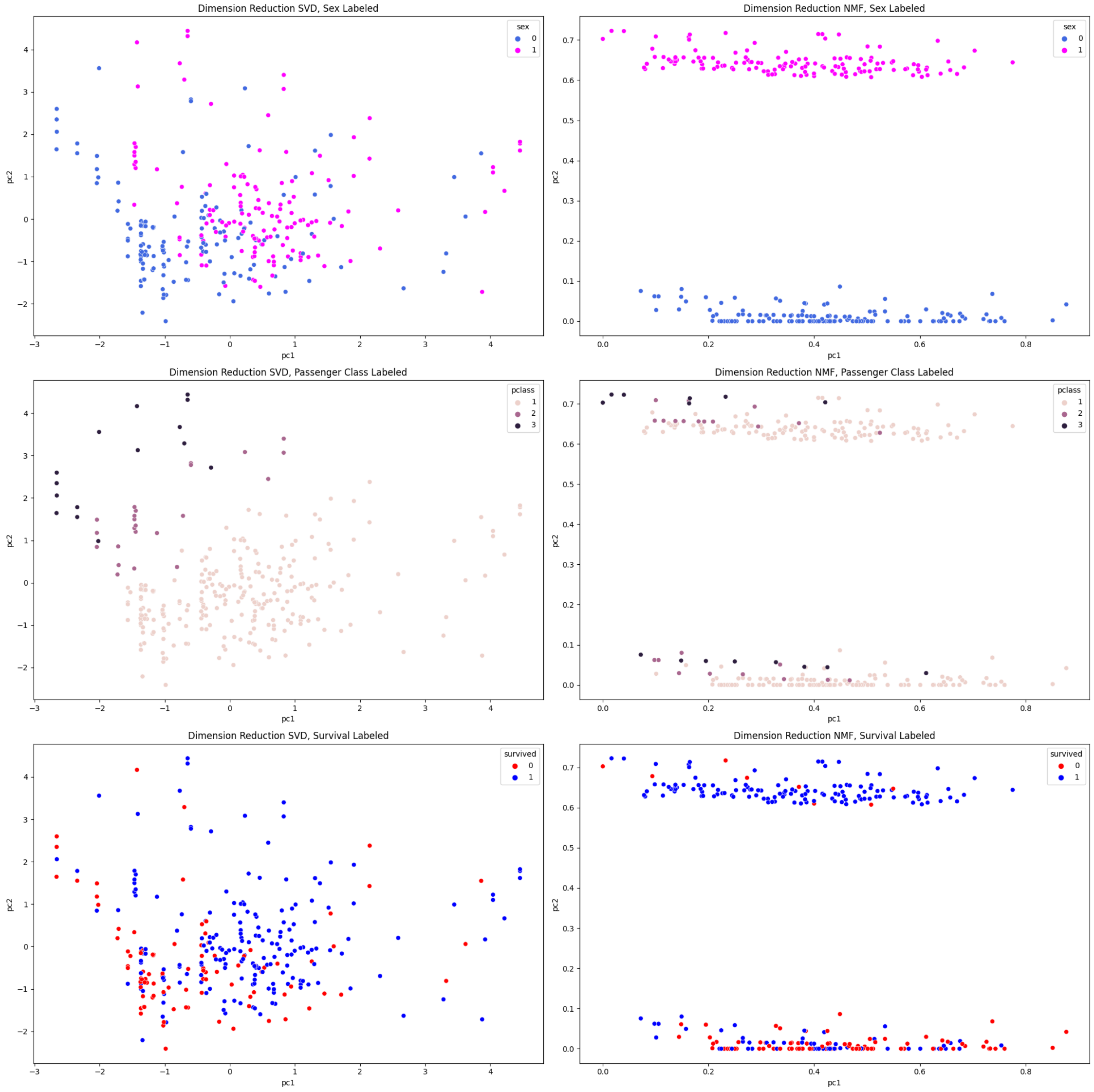
[11]:
```
     pclass  sex      fare  embarked        age  parch  sibsp
0         1    1  211.3375         0  29.000000      0      0
1         1    0  151.5500         0   0.916700      2      1
2         1    1  151.5500         0   2.000000      2      1
3         1    0  151.5500         0  30.000000      2      1
4         1    1  151.5500         0  25.000000      2      1
..      …   …        …        …         …    …     …
290       3    1   16.7000         0   4.000000      1      1
291       3    0    7.6500         0  19.000000      0      0
292       3    1   10.4625         0   2.000000      1      0
293       3    1   10.4625         0  29.000000      1      1
294       3    0    7.7500         2  30.585233      0      0

[295 rows x 7 columns]
```

[12]:
```
for column in  list(tdc.columns):
    print(column, len(tdc[column].unique()))
```

```
pclass 3
sex 2
fare 112
embarked 3
age 73
parch 5
sibsp 4
```

[15]:
```
from sklearn.decomposition import PCA

X_std = StandardScaler(with_mean=True, with_std=True).fit_transform(tdc)
X_zo = MinMaxScaler(feature_range=(0, 1)).fit_transform(tdc)

X_svd = pd.DataFrame(PCA(n_components=2, svd_solver='full',␣
 ↪random_state=RANDOMSTATE).fit_transform(X_std),
                     columns=['pc1', 'pc2'])
X_nmf = pd.DataFrame(NMF(n_components=2, init='random',␣
 ↪random_state=RANDOMSTATE).fit_transform(X_zo),
                     columns=['pc1', 'pc2'])

fig, axes = plt.subplots(3, 2, figsize=(20, 20), sharey=False)
```

```
sns.scatterplot(x='pc1', y='pc2', data=X_svd, hue=tdc.sex, palette={0:␣
 ↪'royalblue', 1: 'magenta'}, ax=axes[0][0])
axes[0][0].set_title('Dimension Reduction SVD, Sex Labeled')

sns.scatterplot(x='pc1', y='pc2', data=X_nmf, hue=tdc.sex, palette={0:␣
 ↪'royalblue', 1: 'magenta'}, ax=axes[0][1])
axes[0][1].set_title('Dimension Reduction NMF, Sex Labeled')

sns.scatterplot(x='pc1', y='pc2', data=X_svd, hue=tdc.pclass, ax=axes[1][0])
axes[1][0].set_title('Dimension Reduction SVD, Passenger Class Labeled')

sns.scatterplot(x='pc1', y='pc2', data=X_nmf, hue=tdc.pclass, ax=axes[1][1])
axes[1][1].set_title('Dimension Reduction NMF, Passenger Class Labeled')

sns.scatterplot(x='pc1', y='pc2', data=X_svd, hue=df[df.cabin.notna()].survived.
 ↪reset_index(drop=True), palette={0: 'red', 1: 'blue'}, ax=axes[2][0])
axes[2][0].set_title('Dimension Reduction SVD, Survival Labeled')

sns.scatterplot(x='pc1', y='pc2', data=X_nmf, hue=df[df.cabin.notna()].survived.
 ↪reset_index(drop=True), palette={0: 'red', 1: 'blue'}, ax=axes[2][1])
axes[2][1].set_title('Dimension Reduction NMF, Survival Labeled')

plt.tight_layout()
plt.show()
```


Dimension Reduction SVD, Sex Labeled
Dimension Reduction NMF, Sex Labeled
Dimension Reduction SVD, Passenger Class Labeled
Dimension Reduction NMF, Passenger Class Labeled
Dimension Reduction SVD, Survival Labeled
Dimension Reduction NMF, Survival Labeled
sex
● 0
● 1
pclass
● 1
● 2
● 3
survived
● 0
● 1
pc1
pc2
−3
−2
−1
0
1
2
3
4
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

[ ]: