Chapter 4

Basics of Feature Engineering

OBJECTIVE OF THE CHAPTER

In the last three chapters, you have been introduced to the basic concepts of machine learning. Also, the process to start modelling a problem has also been discussed in details. With this context in mind, in this chapter, we will introduce you to another very important aspect of machine learning, that is feature engineering. Though not a core part of the machine learning processes, feature engineering is a critical allied task that we need to perform to make learning more effective. It has three key components – feature construction, feature selection, and feature transformation, each of which will be covered in details in this chapter.

4.1 INTRODUCTION

In the last three chapters, we had a jumpstart to the machine learning process. We first started with what human learning is and how the different types of machine learning emulate the aspects of human learning. We had a detailed view of the different types of problem that can be solved using machine learning techniques. Before applying machine learning to solve the problems, there are certain preparatory steps. These preparatory steps have been covered in details. After that, we have done a step-by-step navigation of the different activities of modelling a problem using machine learning. Modelling alone doesn't help us to realize the effectiveness of machine learning as a problem- solving tool. So we also learnt how to measure the effectiveness of machine learning models in solving problems. In case a specific model is not effective, we can use different levers to boost the effectiveness. Those levers of boosting the model performance were also covered.

Now that we are ready (well almost ready!) to start solving problems using machine learning, we need to touch upon another key aspect which plays a critical role in solving any machine learning problem – feature engineering. Though feature engineering is a part of the preparatory activities which have already been covered in Chapter 2, the criticality and vastness of the area call for treating it separately. This area deals with features of the data set, which form an important input of any machine learning problem – be supervised or unsupervised learning. Feature engineering is a critical preparatory process in machine learning. It is responsible for taking raw input data and converting that to well-aligned features which are ready to be used by the machine learning models.

But before we start discussing feature engineering, let's try to understand more clearly what feature is.

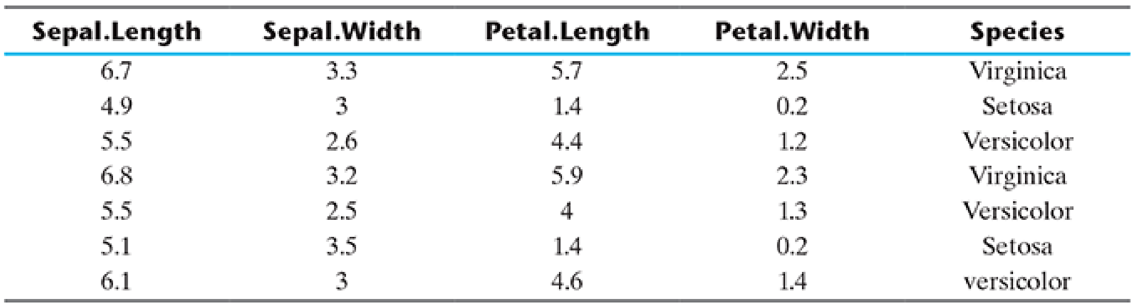
Did you know?

Unstructured data is raw, unorganized data which doesn't follow a specific format or hierarchy. Typical examples of unstructured data include text data from social networks, e.g. Twitter, Facebook, etc. or data from server logs, etc.

4.1.1 What is a feature?

A feature is an attribute of a data set that is used in a machine learning process. There is a view amongst certain machine learning practitioners that only those attributes which are meaningful to a machine learning problem are to be called as features, but this view has to be taken with a pinch of salt. In fact, selection of the subset of features which are meaningful for machine learning is a sub-area of feature engineering which draws a lot of research interest. The features in a data set are also called its dimensions. So a data set having '*n*' features is called an *n*-dimensional data set.

Let's take the example of a famous machine learning data set, Iris, introduced by the British statistician and biologist Ronald Fisher, partly shown in Figure 4.1. It has five attributes or features namely Sepal.Length, Sepal.Width, Petal.Length, Petal. Width and Species. Out of these, the feature 'Species' represent the class variable and the remaining features are the predictor variables. It is a five-dimensional data set.



Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
6.7	3.3	5.7	2.5	Virginica
4.9	3	1.4	0.2	Setosa
5.5	2.6	4.4	1.2	Versicolor
6.8	3.2	5.9	2.3	Virginica
5.5	2.5	4	1.3	Versicolor
5.1	3.5	1.4	0.2	Setosa
6.1	3	4.6	1.4	versicolor

FIG. 4.1 Data set features

4.1.2 What is feature engineering?

Feature engineering refers to the process of translating a data set into features such that these features are able to represent the data set more effectively and result in a better learning performance.

As we know already, feature engineering is an important pre-processing step for machine learning. It has two major elements:

1. feature transformation
2. feature subset selection

Feature transformation transforms the data – structured or unstructured, into a new set of features which can represent the underlying problem which machine learning is trying to solve. There are two variants of feature transformation:

1. feature construction
2. feature extraction

Both are sometimes known as feature discovery.

Feature construction process discovers missing information about the relationships between features and augments the feature space by creating additional features. Hence, if there are '*n*' features or dimensions in a data set, after feature construction '*m*' more features or dimensions may get added. So at the end, the data set will become '*n* + *m*' dimensional.

Feature extraction is the process of extracting or creating a new set of features from the original set of features using some functional mapping.

Unlike feature transformation, in case of **feature subset selection** (or simply **feature selection**) no new feature is generated. The objective of feature selection is to derive a subset of features from the full feature set which is most meaningful in the context of a specific machine learning problem. So, essentially the job of feature selection is to derive a subset F_j (F_1, F_2, …, F_m) of F_i (F_1, F_2, …, F_n), where $m < n$, such that F_j is most meaningful and gets the best result for a machine learning problem. We will discuss these concepts in detail in the next section.

Points to Ponder

Data scientists and machine learning practitioners spend significant amount of time in different feature engineering activities. Selecting the right features has a critical role to play in the success of a machine learning model.

It is quite evident that feature construction expands the feature space, while feature extraction and feature selection reduces the feature space.

4.2 FEATURE TRANSFORMATION

Engineering a good feature space is a crucial prerequisite for the success of any machine learning model. However, often it is not clear which feature is more important. For that reason, all available attributes of the data set are used as features and the problem of identifying the important features is left to the learning model. This is definitely not a feasible approach, particularly for certain domains e.g. medical image classification, text categorization, etc. In case a model has to be trained to classify a document as spam or non-spam, we can represent a document as a bag of words. Then the feature space will contain all unique words occurring across all documents. This will easily be a feature space of a few hundred thousand features. If we start including bigrams or trigrams along with words, the count of features will run in millions. To deal with this problem, feature transformation comes into play. Feature transformation is used as an effective tool for dimensionality reduction and hence for boosting learning model performance. Broadly, there are two distinct goals of feature transformation:

- Achieving best reconstruction of the original features in the data set
- Achieving highest efficiency in the learning task

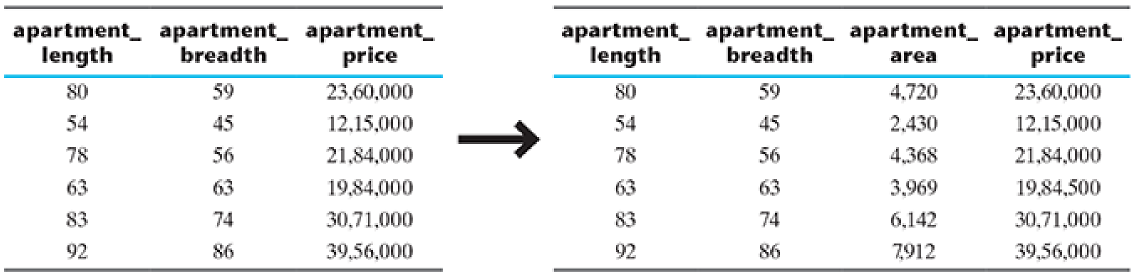
Did you know?

In the field of natural language processing, '*n*-gram' is a contiguous set of *n* items for example words in a text block or document. Using numerical prefixes, *n*-gram of size 1 is called unigram (i.e. a single word), size 2 is called bigram (i.e. a two-word phrase), size 3 is called trigram (i.e. a three-word phrase) etc.

4.2.1 Feature construction

Feature construction involves transforming a given set of input features to generate a new set of more powerful features. To understand more clearly, let's take the example of a real estate data set having details of all apartments sold in a specific region.

The data set has three features – apartment length, apartment breadth, and price of the apartment. If it is used as an input to a regression problem, such data can be training data for the regression model. So given the training data, the model should be able to predict the price of an apartment whose price is not known or which has just come up for sale. However, instead of using length and breadth of the apartment as a predictor, it is much convenient and makes more sense to use the area of the apartment, which is not an existing feature of the data set. So such a feature, namely apartment area, can be added to the data set. In other words, we transform the three-dimensional data set to a four-dimensional data set, with the newly 'discovered' feature apartment area being added to the original data set. This is depicted in Figure 4.2.



apartment_ length	apartment_ breadth	apartment_ price
80	59	23,60,000
54	45	12,15,000
78	56	21,84,000
63	63	19,84,000
83	74	30,71,000
92	86	39,56,000

apartment_ length	apartment_ breadth	apartment_ area	apartment_ price
80	59	4,720	23,60,000
54	45	2,430	12,15,000
78	56	4,368	21,84,000
63	63	3,969	19,84,500
83	74	6,142	30,71,000
92	86	7,912	39,56,000

FIG. 4.2 Feature construction (example 1)

Note: Though for the sake of simplicity the features apartment length and apartment breadth have been retained in Figure 4.2, in reality, it makes more sense to exclude these features when building the model.

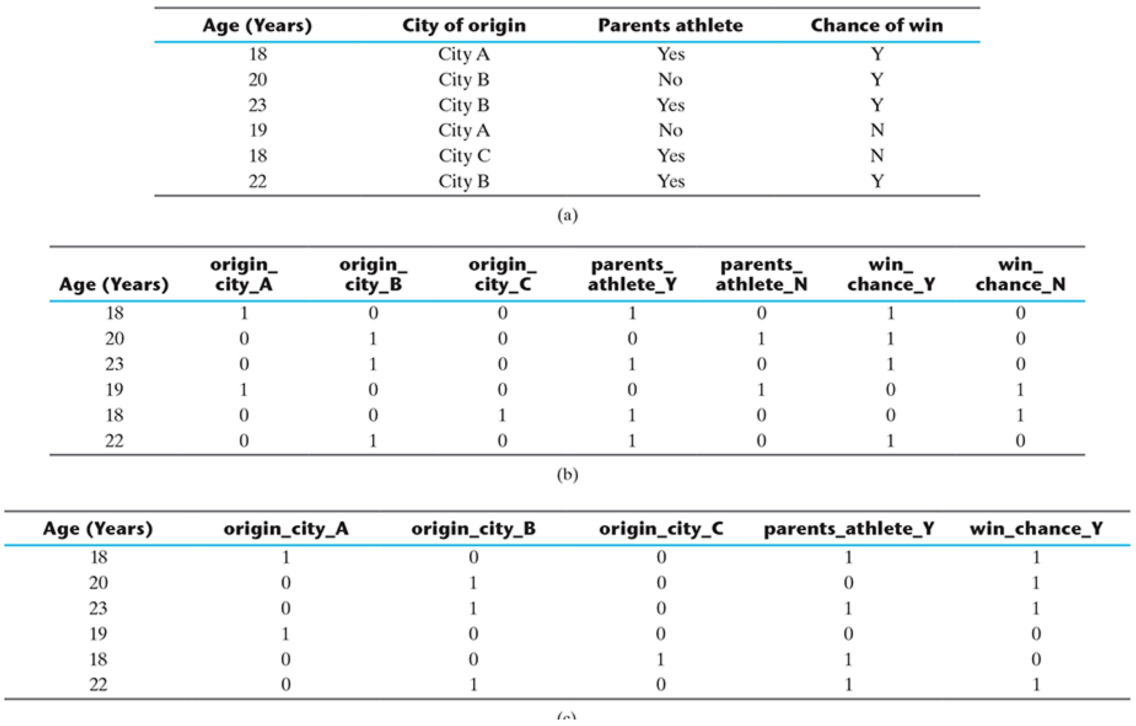
There are certain situations where feature construction is an essential activity before we can start with the machine learning task. These situations are

- when features have categorical value and machine learning needs numeric value inputs
- when features having numeric (continuous) values and need to be converted to ordinal values
- when text-specific feature construction needs to be done

4.2.1.1 Encoding categorical (nominal) variables

Let's take the example of another data set on athletes, as presented in Figure 4.3a. Say the data set has features age, city of origin, parents athlete (i.e. indicate whether any one of the parents was an athlete) and Chance of Win. The feature chance of a win is a class variable while the others are predictor variables. We know that any machine learning algorithm, whether it's a classification algorithm (like *k*NN) or a regression algorithm, requires numerical figures to learn from. So there are three features – City of origin, Parents athlete, and Chance of win, which are categorical in nature and cannot be used by any machine learning task.

In this case, feature construction can be used to create new dummy features which are usable by machine learning algorithms. Since the feature 'City of origin' has three unique values namely City A, City B, and City C, three dummy features namely origin_ city_A, origin_city_B, and origin_city_C is created. In the same way, dummy features parents_athlete_Y and parents_athlete_N are created for feature 'Parents athlete' and win_chance_Y and win_chance_N are created for feature 'Chance of win'. The dummy features have value 0 or 1 based on the categorical value for the original feature in that row. For example, the second row had a categorical value 'City B' for the feature 'City of origin'. So, the newly created features in place of 'City of origin', i.e. origin_city_A, origin_city_B and origin_city_C will have values 0, 1 and 0, respectively. In the same way, parents_athlete_Y and parents_athlete_N will have values 0 and 1, respectively in row 2 as the original feature 'Parents athlete' had a categorical value 'No' in row 2. The entire set of transformation for athletes' data set is shown in Figure 4.3b.



Age (Years)	City of origin	Parents athlete	Chance of win
18	City A	Yes	Y
20	City B	No	Y
23	City B	Yes	Y
19	City A	No	N
18	City C	Yes	N
22	City B	Yes	Y

(a)

Age (Years)	origin_ city_A	origin_ city_B	origin_ city_C	parents_ athlete_Y	parents_ athlete_N	win_ chance_Y	win_ chance_N
18	1	0	0	1	0	1	0
20	0	1	0	0	1	1	0
23	0	1	0	1	0	1	0
19	1	0	0	0	1	0	1
18	0	0	1	1	0	0	1
22	0	1	0	1	0	1	0

(b)

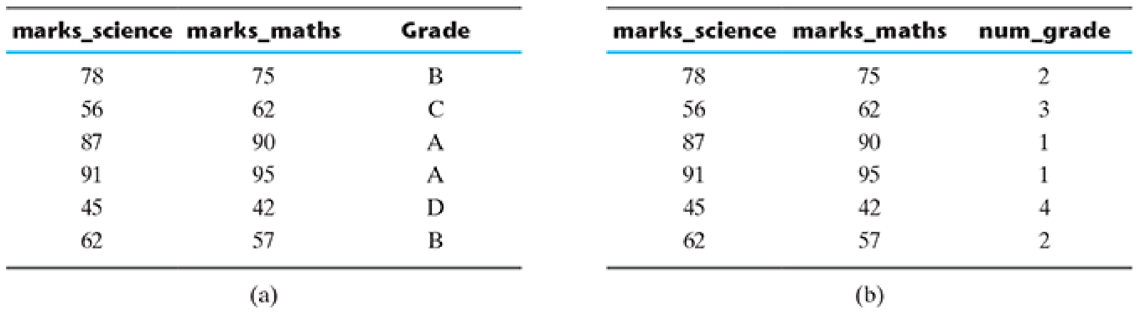
Age (Years)	origin_city_A	origin_city_B	origin_city_C	parents_athlete_Y	win_chance_Y
18	1	0	0	1	1
20	0	1	0	0	1
23	0	1	0	1	1
19	1	0	0	0	0
18	0	0	1	1	0
22	0	1	0	1	1

FIG. 4.3 Feature construction (encoding nominal variables)

However, examining closely, we see that the features 'Parents athlete' and 'Chance of win' in the original data set can have two values only. So creating two features from them is a kind of duplication, since the value of one feature can be decided from the value of the other. To avoid this duplication, we can just leave one feature and eliminate the other, as shown in Figure 4.3c.

4.2.1.2 Encoding categorical (ordinal) variables

Let's take an example of a student data set. Let's assume that there are three variable – science marks, maths marks and grade as shown in Figure 4.4a. As we can see, the grade is an ordinal variable with values A, B, C, and D. To transform this variable to a numeric variable, we can create a feature num_grade mapping a numeric value against each ordinal value. In the context of the current example, grades A, B, C, and D in Figure 4.4a is mapped to values 1, 2, 3, and 4 in the transformed variable shown in Figure 4.4b.



marks_science	marks_maths	Grade
78	75	B
56	62	C
87	90	A
91	95	A
45	42	D
62	57	B

(a)

marks_science	marks_maths	num_grade
78	75	2
56	62	3
87	90	1
91	95	1
45	42	4
62	57	2

(b)

FIG. 4.4 Feature construction (encoding ordinal variables)

4.2.1.3 Transforming numeric (continuous) features to categorical features

Sometimes there is a need of transforming a continuous numerical variable into a categorical variable. For example, we may want to treat the real estate price prediction problem, which is a regression problem, as a real estate price category prediction, which is a classification problem. In that case, we can 'bin' the numerical data into multiple categories based on the data range. In the context of the real estate price prediction example, the original data set has a numerical feature apartment_price as shown in Figure 4.5a. It can be transformed to a categorical variable price-grade either as shown in Figure 4.5b or as shown in Figure 4.5c.

4.2.1.4 Text-specific feature construction

In the current world, text is arguably the most predominant medium of communication. Whether we think about social networks like Facebook or micro-blogging channels like Twitter or emails or short messaging services such as Whatsapp, text plays a major role in the flow of information. Hence, text mining is an important area of research – not only for technology practitioners but also for industry practitioners. However, making sense of text data, due to the inherent unstructured nature of the data, is not so straightforward. In the first place, the text data chunks that we can think about do not have readily available features, like structured data sets, on which machine learning tasks can be executed. All machine learning models need numerical data as input. So the text data in the data sets need to be transformed into numerical features.

apartment_ area	apartment_ price
4,720	23,60,000
2,430	12,15,000
4,368	21,84,000
3,969	19,84,500
6,142	30,71,000
7,912	39,56,000

(a)

apartment_ area	apartment_ grade
4,720	Medium
2,430	Low
4,368	Medium
3,969	Low
6,142	High
7,912	High

(b)

apartment_ area	apartment_ grade
4,720	2
2,430	1
4,368	2
3,969	1
6,142	3
7,912	3

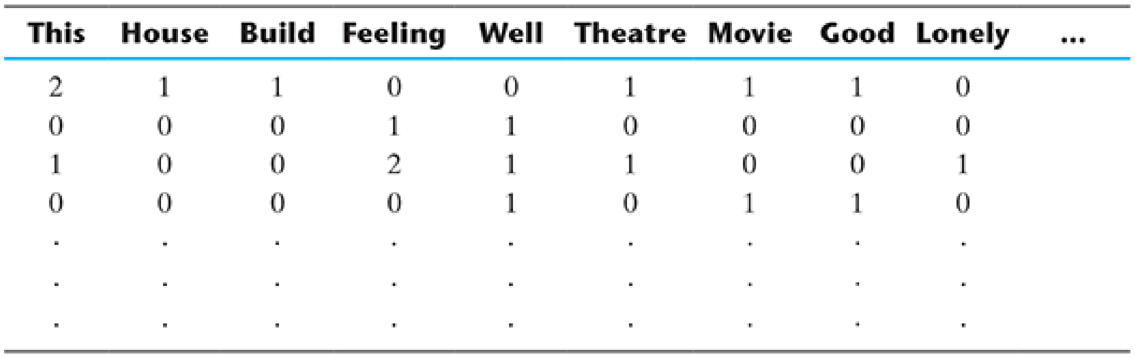
(c)

FIG. 4.5 Feature construction (numeric to categorical)

Text data, or corpus which is the more popular keyword, is converted to a numerical representation following a process is known as vectorization. In this process, word occurrences in all documents belonging to the corpus are consolidated in the form of bag-of-words. There are three major steps that are followed:

1. tokenize
2. count
3. normalize

In order to tokenize a corpus, the blank spaces and punctuations are used as delimiters to separate out the words, or tokens. Then the number of occurrences of each token is counted, for each document. Lastly, tokens are weighted with reducing importance when they occur in the majority of the documents. A matrix is then formed with each token representing a column and a specific document of the corpus representing each row. Each cell contains the count of occurrence of the token in a specific document. This matrix is known as a document-term matrix (also known as a term-document matrix). Figure 4.6 represents a typical document-term matrix which forms an input to a machine learning model.



This	House	Build	Feeling	Well	Theatre	Movie	Good	Lonely	...
2	1	1	0	0	1	1	1	0	
0	0	0	1	1	0	0	0	0	
1	0	0	2	1	1	0	0	1	
0	0	0	0	1	0	1	1	0	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	

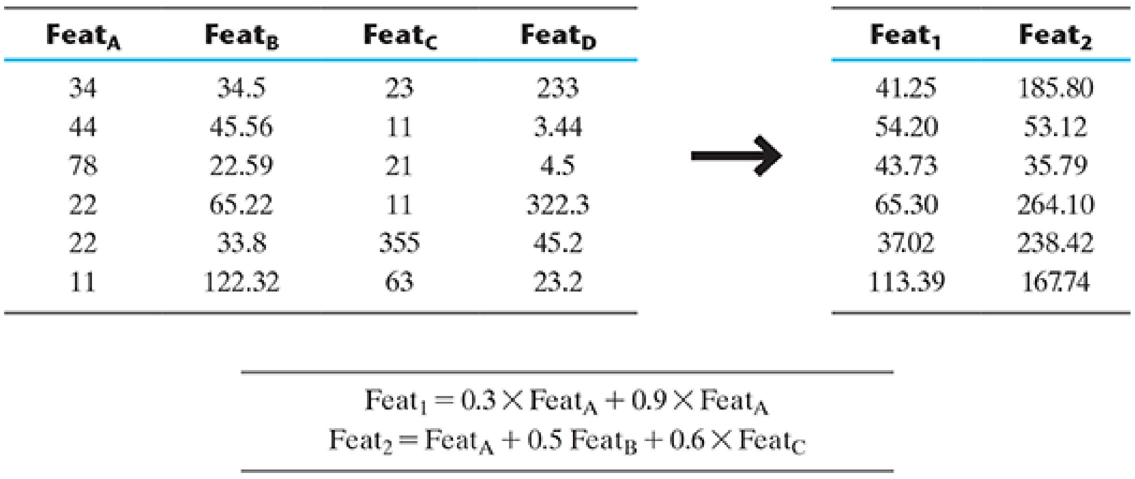
FIG. 4.6 Feature construction (text-specific)

4.2.2 Feature extraction

In feature extraction, new features are created from a combination of original features. Some of the commonly used operators for combining the original features include

1. For Boolean features: Conjunctions, Disjunctions, Negation, etc.
2. For nominal features: Cartesian product, M of N, etc.
3. For numerical features: Min, Max, Addition, Subtraction, Multiplication, Division, Average, Equivalence, Inequality, etc.

Let's take an example and try to understand. Say, we have a data set with a feature set F_i (F_1, F_2, ..., F_n). After feature extraction using a mapping function f (F_1, F_2, ..., F_n) say, we will have a set of features $\acute{F}_i(\acute{F}_1, \acute{F}_2, \ldots, \acute{F}_m)$ such that $\acute{F}_i = f(F_i)$ and $m < n$. For example, $\acute{F}_1 = k_1F_1 + k_2F_2$. This is depicted in Figure 4.7.



$Feat_A$	$Feat_B$	$Feat_C$	$Feat_D$
34	34.5	23	233
44	45.56	11	3.44
78	22.59	21	4.5
22	65.22	11	322.3
22	33.8	355	45.2
11	122.32	63	23.2

$\rightarrow$

$Feat_1$	$Feat_2$
41.25	185.80
54.20	53.12
43.73	35.79
65.30	264.10
37.02	238.42
113.39	167.74

$$Feat_1 = 0.3 \times Feat_A + 0.9 \times Feat_A$$
$$Feat_2 = Feat_A + 0.5\ Feat_B + 0.6 \times Feat_C$$

FIG. 4.7 Feature extraction

Let's discuss the most popular feature extraction algorithms used in machine learning:

4.2.2.1 Principal Component Analysis

Every data set, as we have seen, has multiple attributes or dimensions – many of which might have similarity with each other. For example, the height and weight of a person, in general, are quite related. If the height is more, generally weight is more and vice versa. So if a data set has height and weight as two of the attributes, obviously they are expected to be having quite a bit of similarity. In general, any machine learning algorithm performs better as the number of related attributes or features reduced. In other words, a key to the success of machine learning lies in the fact that the features are less in number as well as the similarity between each other is very less. This is the main guiding philosophy of principal component analysis (PCA) technique of feature extraction.

In PCA, a new set of features are extracted from the original features which are quite dissimilar in nature. So an *n*-dimensional feature space gets transformed to an *m*-dimensional feature space, where the dimensions are orthogonal to each other, i.e. completely independent of each other. To understand the concept of orthogonality, we have to step back and do a bit of dip dive into vector space concept in linear algebra.

We all know that a vector is a quantity having both magnitude and direction and hence can determine the position of a point relative to another point in the Euclidean space (i.e. a two or three or ‘n’ dimensional space). A vector space is a set of vectors. Vector spaces have a property that they can be represented as a linear combination of a smaller set of vectors, called basis vectors. So, any vector ‘v’ in a vector space can be represented as

$$v = \sum_{i=1}^{n} a_i u_i$$

where, a_i represents ‘*n*’ scalars and u_i represents the basis vectors. Basis vectors are orthogonal to each other. Orthogonality of vectors in *n*-dimensional vector space can be thought of an extension of the vectors being perpendicular in a two-dimensional vector space. Two orthogonal vectors are completely unrelated or independent of each other. So the transformation of a set of vectors to the corresponding set of basis vectors such that each vector in the original set can be expressed as a linear combination of basis vectors helps in decomposing the vectors to a number of independent components.

Now, let’s extend this notion to the feature space of a data set. The feature vector can be transformed to a vector space of the basis vectors which are termed as principal components. These principal components, just like the basis vectors, are orthogonal to each other. So a set of feature vectors which may have similarity with each other is transformed to a set of principal components which are completely unrelated. However, the principal components capture the variability of the original feature space. Also, the number of principal component derived, much like the basis vectors, is much smaller than the original set of features.

The objective of PCA is to make the transformation in such a way that

1. The new features are distinct, i.e. the covariance between the new features, i.e. the principal components is 0.

2. The principal components are generated in order of the variability in the data that it captures. Hence, the first principal component should capture the maximum variability, the second principal component should capture the next highest variability etc.
3. The sum of variance of the new features or the principal components should be equal to the sum of variance of the original features.

PCA works based on a process called eigenvalue decomposition of a covariance matrix of a data set. Below are the steps to be followed:

1. First, calculate the covariance matrix of a data set.
2. Then, calculate the eigenvalues of the covariance matrix.
3. The eigenvector having highest eigenvalue represents the direction in which there is the highest variance. So this will help in identifying the first principal component.
4. The eigenvector having the next highest eigenvalue represents the direction in which data has the highest remaining variance and also orthogonal to the first direction. So this helps in identifying the second principal component.
5. Like this, identify the top ‘*k*’ eigenvectors having top ‘*k*’ eigenvalues so as to get the ‘*k*’ principal components.

4.2.2.2 Singular value decomposition

Singular value decomposition (SVD) is a matrix factorization technique commonly used in linear algebra. SVD of a matrix A ($m \times n$) is a factorization of the form:

$$A = U \sum V$$

where, U and V are orthonormal matrices, U is an $m \times m$ unitary matrix, V is an $n \times n$ unitary matrix and $\sum$ is an $m \times n$ rectangular diagonal matrix. The diagonal entries of $\sum$ are known as singular values of matrix A. The columns of U and V are called the left-singular and right-singular vectors of matrix A, respectively.

SVD is generally used in PCA, once the mean of each variable has been removed. Since it is not always advisable to remove the mean of a data attribute, especially when the data set is sparse (as in case of text data), SVD is a good choice for dimensionality reduction in those situations.

SVD of a data matrix is expected to have the properties highlighted below:

1. Patterns in the attributes are captured by the right-singular vectors, i.e. the columns of *V*.
2. Patterns among the instances are captured by the left-singular, i.e. the columns of *U*.
3. Larger a singular value, larger is the part of the matrix A that it accounts for and its associated vectors.
4. New data matrix with '*k*' attributes is obtained using the equation

$$D' = D \times [v_{\mathbf{1}}, v_{\mathbf{2}}, \dots , v_{\boldsymbol{k}}]$$

   Thus, the dimensionality gets reduced to *k*
   SVD is often used in the context of text data.

4.2.2.3 Linear Discriminant Analysis

Linear discriminant analysis (LDA) is another commonly used feature extraction technique like PCA or SVD. The objective of LDA is similar to the sense that it intends to transform a data set into a lower dimensional feature space. However, unlike PCA, the focus of LDA is not to capture the data set variability. Instead, LDA focuses on class separability, i.e. separating the features based on class separability so as to avoid over-fitting of the machine learning model.

Unlike PCA that calculates eigenvalues of the covariance matrix of the data set, LDA calculates eigenvalues and eigenvectors within a class and inter-class scatter matrices. Below are the steps to be followed:

1. Calculate the mean vectors for the individual classes.
2. Calculate intra-class and inter-class scatter matrices.

3. Calculate eigenvalues and eigenvectors for S_W^{-1} and S_B, where S_W is the intra-class scatter matrix and S_B is the inter-class scatter matrix

$$S_W = \sum_{i=1}^{c} S_i;$$

$$S_i = \sum_{x \in D_i}^{n} (x - m_i)(x - m_i)^T$$

where, m_i is the mean vector of the i-th class

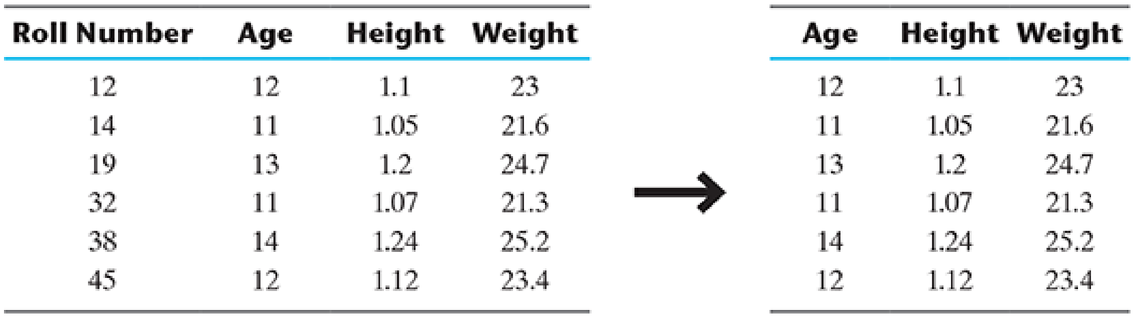
$$S_B = \sum_{i=1}^{c} N_i (m_i - m)(m_i - m)^T$$

where, mi is the sample mean for each class, m is the overall mean of the data set, *Ni* is the sample size of each class

4. Identify the top '*k*' eigenvectors having top '*k*' eigenvalues

4.3 FEATURE SUBSET SELECTION

Feature selection is arguably the most critical pre-processing activity in any machine learning project. It intends to select a subset of system attributes or features which makes a most meaningful contribution in a machine learning activity. Let's quickly discuss a practical example to understand the philosophy behind feature selection. Say we are trying to predict the weight of students based on past information about similar students, which is captured in a 'student weight' data set. The student weight data set has features such as Roll Number, Age, Height, and Weight. We can well understand that roll number can have no bearing, whatsoever, in predicting student weight. So we can eliminate the feature roll number and build a feature subset to be considered in this machine learning problem. The subset of features is expected to give better results than the full set. The same has been **depicted i**n Figure 4.8.



Roll Number	Age	Height	Weight
12	12	1.1	23
14	11	1.05	21.6
19	13	1.2	24.7
32	11	1.07	21.3
38	14	1.24	25.2
45	12	1.12	23.4

$\rightarrow$

Age	Height	Weight
12	1.1	23
11	1.05	21.6
13	1.2	24.7
11	1.07	21.3
14	1.24	25.2
12	1.12	23.4

FIG. 4.8 Feature selection

But before we go forward with more detailed discussion on feature selection, let's try to understand the issues which have made feature selection such a relevant problem to be solved.

4.3.1 Issues in high-dimensional data

With the rapid innovations in the digital space, the volume of data generated has increased to an unbelievable extent. At the same time, breakthroughs in the storage technology area have made storage of large quantity of data quite cheap. This has further motivated the storage and mining of very large and high-dimensionality data sets.

Points to Ponder

'High-dimensional' refers to the high number of variables or attributes or features present in certain data sets, more so in the domains like DNA analysis, geographic information systems (GIS), social networking, etc. The high-dimensional spaces often have hundreds or thousands of dimensions or attributes, e.g. DNA microarray data can have up to 450,000 variables (gene probes).

Alongside, two new application domains have seen drastic development. One is that of biomedical research, which includes gene selection from microarray data. The other one is text categorization which deals with huge volumes of text data from social networking sites, emails, etc. The first domain, i.e. biomedical research generates data sets having a number of features in the range of a few tens of thousands. The text data generated from different sources also have extremely high dimensions. In a large document corpus having few thousand documents embedded, the number of unique word tokens which represent the feature of the text data set, can also be in the range of a few tens of thousands. To get insight from such high-dimensional data may be a big challenge for any machine learning algorithm. On one hand, very high quantity of computational resources and high amount of time will be required. On the other hand the performance of the model – both for supervised and unsupervised machine learning task, also degrades sharply due to unnecessary noise in the data. Also, a model built on an extremely high number of features may be very difficult to understand. For this reason, it is necessary to take a subset of the features instead of the full set.

The objective of feature selection is three-fold:

- Having faster and more cost-effective (i.e. less need for computational resources) learning model
- Improving the efficiency of the learning model
- Having a better understanding of the underlying model that generated the data

4.3.2 Key drivers of feature selection – feature relevance and redundancy

4.3.2.1 Feature relevance

In supervised learning, the input data set which is the training data set, has a class label attached. A model is inducted based on the training data set – so that the inducted model can assign class labels to new, unlabelled data. Each of the predictor variables, is expected to contribute information to decide the value of the class label. In case a variable is not contributing any information, it is said to be irrelevant. In case the information contribution for prediction is very little, the variable is said to be weakly relevant. Remaining variables, which make a significant contribution to the prediction task are said to be strongly relevant variables.

In unsupervised learning, there is no training data set or labelled data. Grouping of similar data instances are done and similarity of data instances are evaluated based on the value of different variables. Certain variables do not contribute any useful information for deciding the similarity of dissimilarity of data instances. Hence, those variables make no significant information contribution in the grouping process. These variables are marked as irrelevant variables in the context of the unsupervised machine learning task.

To get a perspective, we can think of the simple example of the student data set that we discussed at the beginning of this section. Roll number of a student doesn't contribute any significant information in predicting what the Weight of a student would be. Similarly, if we are trying to group together students with similar academic capabilities, Roll number can really not contribute any information whatsoever. So, in context of the supervised task of predicting student Weight or the unsupervised task of grouping students with similar academic merit, the variable Roll number is quite irrelevant.

Any feature which is irrelevant in the context of a machine learning task is a candidate for rejection when we are selecting a subset of features. We can consider whether the weakly relevant features are to be rejected or not on a case-to-case basis.

4.3.2.2 Feature redundancy

A feature may contribute information which is similar to the information contributed by one or more other features. For example, in the weight prediction problem referred earlier in the section, both the features Age and Height contribute similar information. This is because with an increase in Age, Weight is expected to increase. Similarly, with the increase of Height also Weight is expected to increase. Also, Age and Height increase with each other. So, in context of the Weight prediction problem, Age and Height contribute similar information. In other words, irrespective of whether the feature height is present as a part of the feature subset, the learning model will give almost same results. In the same way, without age being part of the predictor variables, the outcome of the learning model will be more or less same. In this kind of a situation when one feature is similar to another feature, the feature is said to be potentially redundant in the context of the learning problem.

All features having potential redundancy are candidates for rejection in the final feature subset. Only a small number of representative features out of a set of potentially redundant features are considered for being a part of the final feature subset.

So, in a nutshell, the main objective of feature selection is to remove all features which are irrelevant and take a representative subset of the features which are potentially redundant. This leads to a meaningful feature subset in context of a specific learning task.

Now, the question is how to find out which of the features are irrelevant or which features have potential redundancy. For that multiple measures are being used, some of which have been covered in the next subsection.

4.3.3 Measures of feature relevance and redundancy

4.3.3.1 Measures of feature relevance

As mentioned earlier, feature relevance is to be gauged by the amount of information contributed by a feature. For supervised learning, mutual information is considered as a good measure of information contribution of a feature to decide the value of the class label. That's why it is a good indicator of the relevance of a feature with respect to the class variable. Higher the value of mutual information of a feature, more relevant is that feature. Mutual information can be calculated as follows:

$$MI(C, f) = H(C) + H(f) - H(C, f)$$

where, marginal entropy of the class, H(C) = $-\sum_{i=1}^{k} p(C_i) \log_2 p(C_i)$

marginal entropy of the feature '*x*', *H*(*f*) = $-\sum_{c} p(f = x) \log_2 p(f = x)$

and *K* = number of classes, *C* = class variable, *f* = feature set that take discrete values.

In case of unsupervised learning, there is no class variable. Hence, feature-to-class mutual information cannot be used to measure the information contribution of the features. In case of unsupervised learning, the entropy of the set of features without one feature at a time is calculated for all the features. Then, the features are ranked in a descending order of information gain from a feature and top 'β' percentage (value of 'β' is a design parameter of the algorithm) of features are selected as relevant features. The entropy of a feature f is calculated using Shannon's formula below:

$$H(f) = -\sum_{x} p(f = x) \log_2 p(f = x)$$

$\sum\limits_{x}$ is used only for features that take discrete values. For continuous features, it should be replaced by discretization performed first to estimate probabilities *p*(*f* = *x*).

4.3.3.2 Measures of Feature redundancy

Feature redundancy, as we have already discussed, is based on similar information contribution by multiple features. There are multiple measures of similarity of information contribution, salient ones being

1. Correlation-based measures
2. Distance-based measures, and
3. Other coefficient-based measure

1. Correlation-based similarity measure

Correlation is a measure of linear dependency between two random variables. Pearson's product moment correlation coefficient is one of the most popular and accepted measures of correlation between two random variables. For two random feature variables F_1 and F_2, Pearson correlation coefficient is defined as:

Correlation values range between +1 and –1. A correlation of 1 (+ / –) indicates perfect correlation, i.e. the two features having a perfect linear relationship. In case the correlation is 0, then the features seem to have no linear relationship. Generally, for all feature selection problems, a threshold value is adopted to decide whether two features have adequate similarity or not.

2. Distance-based similarity measure

The most common distance measure is the **Euclidean distance**, which, between two features F_1 and F_2 are calculated as:

$$d(F_1,\ F_2) = \sqrt{\sum_{i=1}^{n} (F_{1_i} - F_{2_i})^2}$$

where F_1 and F_2 are features of an n-dimensional data set. Refer to the Figure 4.9. The data set has two features, aptitude (F_1) and communication (F_2) under consideration. The Euclidean distance between the features has been calculated using the formula provided above.

Aptitude (F_1)	**Communication (F_2)**	**($F_1 - F_2$)**	**($F_1 - F_2$)^2**
2	6	–4	16
3	5.5	–2.5	6.25
6	4	2	4
7	2.5	4.5	20.25
8	3	5	25
6	5.5	0.5	0.25
6	7	–1	1
7	6	1	1
8	6	2	4
9	7	2	4
			81.75

FIG. 4.9 Distance calculation between features

A more generalized form of the Euclidean distance is the **Minkowski distance**, measured as

$$d(F_1, F_2) = \sqrt{\sum_{i=1}^{n} (F_{1_i} - F_{2_i})^r}$$

Minkowski distance takes the form of Euclidean distance (also called **L_2 norm**) when $r = 2$.

At $r = 1$, it takes the form of **Manhattan distance** (also called **L_1 norm**), as shown below:

$$d(F_1, F_2) = \sum_{i=1}^{n} |F_{1_i} - F_{2_i}|$$

A specific example of Manhattan distance, used more frequently to calculate the distance between binary vectors is the **Hamming distance**. For example, the Hamming distance between two vectors 01101011 and 11001001 is 3, as illustrated in Figure 4.10a.

3. Other similarity measures

Jaccard index/coefficient is used as a measure of similarity between two features. The **Jaccard distance**, a measure of dissimilarity between two features, is complementary of Jaccard index.

FIG. 4.10 Distance measures between features

For two features having binary values, Jaccard index is measured as

where, n_{11} = number of cases where both the features have value 1

n_{01} = number of cases where the feature 1 has value 0 and feature 2 has value 1

n_{10} = number of cases where the feature 1 has value 1 and feature 2 has value 0

Jaccard distance, $d_J = 1 - J$

Let's consider two features F_1 and F_2 having values (0, 1, 1, 0, 1, 0, 1, 0) and (1, 1, 0, 0, 1, 0, 0, 0). Figure 4.10b shows the identification of the values of n_{11}, n_{01} and n_{10}. As shown, the cases where both the values are 0 have been left out without border – as an indication of the fact that they will be excluded in the calculation of Jaccard coefficient.

Jaccard coefficient of F_1 and F_2, J =

∴ Jaccard distance between F_1 and F_2, $d_J = 1 - J =$ or 0.6.

Simple matching coefficient (SMC) is almost same as Jaccard coeficient except the fact that it includes a number of cases where both the features have a value of 0.

where, n_{11} = number of cases where both the features have value 1

n_{01} = number of cases where the feature 1 has value 0 and feature 2 has value 1

n_{10} = number of cases where the feature 1 has value 1 and feature 2 has value 0

n_{00} = number of cases where both the features have value 0

Quite understandably, the total count of rows, $n = n_{00} + n_{01} + n_{10} + n_{11}$. As shown in Figure 4.10c, all values have been included in the calculation of SMC.

One more measure of similarity using similarity coefficient calculation is **Cosine Similarity**. Let's take the example of a typical text classification problem. The text corpus needs to be first transformed into features with a word token being a feature and the number of times the word occurs in a document comes as a value in each row. There are thousands of features in such a text data set. However, the data set is sparse in nature as only a few words do appear in a document, and hence in a row of the data set. So each row has very few non-zero values. However, the non-zero values can be anything integer value as the same word may occur any number of times. Also, considering the sparsity of the data set, the 0-0 matches (which obviously is going to be pretty high) need to be ignored. Cosine similarity which is one of the most popular measures in text classification is calculated as:

where, *x.y* = vector dot product of *x* and *y* =

Let's calculate the cosine similarity of *x* and *y*, where *x* = (2, 4, 0, 0, 2, 1, 3, 0, 0) and *y* = (2, 1, 0, 0, 3, 2, 1, 0, 1).

In this case, *x.y* = 2*2 + 4*1 + 0*0 + 0*0 + 2*3 + 1*2 + 3*1 + 0*0 + 0*1 = 19

Cosine similarity actually measures the angle (refer to Fig. 4.11) between *x* and *y* vectors. Hence, if cosine similarity has a value 1, the angle between *x* and *y* is 0° which means *x* and *y* are same except for the magnitude. If cosine similarity is 0, the angle between *x* and *y* is 90°. Hence, they do not share any similarity (in case of text data, no term/word is common). In the above example, the angle comes to be 43.2°.

FIG. 4.11 Cosine similarity

4.3.4 Overall feature selection process

Feature selection is the process of selecting a subset of features in a data set. As depicted in Figure 4.12, a typical feature selection process consists of four steps:

1. generation of possible subsets
2. subset evaluation
3. stop searching based on some stopping criterion
4. validation of the result

FIG. 4.12 Feature selection process

Subset generation, which is the first step of any feature selection algorithm, is a search procedure which ideally should produce all possible candidate subsets. However, for an *n*-dimensional data set, $2^{\boldsymbol{n}}$ subsets can be generated. So, as the value of ‘*n*’ becomes high, finding an optimal subset from all the $2^{\boldsymbol{n}}$ candidate subsets becomes intractable. For that reason, different approximate search strategies are employed to find candidate subsets for evaluation. On one hand, the search may start with an empty set and keep adding features. This search strategy is termed as a sequential forward selection. On the other hand, a search may start with a full set and successively remove features. This strategy is termed as sequential backward elimination. In certain cases, search start with both ends and add and remove features simultaneously. This strategy is termed as a bi-directional selection.

Each candidate subset is then evaluated and compared with the previous best performing subset based on certain **evaluation criterion**. If the new subset performs better, it replaces the previous one.

This cycle of subset generation and evaluation continues till a pre-defined **stopping criterion** is fulfilled. Some commonly used stopping criteria are

1. the search completes
2. some given bound (e.g. a specified number of iterations) is reached
3. subsequent addition (or deletion) of the feature is not producing a better subset

4. a sufficiently good subset (e.g. a subset having better classification accuracy than the existing benchmark) is selected

Then the selected best subset is **validated** either against prior benchmarks or by experiments using real-life or synthetic but authentic data sets. In case of supervised learning, the accuracy of the learning model may be the performance parameter considered for validation. The accuracy of the model using the subset derived is compared against the model accuracy of the subset derived using some other benchmark algorithm. In case of unsupervised, the cluster quality may be the parameter for validation.

4.3.5 Feature selection approaches

There are four types of approach for feature selection:

1. Filter approach
2. Wrapper approach
3. Hybrid approach
4. Embedded approach

In the **filter approach** (as depicted in Fig. 4.13), the feature subset is selected based on statistical measures done to assess the merits of the features from the data perspective. No learning algorithm is employed to evaluate the goodness of the feature selected. Some of the common statistical tests conducted on features as a part of filter approach are – Pearson's correlation, information gain, Fisher score, analysis of variance (ANOVA), Chi-Square, etc.

FIG. 4.13 Filter approach

In the **wrapper approach** (as depicted in Fig. 4.14), identification of best feature subset is done using the induction algorithm as a black box.

The feature selection algorithm searches for a good feature subset using the induction algorithm itself as a part of the evaluation function. Since for every candidate subset, the learning model is trained and the result is evaluated by running the learning algorithm, wrapper approach is computationally very expensive. However, the performance is generally superior compared to filter approach.

FIG. 4.14 Wrapper approach

Hybrid approach takes the advantage of both filter and wrapper approaches. A typical hybrid algorithm makes use of both the statistical tests as used in filter approach to decide the best subsets for a given cardinality and a learning algorithm to select the final best subset among the best subsets across different cardinalities.

Embedded approach (as depicted in Fig. 4.15) is quite similar to wrapper approach as it also uses and inductive algorithm to evaluate the generated feature subsets. However, the difference is it performs feature selection and classification simultaneously.

FIG. 4.15 Embedded approach

4.4 SUMMARY

- A feature is an attribute of a data set that is used in a machine learning process.
- Feature engineering is an important pre-processing step for machine learning, having two major elements:
  1. feature transformation
  2. feature subset selection
- Feature transformation transforms data into a new set of features which can represent the underlying machine learning problem
- There are two variants of feature transformation:
  1. feature construction
  2. feature extraction
- Feature construction process discovers missing information about the relationships between features and augments the feature space by creating additional features.
- Feature extraction is the process of extracting or creating a new set of features from the original set of features using some functional mapping.
- Some popular feature extraction algorithms used in machine learning:
  1. Principal Component Analysis (PCA)
  2. Singular Value Decomposition (SVD)
  3. Linear Discriminant Analysis (LDA)
- Feature subset selection is intended to derive a subset of features from the full feature set. No new feature is generated.

- The objective of feature selection is three-fold:
  1. Having faster and more cost-effective (i.e. less need for computational resources) learning model
  2. Improving the efficiency of the learning model
  3. Having a better understanding of the underlying model that generated the data

     Feature selection intends to remove all features which are irrelevant and take a representative subset of the features which are potentially redundant. This leads to a meaningful feature subset in context of a specific learning task.
- Feature relevance is indicated by the information gain from a feature measured in terms of relative entropy.
- Feature redundancy is based on similar information contributed by multiple features measured by feature-to-feature:
  1. Correlation
  2. Distance (Minkowski distances, e.g. Manhattan, Euclidean, etc. used as most popular measures)
  3. Other coefficient-based (Jaccard, SMC, Cosine similarity, etc.)
- Main approaches for feature selection are
  1. Filter
  2. Wrapper
  3. Hybrid
  4. Embedded

SAMPLE QUESTIONS

MULTIPLE-CHOICE QUESTIONS (1 MARK QUESTIONS)

1. Engineering a good feature space is a crucial ___ for the success of any machine learning model.
   1. Pre-requisite
   2. Process
   3. Objective
   4. None of the above
2. n-gram of size 1 is called

   1. Bigram
   2. Unigram
   3. Trigram
   4. None of the above
3. Feature ___ involves transforming a given set of input features to generate a new set of more powerful features.
   1. Selection
   2. Engineering
   3. Transformation
   4. Re-engineering
4. Conversion of a text corpus to a numerical representation is done using ___ process.
   1. Tokenization
   2. Normalization
   3. Vectorization
   4. None of the above
5. ___ approach uses induction algorithm for subset validation.
   1. Filter
   2. Hybrid
   3. Wrapper
   4. Embedded
6. In feature extraction, some of the commonly used ___ are used for combining the original features.
   1. Operators
   2. Delimiters
   3. Words
   4. All of the above
7. Hamming distance between binary vectors 1001 and 0101 is
   1. 1
   2. 2
   3. 3
   4. 4
8. PCA is a technique for
   1. Feature extraction
   2. Feature construction

   3. Feature selection
   4. None of the above
9. The new features created in PCA are known as
   1. Principal components
   2. Eigenvectors
   3. Secondary components
   4. None of the above
10. In LDA, intra-class and inter-class ___ matrices are calculated.
    1. Scatter
    2. Adjacency
    3. Similarity
    4. None of the above
11. Cosine similarity is most popularly used in
    1. Text classification
    2. Image classification
    3. Feature selection
    4. None of the above
12. This approach is quite similar to wrapper approach as it also uses and inductive algorithm to evaluate the generated feature subsets.
    1. Embedded approach
    2. Filter approach
    3. Pro Wrapper approach
    4. Hybrid approach
13. In ___ approach, identification of best feature subset is done using the induction algorithm as a black box.
    1. Embedded
    2. Filter
    3. Wrapper
    4. Hybrid

SHORT-ANSWER TYPE QUESTIONS (5 MARKS QUESTIONS)

1. What is a feature? Explain with an example.
2. What are the different situations which necessitate feature construction?

3. Explain the process of encoding nominal variables.
4. Explain the process of transforming numeric features to categorical features.
5. Explain the wrapper approach of feature selection. What are the merits and de-merits of this approach?
6. When can a feature be termed as irrelevant? How can it be measured?
7. When can a feature be termed as redundant? What are the measures to determine the potentially redundant features?
8. What are the different distance measures that can be used to determine similarity of features?
9. Compare Euclidean distance with Manhattan distance?
10. Differentiate feature transformation with feature selection
11. Write short notes on any two:
    1. SVD
    2. Hybrid method of feature selection
    3. Silhouette width
    4. ROC curve

LONG-ANSWER TYPE QUESTIONS (10 MARKS QUESTIONS)

1. What is feature engineering? Explain, in details, the different aspects of feature engineering?
2. What is feature selection? Why is it needed? What are the different approaches of feature selection?
3. Explain the filter and wrapper approaches of feature selection. What are the merits and demerits of these approaches?
4. 1. Explain the overall process of feature selection
   2. Explain, with an example, the main underlying concept of feature extraction. What are the most popular algorithms for feature extraction?
5. Explain the process of feature engineering in context of a text categorization problem.
6. Why is cosine similarity a suitable measure in context of text categorization? Two rows in a document-term matrix have values - (2, 3, 2, 0, 2, 3, 3, 0, 1) and (2, 1, 0, 0, 3, 2, 1, 3, 1). Find the cosine similarity.

7. 1. How can we calculate Hamming distance? Find the Hamming distance between 10001011 and 11001111.
   2. Compare the Jaccard index and similarity matching coefficient of two features having values (1, 1, 0, 0, 1, 0, 1, 1) and (1, 0, 0, 1, 1, 0, 0, 1).
8. What do you understand by a high-dimensional data set? Give a few practical examples? What is the challenge while applying machine learning technique on a high-dimensional data set? How can that be addressed?
9. 1. Write short notes on any <u>two</u>:
      1. PCA
      2. Vectorization
      3. Embedded method
   2. Write the difference between (any <u>two</u>):
      1. Sequential forward selection vs. sequential backward elimination
      2. Filter vs. wrapper method of feature selection
      3. Jaccard coefficient vs. SMC