Artificial Intelligence & Machine Learning

Course Code: 302049

Unit 2: Feature Extraction & Selection

Third Year Bachelor of Engineering (Choice Based Credit System)
Mechanical Engineering (2019 Course)
Board of Studies – Mechanical and Automobile Engineering, SPPU, Pune
(With Effect from Academic Year 2021-22)

Question bank and its solution

by

Abhishek D. Patange, Ph.D.
Department of Mechanical Engineering
College of Engineering Pune (COEP)

Unit 2: Feature Extraction and Selection

Syllabus:

Content	Theory	Mathematics	Numerical
• **Feature extraction**			
Statistical Features	✓	✓	✓
Principal Component Analysis	✓	✓	✓
• **Feature selection**			
Ranking	✓	✗	✗
Decision Tree – Entropy, Information Gain	✓	✓	✓
Exhaustive	✓	✗	✗
Best First	✓	✗	✗
Greedy Forward, Backward	✓	✗	✗
• **Application of Feature Extraction and Selection Algorithms in Mechanical Engineering**			

Type of question and marks:

Type	Theory	Mathematics	Numerical
Marks	2 or 4 or 6	4 marks	2 or 4 marks

QUESTION BANK FOR UNIT 2: FEATURE EXTRACTION AND SELECTION

Topic: Feature extraction

Theory	Mathematics	Numerical
✓	✓	✓

Theory questions

1. What is feature?

Feature is defined as a function of the basic measurement variables or attributes that specifies some quantifiable property of an object and is useful for classification and/or pattern recognition. Obtaining a good data representation is a very domain specific task and it is related to the available measurements.

2. Define: low level feature, high level feature, general feature, global feature and local feature.

- **Low-Level Features:** The fundamental features that can be extracted directly from an image without any object description.
- **High-Level Features:** Features that concern with finding shapes and objects in computer images and it is based on low level features.
- **General Features:** Application independent features such as color, texture, and shape.
- **Global Features:** Features that are calculated over the entire image or just regular sub-area of an image.
- **Local Features:** Features computed over a subdivision of the image bands that are resulted from image segmentation or edge detection.

3. What is feature extraction?

Feature extraction is the process of transforming raw data into more informative signatures or characteristics of a system, which will most efficiently or meaningfully represent the information that is important for analysis and classification. Following are typical examples.

- A model for predicting the **risk of cardiac disease** may have features such as age, gender, weight, whether the person smokes, whether the person is suffering from diabetic disease, etc.
- A model for predicting whether the person is **suitable for a job** may have features such as the education qualification, number of years of experience, experience working in the field etc.
- A model for predicting the **size of a shirt** for a person may have features such as age, gender, height, weight, etc.
- In **character recognition**, features may include histograms counting the number of black pixels along horizontal and vertical directions, number of internal holes, stroke detection and many others.

- In **speech recognition**, features for recognizing phonemes can include noise ratios, length of sounds, relative power, filter matches and many others.
- In **spam detection algorithms**, features may include the presence or absence of certain email headers, the email structure, the language, the frequency of specific terms, the grammatical correctness of the text.
- In **computer vision**, there are a large number of possible features, such as edges and objects.

4. Explain: feature vector, feature space, and feature construction.

- In pattern recognition and machine learning, a **feature vector** is an n-dimensional vector of numerical features that represent some object. Many algorithms in machine learning require a numerical representation of objects, since such representations facilitate processing and statistical analysis. When representing images, the feature values might correspond to the pixels of an image, while when representing texts the features might be the frequencies of occurrence of textual terms.
- Feature vectors are equivalent to the vectors of explanatory variables used in statistical procedures such as linear regression. Feature vectors are often combined with weights using a dot product in order to construct a linear predictor function that is used to determine a score for making a prediction.
- The vector space associated with these vectors is often called the **feature space**. In order to reduce the dimensionality of the feature space, a number of dimensionality reduction techniques can be employed.
- Higher-level features can be obtained from already available features and added to the feature vector; for example, for the study of diseases the feature 'Age' is useful and is defined as *Age = 'Year of death' minus 'Year of birth'* .
- This process is referred to as **feature construction**. Feature construction is the application of a set of constructive operators to a set of existing features resulting in construction of new features. Examples of such constructive operators include checking for the equality conditions {=, ≠}, the arithmetic operators {+,−,×, /}, the array operators {max(S), min(S), average(S)} as well as other more sophisticated operators, for example count(S,C) that counts the number of features in the feature vector S satisfying some condition C or, for example, distances to other recognition classes generalized by some accepting device.
- Feature construction has long been considered a powerful tool for increasing both accuracy and understanding of structure, particularly in high-dimensional problems. Applications include studies of disease and emotion recognition from speech.

5. What are characteristic of good features?

Here are some characteristics of good features:

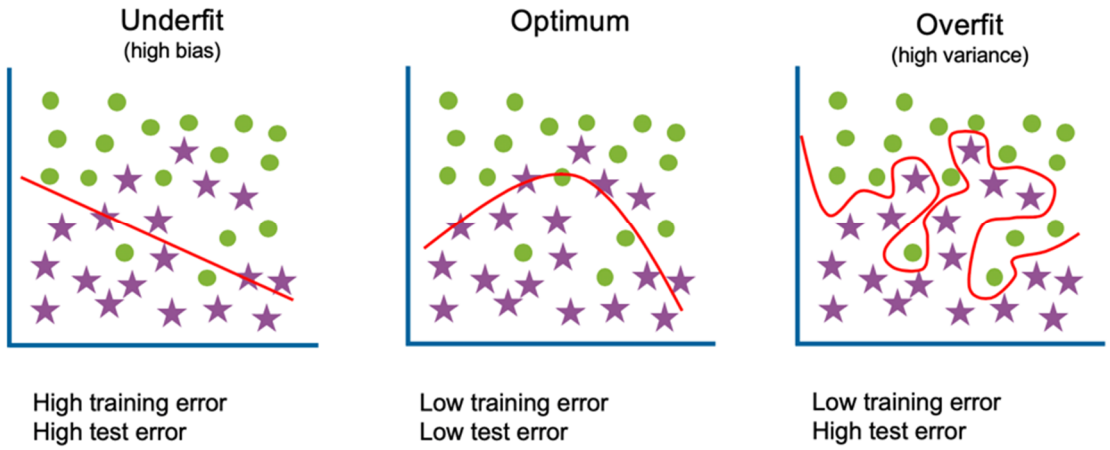
- **Features must be found in most of the data samples**: Great features represent unique characteristics which can be applied across different types of data samples and are not limited to just one data sample. For example, can the "red" color of apple act as a feature? Not really. Because apple can be found in different colors. It might have happened that the sample of apples that was taken for evaluation contained apple of just "red" color. If not found, we may end up creating models having **high bias**.
- **Features must be unique and may not be found prevalent with other (different) forms**: Great features are the ones which is unique to apple and should not be applicable for other fruits. The toughness characteristic of apple such as "hard to teeth" may not be good feature. This is because a guava can also be explained using this feature.
- **Features in reality**: There can be features which can be accidental in nature and is not a feature at all when considering the population. For example, in a particular sample of data, a particular kind of feature can be found to be prevalent. However, when multiple data samples are taken, the feature goes missing.

6. How to determine good features?

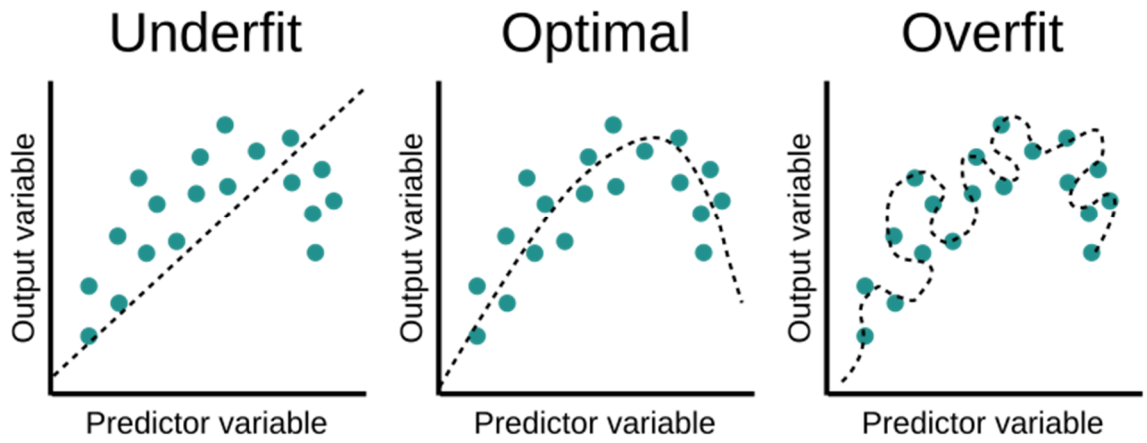
Reasoning by first principles can be of great help when analysing the features of your model. In first principles thinking, we break down the problem into its constituent's parts and work to arrive at the most basic causes or first causes. In relation to machine learning model representing the real-world problem, these first causes can become the features of the model. For example, if there is a need to model a real world situation of student scoring good or bad marks in the examination, we can apply first principles thinking and try and arrive at the most basic causes such as the following in relation to the student getting marks in the examination:

- What are the books referred? Are they the ones recommended by the school or teachers, or, they are extra books as well?
- Does the student take extra coaching?
- Does the student make the notes diligently in the class?
- Does the student refer the class notes as preparatory material?
- Does the student get help from his/her parents?
- Does the student get help from his/her siblings?
- Is the student introvert or extrovert?

7. Represent over-fitting, under-fitting and optimum fitting in classification problem pictorially.



8. Represent over-fitting, under-fitting and optimum fitting in regression problem pictorially.



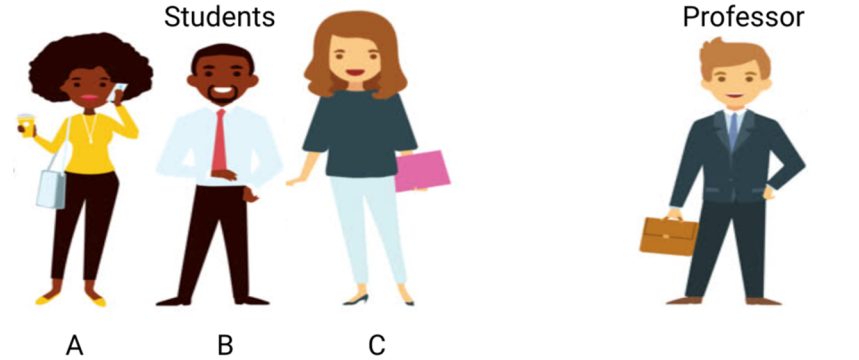
9. Write a note on over-fitting, under-fitting and optimum fitting.

- **Overfitting** and **underfitting** are two of the most common causes of poor model accuracy. The model fit can be predicted by taking a look at the prediction error on the training and test data.
- An **underfit model** results in high prediction errors for both training and test data. An **overfit model** gives a very low prediction error on training data, but a very high prediction error on test data. Both types of models result in poor accuracy.
- An underfit model fails to significantly grasp the relationship between the input values and target variables. This may be the case when the model is too simple (i.e., the input features are not explanatory enough to describe the target well).

- An overfit model has overly memorized the data set it has seen and is unable to generalize the learning to an unseen data set. That is why an overfit model results in very poor test accuracy. Poor test accuracy may occur when the model is highly complex, i.e., the input feature combinations are in a large number and affect the model's flexibility.

10. Explain over-fitting, under-fitting by any practical example.

Consider a math class consisting of 3 students and a professor.



Now, in any classroom, we can broadly divide the students into 3 categories. We'll talk about them one-by-one.

- Hobby = chating
- Not interested in class
- Doesn't pay much attention to professor

Let's say that student A resembles a student who does not like math. She is not interested in what is being taught in the class and therefore does not pay much attention to the professor and the content he is teaching.

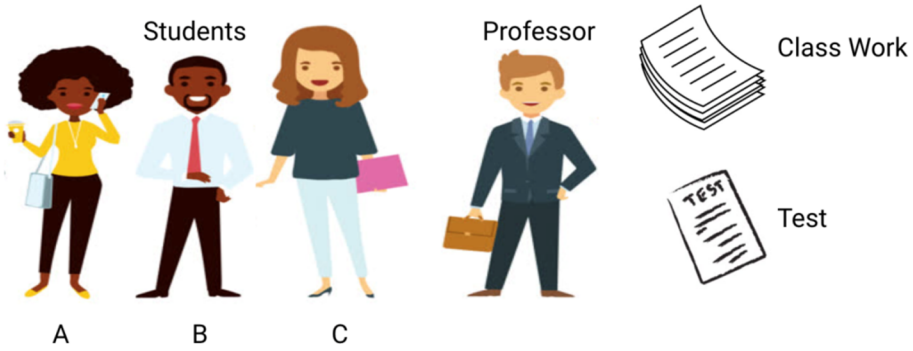
- Hobby = to be best in class.
- Mugs up everything professor says.
- Too much attention to the class work.

Let's consider student B. He is the most competitive student who focuses on memorizing each and every question being taught in class instead of focusing on the key concepts. Basically, he isn't interested in learning the problem-solving approach.

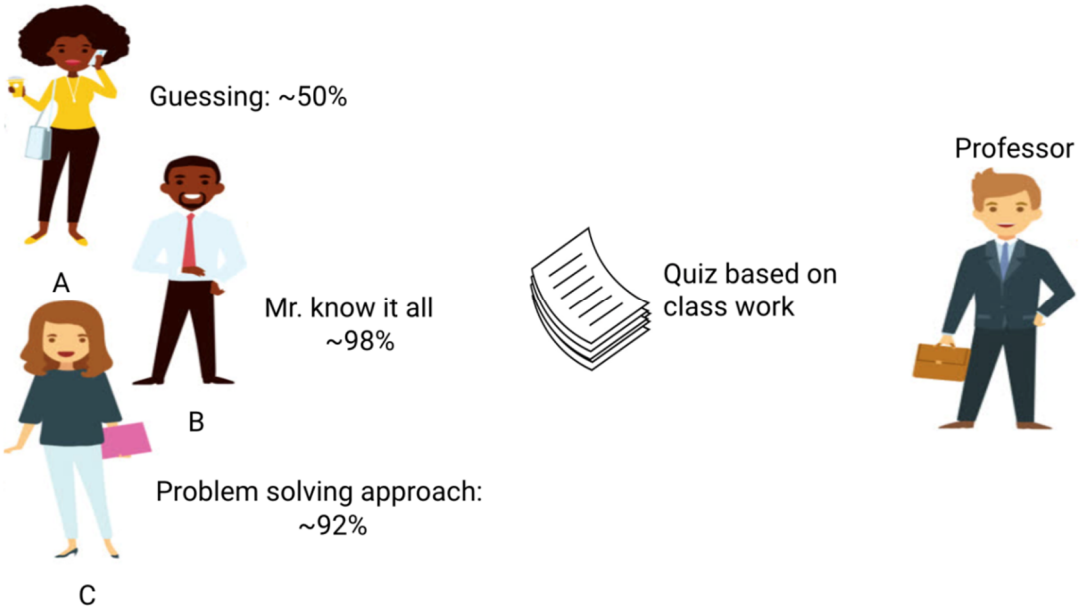
C

- Hobby = learning new things
- Eager to learn concepts.
- Pays attention to class and learns the idea behind solving a problem.

Finally, we have the ideal student C. She is purely interested in learning the key concepts and the problem-solving approach in the math class rather than just memorizing the solutions presented.



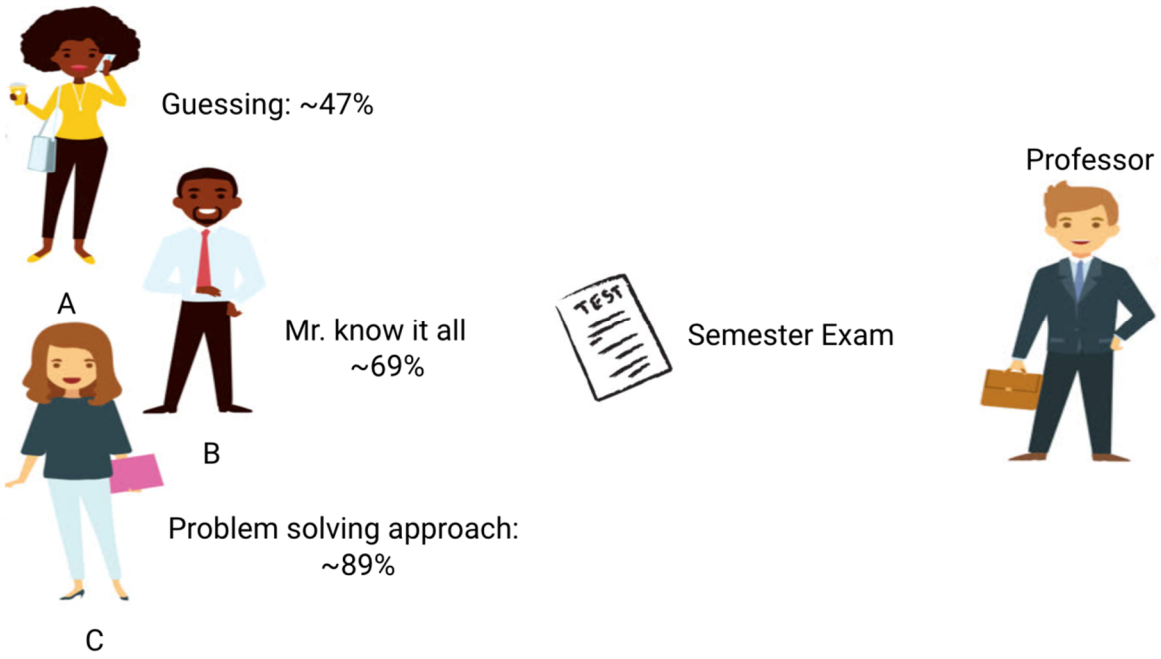
We all know from experience what happens in a classroom. The professor first delivers lectures and teaches the students about the problems and how to solve them. At the end of the day, the professor simply takes a quiz based on what he taught in the class. The obstacle comes in the semester3 tests that the school lays down. This is where new questions (unseen data) come up. The students haven't seen these questions before and certainly haven't solved them in the classroom. Sounds familiar? So, let's discuss what happens when the teacher takes a classroom test at the end of the day:



- Student A, who was distracted in his own world, simply guessed the answers and got approximately 50% marks in the test
- On the other hand, the student who memorized each and every question taught in the classroom was able to answer almost every question by memory and therefore obtained 98% marks in the class test
- For student C, she actually solved all the questions using the problem-solving approach she learned in the classroom and scored 92%

We can clearly infer that the student who simply memorizes everything is scoring better without much difficulty.

Now here's the twist. Let's also look at what happens during the monthly test, when students have to face new unknown questions which are not taught in the class by the teacher.



- In the case of student A, things did not change much and he still randomly answers questions correctly ~50% of the time.
- In the case of Student B, his score dropped significantly. Can you guess why? This is because he always memorized the problems that were taught in the class but this monthly test contained questions which he has never seen before. Therefore, his performance went down significantly
- In the case of Student C, the score remained more or less the same. This is because she focused on learning the problem-solving approach and therefore was able to apply the concepts she learned to solve the unknown questions

You might be wondering how this example relates to the problem which we encountered during the train and test scores of the decision tree classifier? Good question!

A

Not interested in learning

Class test ~50%
Test ~47%

B

Memorizing the lessons

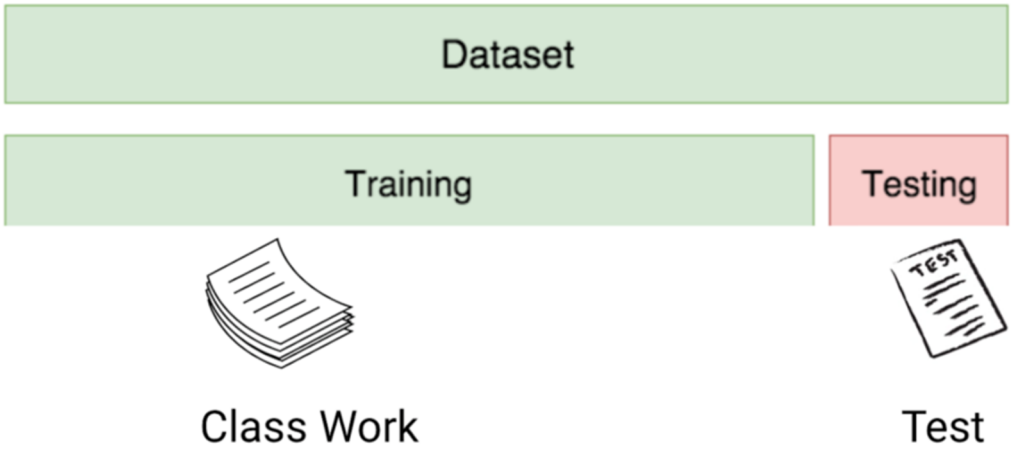
Class test ~98%
Test ~69%

C

Conceptual Learning

Class test ~92%
Test ~89%

So, let's work on connecting this example with the results of the decision tree classifier that I showed you earlier.



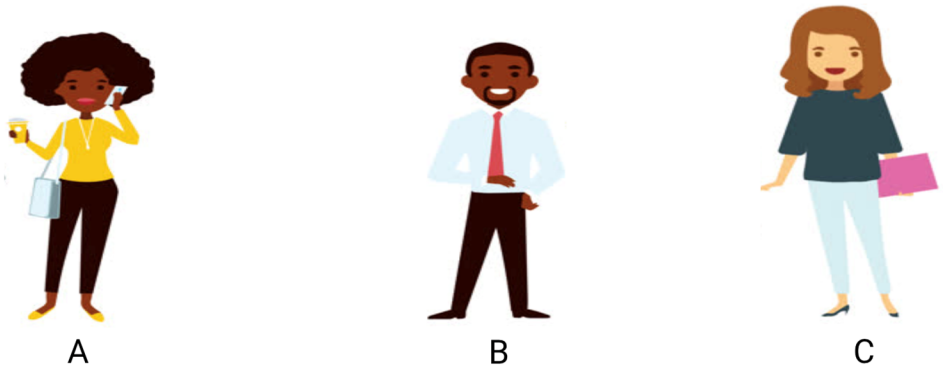
First, the classwork and class test resemble the training data and the prediction over the training data itself respectively.

On the other hand, the semester test represents the test set from our data which we keep aside before we train our model (or unseen data in a real-world machine learning project).

Now, recall our decision tree classifier I mentioned earlier. It gave a perfect score over the training set but struggled with the test set.

Comparing that to the student examples we just discussed, the classifier establishes an analogy with student B who tried to memorize each and every question in the training set.

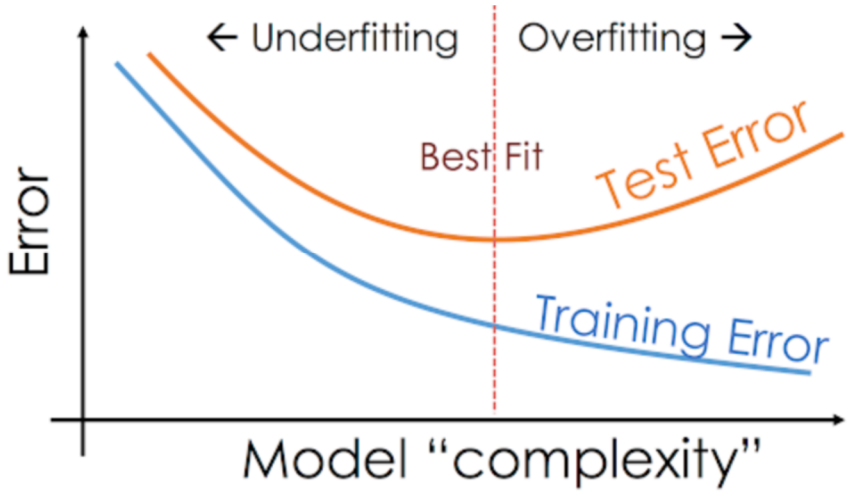
Similarly, our decision tree classifier tries to learn each and every point from the training data but suffers radically when it encounters a new data point in the test set. It is not able to generalize it well.



A	B	C
Not interested in learning	Memorizing the lessons	Conceptual Learning
Class test ~50% Test ~47%	Class test ~98% Test ~69%	Class test ~92% Test ~89%
Under-fit/ biased learning	**Over-fit/ Memorizing**	**Best-fit**

This situation where any given model is performing too well on the training data but the performance drops significantly over the test set is called an over-fitting model.

For example, non-parametric models like decision trees, KNN, and other tree-based algorithms are very prone to over-fitting. These models can learn very complex relations which can result in over-fitting. The graph below summarises this concept:



On the other hand, if the model is performing poorly over the test and the train set, then we call that an under-fitting model. An example of this situation would be building a linear regression model over non-linear data.

11. Why use Principal component analysis (PCA)?

- Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning. It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the

help of orthogonal transformation. These new transformed features are called the **Principal Components**.

- It is one of the popular tools that is used for exploratory data analysis and predictive modeling. It is a technique to draw strong patterns from the given dataset by reducing the variances.
- PCA generally tries to find the lower-dimensional surface to project the high-dimensional data.
- PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality.
- Some real-world applications of PCA are ***image processing, movie recommendation system, optimizing the power allocation in various communication channels.***
- It is a feature extraction technique, so it contains the important variables and drops the least important variable.

12. Explain common terms used in PCA algorithm?

Some common terms used in PCA algorithm:

- **Dimensionality:** It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.
- **Correlation:** It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.
- **Orthogonal:** It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.
- **Eigenvectors:** If there is a square matrix M, and a non-zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v.
- **Covariance Matrix:** A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

13. What are Principal Components in PCA?

As described above, the transformed new features or the output of PCA are the Principal Components. The number of these PCs is either equal to or less than the original features present in the dataset. Some properties of these principal components are given below:

- The principal component must be the linear combination of the original features.
- These components are orthogonal, i.e., the correlation between a pair of variables is zero.
- The importance of each component decreases when going to 1 to n, it means the 1 PC has the most importance, and n PC will have the least importance.

14. What are the advantages of using PCA on your dataset?

Here are several reasons why you want to use PCA:

- **Removes correlated features**. PCA will help you remove all the features that are correlated, a phenomenon known as *multi-collinearity*. Finding features that are correlated is time consuming, especially if the number of features is large.**Improves machine learning algorithm performance**. With the number of features reduced with PCA, the time taken to train your model is now significantly reduced.
- **Reduce overfitting**. By removing the unnecessary features in your dataset, PCA helps to overcome overfitting.

On the other hand, PCA has its disadvantages:

- **Independent variables are now less interpretable**. PCA reduces your features into smaller number of components. Each component is now a linear combination of your original features, which makes it less readable and interpretable.
- **Information loss**. Data loss may occur if you do not exercise care in choosing the right number of components.
- **Feature scaling**. Because PCA is a *variance maximizing* exercise, PCA requires features to be scaled prior to processing.

15. What are the assumptions and limitations of PCA?

PCA is related to the set of operations in the Pearson correlation, so it inherits similar assumptions and limitations:

- **PCA assumes a correlation between features**. If the features (or dimensions or columns, in tabular data) are not correlated, PCA will be unable to determine principal components.
- **PCA is sensitive to the scale of the features**. Imagine we have two features - one takes values between 0 and 1000, while the other takes values between 0 and 1. PCA will be extremely biased towards the first feature being the first principle component, regardless of the *actual* maximum variance within the data. This is why it's so important to standardize the values first.
- **PCA is not robust against outliers**. Similar to the point above, the algorithm will be biased in datasets with strong outliers. This is why it is recommended to remove outliers before performing PCA.
- **PCA assumes a linear relationship between features**. The algorithm is not well suited to capturing non-linear relationships. That's why it's advised to turn non-linear features or relationships between features into linear, using the standard methods such as log transforms.

- **Technical implementations often assume no missing values**. When computing PCA using statistical software tools, they often assume that the feature set has no missing values (no empty rows). Be sure to remove those rows and/or columns with missing values, or impute missing values with a close approximation (e.g. the mean of the column).

Mathematics based questions

16. What are different statistical features? Write their mathematical expressions.

The mathematical expressions for descriptive statistical attributes:

Sr. No.	Attribute	Mathematical Expression
1	Kurtosis	It is an estimate of the 'tailedness' of the probability distribution of a real-valued random variable. $\left\{\frac{n(n+1)}{(n-1)(n-2)(n-3)}\sum\left(\frac{x_i-\bar{x}}{S_d}\right)^4\right\}-\frac{3(n-1)^2}{(n-2)(n-3)}$
2	Standard Error	Standard Error is a measure of the deviation of the sample means from the population. The standard error of a sample statistic is an estimate of the standard deviation of the sampling distribution of that sample statistic. It helps you to find out confidence intervals for that statistic at different significance levels. $\sqrt{\frac{1}{n-2}\left(\sum(y-\bar{y})^2-\frac{\sum[(x-\bar{x})(y-\bar{y})]^2}{\sum(x-\bar{x})^2}\right)}$
3	Maximum value	It is the highest data point value.
4	Skewness	It defines the inclination of the spread of data on either side. $\frac{n}{(n-1)(n-2)}\sum\left(\frac{x_i-\bar{x}}{S_d}\right)^3$
5	Minimum value	It is the lowest data point value.
6	Range	It is an estimate of subtraction between maximum and minimum values of data points.
7	Count	It is an estimate of the number of data points in each sample.
8	Summation	It is an estimate of the sum of all feature values for each sample.
9	Variance	It is the expectation of the squared deviation of a random variable from its mean. $\frac{n\sum x^2-(\sum x)^2}{n(n-1)}$

10	Standard Deviation	It is a measure of the amount of variation or dispersion of a set of values. $\sqrt{\frac{n\sum x^2 - (\sum x)^2}{n(n-1)}}$
11	Mode	It is an estimate of the number which occurs most frequently in a set of data points.
12	Median	It is an estimate of the middle value segregating the higher and lower splits of a data set.
13	Mean	It is an estimate of the average (arithmetic) of a set of data points. $\bar{x} = \frac{x_1 + x_2 + x_3 + \dots\dots + x_n}{n}$
14	Impulse factor	The impulse factor estimates the impact created by considering the ratio of maximum value and average value. $\frac{Maximum\ value\ of\ x}{\bar{x}}$
15	K-factor	The K-factor is a product of root mean square value and maximum value. $\frac{\sqrt{\frac{x_1^2 + x_2^2 + x_3^2 + \dots\dots + x_n^2}{n}}}{Maximum\ value\ of\ x}$
16	Shape factor	The shape factor is the ratio of root mean square value and average value. $\frac{\sqrt{\frac{x_1^2 + x_2^2 + x_3^2 + \dots\dots + x_n^2}{n}}}{\bar{x}}$

17. Explain mathematics behind PCA.

The central idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of a large number of interrelated variables while retaining as much as possible of the variation present in the data set. This is achieved by transforming to a new set of variables, the *principal components (PCs)*, which are uncorrelated, and which are ordered so that the first few retain most of the variation present in all of the original variables. PCA can be thought of as an unsupervised learning problem. The whole process of obtaining principle components from a raw dataset can be simplified in six parts :

- Take the whole dataset consisting of *d+1 dimensions* and ignore the labels such that our new dataset becomes *d dimensional.*
- Compute the *mean* for every dimension of the whole dataset.

- Compute the *covariance matrix* of the whole dataset.
- Compute *eigenvectors* and the corresponding *eigenvalues*.
- Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a *d* × *k dimensional* matrix **W.**
- Use this *d* × *k eigenvector matrix* to transform the samples onto the new subspace.

So, let's unfurl the maths behind each of this one by one.

1. Take the whole dataset consisting of *d+1 dimensions* and ignore the labels such that our new dataset becomes *d dimensional.*

Let's say we have a dataset which is *d+1 dimensional*. Where d could be thought as *X_train* and 1 could be thought as y_train *(labels)* in modern machine learning paradigm. So, *X_train + y_train* make up our complete train dataset. So, after we drop the labels we are left with *d dimensional* dataset and this would be the dataset we will use to find the principal components. Also, let's assume we are left with a three-dimensional dataset after ignoring the labels i.e d = 3. We will assume that the samples stem from two different classes, where one-half samples of our dataset are labeled class 1 and the other half class 2.

Let our data matrix **X** be the score of three students:

Student	Math	English	Art
1	90	60	90
2	90	90	30
3	60	60	60
4	60	60	90
5	30	30	30

2. Compute the mean of every dimension of the whole dataset.

The data from the above table can be represented in matrix **A,** where each column in the matrix shows scores on a test and each row shows the score of a student.

$$\mathbf{A} = \begin{bmatrix} 90 & 60 & 90 \\ 90 & 90 & 30 \\ 60 & 60 & 60 \\ 60 & 60 & 90 \\ 30 & 30 & 30 \end{bmatrix}$$

Matrix A

So, The mean of matrix **A** would be

$$\bar{\mathbf{A}} = [\,66 \quad 60 \quad 60\,]$$

Mean of Matrix A

3. Compute the *covariance matrix* of the whole dataset (sometimes also called as the variance-covariance matrix)

So, we can compute the covariance of two variables **X** and **Y** using the following formula

$$cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(Xi-\bar{x})(Yi-\bar{y})$$

Using the above formula, we can find the covariance matrix of **A.** Also, the result would be a *square matrix of d ×d dimensions.*

Let's rewrite our original matrix like this

	Math	*English*	*Arts*
1	90	60	90
2	90	90	30
3	60	60	60
4	60	60	90
5	30	30	30

Matrix A

Its c*ovariance matrix* would be

	Math	*English*	*Art*
Math	504	360	180
English	360	360	0
Art	180	0	720

Covariance Matrix of A

Few points that can be noted here is:

- Shown in *Blue* along the diagonal, we see the variance of scores for each test. The art test has the biggest variance (720); and the English test, the smallest (360). So we can say that art test scores have more variability than English test scores.
- The covariance is displayed in black in the off-diagonal elements of the matrix **A**

a) The covariance between math and English is positive (360), and the covariance between math and art is positive (180). This means the scores tend to covary in a positive way. As scores on math go up, scores on art and English also tend to go up; and vice versa.

b) The covariance between English and art, however, is zero. This means there tends to be no predictable relationship between the movement of English and art scores.

4. Compute Eigenvectors and corresponding Eigenvalues

Intuitively, an eigenvector is a vector whose direction remains unchanged when a linear transformation is applied to it.

Now, we can easily compute eigenvalue and eigenvectors from the covariance matrix that we have above.

Let ***A*** be a square matrix, **v** a vector and **λ** a scalar that satisfies ***A***v **= λ**v, then **λ** is called eigenvalue associated with eigenvector **v** of ***A***. The eigenvalues of ***A*** are roots of the characteristic equation

$$\det(A-\lambda I) = 0$$

Calculating *det(A-λI)* first, *I* is an identity matrix :

$$\det\left(\begin{pmatrix}504 & 360 & 180\\ 360 & 360 & 0\\ 180 & 0 & 720\end{pmatrix} - \lambda\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}\right)$$

Simplifying the matrix first, we can calculate the determinant later,

$$\begin{pmatrix}504 & 360 & 180\\ 360 & 360 & 0\\ 180 & 0 & 720\end{pmatrix} - \begin{pmatrix}\lambda & 0 & 0\\ 0 & \lambda & 0\\ 0 & 0 & \lambda\end{pmatrix}$$

$$\begin{pmatrix}504-\lambda & 360 & 180\\ 360 & 360-\lambda & 0\\ 180 & 0 & 720-\lambda\end{pmatrix}$$

Now that we have our simplified matrix, we can find the determinant of the same :

$$\det\begin{pmatrix}504-\lambda & 360 & 180\\ 360 & 360-\lambda & 0\\ 180 & 0 & 720-\lambda\end{pmatrix}$$

$$-\lambda^3 + 1584\lambda^2 - 641520\lambda + 25660800$$

We now have the equation and we need to solve for *λ,* so as to get the *eigenvalue of the matrix.* So, equating the above equation to zero :

$$-\lambda^3 + 1584\lambda^2 - 641520\lambda + 25660800 = 0$$

After solving this equation for the value of ***λ,*** we get the following value

$$\lambda \approx 44.81966\ldots, \lambda \approx 629.11039\ldots, \lambda \approx 910.06995\ldots$$

Eigenvalues

Now, we can calculate the eigenvectors corresponding to the above eigenvalues. So, after solving for *eigenvectors* we would get the following solution for the corresponding *eigenvalues*

$$\begin{pmatrix}-3.75100\ldots\\ 4.28441\ldots\\ 1\end{pmatrix}, \begin{pmatrix}-0.50494\ldots\\ -0.67548\ldots\\ 1\end{pmatrix}, \begin{pmatrix}1.05594\ldots\\ 0.69108\ldots\\ 1\end{pmatrix}$$

5. Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a *d* × *k dimensional* matrix W.

We started with the goal to reduce the dimensionality of our feature space, i.e., projecting the feature space via PCA onto a smaller subspace, where the eigenvectors will form the axes

of this new feature subspace. However, the eigenvectors only define the directions of the new axis, since they have all the same unit length 1. So, in order to decide which eigenvector(s) we want to drop for our lower-dimensional subspace, we have to take a look at the corresponding eigenvalues of the eigenvectors. Roughly speaking, the eigenvectors with the lowest eigenvalues bear the least information about the distribution of the data, and those are the ones we want to drop. The common approach is to rank the eigenvectors from highest to lowest corresponding eigenvalue and choose the top *k* eigenvectors. So, after sorting the eigenvalues in decreasing order, we have

$$\begin{pmatrix} 910.06995 \\ 629.11039 \\ 44.81966 \end{pmatrix}$$

For our simple example, where we are reducing a 3-dimensional feature space to a 2-dimensional feature subspace, we are combining the two eigenvectors with the highest eigenvalues to construct our *d*×*k* dimensional eigenvector matrix **W.** So, *eigenvectors* corresponding to two maximum eigenvalues are:

$$\mathbf{W} = \begin{bmatrix} 1.05594 & -0.50494 \\ 0.69108 & -0.67548 \\ 1 & 1 \end{bmatrix}$$

6. Transform the samples onto the new subspace

In the last step, we use the 2×3 dimensional matrix ***W*** that we just computed to transform our samples onto the new subspace via the equation ***y = W′ × x*** where ***W′*** is the *transpose* of the matrix ***W.*** *So lastly, we have computed our two principal components and projected the data points onto the new subspace.*

Problems/Numerical

18. Problems on PCA

Problem 1:

Consider the following dataset

x1	2.5	0.5	2.2	1.9	3.1	2.3	2.0	1.0	1.5	1.1
x2	2.4	0.7	2.9	2.2	3.0	2.7	1.6	1.1	1.6	0.9

Step 1: Standardize the Dataset

Mean for x_1= 1.81 = x_{1mean}

Mean for x_2= 1.91 = x_{2mean}

We will change the dataset.

x1	0.69	-1.31	0.39	0.09	1.29	0.49	0.19	-0.81	-0.31	-0.71
x2	0.49	-1.21	0.99	0.29	1.09	0.79	-0.31	-0.81	-0.31	-1.01

Step 2: Find the Eigenvalues and eigenvectors

Correlation Matrix c = $C = \left(\frac{X \cdot X^T}{N-1}\right)$

where, X is the Dataset Matrix (In this numerical, it is a 10 X 2 matrix)

X^T is the transpose of the X (In this numerical, it is a 2 X 10 matrix) and N is the number of elements = 10

So, $C = \left(\frac{X \cdot X^T}{10-1}\right) = \left(\frac{X \cdot X^T}{9}\right)$

{So in order to calculate the Correlation Matrix, we have to do the multiplication of the Dataset Matrix with its transpose}

$$C = \begin{bmatrix} 0.616556 & 0.615444 \\ 0.615444 & 0.716556 \end{bmatrix}$$

Using the equation, **| C –** λ**I | = 0– equation (i)** where { \lambda is the eigenvalue and I is the Identity Matrix }

So solving equation (i)

$$\begin{bmatrix} 0.616556 & 0.615444 \\ 0.615444 & 0.716556 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{vmatrix} 0.616556 - \lambda & 0.615444 \\ 0.615444 & 0.716556 - \lambda \end{vmatrix} = 0$$

Taking the determinant of the left side, we get

$$0.44180 - 0.616556\lambda - 0.716556\lambda + \lambda^2 - 0.37877 = 0$$

$$\lambda^2 - 1.33311\lambda + 0.06303 = 0$$

We get two values for λ, that are **(**λ_1**) = 1.28403 and (**λ_2**) = 0.0490834**. Now we have to find the eigenvectors for the eigenvalues λ_1 **and** λ_2

To find the eigenvectors from the eigenvalues, we will use the following approach:

First, we will find the eigenvectors for the eigenvalue 1.28403 by using the equation $C \cdot X = \lambda \cdot X$

$$\begin{bmatrix} 0.616556 & 0.615444 \\ 0.615444 & 0.716556 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = 1.28403 \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} 0.616556x + 0.615444y \\ 0.615444x + 0.716556y \end{bmatrix} = \begin{bmatrix} 1.28403x \\ 1.28403y \end{bmatrix}$$

Solving the matrices, we get

0.616556x + 0.615444y = 1.28403x ; **x = 0.922049 y**

(x and y belongs to the matrix X) so if we put y = 1, x comes out to be 0.922049. So now the updated X matrix will look like:

$$X = \begin{bmatrix} 0.922049 \\ 1 \end{bmatrix}$$

IMP: Till now we haven't reached to the eigenvectors, we have to a bit of modifications in the X matrix. They are as follows:

A. Find the square root of the sum of the squares of the element in X matrix i.e.

$$\sqrt{0.922049^2 + 1^2} = \sqrt{0.850174 + 1} = \sqrt{1.850174} = 1.3602$$

B. Now divide the elements of the X matrix by the number 1.3602 (just found that)

$$\begin{bmatrix} \frac{0.922049}{1.3602} \\ \\ \frac{1}{1.3602} \end{bmatrix} = \begin{bmatrix} 0.67787 \\ 0.73518 \end{bmatrix}$$

So now we found the eigenvectors for the eigenvector λ_1, they are 0.67787 and 0.73518

Secondly, we will find the eigenvectors for the eigenvalue 0.0490834 by using the equation {Same approach as of previous step)

$$\begin{bmatrix} 0.616556 & 0.615444 \\ 0.615444 & 0.716556 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = 0.0490834 \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} 0.616556x + 0.615444y \\ 0.615444x + 0.716556y \end{bmatrix} = \begin{bmatrix} 0.0490834x \\ 0.0490834y \end{bmatrix}$$

Solving the matrices, we get

0.616556x + 0.615444y = 0.0490834x; **y = -0.922053**

(x and y belongs to the matrix X) so if we put x = 1, y comes out to be -0.922053 So now the updated X matrix will look like:

$$X = \begin{bmatrix} 1 \\ -0.922053 \end{bmatrix}$$

IMP: Till now we haven't reached to the eigenvectors, we have to a bit of modifications in the X matrix. They are as follows:

A. Find the square root of the sum of the squares of the elements in X matrix i.e.

$$\sqrt{1^2 + (-0.922053)^2} = \sqrt{1 + 0.85018} = \sqrt{1.85018} = 1.3602$$

B. Now divide the elements of the X matrix by the number 1.3602 (just found that)

$$\begin{bmatrix} \frac{1}{1.3602} \\ \frac{-0.922053}{1.3602} \end{bmatrix} = \begin{bmatrix} 0.735179 \\ 0.677873 \end{bmatrix}$$

So now we found the eigenvectors for the eigenvector \lambda_2, they are 0.735176 and 0.677873

Sum of eigenvalues (λ_1) and (λ_2) = 1.28403 + 0.0490834 = 1.33 = Total Variance {Majority of variance comes from λ_1}

Step 3: Arrange Eigenvalues

The eigenvector with the highest eigenvalue is the Principal Component of the dataset. So in this case, eigenvectors of lambda1 are the principal components.

{Basically in order to complete the numerical we have to only solve till this step, but if we have to prove why we have chosen that particular eigenvector we have to follow the steps from 4 to 6}

Step 4: Form Feature Vector

$\begin{bmatrix} 0.677873 & 0.735179 \\ 0.735179 & -0.677879 \end{bmatrix}$ This is the FEATURE VECTOR for Numerical

Where first column are the eigenvectors of λ_1 & second column are the eigenvectors of λ_2

Step 5: Transform Original Dataset

Use the equation Z = X V

$$Z = \begin{bmatrix} 0.69 & 0.49 \\ -1.31 & -1.21 \\ 0.39 & 0.99 \\ 0.09 & 0.29 \\ 1.29 & 1.09 \\ 0.49 & 0.79 \\ 0.19 & -0.31 \\ -0.81 & -0.81 \\ -0.31 & -0.31 \\ -0.71 & -1.01 \end{bmatrix} \cdot \begin{bmatrix} 0.677873 & 0.735179 \\ 0.735179 & -0.677879 \end{bmatrix} = \begin{bmatrix} 0.8297008 & 0.17511574 \\ -1.77758022 & -0.14285816 \\ 0.99219768 & -0.38437446 \\ 0.27421048 & -0.13041706 \\ 1.67580128 & 0.20949934 \\ 0.91294918 & -0.17528196 \\ -1.14457212 & -0.04641786 \\ -0.43804612 & -0.01776486 \\ -1.22382.62 & 0.16267464 \end{bmatrix} =$$

Step 6: Reconstructing Data

Use the equation X = $Z * V^T$ ($V^T$is Transpose of V), X = Row Zero Mean Data

$$\begin{bmatrix} 0.8297008 & 0.17511574 \\ -1.77758022 & -0.14285816 \\ 0.99219768 & -0.38437446 \\ 0.27421048 & -0.13041706 \\ 1.67580128 & 0.20949934 \\ 0.91294918 & -0.17528196 \\ -1.14457212 & -0.04641786 \\ -0.43804612 & -0.01776486 \\ -1.22382.62 & 0.16267464 \end{bmatrix} \cdot \begin{bmatrix} 0.677873 & 0.735179 \\ 0.735176 & -0.677879 \end{bmatrix} = \begin{bmatrix} 0.6899999766573 & 0.4899999834233 \\ -1.3099999556827 & -1.2099999590657 \\ 0.389999968063 & 0.9899999665083 \\ 0.0899999969553 & 0.2899999901893 \\ 0.61212695653593 & 0.35482096313253 \\ 0.4899999834233 & 0.7899999732743 \\ 0.189999935723 & -0.309999995127 \\ -0.8099999725977 & -0.8099999725977 \\ -0.3099999895127 & -0.3099999895127 \\ -0.7099999759807 & -1.0099999658317 \end{bmatrix}$$

So in order to reconstruct the original data, we follow:

Row Original DataSet = Row Zero Mean Data + Original Mean

$$\begin{bmatrix} 0.6899999766573 & 0.4899999834233 \\ -1.3099999556827 & -1.2099999590657 \\ 0.389999968063 & 0.9899999665083 \\ 0.0899999969553 & 0.2899999901893 \\ 0.61212695653593 & 0.35482096313253 \\ 0.4899999834233 & 0.7899999732743 \\ 0.189999935723 & -0.309999995127 \\ -0.8099999725977 & -0.8099999725977 \\ -0.3099999895127 & -0.3099999895127 \\ -0.7099999759807 & -1.0099999658317 \end{bmatrix} + \begin{bmatrix} 1.81 & 1.91 \end{bmatrix} = \begin{bmatrix} 2.49 & 2.39 \\ 0.5 & 0.7 \\ 2.19 & 2.89 \\ 1.89 & 2.19 \\ 3.08 & 2.99 \\ 2.30 & 2.7 \\ 2.01 & 1.59 \\ 1.01 & 1.11 \\ 1.5 & 1.6 \\ 1.1 & 0.9 \end{bmatrix}$$

So for the eigenvectors of first eigenvalue, data can be reconstructed similar to the original dataset. Thus we can say that the Principal Component of the dataset is $\lambda_1$is 1.28403 followed by λ_2**that is 0.0490834**

Problem 2:

Consider the following dataset

x	2.5	0.5	2.2	1.9	3.1	2.3	2	1	1.5	1.1
y	2.4	0.7	2.9	2.2	3	2.7	1.6	1.1	1.6	0.9

Calculate mean for x & y and differences.

X	$\overline{X}$	$x-\overline{X}$	***y***	$y-\overline{Y}$	$\overline{Y}$
2.5		0.69	2.4		0.49
0.5		-1.31	0.7		-1.21
2.2		0.39	2.9		0.99
1.9		0.09	2.2		0.29
3.1	**1.81**	1.29	3.0	**1.91**	1.09
2.3		0.49	2.7		0.79
2		0.19	1.6		-0.31
1		-0.81	1.1		-0.81
1.5		-0.31	1.6		-0.31
1.1		-0.71	0.9		-1.01

$$C = \begin{bmatrix} \frac{\sum_{i=1}^{n}(X_i-\overline{X})(X_i-\overline{X})}{n-1} & \frac{\sum_{i=1}^{n}(X_i-\overline{X})(Y_i-\overline{Y})}{n-1} \\ \frac{\sum_{i=1}^{n}(Y_i-\overline{Y})(X_i-\overline{X})}{n-1} & \frac{\sum_{i=1}^{n}(Y_i-\overline{Y})(Y_i-\overline{Y})}{n-1} \end{bmatrix}$$

$$C = \begin{bmatrix} 0.6165 & 0.6154 \\ 0.6154 & 0.7165 \end{bmatrix}$$

$$\text{Eigenvalues} = \begin{bmatrix} 0.0491 \\ 1.2840 \end{bmatrix}$$

$$\text{Eigenvectors} = \begin{bmatrix} -0.7352 & -0.6779 \\ 0.6779 & -0.7352 \end{bmatrix}$$

$$\text{Eigenvalues (Highest to Lowest)} = \begin{bmatrix} 1.284 \\ 0.0491 \end{bmatrix}$$

$$\text{Feature Vectors} = \begin{bmatrix} -0.6779 & -0.7352 \\ -0.7352 & 0.6779 \end{bmatrix}$$

Topic: Feature selection

Theory	Mathematics	Numerical
✓	✓	✓

Theory questions

19. What is feature selection?

Feature selection is the process of selecting a subset of the constructed features according to a certain criteria. Reducing the number of features as well as removing the irrelevant, redundant, or noisy data increases the efficiency and effectiveness of the implemented algorithms. This is an important and frequently used dimensionality reduction technique for various applications, e.g. data mining, face recognition

20. Explain Filters, Wrappers, and embedded methods in detail.

Filter Methods

- In this method, select subsets of variables as a pre-processing step, independently of the used classifier. The **Variable Ranking**-***Feature Selection*** is a Filter Method.

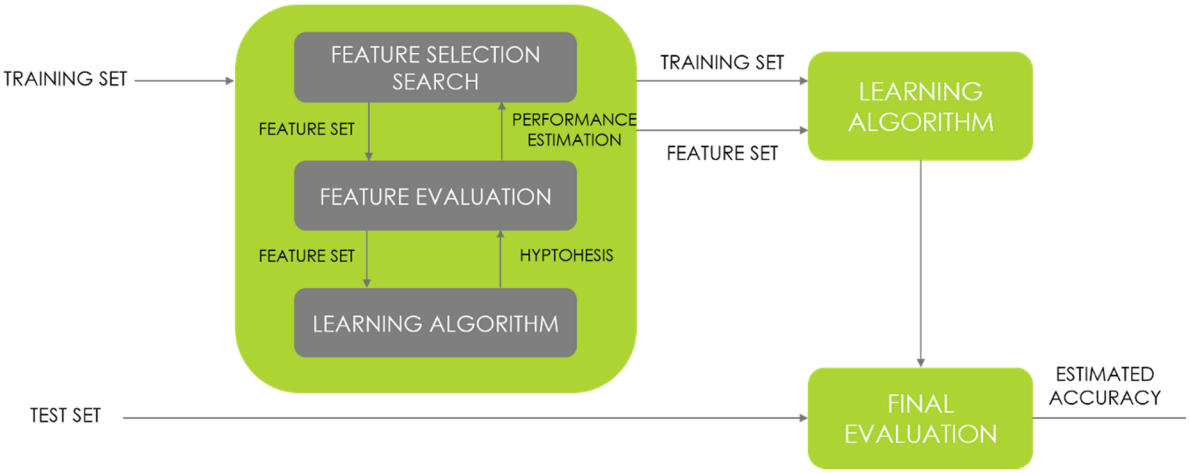
Filter Methods: Feature Subset Selection Method

Key features of Filter Methods for **Feature Subset Selection:**

- Filter Methods are usually fast
- They provide generic selection of features, not tuned by given learner (universal)
- Filter Methods are also often criticized (feature set not optimized for used classifier)
- Filter Methods are sometimes used as a pre-processing step for other methods.

Wrapper Methods

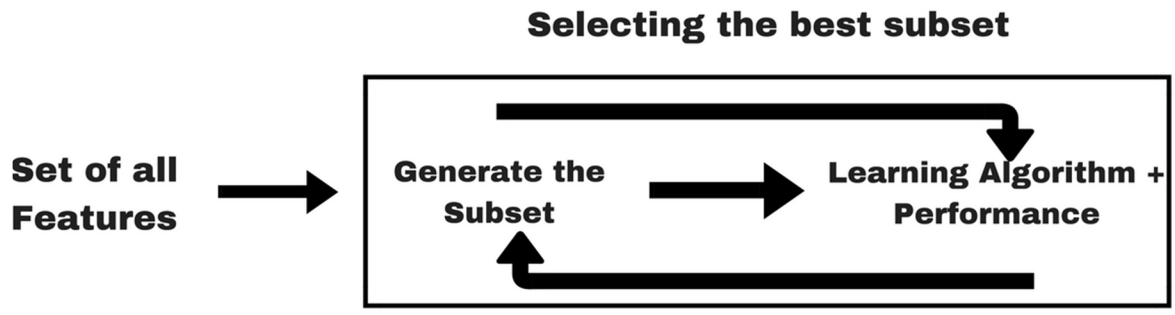
- The Learner is considered a black-box. Interface of the black-box is used to score subsets of variables according to the predictive power of the learner when using the subsets.
- Results vary for different learners
- One needs to define: – how to search the space of all possible variable subsets ?– how to assess the prediction performance of a learner ?



Wrapper Methods: Feature Subset Selection Method

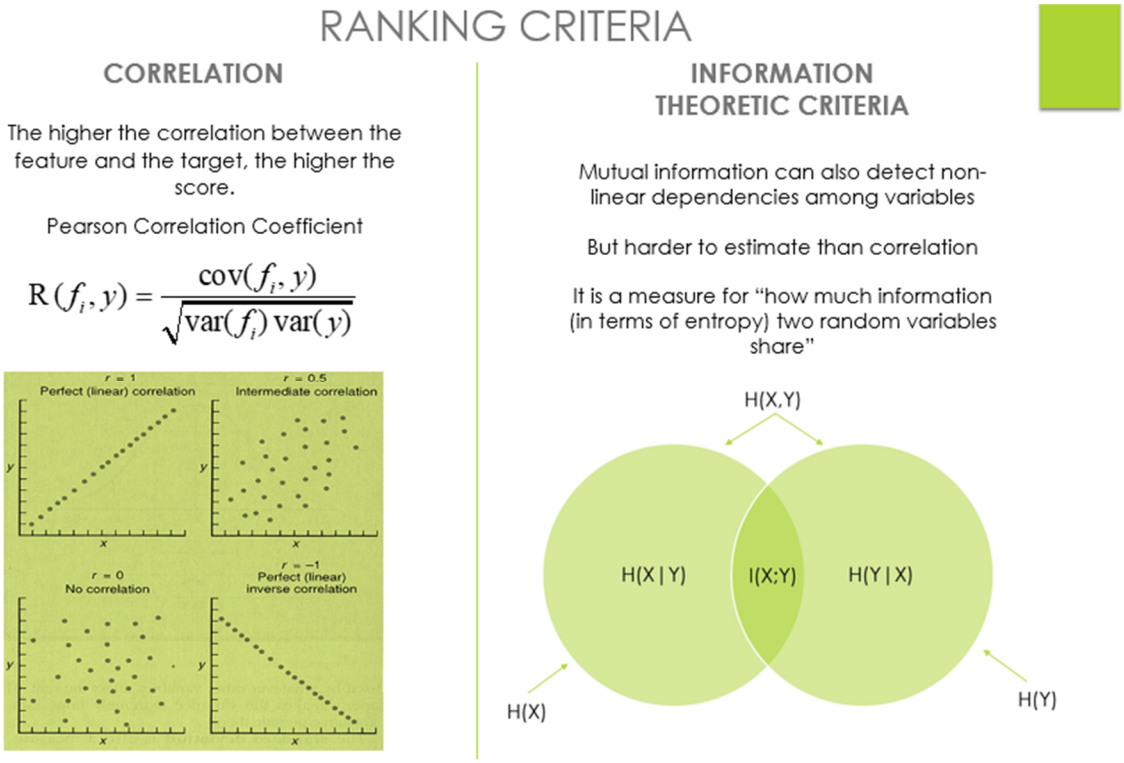
Embedded Methods

- Embedded methods are iterative in a sense that takes care of each iteration of the model training process and carefully extract those features which contribute the most to the training for a particular iteration.
- Regularization methods are the most commonly used embedded methods which penalize a feature given a coefficient threshold.
- This is why Regularization methods are also called penalization methods that introduce additional constraints into the optimization of a predictive algorithm (such as a regression algorithm) that bias the model toward lower complexity (fewer coefficients).
- Some of the most popular examples of these methods are LASSO and RIDGE regression which have inbuilt penalization functions to reduce overfitting.
- Embedded methods can be explained with the help of following graphic:

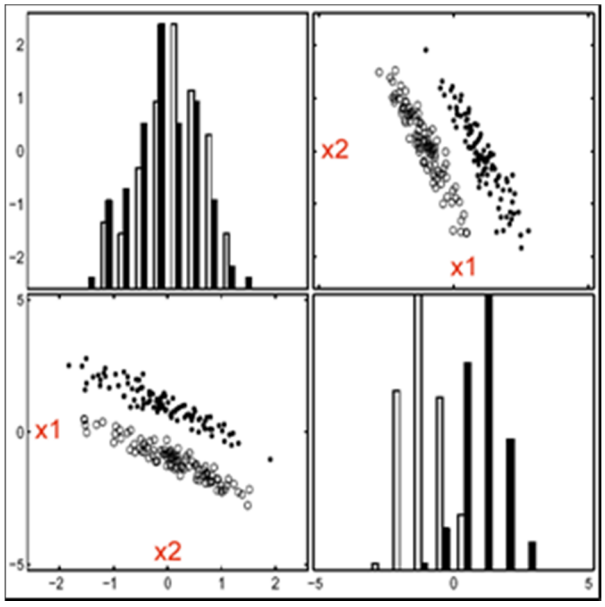


21. Explain feature ranking as a feature selection method.

Variable Ranking is the process of ordering the features by the value of some scoring function, which usually measures feature-relevance. Resulting set: The score *S(fi)* is computed from the training data, measuring some criteria of feature *fi*. By convention a high score is indicative for a valuable (relevant) feature. A simple method for *feature selection* using **variable ranking** is to select the *k* highest ranked features according to *S*. This is usually not optimal, but often preferable to other, more complicated methods. It is computationally efficient — only calculation and sorting of *n* scores.



Ranking Criteria poses some questions:

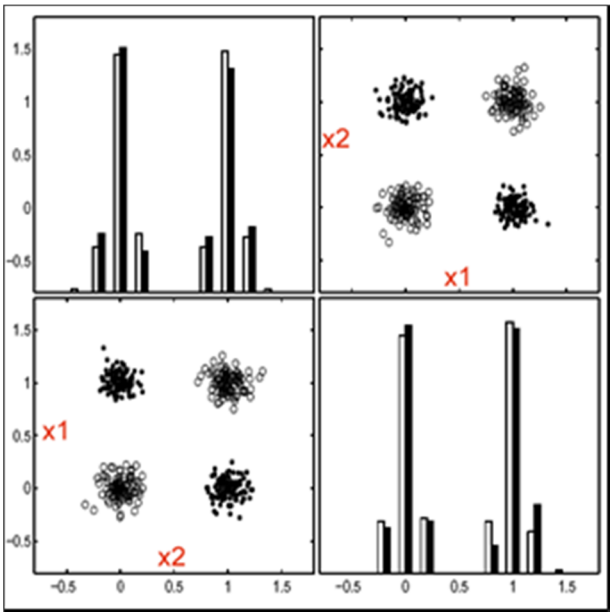


Can variables with small score be automatically discarded? NO!

1. Can variables with small score be automatically discarded? The answer is **NO!**

- Even variables with small score can improve class seperability
- Here, this depends on the correlation between *x1* and *x2*

Here, the class conditional distributions have a high co-variance in the direction orthogonal to the line between the two class centers.



Can a useless variable (i.e. one with a small score) be useful together with others? YES!

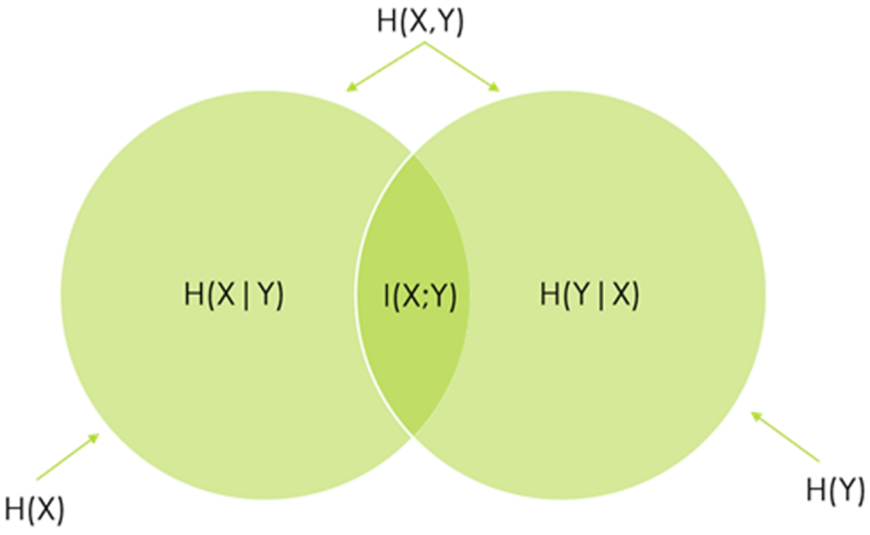
2. Can a useless variable (i.e. one with a small score) be useful together with others?

The answer is **YES!**

• The correlation between variables and target are not enough to assess relevance

• The correlation / co-variance between pairs of variables has to be considered too (potentially difficult)

- Also, the **diversity** of features needs to be considered.



Information Theoretic Criteria

3. ***Can two variables that are useless by themselves can be useful together?***

The answer is **YES!**

This can be done using the Information Theoretic Criteria.

Information Theoretic Criteria

- Mutual information can also detect non-linear dependencies among variables
- But, it is harder to estimate than correlation
- It is a measure for "how much information (in terms of entropy) two random variables share".

Variable Ranking — Single Variable Classifiers

- Idea: Select variables according to their **individual predictive power**
- Criterion: Performance of a classifier built with 1 variable e.g. the value of the variable itself
- The Predictive power is usually measured in terms of error rate (or criteria using False Positive Rate, False Negative Rate)
- Also, a combination of SVC's can be deployed using ensemble methods (boosting,...).

22. Explain exhaustive feature selection strategy.

In an exhaustive feature selection the best subset of features is selected, over all possible feature subsets, by optimizing a specified performance metric for a certain machine learning algorithm. For example, if the classifier is a logistic regression and the dataset consists of 4 features, the algorithm will evaluate all 15 feature combinations as follows:

- all possible combinations of 1 feature
- all possible combinations of 2 features
- all possible combinations of 3 features
- all the 4 features

and select the one that results in the best performance (e.g., classification accuracy) of the logistic regression classifier.

- This is another greedy algorithm as it evaluates all possible feature combinations. It is quite computationally expensive, and sometimes, if feature space is big, even unfeasible.
- There is a special package for python that implements this type of feature selection: mlxtend.
- In the mlxtend implementation of the exhaustive feature selection, the stopping criterion is an arbitrarily set number of features. So the search will finish when we reach the desired number of selected features.
- This is somewhat arbitrary because we may be selecting a subopimal number of features, or likewise, a high number of features.

23. Explain greedy forward and backward feature selection strategy.

- Forward selection is an iterative method in which we start with having no feature in the model. In each iteration, we keep adding the feature which best improves our model till an addition of a new variable does not improve the performance of the model.
- The procedure starts with an empty set of features [reduced set]. The best of the original features is determined and added to the reduced set. At each subsequent iteration, the best of the remaining original attributes is added to the set.
- Step forward feature selection starts by evaluating all features individually and selects the one that generates the best performing algorithm, according to a pre-set evaluation criteria. In the second step, it evaluates all possible combinations of the selected feature and a second feature, and selects the pair that produce the best performing algorithm based on the same pre-set criteria.
- In backward elimination, we start with all the features and removes the least significant feature at each iteration which improves the performance of the model. We repeat this until no improvement is observed on removal of features.
- The procedure starts with the full set of attributes. At each step, it removes the worst attribute remaining in the set.

24. Explain best first feature selection strategy.

Best-first search algorithm always selects the path which appears best at that moment. It is the combination of depth-first search and breadth-first search algorithms. It uses the heuristic function and search. Best-first search allows us to take the advantages of both algorithms. With the help of best-first search, at each step, we can choose the most promising node. In the best first search algorithm, we expand the node which is closest to the goal node and the closest cost is estimated by heuristic function, i.e. f(n)= g(n). Where, h(n)= estimated cost from node n to the goal. The greedy best first algorithm is implemented by the priority queue.

Best first search algorithm:

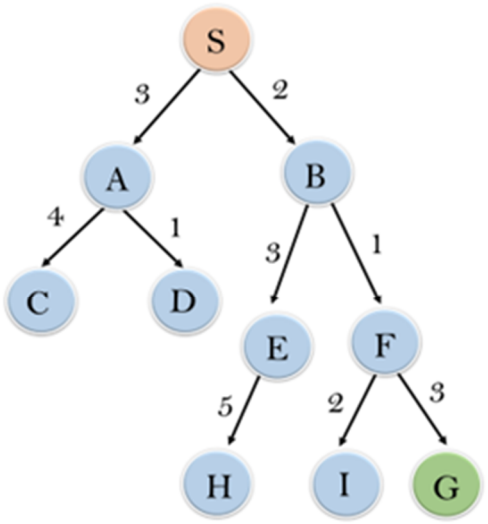
- **Step 1:** Place the starting node into the OPEN list.
- **Step 2:** If the OPEN list is empty, Stop and return failure.
- **Step 3:** Remove the node n, from the OPEN list which has the lowest value of h(n), and places it in the CLOSED list.
- **Step 4:** Expand the node n, and generate the successors of node n.
- **Step 5:** Check each successor of node n, and find whether any node is a goal node or not. If any successor node is goal node, then return success and terminate the search, else proceed to Step 6.

- **Step 6:** For each successor node, algorithm checks for evaluation function f(n), and then check if the node has been in either OPEN or CLOSED list. If the node has not been in both list, then add it to the OPEN list.
- **Step 7:** Return to Step 2.

Advantages: Best first search can switch between BFS and DFS by gaining the advantages of both the algorithms. This algorithm is more efficient than BFS and DFS algorithms.

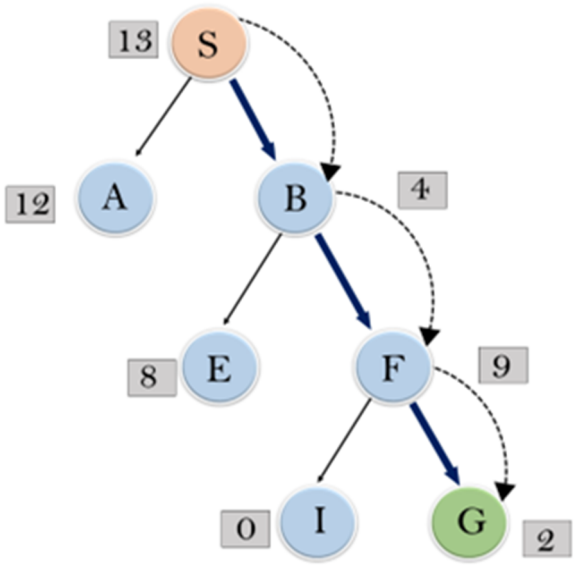
Disadvantages: It can behave as an unguided depth-first search in the worst case scenario. It can get stuck in a loop as DFS. This algorithm is not optimal.

Example: Consider the below search problem, and we will traverse it using greedy best-first search. At each iteration, each node is expanded using evaluation function f(n)=h(n) , which is given in the below table.



node	H (n)
A	12
B	4
C	7
D	3
E	8
F	2
H	4
I	9
S	13
G	0

In this search example, we are using two lists which are **OPEN** and **CLOSED** Lists. Following are the iteration for traversing the above example.



Expand the nodes of S and put in the CLOSED list

Initialization: Open [A, B], Closed [S]

Iteration 1: Open [A], Closed [S, B]

Iteration 2: Open [E, F, A], Closed [S, B]: Open [E, A], Closed [S, B, F]

Iteration 3: Open [I, G, E, A], Closed [S, B, F]: Open [I, E, A], Closed [S, B, F, G]

Hence the final solution path will be: **S----> B----->F----> G**

Time Complexity: The worst case time complexity of Greedy best first search is $O(b^m)$.

Space Complexity: The worst case space complexity of Greedy best first search is $O(b^m)$. Where, m is the maximum depth of the search space.

Complete: Greedy best-first search is also incomplete, even if the given state space is finite.

Optimal: Greedy best first search algorithm is not optimal.

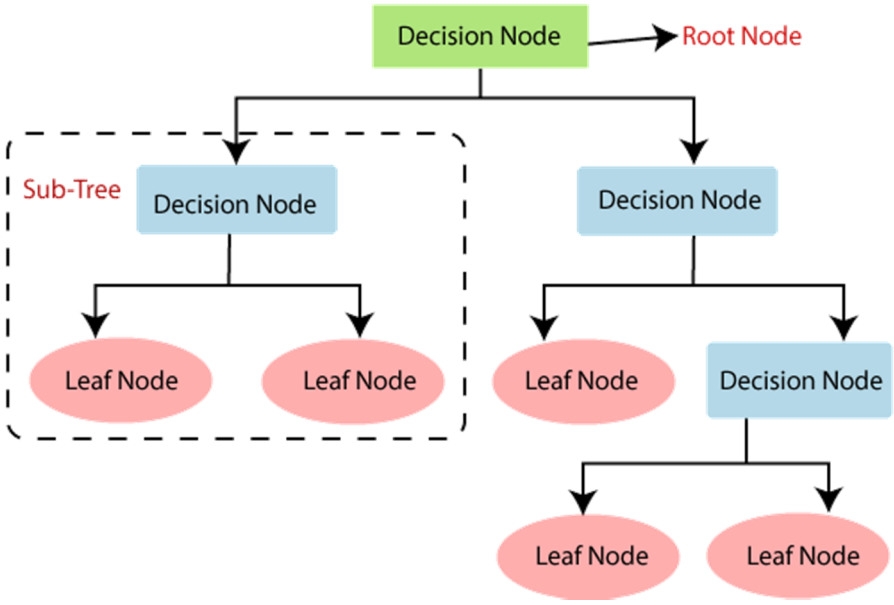
25. Compare filter, wrapper and embedded methods.

Filter methods	Wrapper methods	Embedded methods
Generic set of methods which do not incorporate a **specific machine learning algorithm.**	Evaluates on a **specific machine learning algorithm** to find optimal features.	Embeds (fix) features during **model building process.** Feature selection is done by observing each iteration of model training phase.
Much **faster** compared to Wrapper methods in terms of time complexity	**High computation time** for a dataset with many features	Sits **between Filter methods and Wrapper methods** in terms of time complexity
Less prone to **over-fitting**	High chances of **over-fitting** because it involves training of machine learning models with different combination of features	Generally used to reduce **over-fitting** by **penalizing** the coefficients of a model being too large.
Examples – **Correlation, Chi-Square test, ANOVA, Information gain** etc.	Examples - **Forward Selection, Backward elimination, Stepwise selection** etc.	Examples - **LASSO, Elastic Net, Ridge Regression** etc.

26. Why use decision tree?

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems.
- It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

- The decisions or the test are performed on the basis of features of the given dataset.
- **It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.**
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into sub-trees. Below diagram explains the general structure of a decision tree.



- It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

27. Explain decision tree terminology.

The decision tree comprises of root node, leaf node, branch nodes, parent/child node etc. following is the explanation of this terminology.

- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.

- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

28. Explain use of decision tree for feature selection.

Feature selection techniques are majorly inspired from **decision tree.** The easiest one to understand and probably the most straight forward one is obtaining a **feature ranking** based on the sum of the improvements in all nodes in which the attribute appears as a *splitter* (weighted by the fraction of the training data in each node split). The *surrogate splitters* (used in CART implementation as a way to deal with missing values in the data) are also considered while calculating variable importance which means that even a variable that never splits a node may be assigned a large importance score. This allows the variable importance rankings to reveal variable masking and nonlinear correlation among the attributes. Importance scores may optionally be confined to splitters; comparing the splitters-only and the full (splitters and surrogates) importance rankings is a useful diagnostic.

It is clear that this kind of variable ranking criteria is heavily influenced by the splitting criteria used and is prone to giving different variable rankings in the presence of certain highly correlated features, when run multiple times.

Another fairly intuitive and somewhat naive approach followed for feature selection exploits the greedy nature of the *tree growing algorithm (C4.5)*. As the best splits are performed early while growing the decision tree, only the features which appear till depth d (usually set to 3) in the decision tree constructed by using a small bootstrapped subset (~10%) of the data are considered important.

This is repeated k number of times using a freshly bootstrapped sample each time and it is hypothesized that if a feature in the deeper levels on any one execution of C4.5 is relevant enough, it will finally rise up and appear in one of the top levels of the tree in some other executions of C4.5. We form a union of all the attributes from each run and call this set as the set of selected features.

Another method is to use the **Separability of Split Value** (SSV) criterion for feature selection. The algorithm for feature selection from a single SSV tree works as follows:

1. T ← the SSV decision tree built for X, Y.
2. For each non-leaf node N of T , G(N) ← the classification error reduction of node N.
3. F ← the set of all the features of the input space.
4. i ← 0

5. While F = ∅ do:

(a) For each feature f ∈ F not used by T define its rank R(f) ← i. Remove these features from F.

(b) Prune T by deleting all the final splits of nodes N for which G(N) is minimal.

(c) Prune T by deleting all the final splits of nodes N for which G(N) = 0.

(d) i ← i + 1

6. Return the list of features in decreasing order of R(f).

As the decision tree building algorithm selects the splits locally, i.e. with respect to the splits selected in earlier stages, so that the features occurring in the decision tree, are complementary. In some cases, the full classification decision trees use only a small part of the features. It does not allow to select any number of features – the maximum is the number of features used by the tree, because the algorithm gives no information about the ranking of the rest of the features. Also, certain good masked features suffer due to this type of strict ranking criteria.

As we move on to some slightly more complex techniques, the ensembles of trees play an important role. Another fairly intuitive algorithm for obtaining variable importance using Random forests first grows a complete forest using a bootstrapped sample containing on an average 63% of the unique data. Then, for evaluating the variable importance, the OOB (Out Of Box) data is run down each decision tree of the ensemble, making a random assignment to one of the child nodes if the split is on the variable under consideration.

The corresponding prediction error of this noised up ensemble is noted which gives us the increase in the prediction error by “noising up” the variable v. This can be used to find out the relative importance of each feature. One caveat, however is that this sort of variable importance is dominated by the majority class. Also, the importance of highly correlated variables is overestimated by this technique of variable importance.

Mathematics based questions

29. Explain entropy reduction, information gain and Gini index in decision tree.

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.** There are two popular techniques for ASM, which are:

Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute. It calculates how much information a feature provides us about a class.

- According to the value of information gain, we split the node and build the decision tree. A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula: **Information Gain= Entropy(S) – [(Weighted Average) * Entropy (each feature)]**

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as: **Entropy(s) = – P(yes)log2 P(yes) – P(no) log2 P(no)** where, S= Total number of samples, P(yes)= probability of yes, P(no)= probability of no

Gini Index:

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART (Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index. Gini index can be calculated using the formula: **Gini Index= 1 – $\sum_j P_j^2$**
- It only creates binary splits, and the CART uses Gini index to create binary splits.

Problems/Numerical

30. Problems on calculating entropy and information gain

Problem 1:

If we decided to arbitrarily label all 4 gumballs as red, how often would one of the gumballs is incorrectly labelled?

4 red and 0 blue:

$$Gini\ Index = 1\text{-}(probability\_red^2 + probability\_blue^2) = 1\text{-}(1^2+0^2) = 0$$

The impurity measurement is 0 because we would never incorrectly label any of the 4 red gumballs here. If we arbitrarily chose to label all the balls 'blue', then our index would still be 0, because we would always incorrectly label the gumballs. The gini score is always the same no matter what arbitrary class you take the probabilities of because they always add to 0 in the formula above. A gini score of 0 is the most pure score possible.

2 red and 2 blue:

$$Gini\ Index = 1\text{-}(probability\_red^2 + probability\_blue^2) = 1\text{-}(0.5^2+0.5^2) = 0.5$$

The impurity measurement is 0.5 because we would incorrectly label gumballs wrong about half the time. Because this index is used in binary target variables (0,1), a gini index of 0.5 is the least pure score possible. Half is one type and half is the other. **Dividing gini scores by 0.5 can help intuitively understand what the score represents. 0.5/0.5 = 1, meaning the grouping is as impure as possible (in a group with just 2 outcomes).**

3 red and 1 blue:

$$Gini\ Index = 1-(probability\_red^2+probability\_blue^2)=1-(0.75^2+0.25^2)=0.375$$

The impurity measurement here is 0.375. If we divide this by 0.5 for more intuitive understanding we will get 0.75, which is the probability of incorrectly/correctly labeling.

Problem 2:

How does entropy work with the same gumball scenarios stated in problem 1?

4 red and 0 blue:

$$Entropy = [(probability\_red) * log2(probability\_red)]—[(probability\_blue) * log2(probability\ of\ blue)] = [(4/4) * log2(4/4)] -[(0/4) * log2(0/4)] = 0$$

Unsurprisingly, the impurity measurement is 0 for entropy as well. This is the max purity score using information entropy.

2 red and 2 blue:

$$Entropy = [(probability\_red) * log2(probability\_red)]—[(probability\_blue) * log2(probability\ of\ blue)] = [(2/4) * log2(2/4)] -[(2/4) * log2(2/4)] = 1$$

The impurity measurement is 1 here, as it's the maximum impurity obtainable.

3 red and 1 blue:

$$Entropy = [(probability\_red) * log2(probability\_red)]—[(probability\_blue) * log2(probability\ of\ blue)] = [(3/4) * log2(3/4)] -[(1/4) * log2(1/4)] = 0.811$$

The purity/impurity measurement is 0.811 here, a bit worse than the gini score.

Problem 3:

Calculate entropy for following example. For the set X = {a,a,a,b,b,b,b,b}, Total instances: 8, Instances of b: 5, Instances of a: 3

$$EntropyH(X) = -\left[\left(\frac{3}{8}\right) log_2\frac{3}{8} + \left(\frac{5}{8}\right) log_2\frac{5}{8}\right]$$

= - [0.375 * (-1.415) + 0.625 * (-0.678)]

= 0.954

Problem 4:

In the below mini-dataset, the label we're trying to predict is the type of fruit. This is based off the size, color, and shape variables.

Fruit	Size	Color	Shape
Watermelon	Big	Green	Round
Apple	Medium	Red	Round
Banana	Medium	Yellow	Thin
Grape	Small	Green	Round
Grapefruit	Medium	Yellow	Round
Lemon	Small	Yellow	Round

Calculate the information gained if we select the *color* variable.

3 out of the 6 records are yellow, 2 are green, and 1 is red. Proportionally, the probability of a yellow fruit is 3 / 6 = 0.5; 2 / 6 = 0.333 for green, and 1 / 6 = 0.1666 for red. Using the formula from above, we can calculate it like this:

Information gain = - ([3/6 * $\log_2$(3/6)] + [2/6 * $\log_2$(2/6)] + [1/6 * $\log_2$(1/6)]) **= 1.459148**

Calculate the information gained if we select the *size* variable.

Information gain = - ([3/6 * $\log_2$(3/6)] + [2/6 * $\log_2$(2/6)] + [1/6 * $\log_2$(1/6)]) **= 1.459148**

In this case, 3 / 6 of the fruits are medium-sized, 2 / 6 are small, 1 / 6 is big.

Calculate the information gained if we select the *shape* variable.

Here, 5 / 6 of the fruits are round and 1 / 6 is thin.

Information gain = - ([5/6 * $\log_2$(5/6)] + [1/6 * $\log_2$(1/6)]) **= 0.650022**

Problem 5:

Consider the training examples shown in Table below for a binary classification problem.

Customer ID	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

(a) Compute the Gini index for the overall collection of training examples.

Answer:

Gini $= 1 - 2 \times 0.5^2 = 0.5$.

(b) Compute the Gini index for the `Customer ID` attribute.

Answer:

The gini for each `Customer ID` value is 0. Therefore, the overall gini for `Customer ID` is 0.

(c) Compute the Gini index for the `Gender` attribute.

Answer:

The gini for `Male` is $1 - 2 \times 0.5^2 = 0.5$. The gini for `Female` is also 0.5. Therefore, the overall gini for `Gender` is $0.5 \times 0.5 + 0.5 \times 0.5 = 0.5$.

(d) Compute the Gini index for the `Car Type` attribute using multiway split.

Answer:

The gini for `Family` car is 0.375, `Sports` car is 0, and `Luxury` car is 0.2188. The overall gini is 0.1625.

(e) Compute the Gini index for the `Shirt Size` attribute using multiway split.

Answer:

The gini for `Small` shirt size is 0.48, `Medium` shirt size is 0.4898, `Large` shirt size is 0.5, and `Extra Large` shirt size is 0.5. The overall gini for `Shirt Size` attribute is 0.4914.

(f) Which attribute is better, `Gender`, `Car Type`, or `Shirt Size`?

Answer:

`Car Type` because it has the lowest gini among the three attributes.

(g) Explain why `Customer ID` should not be used as the attribute test condition even though it has the lowest Gini.

Answer:

The attribute has no predictive power since new customers are assigned to new `Customer IDs`.

Problem 6:

Consider the following data set for a binary class problem.

A	B	Class Label
T	F	+
T	T	+
T	T	+
T	F	−
T	T	+
F	F	−
F	F	−
F	F	−
T	T	−
T	F	−

(a) Calculate the information gain when splitting on A and B. Which attribute would the decision tree induction algorithm choose?

Answer:

The contingency tables after splitting on attributes A and B are:

	$A=T$	$A=F$
+	4	0
−	3	3

	$B=T$	$B=F$
+	3	1
−	1	5

The overall entropy before splitting is:

$$E_{orig} = -0.4 \log 0.4 - 0.6 \log 0.6 = 0.9710$$

The information gain after splitting on A is:

$$\begin{aligned} E_{A=T} &= -\frac{4}{7}\log\frac{4}{7} - \frac{3}{7}\log\frac{3}{7} = 0.9852 \\ E_{A=F} &= -\frac{3}{3}\log\frac{3}{3} - \frac{0}{3}\log\frac{0}{3} = 0 \\ \Delta &= E_{orig} - 7/10 E_{A=T} - 3/10 E_{A=F} = 0.2813 \end{aligned}$$

The information gain after splitting on B is:

$$\begin{aligned} E_{B=T} &= -\frac{3}{4}\log\frac{3}{4} - \frac{1}{4}\log\frac{1}{4} = 0.8113 \\ E_{B=F} &= -\frac{1}{6}\log\frac{1}{6} - \frac{5}{6}\log\frac{5}{6} = 0.6500 \\ \Delta &= E_{orig} - 4/10E_{B=T} - 6/10E_{B=F} = 0.2565 \end{aligned}$$

Therefore, attribute A will be chosen to split the node.

(b) Calculate the gain in the Gini index when splitting on A and B. Which attribute would the decision tree induction algorithm choose?

Answer:

The overall gini before splitting is:

$$G_{orig} = 1 - 0.4^2 - 0.6^2 = 0.48$$

The gain in gini after splitting on A is:

$$\begin{aligned} G_{A=T} &= 1 - \left(\frac{4}{7}\right)^2 - \left(\frac{3}{7}\right)^2 = 0.4898 \\ G_{A=F} &= 1 = \left(\frac{3}{3}\right)^2 - \left(\frac{0}{3}\right)^2 = 0 \\ \Delta &= G_{orig} - 7/10G_{A=T} - 3/10G_{A=F} = 0.1371 \end{aligned}$$

The gain in gini after splitting on B is:

$$\begin{aligned} G_{B=T} &= 1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2 = 0.3750 \\ G_{B=F} &= 1 = \left(\frac{1}{6}\right)^2 - \left(\frac{5}{6}\right)^2 = 0.2778 \\ \Delta &= G_{orig} - 4/10G_{B=T} - 6/10G_{B=F} = 0.1633 \end{aligned}$$

Therefore, attribute B will be chosen to split the node.

Problem 7:

Consider the training examples shown in Table below for a binary classification problem.

Instance	a_1	a_2	a_3	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	−
4	F	F	4.0	+
5	F	T	7.0	−
6	F	T	3.0	−
7	F	F	8.0	−
8	T	F	7.0	+
9	F	T	5.0	−

(a) What is the entropy of this collection of training examples with respect to the positive class?

Answer:

There are four positive examples and five negative examples. Thus, $P(+) = 4/9$ and $P(-) = 5/9$. The entropy of the training examples is $-4/9 \log_2(4/9) - 5/9 \log_2(5/9) = 0.9911$.

(b) What are the information gains of a_1 and a_2 relative to these training examples?

Answer:

For attribute a_1, the corresponding counts and probabilities are:

a_1	+	-
T	3	1
F	1	4

The entropy for a_1 is

$$\frac{4}{9}\left[-(3/4)\log_2(3/4) - (1/4)\log_2(1/4)\right] + \frac{5}{9}\left[-(1/5)\log_2(1/5) - (4/5)\log_2(4/5)\right] = 0.7616.$$

Therefore, the information gain for a_1 is $0.9911 - 0.7616 = 0.2294$.

For attribute a_2, the corresponding counts and probabilities are:

a_2	+	-
T	2	3
F	2	2

The entropy for a_2 is

$$\frac{5}{9}\left[-(2/5)\log_2(2/5) - (3/5)\log_2(3/5)\right] + \frac{4}{9}\left[-(2/4)\log_2(2/4) - (2/4)\log_2(2/4)\right] = 0.9839.$$

Therefore, the information gain for a_2 is $0.9911 - 0.9839 = 0.0072$.

(c) For a_3, which is a continuous attribute, compute the information gain for every possible split.

Answer:

a_3	Class label	Split point	Entropy	Info Gain
1.0	+	2.0	0.8484	0.1427
3.0	-	3.5	0.9885	0.0026
4.0	+	4.5	0.9183	0.0728
5.0 5.0	- -	5.5	0.9839	0.0072
6.0	+	6.5	0.9728	0.0183
7.0 7.0	+ -	7.5	0.8889	0.1022

The best split for a_3 occurs at split point equals to 2.

(d) What is the best split (among a_1, a_2, and a_3) according to the information gain?

Answer:

According to information gain, a_1 produces the best split.

(e) What is the best split (between a_1 and a_2) according to the classification error rate?

Answer:

For attribute a_1: error rate $= 2/9$.

For attribute a_2: error rate $= 4/9$.

Therefore, according to error rate, a_1 produces the best split.

(f) What is the best split (between a_1 and a_2) according to the Gini index?

Answer:

For attribute a_1, the gini index is

$$\frac{4}{9}\left[1-(3/4)^2-(1/4)^2\right]+\frac{5}{9}\left[1-(1/5)^2-(4/5)^2\right]=0.3444.$$

For attribute a_2, the gini index is

$$\frac{5}{9}\left[1-(2/5)^2-(3/5)^2\right]+\frac{4}{9}\left[1-(2/4)^2-(2/4)^2\right]=0.4889.$$

Since the gini index for a_1 is smaller, it produces the better split.

Feel free to contact me on +91-8329347107 calling / +91-9922369797 whatsapp, email ID: adp.mech@coep.ac.in and abhipatange93@gmail.com

****