Artificial Intelligence
Chapter 2 - Feature Engineering

Dr. Adnan Ismail Al-Sulaifanie

Department of Electrical and Computer Engineering
University of Duhok, Iraq

Outline

- ➤ Definition of features
- ➤ Necessity of feature engineering.
- ➤ Data preprocessing - imputation, scaling, and encoding.
- ➤ Feature selection methods – filter based, wrapper based, and embedded feature selection.
- ➤ Feature extraction.
- ➤ Dimensionality reduction methods – PCA and LDA

References

1. Max Kuhn and Kjell Johnson, **Feature Engineering and Selection - A Practical Approach for Predictive Models**, 2021
2. Jason Brownlee **Data Preparation for Machine Learning – Data Cleaning, Feature Selection, and Data Transforms in Python**, 2020
3. scikit-learn user guide

1 Features

- Feature is an attribute or variable used to describe some aspect of individual data objects.

  Examples – age and eye color for persons, major and grade point average for students, length and width of house, ...

- These features serve as input to the ML model and play a vital in producing an accurate model.

- Types of features:

  1. Numerical features have continuous values measurable on a scale, such as height, weight, or income.
  2. Categorical features take a limited number of values. These values are discrete and belong to specific categories, such as gender, color, city, etc.

- Feature challenges:

  - Some dataset have missing values.
  - Some ML algorithms cannot handle categorical features
  - Not all features are on the same scale.
  - Some features are irrelevant, while others impact the target.
  - Not all features are equally importance for a prediction task, and some may introduce noise into the model.
  - Some features are highly correlated (correlated).

- Directly applying raw input data to the model may be insufficient.

- Choosing the right features can improve the performance accuracy and interpretability of ML algorithms.

- The success of ML models depends on the quality of features used during training phase.

- **Feature engineering** is the processes applied to raw data/features to address previous problems. These processes include imputation, scaling, extraction, selection, transformation, ...

- Feature Engineering aims to enhance the performance of ML models by providing it with more informative and discriminative input feature. It improves the model's ability to understand patterns, make accurate predictions, and generalize well to new, unseen data.

2 Feature Imputation

- For various reasons, many real world datasets contain missing values, often encoded as blanks, NaNs or other placeholders.

- Such datasets however are incompatible with scikit-learn estimators which assume that all values in an array are numerical and available.
- A basic strategy to use incomplete datasets is to discard entire rows and/or columns containing missing values. However, this comes at the price of losing data which may be valuable (even though incomplete).
- A better strategy is to impute the missing values. **Feature imputation** is the process applied to estimate or replace those missing values.
- Common methods for feature imputation include statistical measures such as constant, mean, median, or most frequent value of the column.
- **SimpleImputer** class in scikit-learn can replace missing values in each column of the datasets.

```
import numpy as np
from sklearn.impute import SimpleImputer
X = np.array([1, 2, 2,4, np.nan, 6])
X = X.reshape(-1,1)
imp = SimpleImputer(missing_values = np.nan, strategy='mean') # or median
imp.fit(X)
Xnew = imp.transform(X)
print(Xnew)

[1. 2. 2. 4. 3. 6.]
```

- If the dataset is large enough, it is preferred to remove raws with missing value(s).

3 Feature Encoding

- Machine learning models can only work with numerical values. For this reason, it is necessary to transform the categorical features into numerical ones. This process is called feature encoding.
- **Label encoding** assigns a unique numerical to each category. These numerical values are usually assigned in alphabetical order or the order of appearance.

  Blue 0
  Green 1
  Red 2
- The model might interpret the categories as being ordered (0 < 1 < 2), which is not necessarily the case. Label encoding is particularly useful when the categories have a natural ordered relationship (like 'Low', 'Medium', 'High').
- For nominal data where there is no intrinsic order (like 'Red', 'Green', 'Blue'), one-hot encoding might be more appropriate
- **One-Hot Encoding** creates a binary column for each category and assigns a **1** or **0** to indicate the presence of a feature.

```
from numpy import asarray
from sklearn.preprocessing import OneHotEncoder

data = asarray([['red'], ['green'], ['blue']])
print(data)
encoder = OneHotEncoder(sparse_output=False)
onehot = encoder.fit_transform(data)
print(onehot)
```

```
'red'     0  0  1
'green'   0  1  0
'blue'    1  0  0
```

➤ **Ordinal Encoding** is similar to label encoding, but specifically used for ordinal data where the categories have a meaningful order.

```
from numpy import asarray
from sklearn.preprocessing import OrdinalEncoder
data = asarray([['red'], ['green'], ['blue']])
print(data) # red   green  blue
encoder = OrdinalEncoder()
result = encoder.fit_transform(data)
print(result) #  2   1   0
```

```
'red'     2
'green'   1
'blue'    0
```

4 Feature Scaling

➤ Different variables affect prediction outcomes based on their values.

➤ Features with larger ranges can dominate those with smaller ranges in machine learning models, leading to biased results.

➤ **Feature Scaling –** transforms the range and/or distribution of features. It is applied independently to each column.

➤ It is a crucial preprocessing step in machine learning that involves transforming the features of a dataset to fit within a specific range or distribution. This process ensures that all features contribute equally to the model, preventing any single feature from dominating the learning process due to its scale.

➤ Advantages:

- ➢ Ensures equal contribution of each feature to the model.
- ➢ Crucial for algorithms using gradient descent.
- ➢ Mitigates the impact of outliers by standardizing features.

➤ Scaling methods:

1. Normalization

2. Standardization
3. Transformation

➤ The selection of the appropriate scaling method depends on factors such as the type and distribution of features, and the machine learning algorithm that will utilize the scaled data.

4.1 Normalization

➤ Transforms the feature values to have a minimum of zero and a maximum of one.

$$xn == \frac{x - x_{min}}{x_{max} - x_{min}}$$

➤ It also can transform the feature values to a specified range, such as [-1, 1] or [0, 100].

➤ It is useful for for algorithms that are sensitive to the scale of the features.

Example 1 - Write a python code to transform the following array into [0, 1].

```
from sklearn.preprocessing  import MinMaxScaler
import  numpy as np

x = np.arange(10).reshape(-1,1) - 5
scaler  = MinMaxScaler(feature_range=(0, 1), copy=True)
xn = scaler.fit_transform(x)
print('x = ' ,  x.T)
print('xn = ', xn.T)
print('mean-x = ', x.mean(), '      mean-xn = ', xn.mean())
print('std-x = ', x.std(),'     std-xn = ', xn.std())
```

```
x =  [[-5 -4 -3 -2 -1  0  1  2  3  4]]
xn =  [[0.0   0.111  0.222   0.333   0.444   0.556   0.667   0.778   0.889   1.]]
mean-x =  -0.5       mean-xn =  0.5
std-x =  2.8722813232690143      std-xn =  0.31914236925211265
```

How we can change the range to [-1,+1]

Homework 1 - Repeat previous program after setting x[0] = -50 and x[0] = 40. Analyze results.

4.2 Maximum Absolute Scaler

➤ scale each feature by its maximum absolute value.

$$xs = \frac{x}{max(|x|)}$$

```
from sklearn.preprocessing  import MaxAbsScaler
import  numpy as np

x = np.arange(10).reshape(-1,1) - 5
scaler  = MaxAbsScaler()
xs = scaler.fit_transform(x)
print(xs)
print('std = ', xs.std())
print('mean = ', xs.mean())

x =  [[-5 -4 -3 -2 -1  0  1  2  3  4]]
xs =  [[-1.  -0.8 -0.6 -0.4 -0.2  0.   0.2  0.4  0.6  0.8]]
mean-x =  -0.5        mean-xs =  -0.099
std-x =  2.87228      std-xs =  0.5745
```

4.3 Standardization

- ➤ transforms the feature values to have a mean of zero and a standard deviation of one. After standardization, all columns are important equally.

$$xs = \frac{x - \mu_x}{\sigma_x}$$

- ➤ This method is essential for algorithms that assume the features are normally distributed, such as linear regression, logistic regression, and support vector machines.

```
from sklearn.preprocessing  import StandardScaler
import  numpy as np

x = np.arange(10).reshape(-1,1) - 5
scaler  = StandardScaler()
xs = scaler.fit_transform(x)
print('x = ' ,  x.T)
print('xs = ', xs.T)
print('std = ', xs.std())
print('mean = ', xs.mean())

x =  [[-5 -4 -3 -2 -1  0  1  2  3  4]]
xs =  [[-1.567   -1.2185   -0.870   -0.522  -0.174   0.174   0.522   0.870   1.219   1.567 ]]
mean-x =  -0.5        mean-xs =  -6.661338147750939e-17
std-x =  2.8723       std-xs =  1.0
```

4.4 Features Transformation

- ➤ Commonly used mathematical transformation

1. Logarithmic $\mathbf{log}(x)$
2. Square root $\sqrt{x}$
3. Reciprocal $\frac{1}{x}$
4. Exponential e^x

5 Feature Selection

- ➤ Not all features in a dataset contribute equally to a model's predictive power.

- Only a subset of features are typically relevant to the target variable.
- Least significant features might be:
  1. **Constant features –** have identical values across all instances, offering no discriminatory information.
  2. **Features with low variance –** exhibit minimal variability and are unlikely to significantly impact model outcomes.
  3. **Redundant features –** refer to features that are highly correlated with others, essentially offering duplicate information.
  4. **Irrelevant features –** have no meaningful connection to the target variable, adding noise to the model.
- Correlated features are **multicollinear**, share a strong and linear relationship, providing redundant information to the model.
- Including irrelevant/correlated features into the model leads to:
  1. Increases model complexity and interpretation difficulty.
  2. Extends training and prediction times.
  3. Results in inaccurate and less reliable models.
- **Feature selection –** is the process of choosing a subset of relevant features for model building, aiming to retain the most pertinent information by eliminating redundant, irrelevant, or noisy features from the original set.
- Advantages of feature selection:
  - Simplifies models, reducing training and prediction times.
  - Optimizes model performance by removing noise.
  - Improves model accuracy with the right subset.
  - Enhances model interpretability.
  - Reduces overfitting.
- Feature selection methods: filter-based, wrapper-based, and embedded methods.

5.1 Filter Methods

- These methods can be applied without considering the ML algorithm to be used.
- Features are chosen based on their statistical properties.
- Each feature is assigned a score based on one of the following:
  1. The amount of information contained in each feature (variance).
  2. The relationship between each individual feature and the target (correlation).

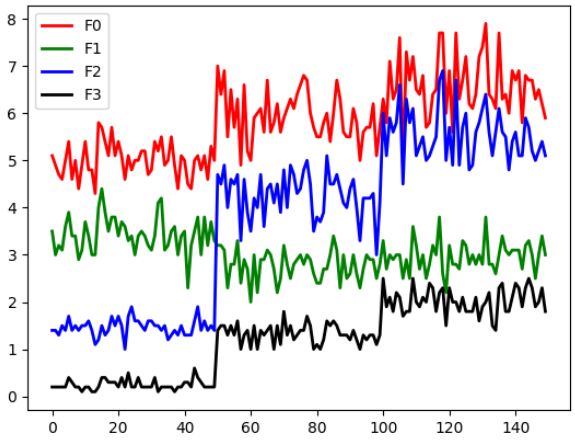
- The obtained scores are utilized to rank the features, thereby limiting the number to be incorporated into the model.
- Advantages of filter-based feature selection:
    1. These methods are easy to implement
    2. Computationally fast and less expensive compared to wrapper methods.
    3. Handles large datasets well.
    4. Independent of learning algorithm. They assess feature relevance independently of the model being used.
    5. Reducing feature space making it simpler to understand the impact of each feature on the model.
- Disadvantages:
    1. Doesn't consider relationships between features. Thus, it cannot remove redundant features.
    2. The scores of feature relevance may not perfectly align with model performance.

5.1.1 Variance thresholding

- Selects features based on their variance, assuming that features with higher variance are more likely to contain valuable information.
- It eliminates all features with a variance below a specified threshold.
- By default, it excludes features with zero variance, meaning those with identical values across all samples.
- Variance thresholding is applicable to unsupervised learning as it focuses solely on the features without considering the desired outputs.

Example 2 - write a python code to remove the features with standard deviation less than 0.5 from iris dataset.

```
from sklearn import datasets
from sklearn.feature_selection import VarianceThreshold
import numpy as np

iris = datasets.load_iris()
X = iris.data
y = iris.target
thresholder = VarianceThreshold(threshold = 0.5)
Xnew = thresholder.fit_transform(X)
print('X-size = ', X.shape)  # X-size =  (150, 4)
print('Xnew-size = ', Xnew.shape)  # Xnew-size =  (150, 3)
print('X[0] = ', X[0])  #X[0] =  [5.1 3.5 1.4 0.2]
print('Xnew[0] = ', Xnew[0])  # Xnew[0] =  [5.1 1.4 0.2]
print(np.std(X,axis = 0)) # [0.82530129 0.43441097 1.75940407 0.75969263]
```



If we keep two features only, which one should be removed?

Example 3 - write a python code to remove the binary feature with probability = 0.8 using variance threshold.

```
from sklearn.feature_selection import VarianceThreshold

X = [[0, 0, 1], [0, 1, 0], [1, 0, 0], [0, 1, 1], [0, 1, 0], [0, 1, 1]]
thresholder = VarianceThreshold(threshold=(.8 * (1 - .8)))
Xnew = thresholder.fit_transform(X)
```

X			Xnew	
0	0	1	0	1
0	1	0	1	0
1	0	0	0	0
0	1	1	1	1
0	1	0	1	0
0	1	1	1	1

5.1.2 P-value

- measures the significance of the relationship between a feature and the target variable in a regression model.
- The p-value indicates the probability of observing a correlation between them, assuming that the null hypothesis (no correlation) is true.

$$\text{p-vale}(x, y) = \begin{cases} < 0.05 & x \quad \text{significant} \\ \geq 0.05 & x \quad \text{insignificant} \end{cases}$$

- A small p-value (p-value < 0.05) means that the correlation is unlikely to occur by chance, the relationship is significant.
- A high p-value (p-value $\geq$ 0.05) means that they are not significantly correlated. Therefore, they can be eliminated.

5.1.3 Pearson correlation coefficient

➤ shows direction and strength between two variables (x, y).

$$r = \frac{cov(x,y)}{\sigma_x * \sigma_y} = \frac{\sum_{i=1}^{N}(x_i - \mu_x).(y_i - \mu_y)}{\sqrt{\sum_{i=1}^{N}(x_i - \mu_x)^2} * \sqrt{\sum_{i=1}^{N}(y_i - \mu_y)^2}}$$

$r \in [-1, 1]$

where -1 means there is a strong negative relation between x and y, while +1 indicates strong positive relation. It is best suited when there is a linear relation.

Example 4 - write python code to compute the correlation between features and the target in diabetes dataset

```
import numpy as np
import scipy.stats as stats
from sklearn.datasets import load_diabetes

data = load_diabetes()()
X, y = data.data, data.target
N = X.shape[1]
R , P= np.zeros(N) , np.zeros(N)
for n in range(N):
   R[n], P[n] = stats.pearsonr(X[:,n], y)
print('Correlation =', np.round(R,3))
print('P-values = ', np.round(P,3))
```

```
Correlation = [0.188   0.043  0.586  0.441  0.212  0.174  -0.395  0.43   0.566  0.382]
P-values =   [0.       0.366  0.     0.     0.     0.      0.     0.     0.     0.   ]
```

Example 5 - repeat previous example using pandas library.

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_diabetes

data = load_diabetes()
X, y = data.data, data.target
df = pd.DataFrame()
for n in range(X.shape[1]):
      df['F'+str(n)] = pd.Series(X[:,n])

df['Y'] = pd.Series(y)
print(np.round(df.corr(),3))
```

	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	Y
F0	1.000	0.174	0.185	0.335	0.260	0.219	-0.075	0.204	0.271	0.302	0.188
F1	0.174	1.000	0.088	0.241	0.035	0.143	-0.379	0.332	0.150	0.208	0.043
F2	0.185	0.088	1.000	0.395	0.250	0.261	-0.367	0.414	0.446	0.389	0.586
F3	0.335	0.241	0.395	1.000	0.242	0.186	-0.179	0.258	0.393	0.390	0.441
F4	0.260	0.035	0.250	0.242	1.000	0.897	0.052	0.542	0.516	0.326	0.212
F5	0.219	0.143	0.261	0.186	0.897	1.000	-0.196	0.660	0.318	0.291	0.174
F6	-0.075	-0.379	-0.367	-0.179	0.052	-0.196	1.000	-0.738	-0.399	-0.274	-0.395
F7	0.204	0.332	0.414	0.258	0.542	0.660	-0.738	1.000	0.618	0.417	0.430
F8	0.271	0.150	0.446	0.393	0.516	0.318	-0.399	0.618	1.000	0.465	0.566
F9	0.302	0.208	0.389	0.390	0.326	0.291	-0.274	0.417	0.465	1.000	0.382
Y	0.188	0.043	0.586	0.441	0.212	0.174	-0.395	0.430	0.566	0.382	1.000

Pandas correlation function is efficient because

1. It computes the correlation between features and target which helps to select best subset of features.
2. It computes the correlation between features which enables removing redundant and highly correlated features.

5.1.4 Chi-square test

➤ Is a statistical test used to determine if there is a significant association between two categorical variables.

The sklearn library provides **SelectKBest** function to select features according to the k highest scores

sklearn.feature_selection.SelectKBest(score_func = 'method', k = 10)

Supported score functions for regression: f_regression, mutual_info_regression

and for classification: chi2, f_classif, mutual_info_classif

Example 6 - write python code to keep K highest scoring features in iris dataset.

```
from sklearn.datasets import load_iris
from sklearn.feature_selection import SelectKBest, f_classif

iris = load_iris()
X = iris.data
y = iris.target
X = X.astype(int)
selector = SelectKBest(f_classif, k = 2)
Xnew = selector.fit_transform(X, y)
print('X-size = ', X.shape)  # X-size =  (150, 4)
print('Xnew-size = ', Xnew.shape) # # Xnew-size =  (150, 2)
```

5.2 Wrapper Methods

- ➤ Use model performance to find the optimal subset of features.
- ➤ Train models to evaluate different feature subsets.
- ➤ More accurate than filter methods.
- ➤ Test all possible feature combinations to find the best subset.
- ➤ Characteristics
  - ➢ Model dependent – tailored to specific models.
  - ➢ Computationally expensive – time-consuming due to evaluating numerous models.
- ➤ Advantages
  - ➢ Better performance compared to filter methods.
  - ➢ Provide the best-performing feature set for a specific model.
- ➤ Disadvantages
  - ➢ High computational complexity due to evaluating many models.
  - ➢ Increased risk of overfitting.

5.2.1 Forward Feature Selection

- ➤ Find best features those can produce highest increase in the performance
- ➤ Procedure

1. Start with an empty set of selected features.
2. For each iteration, add one feature and train the model using the current set of selected features plus the new feature. Evaluate the model's performance using the chosen performance metric.
3. Identify the feature that, when added to the selected set, results in the best improvement in the model's performance. Add this feature to the set of selected features.
4. Repeat the iterative selection process until a stopping criterion is met. Common stopping criteria include:
   (a) A predefined number of features is selected.
   (b) The addition of new features does not significantly improve the model's performance.
   (c) The model's performance starts to degrade with the addition of new features.

5.2.2 Backward Feature Elimination

- **Approach –** Starts with all features in the dataset and iteratively removes the least significant feature at each step.
- **Selection Criterion –** The "least significant" feature is usually determined by a statistical measure (e.g., p-value in a linear model) or based on the importance score assigned by a chosen model (e.g., coefficient magnitude in a linear model, feature importance in a tree-based model). After removing the feature, the model is retrained, and the process is repeated until a desired number of features remains or a stopping criterion is met (e.g., no significant improvement in model performance).
- **Key Characteristic –** Evaluates features based on their individual contribution to the model when all other features are present.
- **Computational Cost –** Can be computationally expensive, especially with a large number of features, as the model is retrained after each feature removal.
- **Model Dependency –** Requires a model that provides feature importance or allows for statistical significance testing of features. Common choices include linear regression (using p-values or coefficients), logistic regression, or tree-based models (using feature importance).
- **Procedure –**

  1. Train the model using all features.
  2. Identify the least important feature based on the selected criterion.
  3. Remove the least important feature.
  4. Retrain the model with the remaining features.
  5. Evaluate model performance.
  6. Repeat steps 2-5 until the desired number of features is reached or a stopping criterion is met.

5.2.3 Recursive Feature Elimination (RFE)

- **Approach –** Similar to BFE, RFE starts with all features. However, instead of relying solely on the individual importance of a feature, RFE recursively fits a model (e.g., a linear model, SVM, or a tree-based model) and removes the weakest feature based on its coefficients (or feature importance). The procedure is repeated on the remaining set of features until the desired number of features is eventually reached.
- **Selection Criterion –** The "weakest" feature is determined based on the coefficients or feature importances assigned by the model after each recursive fitting. Features with smaller absolute coefficient values (in linear models) or lower importance scores (in tree-based models) are considered less important.

- **Key Characteristic –** Considers the interaction and relationships between features through the fitted model's coefficients or feature importances at each iteration. It's more focused on the model's learned representation of feature importance than a purely statistical measure.
- **Computational Cost –** Generally more computationally expensive than basic BFE because the model is fit recursively on progressively smaller feature sets.
- **Model Dependency –** Requires a model that provides coefficients or feature importance scores. Common choices include linear regression, logistic regression, support vector machines (SVMs), and tree-based models. Note that RFE with SVMs is a common and powerful approach.
- **Procedure –**

  1. Train the model using all features.
  2. Rank features based on their coefficients (or feature importance).
  3. Eliminate the least important feature (or a subset of least important features).
  4. Retrain the model with the remaining features.
  5. Repeat steps 2-4 recursively until the desired number of features is reached.

- Benefits

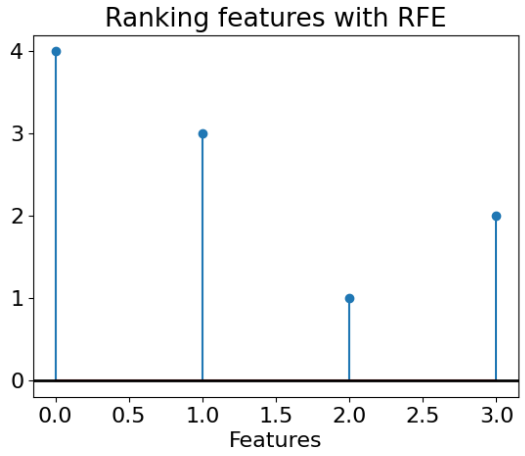
  1. Evaluates each feature's impact on model performance.
  2. Develops a model with the most significant features.

sklearn.feature_selection.RFE(estimator, n_features_to_select = None, step = 1)

Example 7 - write python code using RFE to remove least significant features from iris dataset

```
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE
import matplotlib.pyplot as plt

X, y = load_iris(return_X_y = True)
estimator = LogisticRegression(max_iter=1000)
rfe = RFE(estimator, n_features_to_select=1)
rfe.fit(X, y)
print('Selected features:', rfe.support_)
print('Feature ranking:', rfe.ranking_)
plt.stem(rfe.ranking_)
plt.title("Ranking of pixels with RFE")
plt.show()
```

```
Selected features: [False False  True False]
Feature ranking:  [ 4     3     1     2 ]
```



Homework 2 - repeat previous example using make_classification function to create dataset of 1000 sample and 10 features. Rank features based on their importance.

Example 8 -

```
from sklearn.datasets import make_classification
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression


X,y = make_classification(n_samples =1000, n_features = 100, n_informative = 2)
estimator = LogisticRegression(max_iter = 500)
selector = RFE(estimator, n_features_to_select = 5, step = 1)
X_new = selector.fit_transform(X, y)
```

	BFE	**RFE**
Selection Criterion	Statistical measures (e.g., p-values) or individual feature importance based on a pre-trained model. Focuses on the individual contribution of a feature in the presence of all others.	Model coefficients or feature importance, determined recursively by fitting the model at each iteration. Considers feature interactions learned by the model.
Recursion	Typically does not involve fitting a new model for importance after every removal. It can use the feature importances from the initial model or refit less frequently.	Always involves recursively fitting the model and re-calculating feature importance at each iteration (or a specified number of iterations).
Computational Cost	Generally less expensive than RFE (unless refitting for each step)	Generally more expensive than BFE

5.2.4 Exhaustive Feature Selection

➤ Evaluates each possible combination of features and returns the best performing feature set according to a specified performance metric.

- ➤ It is one of the best feature selection methods, but also one of the most computationally expensive, as it requires testing $2^n - 1$ feature subsets for a dataset with n features.
- ➤ Example – if the dataset consists of 4 features, the algorithm will evaluate all 15 feature combinations.

5.3 Embedded Methods

- ➤ Embedded methods perform feature selection during the model training process.
- ➤ They reduce data dimensionality and improve model performance by selecting relevant features.
- ➤ Examples – LASSO, Decision Trees, Neural Networks.
- ➤ Advantages
    1. No separate feature selection step required.
    2. Optimizes feature selection for the specific model and data.
    3. Regularizes model complexity to avoid overfitting.
- ➤ Limitations
    1. Selected features may not be optimal for other models or datasets.
    2. Feature selection criteria may be unclear.
    3. Feature selection process may not be easily modified or customized.

6 Feature Extraction

- ➤ Using raw pixel data for image recognition can be inefficient. The arrangement of pixels, rather than their absolute values, often holds significant information.
- ➤ **Feature Extraction –** is the process of generating new features from existing ones to enhance model accuracy. Transforms raw data into meaningful and informative features for better model training.
- ➤ **Example –** Raw features like length and width of a house may not be highly informative. Feature extraction can derive more meaningful features, such as the area, to improve model performance.
- ➤ Feature extraction involves operations like addition, subtraction, multiplication, and ratio. In image processing, includes identifying edges, corners, and patterns.
- ➤ Advantages of Feature Extraction:
    1. Reduces data processing time, making model training more efficient.

2. Improves accuracy and generalization by removing irrelevant information.

➤ Common Image Feature Extraction Methods:

1. Fourier Transform
2. Wavelet Transform
3. Convolution
4. Autoencoders
5. Local Binary Pattern
6. Histogram of Oriented Gradients
7. Speed Up Robust Features
8. Color Histogram

7 Dimensionality Reduction

➤ High-Dimensional Data increases computational complexity, risk of overfitting, and difficulties in visualization and interpretation.

➤ The possible solution is to reduce the number of features to improve data manageability and analysis.

➤ **Dimensionality reduction –** transforms a dataset by reducing features while preserving essential information. It creates new features from the original ones.

➤ Benefits of Dimensionality Reduction:

1. Reduces data and model complexity, improving learning algorithm speed and accuracy.
2. Helps avoid overfitting.
3. Enhances data visualization and understanding.

➤ Common Techniques

1. Principal Component Analysis (PCA) – used for regression.
2. Linear Discriminant Analysis (LDA) – used for classification.

7.1 Principal Component Analysis (PCA)

➤ PCA Overview

➢ Statistical procedure using orthogonal transformation to convert correlated variables into uncorrelated ones.

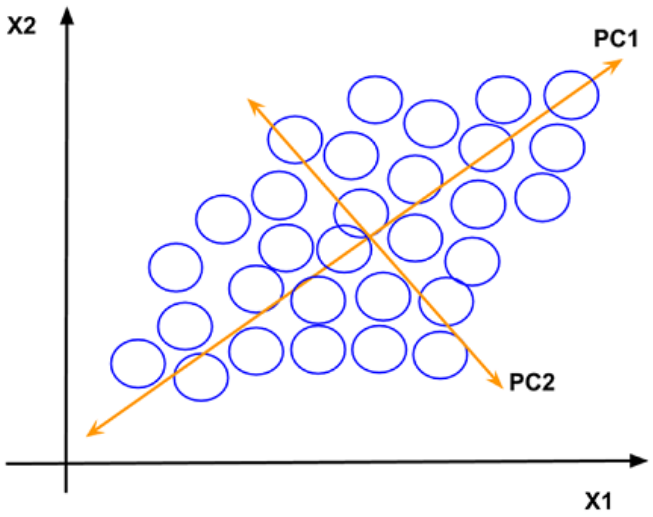
➢ Creates new features called Principal Components (PCs).

➢ Applied to unsupervised datasets.

- ➢ Aims to reduce variables while preserving information, though some accuracy is lost.

➤ Characteristics of Principal Components (PCs)

- ➢ Linear combinations of original features.
- ➢ Orthogonal (uncorrelated) to each other.
- ➢ Ordered by variance: PC1 captures the most variation, PC2 captures the next highest variance, and so on.



➤ Benefits of Using PCs

- ➢ Reduces data dimensionality, aiding visualization, analysis, and modeling.
- ➢ Summarizes data using a limited number of PCs.

➤ PCA Procedure

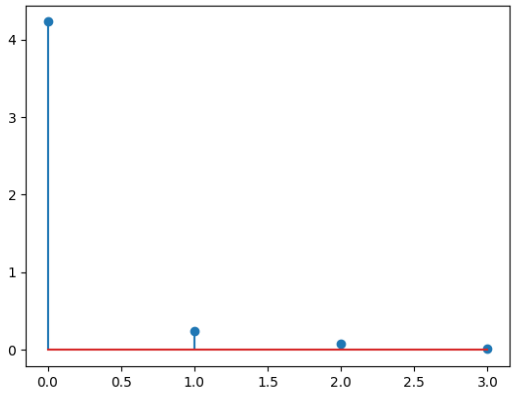
1. Standardize each dataset column – $xs = \frac{x - \mu}{\sigma}$
2. Calculate the covariance matrix to show feature relationships.
3. Decompose the covariance matrix into eigenvectors and eigenvalues.
4. Select principal components based on the highest eigenvalues.
5. Transform original data into a new lower-dimensional space using selected PCs.

➤ Limitations – PCs are less interpretable and lack real-world meaning.

Example 9 - write python code using PCA to reduce the dimensionality of iris dataset.

```
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
import numpy as np

iris = load_iris()
X = iris.data
y = iris.target
pca = PCA()
Xpca = pca.fit_transform(X)
print('pca-variance = ', pca.explained_variance_)
PC0 = pca.explained_variance_[0]
plt.stem(pca.explained_variance_)
plt.show()

counter = 0
for x in pca.explained_variance_:
    if x / PC0 > 0.05:  counter += 1

pca.n_components = counter
Xpca = pca.fit_transform(X)

print('x-shape = ', X.shape)
print('x-pca-shape = ', Xpca.shape)
print('X[0] = ', X[0])
print('Xpca[0] = ', Xpca[0])
```

```
pca-variance =  [4.22824171  0.24267075  0.0782095   0.02383509]
x-shape =  (150, 4)
x-pca-shape =  (150, 2)
X[0] =  [5.1  3.5  1.4  0.2]
Xpca[0] =  [-2.68412563  0.31939725]
```



7.2 Linear Discriminant Analysis (LDA)

- ➤ The goal of LDA is to find the best linear combination of features that separate different classes (categories) in the data.
- ➤ LDA aims to maximize the between-class variance while minimizing the within-class variance, effectively creating a space where classes are as distinguishable as possible.
- ➤ LDA requires class labels (target variables) to perform its calculations. It uses this information to find the optimal directions to project the data.
- ➤ It focus on maximizing class separability. Therefore, it is excellent for classification problems, especially when you want to reduce dimensionality while preserving class information.
- ➤ Example – Imagine you have data about flowers, with features like ”petal length,” ”petal width,” ”sepal length,” and ”sepal width,” and you want to classify them into different species

(e.g., Iris setosa, Iris versicolor, Iris virginica). LDA will find the linear combination of these features that best separates the different species.