Feature Selection

Agenda

In this lesson, we will cover the following concepts with the help of a business use case:

- Feature Selection
- Dimensionality Reduction:
    - Dimensionality Reduction Techniques
    - Pros and Cons of Dimensionality Reduction
- Prominent Methods Used for Feature Selection:
    - Factor Analysis
    - Principal Component Analysis
    - LDA

What Is Feature Selection?

Feature selection is a method that helps in the inclusion of the significant variables that help form a model with good predictive power.

Features or variables that are redundant or irrelevant can negatively impact the performance of the model, thus it becomes necessary to remove them.

Benefits of Feature Selection

- It reduces overfitting as the unwanted variables are removed, and the focus is on the significant variables.
- It removes irrelevant information, which helps to improve the accuracy of the model's predictions.
- It reduces the computation time involved to get the model's predictions.

Dimensionality Reduction

Dimensionality reduction is the method of transforming a collection of data having large dimensions into data of smaller dimensions while ensuring that identical information is conveyed concisely.

Dimensionality Reduction Techniques

Some of the techniques used for dimensionality reduction are:

1. Imputing missing values
2. Dropping low-variance variables
3. Decision trees (DT)
4. Random forest (RF)
5. Reducing highly correlated variables
6. Backward feature elimination
7. Factor analysis
8. Principal component analysis (PCA)

Pros of Dimensionality Reduction

- It helps to compress data, reducing the storage space needed.

- It cuts down on computing time.
- It also aids in the removal of redundant features.
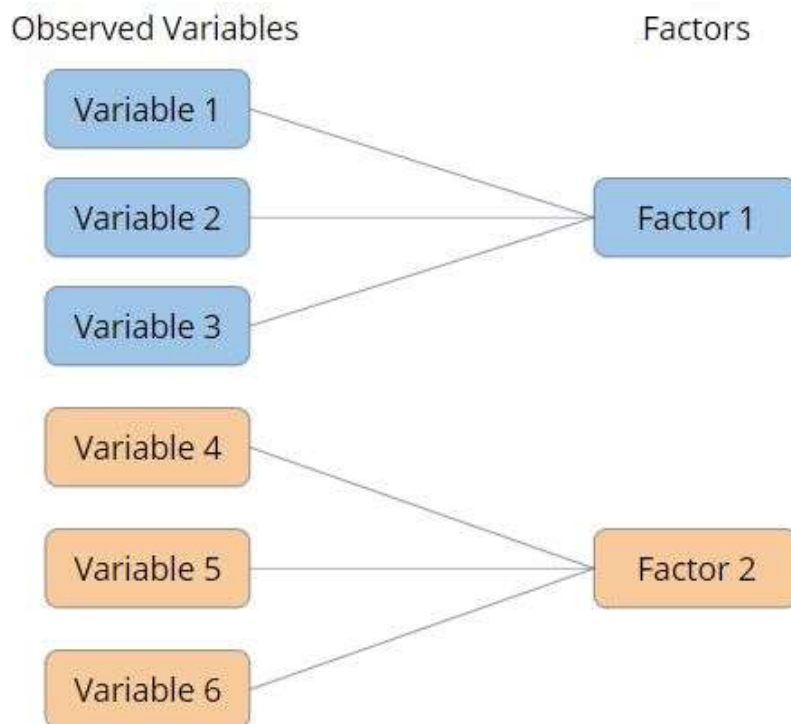
Cons of Dimensionality Reduction

- Some data may will be lost as a result.
- PCA tends to find linear relationships between variables, which is not always ideal.
- When mean and covariance are insufficient to describe datasets, PCA fails.
- We use certain thumb rules when we do not know how many principal components to keep in practice.
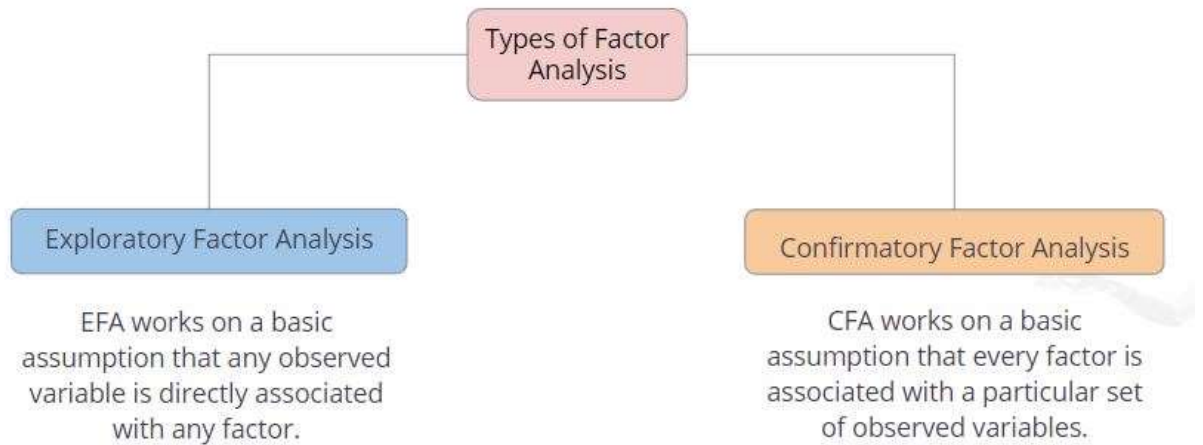
Gist of Factor Analysis

- Factor analysis is used to:
    - Explain variance among the observed variables
    - Condense the set of observed variables into the factors

$$Y_i = \beta_{i0} + \beta_{i1} F_1 + \beta_{i2} F_2 + (1)e_i$$

- Factor explains the amount of variance in the observed variables.
- In other words, factor analysis is a method that investigates linear relation of a number of variables of interest V1, V2,……., Vl, with a smaller number of unobservable factors F1, F2,…….., Fk.

Types of FA

Types of Factor Analysis

Exploratory Factor Analysis

EFA works on a basic assumption that any observed variable is directly associated with any factor.

Confirmatory Factor Analysis

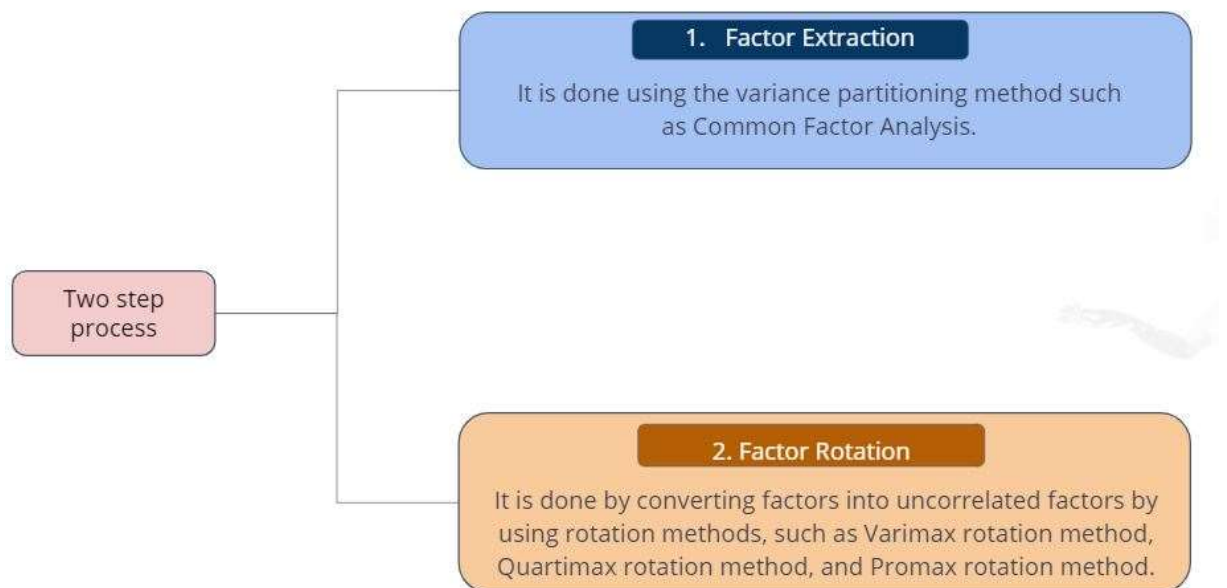CFA works on a basic assumption that every factor is associated with a particular set of observed variables.

Work Process of FA

The objective of the factor analysis is the reduction of the number of observed variables and find the unobservable variables.

It is a two-step process.

Two step process

1. **Factor Extraction**

It is done using the variance partitioning method such as Common Factor Analysis.

2. Factor Rotation

It is done by converting factors into uncorrelated factors by using rotation methods, such as Varimax rotation method, Quartimax rotation method, and Promax rotation method.

Choosing Factors

Note: Let us get accustomed to the term eigenvalue before moving to the selecting the number of factors.

Eigenvalues:

It represents the explained variance of each factor from the total variance and is also known as the characteristic roots.

Ways to Choose Factors:

- The eigenvalues are a good measure for identifying the significant factors. An eigenvalue greater than 1 is considered for the selection criteria of the feature.

- Apart from observing values, the graphical approach is used that visually represents the factors' eigenvalues. This visualization is known as the scree plot. A Scree plot helps in determining the number of factors where the curve makes an elbow.

Use Case: Feature Selection in Cancer Dataset Using FA

Problem Statement

John Cancer Hospital (JCH) is a leading cancer hospital in the USA. It specializes in preventing breast cancer.

Over the last few years, JCH has collected breast cancer data from patients who came for screening or treatment.

However, this data has 32 attributes and is difficult to run and interpret the result. As an ML expert,

you have to reduce the number of attributes so that the results are meaningful and accurate.

Use FA for feature selection.

Dataset

Features of the dataset are computed from a digitized image of a Fine-Needle Aspirate (FNA) of a breast mass.

They describe the characteristics of the cell nuclei present in the image.

Data Dictionary

Dimensions:

- 32 variables
- 569 observations

Attribute Information:

1. ID number
2. Diagnosis (M = malignant, B = benign)
3. Attributes with mean values:
   10 real-valued features are computed for each cell nucleus:

   - radius_mean (mean of distances from center to points on the perimeter)
   - texture_mean (standard deviation of gray-scale values)
   - perimeter_mean
   - area_mean
   - smoothness_mean (local variation in radius lengths)
   - compactness_mean (perimeter^2 / area - 1.0)
   - concavity_mean (severity of concave portions of the contour)
   - concave points_mean (number of concave portions of the contour)
   - symmetry_mean
   - fractal dimension_mean ("coastline approximation" - 1)
4. Attributes with standard error and worst/largest:

   - radius_se
   - texture_se
   - perimeter_se
   - area_se

- smoothness_se
- compactness_se
- concavity_se
- concave points_se
- symmetry_se
- fractal_dimension_se
- radius_worst
- texture_worst
- perimeter_worst
- area_worst
- smoothness_worst
- compactness_worst
- concavity_worst
- concave points_worst
- symmetry_worst
- fractal_dimension_worst

Solution

Import Libraries

In [ ]:
```
1
```

In Python, Numpy is a package that includes multidimensional array objects as well as a number of derived objects. Matplotlib is an amazing visualization library in Python for 2D plots of arrays.\n Pandas is used for data manipulation and analysis\n So these are the core libraries that are used for the EDA process.

These libraries are written with an import keyword.

In [ ]:
```
1  import matplotlib.pyplot as plt
2  import pandas as pd
3  import numpy as np
4  %matplotlib inline
```

Import and Check the Data

Before reading data from a csv file, you need to download the "breast-cancer-data.csv" dataset from the resource section and upload it into the Lab. We will use the Up arrow icon, which is shown on the left side under the View icon. Click on the Up arrow icon and upload the file from wherever it has downloaded into your system.

After this, you will see the downloaded file will be visible on the left side of your lab along with all the .pynb files.

In [ ]:
```
1  df = pd.read_csv('breast-cancer-data.csv')
2  df.head()
```

pd.read_csv function is used to read the "breast-cancer-data.csv" file and df.head() will show the top 5 rows of the dataset.

- dataframe or df is a variable that will store the data read by the csv file.
- head will show the rows and () default take the 5 top rows as output.
- one more example - df.head(3) will show the top 3 rows.

shape function

```
In [ ]:    1  df.shape
```

df.shape will show the number of rows and columns in the dataframe.

info Function

```
In [ ]:    1
```

```
In [ ]:    1  # Check the data , there should be no missing values
           2  df.info()
```

- The dataframe's information is printed using the info() function.
- The number of columns, column labels, column data types, memory use, range index, and the number of cells in each column are all included in the data (non-null values).

```
In [ ]:    1  # defining the array as np.array
           2  feature_names = np.array(['mean radius' 'mean texture' 'mean perimeter' 'mean area'
           3   'mean smoothness' 'mean compactness' 'mean concavity'
           4   'mean concave points' 'mean symmetry' 'mean fractal dimension'
           5   'radius error' 'texture error' 'perimeter error' 'area error'
           6   'smoothness error' 'compactness error' 'concavity error'
           7   'concave points error' 'symmetry error' 'fractal dimension error'
           8   'worst radius' 'worst texture' 'worst perimeter' 'worst area'
           9   'worst smoothness' 'worst compactness' 'worst concavity'
          10   'worst concave points' 'worst symmetry' 'worst fractal dimension'])
```

```
In [ ]:    1  #### Convert diagnosis column to 1/0 and store in new column target
           2  from sklearn.preprocessing import LabelEncoder
```

- The sklearn.preprocessing package contains a number of useful utility methods and transformer classes for converting raw feature vectors into a format that is suitable for downstream estimators.
- LabelEncoder encodes labels with a value between 0 and n_classes-1 where n is the number of distinct labels.
- These libraries are written with an import keyword.

```
In [ ]:    1  # # Encode label diagnosis
           2  #M -> 1
           3  #B -> 0
           4
           5  #Converting diagnosis to numerical variable in df
           6  df['diagnosis'] = df['diagnosis'].map({'M': 1, 'B': 0}).astype(int)
```

In the above code, we are encoding the column diagnosis in which we are encoding M as 1 and B as 0.

Factor Analysis

- A linear statistical model is a factor analysis.
- It is used to condense a group of observable variables into an unobserved variable termed factors and to explain the variance among the observed variables.

Adequacy Test

Before you perform factor analysis, you need to evaluate the "factorability" of our dataset. Can we find the factors in the dataset? Checking factorability or sampling adequacy can be done in two ways:

1- The Bartlett's Test

2- Test of Kaiser-Meyer-Olkin

```
In [ ]:   1  #Install factor analyzer
          2  #!pip install factor_analyzer
```

Bartlett's Test

Bartlett's test of sphericity checks whether or not the observed variables intercorrelate at all using the observed correlation matrix against the identity matrix. If the test is found to be statistically insignificant, you should not employ a factor analysis.

Note: This test checks for the intercorrelation of observed variables by comparing the observed correlation matrix against the identity matrix.

```
In [ ]:   1  !pip install factor_analyzer
```

- In the above code, we are installing the factor analyzer.
- pip is used to install the packages.
- Factor analysis is an exploratory data analysis method used to search for influential underlying factors or latent variables from a set of observed variables.

Now, you are trying to perform factor analysis by using the factor analyzer module. Use the below code for calculating_bartlett_sphericity.

```
In [ ]:   1  from factor_analyzer import FactorAnalyzer
          2  from factor_analyzer.factor_analyzer import calculate_bartlett_sphericity
          3  chi_square_value,p_value=calculate_bartlett_sphericity(df)
          4  chi_square_value, p_value
```

- In the above code, you are importing the factor analyzer and calculate_bartlett_sphericity.
- In this Bartlett's test, the p-value is 0. The test was statistically significant, indicating that the observed correlation matrix is not an identity matrix.

Inference:

The p-value is 0, and this indicates that the test is statistically significant and highlights that the correlation matrix is not an identity matrix.

Kaiser-Meyer-Olkin Test

- The Kaiser-Meyer-Olkin (KMO) test determines if data is suitable for factor analysis.
- It assesses the suitability of each observed variable as well as the entire model.

```
In [ ]:   1  from factor_analyzer.factor_analyzer import calculate_kmo
          2  kmo_all,kmo_model=calculate_kmo(df)
```

```
In [ ]:   1  kmo_model
```

- In the above code, we are calculating the KMO. KMO estimates the proportion of variance among all the observed variables.
- The overall KMO for the data is 0.25, which is excellent. This value indicates that you can proceed with your planned factor analysis.

```
In [ ]:    1  df.head(2)
```

Inference:

The KMO value is less than 0.5, and this indicates that we need to delete the insignificant variables.

Finding Significant Variables

```
In [ ]:    1  corr = df.corr()
           2  corr.style.background_gradient(cmap='coolwarm')
```

If your main goal is to visualize the correlation matrix rather than creating a plot per se, the convenient pandas styling options are a viable built-in solution, as shown in the above code.

Creating Dataset of Significant Variables

```
In [ ]:    1  df_corr = df[['radius_mean','perimeter_mean', 'area_mean','radius_worst','perimeter_
```

Running KMO Test

```
In [ ]:    1  from factor_analyzer.factor_analyzer import calculate_kmo
           2  kmo_all,kmo_model=calculate_kmo(df_corr)
           3  kmo_model
```

- In the above code, we are calculating the KMO. KMO estimates the proportion of variance among all the observed variables.
- The overall KMO for the data is 0.82, which is excellent. This value indicates that you can proceed with your planned factor analysis.

```
In [ ]:    1
```

Inference:

The value of the KMO model is greater than 0.5. Therefore, the dataset is good enough for factor analysis.

Factor Analysis

```
In [ ]:    1  from factor_analyzer import FactorAnalyzer
           2  fa = FactorAnalyzer(rotation='varimax')
           3  fa.fit(df_corr,11)
```

In the above code, we create a factor analysis object and perform factor analysis.

```
In [ ]:    1  ev, v = fa.get_eigenvalues()
           2  ev
```

- Here, you can see only for 6-factors eigenvalues are greater than one. It means we only need to choose 6 factors (or unobserved variables).
- In the above code, we are checking the Eigenvalues.
- It measures how much of the variance of the variables a factor explains.
- An eigenvalue of more than one means that the factor explains more variance than a unique variable.

```
In [ ]:    1  plt.scatter(range(1,df_corr.shape[1]+1),ev)
           2  plt.plot(range(1,df_corr.shape[1]+1),ev)
           3  plt.title('Scree Plot')
           4  plt.xlabel('Factors')
           5  plt.ylabel('Eigenvalue')
           6  plt.grid()
           7  plt.show()
```

In the above code, we are creating screen plot using matplotlib module.

Inference:

- Referring eigenvalues: There are only two values above 1. Therefore, we will choose only 2 factors (or unobserved variables).
- Referring Scree plot: There are only two values after the elbow. Therefore, we will choose only 2 factors (or unobserved variables).

```
In [ ]:    1  fa1 = FactorAnalyzer(rotation="varimax", n_factors=2)
           2  fa1.fit(df, 2)
```

```
In [ ]:    1  fa1.loadings_
```

Confirmatory Factor Analysis

- It is used to determine the factors and factor loadings of measured variables.
- The general assumption of CFA is that every factor has an association with a specific subset of the measured variables.
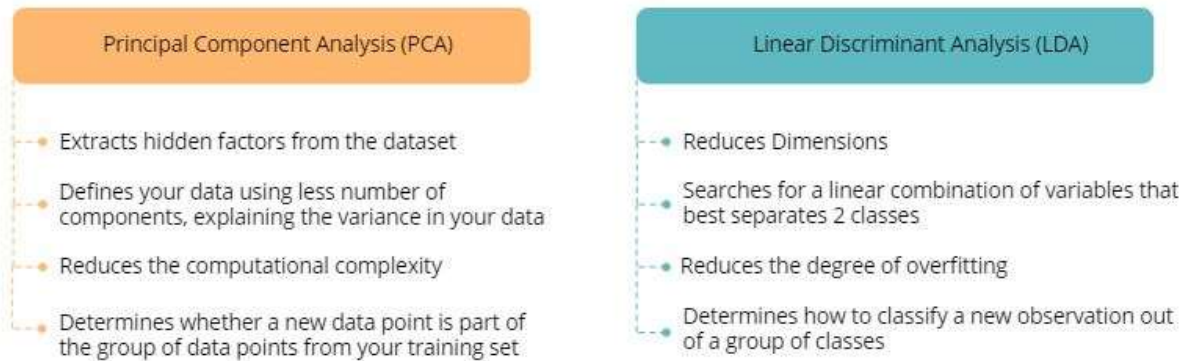
```
In [ ]:    1  from factor_analyzer import (ConfirmatoryFactorAnalyzer, ModelSpecificationParser)
           2
           3  model_dict = {"F1": ['radius_mean','perimeter_mean', 'area_mean','radius_worst','per
           4
           5  model_spec = ModelSpecificationParser.parse_model_specification_from_dict(df_corr,mo
           6
           7
           8  cfa1 = ConfirmatoryFactorAnalyzer(model_spec, disp=False)
           9  cfa1.fit(df_corr.values)
          10
          11  cfa1.loadings_
```
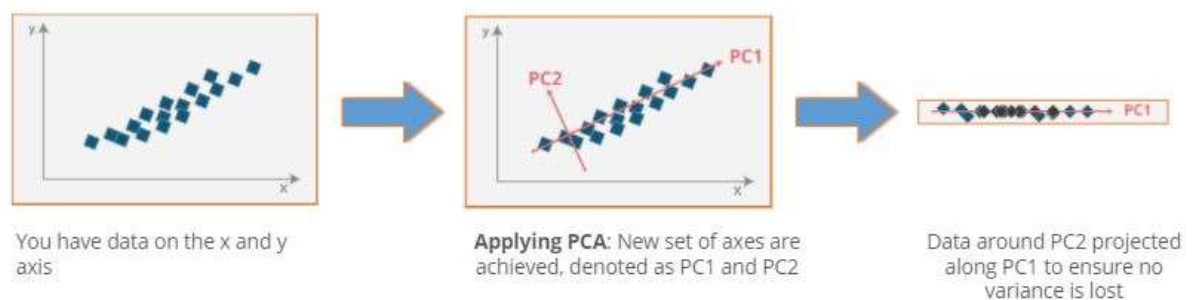
```
In [ ]:    1  cfa1.factor_varcovs_
```

```
In [ ]:    1  cfa1.transform(df_corr.values)
```

Now, let us take a look at PCA and LDA-specific benefits.

PCA and LDA # NO Review

Principal Component Analysis (PCA)	Linear Discriminant Analysis (LDA)
• Extracts hidden factors from the dataset	• Reduces Dimensions
• Defines your data using less number of components, explaining the variance in your data	• Searches for a linear combination of variables that best separates 2 classes
• Reduces the computational complexity	• Reduces the degree of overfitting
• Determines whether a new data point is part of the group of data points from your training set	• Determines how to classify a new observation out of a group of classes

Gist of PCA

You have data on the x and y axis

Applying PCA: New set of axes are achieved, denoted as PC1 and PC2

Data around PC2 projected along PC1 to ensure no variance is lost

What Is Principal Component?

A principal component is a normalized linear combination of the original predictors in a dataset.

First Principal Component (Z^1)

- The aim of PCA is to find components that account for **maximum variance in the data** that includes the error and within-variable variance.
- It finds the direction of the highest variability in the data. Greater the variability captured in the first component, the greater the information captured by the component.
- The first principal component develops a line that is closest to the data points.

  In other words, it minimizes the sum of squared distance between a data point and the line.

Equation for first principal component:

$$Z^1 = \Phi^{11}X^1 + \Phi^{21}X^2 + \Phi^{31}X^3 + ... + \Phi p^1 X p$$

- Z^1 = First principal component
- Φp^1 = Loading vector
- X^1 to Xp = Normalized predictors that have mean equals to **0** and standard deviation equals to **1**.

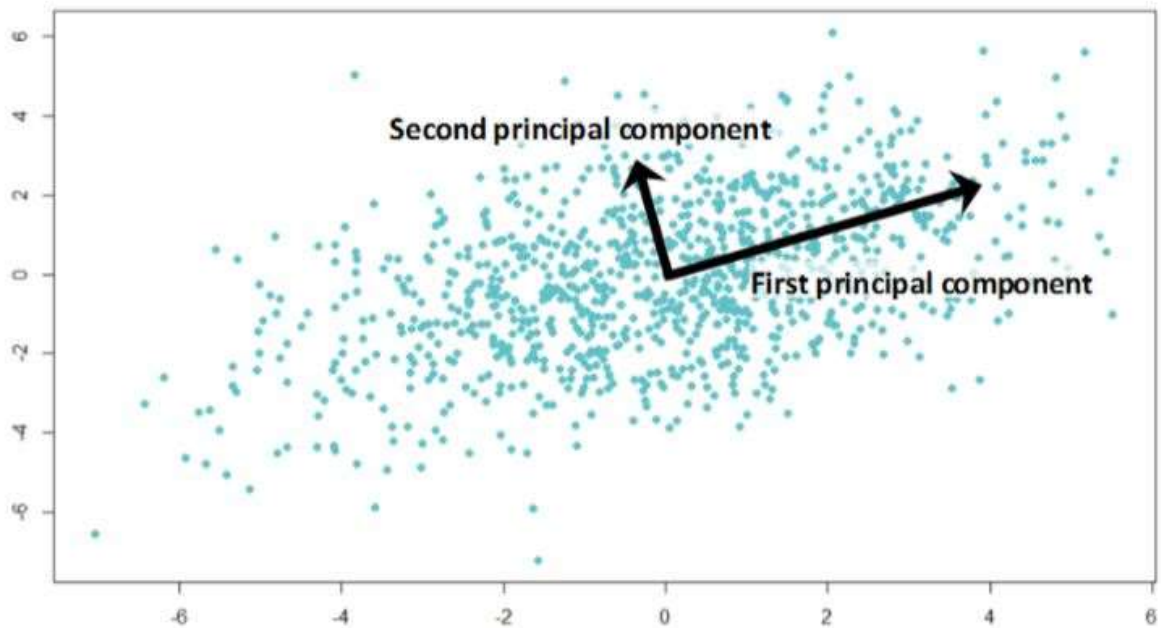### Second Principal Component (Z^2)

The aim of the linear combination of original predictors is to capture the remaining variance in the dataset where Z^2 is uncorrelated to Z^1.

In other words, the correlation between the first and second components should be zero.

Equation for first principal component:

$$Z^2 = \Phi^{12}X^1 + \Phi^{22}X^2 + \Phi^{32}X^3 + ... + \Phi p2Xp$$

Graphical Representation of Uncorrelated Principal Components:

When we talk of all the succeeding principal components, they will follow a similar concept, i.e., capturing the remaining variation without being correlated to the previous component.

How Does PCA work?

PCA is built on the concept of Eigen vectors and Eigen values. So, let us first understand the Eigen vector and value in brief.

Eigenvectors and Eigenvalues

Eigenvectors

When the vectors are plotted on a two-dimensional plane they showcase both the magnitude and direction.

Usually, the direction of a vector is along its span but there are special vectors that on linear transformation get squished or stretched instead of falling off their span.

Eigenvalues

These are the constant values that increase or decrease the Eigenvectors along their span when transformed linearly.

PCA Process

Finding PC1

The first principal component is the direction of maximum variance and is obtained by solving eigenvectors

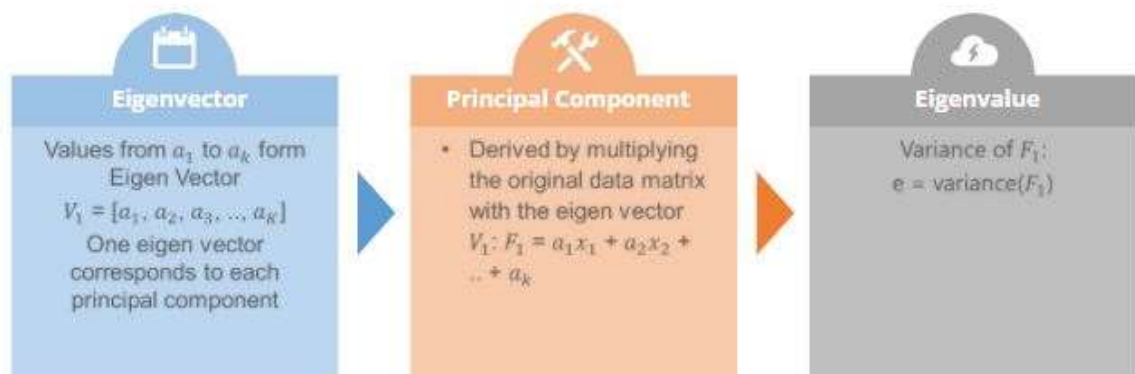PC1: Mathematically,
$$a_1 x_1 + a_2 x_2 + a_3 x_3.. + a_k x_k$$

Constraint:
$$a_1^2 + a_2^2 + .. + a_k^2 = 1$$

Eigen Decomposition is used to solve the above equation

01

02

03

Principal Component 1

Result of Eigen Decomposition

Solution of eigenvectors results into eigenvalues

Eigenvector

Values from a_1 to a_k form Eigen Vector
$$V_1 = [a_1, a_2, a_3, ..., a_k]$$
One eigen vector corresponds to each principal component

Principal Component

- Derived by multiplying the original data matrix with the eigen vector
$$V_1: F_1 = a_1 x_1 + a_2 x_2 + ... + a_k$$

Eigenvalue

Variance of F_1:
$$e = variance(F_1)$$

Eigenvalues are the variances of principal components arranged in descending order

$F_1, F_2, F_3, ..., F_k$

$Var(F_1) > Var(F_2) > Var(F_3) > Var(F_k)$

Eigenvalue 1 > Eigenvalue 2 > Eigenvalue 3 > Eigenvalue k
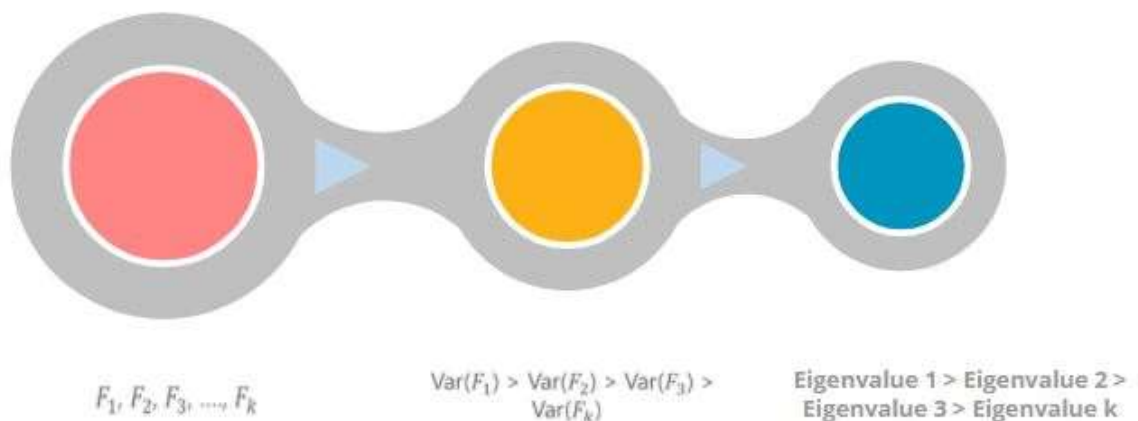
Use Case: Feature Selection in Cancer Dataset Using PCA

Problem Statement

John Cancer Hospital (JCH) is a leading cancer hospital in the USA. It specializes in preventing breast cancer.

Over the last few years, JCH has collected breast cancer data from patients who came for screening or treatment.

However, this data has 32 attributes and is difficult to run and interpret the result. As an ML expert,

you have to reduce the number of attributes so that the results are meaningful and accurate.

Use PCA for feature selection.

Dataset

Features of the dataset are computed from a digitized image of a Fine-Needle Aspirate (FNA) of a breast mass.

They describe the characteristics of the cell nuclei present in the image.

Links to the Dataset

This database is available at:

- The UCI Machine Learning Repository:
  https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29
  (https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29)

Data Dictionary

Dimensions:

- 32 variables
- 569 observations

Attribute Information:

1. ID number
2. Diagnosis (M = malignant, B = benign)
3. Attributes with mean values:
   Ten real-valued features are computed for each cell nucleus:

   - radius_mean (mean of distances from center to points on the perimeter)
   - texture_mean (standard deviation of gray-scale values)
   - perimeter_mean
   - area_mean
   - smoothness_mean (local variation in radius lengths)
   - compactness_mean (perimeter^2 / area - 1.0)
   - concavity_mean (severity of concave portions of the contour)
   - concave points_mean (number of concave portions of the contour)
   - symmetry_mean
   - fractal dimension_mean ("coastline approximation" - 1)

4. Attributes with standard error and worst/largest:

- radius_se
- texture_se
- perimeter_se
- area_se
- smoothness_se
- compactness_se
- concavity_se
- concave points_se
- symmetry_se
- fractal_dimension_se
- radius_worst
- texture_worst
- perimeter_worst
- area_worst
- smoothness_worst
- compactness_worst
- concavity_worst
- concave points_worst
- symmetry_worst
- fractal_dimension_worst

Solution

Import Libraries

> Indented block # CE Review

```
In [ ]:  1  import matplotlib.pyplot as plt
         2  import pandas as pd
         3  import numpy as np
         4  import seaborn as sns
         5  %matplotlib inline
```

Import and Check the Data

Before reading data from a csv file, you need to download the "breast-cancer-data.csv" dataset from the resource section and upload it into the lab. We will use the Up arrow icon which is shown on the left side under the View icon. Click on the Up arrow icon and upload the filewherever it has been downloaded into your system.

After this, you will see the downloaded file will be visible on the left side of your lab along with all the .pynb files.

```
In [ ]:  1  df = pd.read_csv('breast-cancer-data.csv')
         2  df.head()
         3
```

pd.read_csv function is used to read the "breast-cancer-data.csv" file and df.head() will show the top 5 rows of the dataset.

- dataframe or df is a variable that will store the data read by the csv file.
- head will show the rows and () default take the top 5 rows as output.

```
In [ ]:    1  df.shape
```

```
In [ ]:    1  # Check the data , there should be no missing values
           2  df.info()
```

- The dataframe's information is printed using the info() function.
- The number of columns, column labels, column data types, memory use, range index, and the number of cells in each column are all included in the data (non-null values).

```
In [ ]:    1  feature_names = np.array(['mean radius' 'mean texture' 'mean perimeter' 'mean area'
           2   'mean smoothness' 'mean compactness' 'mean concavity'
           3   'mean concave points' 'mean symmetry' 'mean fractal dimension'
           4   'radius error' 'texture error' 'perimeter error' 'area error'
           5   'smoothness error' 'compactness error' 'concavity error'
           6   'concave points error' 'symmetry error' 'fractal dimension error'
           7   'worst radius' 'worst texture' 'worst perimeter' 'worst area'
           8   'worst smoothness' 'worst compactness' 'worst concavity'
           9   'worst concave points' 'worst symmetry' 'worst fractal dimension'])
```

```
In [ ]:    1  #### Convert diagnosis column to 1/0 and store in new column target
           2  from sklearn.preprocessing import LabelEncoder
```

```
In [ ]:    1  # # Encode Label diagnosis
           2  #M -> 1
           3  #B -> 0
```

```
In [ ]:    1  target_data=df["diagnosis"]
           2  encoder = LabelEncoder()
           3  target_data = encoder.fit_transform(target_data)
```

In the above code, in output we are getting all the rows, but only with the last column.

```
In [ ]:    1  df.drop(["diagnosis"],axis = 1, inplace = True)
```

Type *Markdown* and LaTeX: α^2

In the above code, you store the encoded column in a dataframe and drop the diagnosis column for simpilcity.

Principal Component Analysis

Let's use PCA to find the first two principal components and visualize the data in this new, two-dimensional space with a single scatter-plot.

```
In [ ]:    1  from sklearn.preprocessing import StandardScaler
```

In the above code, you will scale data so that each feature has a single unit variance.

```
In [ ]:    1  scaler = StandardScaler()
           2  scaler.fit(df)
```

```
In [ ]:    1  scaled_data = scaler.transform(df)
```

In the above code, you will initially create an object of the StandardScaler() function. Further, you will use fit() along with the object assigned to df and standardize it.

Two Principal Components

```
In [ ]:   1  from sklearn.decomposition import PCA
```

In the above code, you can transform this data into its first 2 principal components.

```
In [ ]:   1  pca = PCA(n_components=2)
```

Now, you will pass the number of components (n_components=2).

```
In [ ]:   1  pca.fit(scaled_data)
```

```
In [ ]:   1  x_pca = pca.transform(scaled_data)
```

Finally, call fit function to aggregate the data. Here, several components represent the lower dimension in which you will project your higher dimension data.

```
In [ ]:   1  #shape of data
          2  scaled_data.shape
```

```
In [ ]:   1  x_pca.shape
```

Plot # NO Review

```
In [ ]:   1  # Reduced 30 dimensions to just 2! Let's plot these two dimensions out!
          2  # Draw inference from the plot?
          3  plt.figure(figsize=(9,6))
          4  plt.scatter(x_pca[:,0],x_pca[:,1],c=target_data,cmap='viridis')
          5  plt.xlabel('First Principal Component')
          6  plt.ylabel('Second Principal Component')
```

Interpreting the components:

Unfortunately, with this great power of dimensionality reduction, comes the cost of being able to easily understand what these components represent.

The components correspond to combinations of the original features. The components themselves are stored as an attribute of the fitted PCA object:

```
In [ ]:   1  pca.components_
```

Explained Variance:

The explained variance tells you how much information (variance) can be attributed to each of the principal components. This is important as you can convert n-dimensional space to 2-dimensional space; you lose some of the variance (information).

```
In [ ]:   1  pca.explained_variance_ratio_
```

Three Principal Components

```
In [ ]:   1  pca_3 = PCA(n_components=3)
          2  pca_3.fit(scaled_data)
          3  x_pca_3 = pca_3.transform(scaled_data)
```

In this Numpy matrix array, each row represents a principal component, and each column relates back to the original features. You can visualize this relationship with a heatmap:

```
In [ ]:    1  x_pca_3.shape
```

What is the total variance attributed by three Components?

```
In [ ]:    1  pca_3.explained_variance_ratio_
```

Gist of LDA

LDA identifies the linear combination of the observed variables that maximize the class separation on the availability of the prior information of the classes.

For example:
A variable in the training dataset that specifies the class of each observation

LDA Process

- Assume a set of D - dimensional samples {X(1, X(2, …, X(N}, N1 of which belong to class ω1 and N2 to class ω2
- Obtain a scalar y by projecting the samples x onto a line: $Y = W^T X$
- Of all the possible lines, select the one that maximizes the separability of the scalars
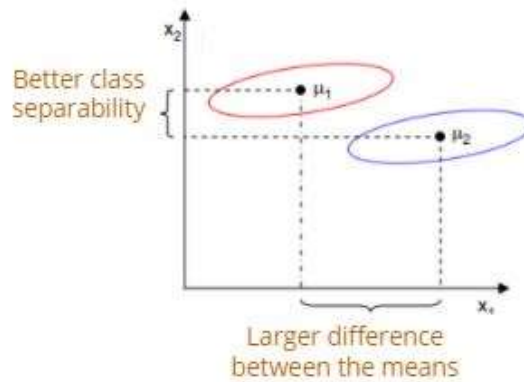
Line of Maximum Separability

The maximum separable line finds out the feature subspace such that class separability is also optimized.
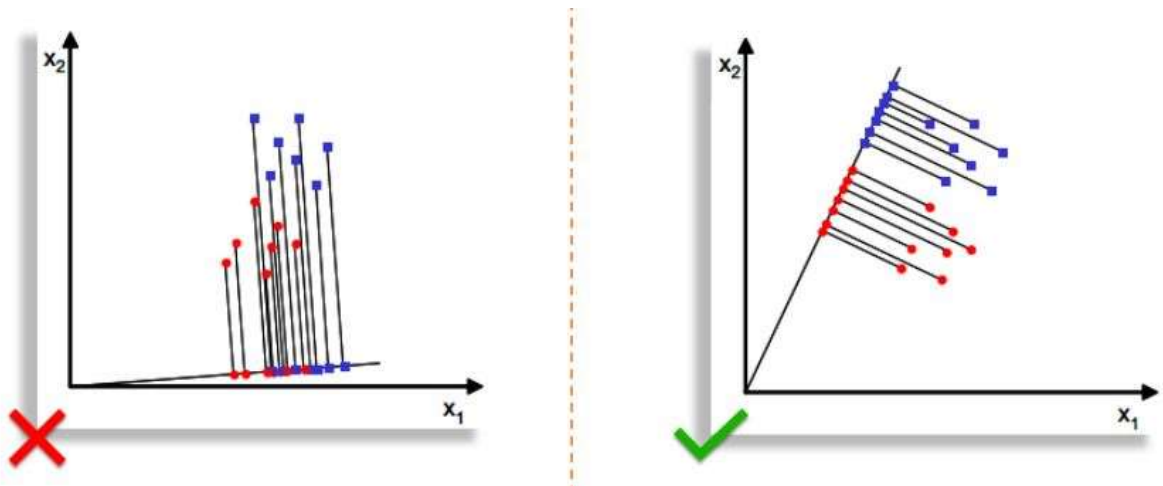
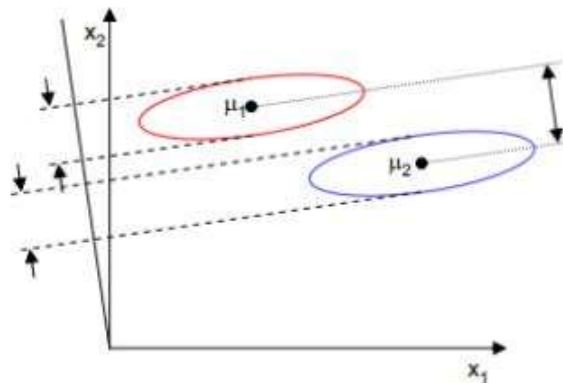### Finding Maximum Separable Line

1. Measure of Separation

- Mean vector within each class of x and y is:

$$\mu_i = \frac{1}{N_i}\sum_{x \in \omega_i} x \quad \text{and} \quad \tilde{\mu}_i = \frac{1}{N_i}\sum_{y \in \omega_i} y = \frac{1}{N_i}\sum_{x \in \omega_i} w^T x = w^T \mu_i$$

- Objective function is the distance between the projected means:

$$J(w) = \left| \tilde{\mu}_1 - \tilde{\mu}_2 \right| = \left| w^T (\mu_1 - \mu_2) \right|$$

2. Linear Discriminant

- Function of difference between the means normalized by measure of scatter (an equivalent variance):

$$J(w) = \frac{\left| \tilde{\mu}_1 - \tilde{\mu}_2 \right|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

Variables from same class are projected very close to each other and at the same time, the projected means are as farther apart as possible

Note: Variables from same class are projected very close to each other and at the same time, the projected means are as farther apart as possible

3. Optimum Projection

- A measure of the scatter in multivariate feature space x:

$$S_i = \sum_{x \in \omega_i} (x - \mu_i)(x - \mu_i)^T$$

$$S_1 + S_2 = S_W$$

S_W is the within class scatter matrix

- Scatter of the projection y can be expressed as a function of the scatter matrix in feature space x

$$\tilde{s}_i^2 = \sum_{y \in \omega_i} (y - \tilde{\mu}_i)^2 = \sum_{x \in \omega_i} \left(w^T x - w^T \mu_i\right)^2 = \sum_{x \in \omega_i} w^T (x - \mu_i)(x - \mu_i)^T w = w^T S_i w$$

$$\tilde{s}_1^2 + \tilde{s}_2^2 = w^T S_W w$$

S_W is the within class scatter

- Difference between the projected means

$$(\tilde{\mu}_1 - \tilde{\mu}_2)^2 = \left(w^T \mu_1 - w^T \mu_2\right)^2 = w^T \underbrace{(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T}_{S_B} w = w^T S_B w$$

4. Obtain the Maxima

- Express the linear discriminant in terms of S_W and S_B:

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

- Find the maxima of J(w) by differentiating and equating to zero

$$w^* = \underset{w}{\mathrm{argmax}} \left\{ \frac{w^T S_B w}{w^T S_W w} \right\} = S_W^{-1}(\mu_1 - \mu_2)$$

Above is the optimal direction for projection

Let us perform LDA using the same dataset that we used for PCA.

Use Case: Feature Selection in Cancer Dataset Using LDA

Note: We are leveraging the problem statement, dataset, and data dictionary from the previous use case.

Problem Statement

John Cancer Hospital (JCH) is a leading cancer hospital in the USA. It specializes in preventing breast cancer.

Over the last few years, JCH has collected breast cancer data from patients who came for screening or treatment.

However, this data has 32 attributes and is difficult to run and interpret the result. As an ML expert,

you have to reduce the number of attributes so that the results are meaningful and accurate.

Use LDA for feature selection.

Dataset

Features of the dataset are computed from a digitized image of a Fine-Needle Aspirate (FNA) of a breast mass.

They describe the characteristics of the cell nuclei present in the image.

Data Dictionary

Dimensions:

- 32 variables
- 569 observations

Attribute Information:

1. ID number
2. Diagnosis (M = malignant, B = benign)
3. Attributes with mean values:
   Ten real-valued features are computed for each cell nucleus:

   - radius_mean (mean of distances from center to points on the perimeter)
   - texture_mean (standard deviation of gray-scale values)
   - perimeter_mean
   - area_mean
   - smoothness_mean (local variation in radius lengths)
   - compactness_mean ($perimeter^2$ / area - 1.0)
   - concavity_mean (severity of concave portions of the contour)
   - concave points_mean (number of concave portions of the contour)
   - symmetry_mean
   - fractal dimension_mean ("coastline approximation" - 1)
4. Attributes with standard error and worst/largest:

   - radius_se
   - texture_se
   - perimeter_se
   - area_se
   - smoothness_se
   - compactness_se
   - concavity_se
   - concave points_se
   - symmetry_se
   - fractal_dimension_se
   - radius_worst
   - texture_worst
   - perimeter_worst

- area_worst
- smoothness_worst
- compactness_worst
- concavity_worst
- concave points_worst
- symmetry_worst
- fractal_dimension_worst

Solution

Import and Check the Data # CE Review

```
In [ ]:   1  import pandas as pd
          2  import numpy as np
          3  from matplotlib import pyplot as plt
          4  from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
          5  from sklearn.model_selection import KFold, cross_val_score, GridSearchCV, train_test
          6  from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
          7  import seaborn as sns
```

In the above code, you are importing different files: pandas, numpy, pyplot, LinearDiscriminantAnalysis, KFold, cross_val_score, GridSearchCV, train_test_split,accuracy_score, classification_report, confusion_matrix and seaborn. To get to know them better, you can start the notebook from the beginning.

```
In [ ]:   1
          2  # load data
          3
          4  df = pd.read_csv('breast-cancer-data.csv')
```

Before reading data from a csv file, you need to download the "breast-cancer-data.csv" dataset from the resource section and upload it into the lab. We will use the Up arrow icon, which is shown on the left side under the View icon. Click on the Up arrow icon and upload the filewhereever it has been downloaded into your system.

After this, you will see the downloaded file will be visible on the left side of your lab with all the .pynb files.

```
In [ ]:   1
          2  df = df.head(559)
```

```
In [ ]:   1  df.shape
```

df.shape will show the number of rows and columns in the dataframe.

```
In [ ]:   1  # map categorical variable 'diagnosis' into numeric
          2
          3  df.diagnosis = df.diagnosis.map({'M': 1, 'B': 0})
```

In the above code, you are encoding the column diagnosis in which we are encoding M as 1 and B as 0.

```
In [ ]:   1  # head will show the rows and () default take 5 top rows as output.
          2  df.head(5)
```

```
In [ ]:   1  df.columns
```

```python
In [ ]:   1  # Drop redundant column 'id'
          2
          3  df.drop('id', axis=1, inplace=True)
```

```python
In [ ]:   1  # Check for NA values
          2
          3  df.isna().any()
```

```python
In [ ]:   1  X_train, X_val, y_train, y_val = train_test_split(df.iloc[:,:-1], df['diagnosis'], t
```

In the above code, you split the data into training and validation sets (since we already have a separate test set).

```python
In [ ]:   1  from sklearn.preprocessing import Normalizer
          2
          3  norm = Normalizer()
          4  norm.fit(X_train)
          5  X_train_norm = norm.transform(X_train)
          6  X_val_norm = norm.transform(X_val)
```

In above code, you are Normalizing the features.

LDA

Let's throw in linear classifiers, i.e., Linear Discriminant Analysis for good measure and because our dataset is small.

```python
In [ ]:   1  lda = LinearDiscriminantAnalysis()
          2  lda.fit(X_train_norm, y_train)
          3  lda_predicted = lda.predict(X_val_norm)
          4
          5  print('LDA Accuracy is: {}'.format(accuracy_score(y_val,lda_predicted)))
          6
          7  print('LDA Classification Report')
          8  print(classification_report(y_val, lda_predicted))
```

- In the above code, you are doing the Linear Discriminant Analysis, in which you find the LDA Accuracy and LDA Classification Report.
- Analogous to the principle of beam search, let us stick with the three best algorithms, LDA, RFC, and GBC, and tune these models further, picking the one that gives the highest accuracy after tuning. Instantiate a new LDA model and check its performance.

```python
In [ ]:   1  confusion_matrix_lda = pd.DataFrame(confusion_matrix(y_val, lda_predicted),
          2                          index = ['Actual Negative','Actual Positive'],
          3                      columns = ['Predicted Negative','Predicted Postive'] )
          4  confusion_matrix_lda
```

```python
In [ ]:   1
```

Note: In this lesson, we saw the use of the feature selection methods, but in the next lesson we are going to use one of these methods as a sub-component of "Supervised Learning - Regression and Classification".