Dimensionality Reduction Techniques: PCA, LDA

Techniques include:

- Principal Component Analysis
- Linear Discriminant Analysis

In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as pex
%matplotlib inline
```

In [2]:

```
from sklearn.datasets import load_breast_cancer

data = load_breast_cancer()
df = pd.DataFrame(data['data'],columns=data['feature_names'])
```

In [3]:

```
# Scale the data
df = (df - df.mean(axis=0)) / df.std(axis=0)
df['status'] = data['target']
```

In [13]:

```
df.head()
```

Out[13]:

an ss	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	
28	2.650542	2.530249	2.215566	2.253764	...	-1.358098	2.301575	1.9
43	-0.023825	0.547662	0.001391	-0.867889	...	-0.368879	1.533776	1.8
00	1.362280	2.035440	0.938859	-0.397658	...	-0.023953	1.346291	1.4
17	1.914213	1.450431	2.864862	4.906602	...	0.133866	-0.249720	-0.5
66	1.369806	1.427237	-0.009552	-0.561956	...	-1.465481	1.337363	1.2

Principal Component Analysis (PCA)

This technique involves projecting the high-dimensional data onto orthogonal axes that maximize the variance in the data.

- We will perform this manually by taking the eigendecomposition of the covariance matrix of the data.
- The eigenvectors represent the orthogonal axes which we will project the data onto.
- Each eigenvalue represents the variance of the data when projected onto the axis represented by its corresponding eigenvector.
- To visualize the data in reduced dimensions, we will choose 3 axes / eigenvectors that preserve the most variance in the data.
- Therefore, these chosen eigenvectors will have the largest eigenvalues.

In [4]:

```
cov_matrix = np.cov(df.drop('status',axis=1), rowvar=False)
cov_matrix.shape
```

Out[4]:

```
(30, 30)
```

In [5]:

```
values, vectors = np.linalg.eig(cov_matrix)
```

In [6]:

```
print("Values shape: " + str(values.shape))
print("Vectors shape: " + str(vectors.shape))
```

```
Values shape: (30,)
Vectors shape: (30, 30)
```

In [7]:

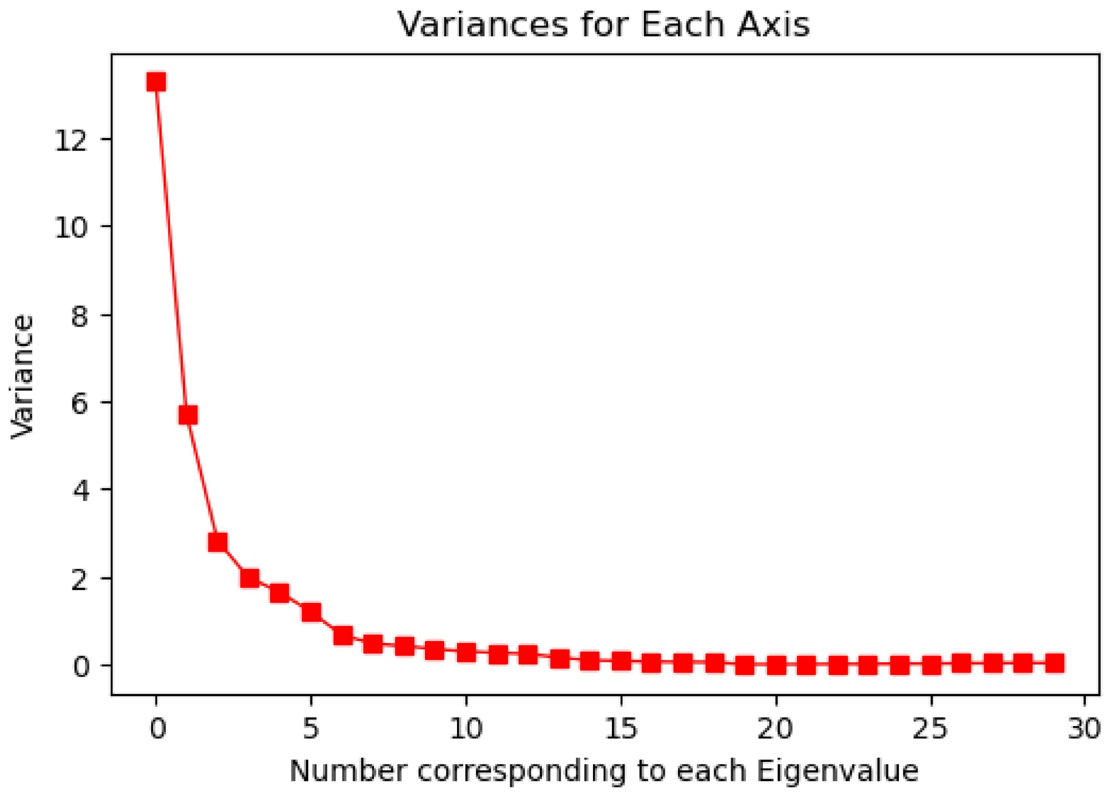
```
"""
- Each eigenvector has 30 entries, one for each feature.
"""

vectors[:,0]
```

Out[7]:

```
array([0.21890244, 0.10372458, 0.22753729, 0.22099499, 0.142
58969,
       0.23928535, 0.25840048, 0.26085376, 0.13816696, 0.064
36335,
       0.20597878, 0.01742803, 0.21132592, 0.20286964, 0.014
53145,
       0.17039345, 0.15358979, 0.1834174 , 0.04249842, 0.102
56832,
       0.22799663, 0.10446933, 0.23663968, 0.22487053, 0.127
95256,
       0.21009588, 0.22876753, 0.25088597, 0.12290456, 0.131
78394])
```

In [8]:

```
plt.figure(figsize=(6,4),dpi=100)
plt.plot(values,marker='s',color='red',lw=1)
plt.xlabel('Number corresponding to each Eigenvalue')
plt.ylabel('Variance')
plt.title('Variances for Each Axis')
plt.show()
```



In [9]:

```
# Let's take the top 3 eigenvectors and project data onto them.
# From the plot we see that the eigenvalues are already sorted, so we c
top_3_vectors = vectors[:,np.array([0,1,2])]
```

In [10]:

```
# Project data down to 3 axes by computing dot product
principal_comp = np.dot(df.drop('status',axis=1).values, top_3_vectors)
```

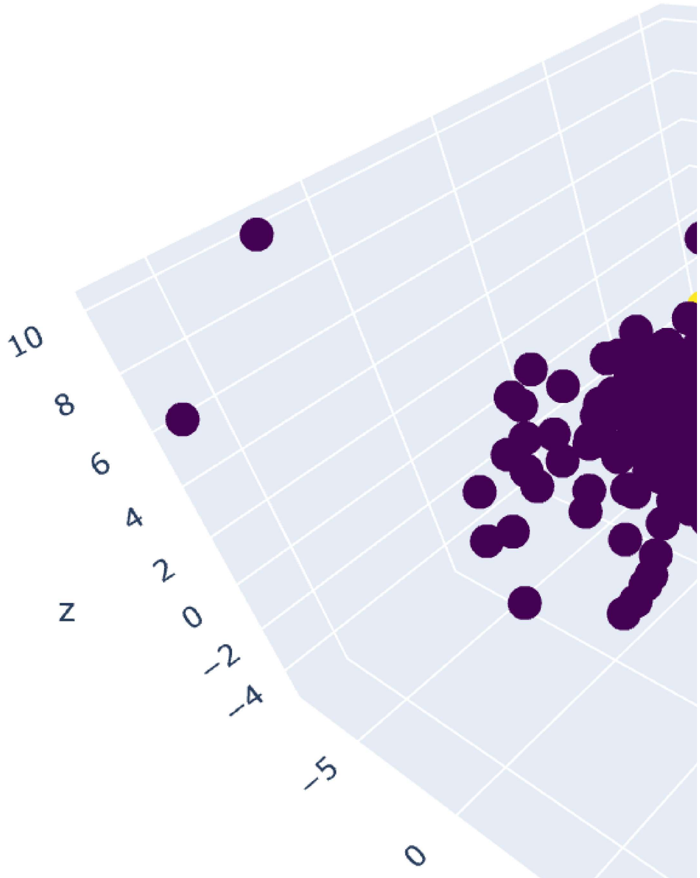
In [11]:

```
print(principal_comp.shape) #Correct shape!
```

```
(569, 3)
```

In [12]:

```
1  pex.scatter_3d(x=principal_comp[:,0],y=principal_comp[:,1],z=principal_
```



We can see the clear separation of classes from this plot, illustrating the benefits of dimensionality reduction. With a few more principal components to maximize the variance, we could train a SVM to take advantage of the relatively clear separation of classes to predict Breast Cancer Status.

Linear Discriminant Analysis

LDA is another dimensionality reduction. Like PCA, it involves projecting the data onto axes; however, its goal is to choose axes that maximize class separability and minimize intra-class scatter simultaneously. With a binary-class dataset, the data will be projected onto a line. With a n-class dataset, the data will be projected onto an (n-1)-dimensional-space.

In [14]:

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as

model = lda(n_components=1) # n_components in this case is 1 less than
```

In [15]:

```
newData = model.fit_transform(df.drop('status', axis=1), df['status'])
newData = pd.DataFrame([newData[:,0],df['status']])

newData = newData.T
newData['y'] = [0 for _ in range(newData.shape[0])]
```

In [16]:

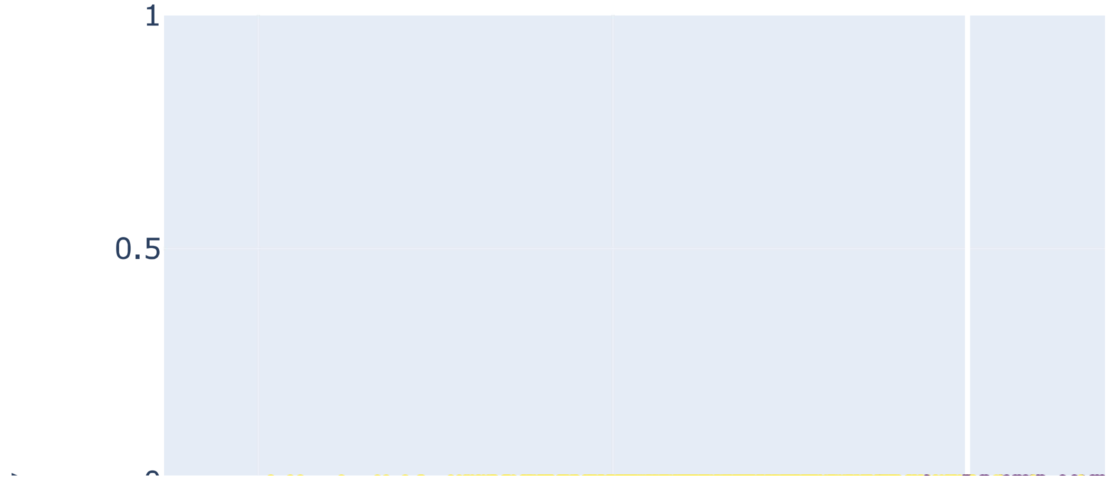
```
model.explained_variance_ratio_ # Variance of data explained by the ax
```

Out[16]:

```
array([1.])
```

In [17]:

```
pex.scatter(newData, x=0, y='y', color=1, color_continuous_scale=pex.c(
```



This plot shows the difference in class distributions for the data projected on a single axis. LDA can also be used as a classifier in addition to a dimensionality reduction technique.