2.1 CONCEPT OF FEATURE

GQ. Define Feature Engineering. Explain the four processes in feature engineering.

- Feature engineering is the pre-processing step of machine learning, which is used to transform raw data into features that can be used for creating a predictive model using Machine learning or statistical Modelling.
- Feature engineering is the pre-processing step of machine learning, which extracts features from raw data.
- It helps to represent an underlying problem to predictive models in a better way, which as a result, improve the accuracy of the model for unseen data.
- The predictive model contains predictor variables and an outcome variable, and while the feature engineering process selects the most useful predictor variables for the model.
- Generally, all machine learning algorithms take input data to generate the output. The input data remains in a tabular form consisting of rows (instances or observations) and columns (variable or attributes), and these attributes are often known as **features**.
- For example, an image is an instance in computer vision, but a line in the image could be the feature. Similarly, in NLP, a document can be an observation, and the word count could be the feature.
- So, we can say a feature is an attribute that impacts a problem or is useful for the problem.
- Feature engineering in ML contains mainly four processes: Feature Creation, Transformations, Feature Extraction, and Feature Selection.
- These processes are described as below :

(1) **Feature Creation :** Feature creation is finding the most useful variables to be used in a predictive model. The process is subjective, and it requires human creativity and intervention. The new features are created by mixing existing features using addition, subtraction, and ration, and these new features have great flexibility.

(2) **Transformations :** The transformation step of feature engineering involves adjusting the predictor variable to improve the accuracy and performance of the model. For example, it ensures that the model is flexible to take input of the variety of data; it ensures that all the variables are on the same scale, making the model easier to understand. It improves the model's accuracy and ensures that all the features are within the acceptable range to avoid any computational error.

(3) **Feature Extraction :** Feature extraction is an automated feature engineering process that generates new variables by extracting them from the raw data. The main aim of this step is to reduce the volume of data so that it can be easily used and managed for data modelling. Feature extraction methods include cluster analysis, text analytics, edge detection algorithms, and principal components analysis (PCA).

(4) **Feature Selection :** While developing the machine learning model, only a few variables in the dataset are useful for building the model, and the rest features are either redundant or irrelevant. If we input the dataset with all these redundant and irrelevant features, it may negatively impact and reduce the overall performance and accuracy of the model. Hence it is very important to identify and select the most appropriate features from the data and remove the irrelevant or less important features, which is done with the help of feature selection in machine learning. "Feature selection is a way of selecting the subset of the most relevant features from the original features set by removing the redundant, irrelevant, or noisy features."

2.2 PREPROCESSING OF DATA

GQ. Define data preprocessing. Explain the steps involved in data preprocessing.

- Data preprocessing is the process of preparing raw data to be used on a machine learning model. It is the first and most important step in developing a machine learning model.
- When developing a machine learning project, we do not always come across clean and formatted data. And, before performing any operation on data, it must be cleaned and formatted. As a result, we use the data preprocessing task for this.
- Real-world data typically contains noise, missing values, and may be in an unusable format that cannot be used directly for machine learning models.
- Data preprocessing is a necessary task for cleaning the data and preparing it for a machine learning model, which improves the accuracy and efficiency of the machine learning model.
- It involves below steps :

(1) Get the Dataset
(2) Importing Libraries
(3) Importing the Datasets
(4) Handling Missing Data
(5) Encoding the Categorical Data
(6) Splitting the Dataset into the Training set and Test set
(7) Feature Scaling

(1) Get the Dataset

- The first thing we need to create a machine learning model is a dataset, because a machine learning model is entirely dependent on data. The dataset is the collection of data for a specific problem in the proper format.
- Datasets can be of various formats for various purposes. For example, if we want to create a machine learning model for business purposes, the dataset will be different from the dataset required for a liver patient. As a result, each dataset is distinct from the others. We usually save the dataset as a CSV file before using it in our code. However, there may be times when we need to use an HTML or xlsx file.

(2) Importing Libraries

- To perform data preprocessing with Python, we must first import some predefined Python libraries.
- These libraries are used to carry out specific tasks. For data preprocessing, we will use three specific libraries, which are: Numpy, Matplotlib and Pandas.

(3) Importing the Datasets

- We must now import the datasets that we have gathered for our machine learning project. However, before importing a dataset, we must make the current directory a working directory.
- To import the dataset, we will use the pandas library's read_csv() function, which reads a csv file and performs various operations on it. We can use this function to read a csv file both locally and via a URL.
- It is essential in machine learning to distinguish the feature matrix (independent variables) from the dataset.

(4) Handling Missing Data

- The next step in data preprocessing is to deal with missing data in the datasets. If our dataset contains some missing data, it may pose a significant challenge to our machine learning model.
- As a result, handling missing values in the dataset is required.
- There are primarily two approaches to dealing with missing data :
  - (i) **By removing the specific row** : The first method is commonly used to deal with null values. In this manner, we simply delete the specific row or column that contains null values. However, this method is inefficient, and removing data may result in information loss, resulting in an inaccurate output.
  - (ii) **By calculating the mean :** In this way, we will calculate the mean of that column or row which contains any missing value and will put it on the place of missing value. This strategy is useful for the features which have numeric data such as age, salary, year, etc. Here, we will use this approach.

(5) Encoding the Categorical Data

- Categorical data is data which has some categories such as Country. Because machine learning models are entirely based on mathematics and numbers, having a categorical variable in our dataset may cause problems when building the model.
- As a result, these categorical variables must be encoded into numbers. We can use OneHotEncoding or Label Encoding technique.

(6) Splitting the Dataset into the Training set and Test set

- In machine learning data preprocessing, we divide our dataset into a training set and a test set.
- This is an important step in data preprocessing because it allows us to improve the performance of our machine learning model.

- Assume we trained our machine learning model on one dataset and then tested it on another. It will then be difficult for our model to understand the correlations between the models.
- **Training Set :** A subset of dataset to train the machine learning model, and we already know the output.
- **Test set :** A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

▶ **(7) Feature Scaling**

- Feature scaling is the final step in machine learning data preprocessing. It is a method for standardizing the independent variables of a dataset within a given range.
- In feature scaling, we place our variables in the same range and scale so that no one variable dominates the other.

⏭ 2.3 NORMALIZATION AND SCALING

GQ. Explain the concept of scaling and normalization with its types.

- Scaling and normalization are so similar that they're often applied interchangeably, but they have different effects on the data.
- In both scaling and normalization, we are transforming the values of numeric variables so that the transformed data points have specific helpful properties. These properties can be exploited to create better features and models.
- In Scaling, we're changing the **range** of the distribution of the data, while in normalization, we're changing the **shape** of the distribution of the data.
- In scaling, we're transforming the data so that it fits within a specific scale, like 0-100 or 0-1.
- Usually, 0-1. You want to scale data especially when you're using methods based on measures of how far apart data points are.
- Normalization is a more radical transformation. The point of normalization is to change your observations so that they can be described as a normal distribution.

2.3.1 Types of Scaling

(1) Simple Feature Scaling : This method simply divides each value by the maximum value for that feature. The resultant values are in the range between zero(0) and one(1). Simple-feature scaling is the defacto scaling method used on image-data, when we scale images by dividing each image by 255 (maximum image pixel intensity). The formula for simple feature scaling is given as :

$$X_{new} = \frac{X_{old}}{X_{max}}$$

(2) Min-max Scaling : This scaler takes each value and subtracts the minimum and then divides by the range(max-min).The resultant values range between zero(0) and one(1). The formula for min-max scaling is given as :

$$X_{new} = \frac{X_{old} - X_{min}}{X_{max} - X_{min}}$$

2.3.2 Types of Normalization

(1) Z-Score or Standard Score

In this technique, values are normalized based on mean and standard deviation of the data A. The formula used is :

$$X_{new} = \frac{X_{old} - \mu_A}{\sigma_A}$$

σ_A, μ_A is the standard deviation and mean of A respectively.

Example : If mean salary is \$54,000 and standard deviation is \$16,000, then the z-score value of salary \$73,600 will be $\frac{73600 - 54000}{16000} = 1.225$

(2) Box-cox Transformation

- A Box-Cox transformation is a transformation of a non-normal dependent variable into a normal shape. The Box-Cox transformation is named after statisticians George Box and Sir David Roxbee Cox.
- At the heart of the box-cox normalization is an exponent *lambda* (λ), which varies from – 5 to 5. All values of λ are considered and the optimal value for your data is selected; The "*optimal value*" is the one which results in the best approximation of a normal distribution curve.

$$w_t = \begin{cases} \log(y_t); & \text{if } \lambda = 0 \\ \frac{(y_t^{\lambda} - 1)}{\lambda}; & \text{otherwise} \end{cases}$$

2.4 STANDARDIZATION

- Standardization is necessary when features of the input data set have wide ranges or are simply measured in different measurement units (such as pounds, metres, miles, etc.).
- For many machine learning models, these variations in the initial feature ranges are problematic. For models that compute distance, for instance, if one of the features has a wide range of values, the distance will be determined by this specific feature.
- **Example :** Let's say we have a 2-dimensional data set with the variables Height in Meters and Weight in Pounds, both of which have ranges of [1 to 2] Meters and [10 to 200] Pounds, respectively. No matter what distance-based model you run on this set of data, the Weight feature will take precedence over the Height feature and contribute more to the distance calculation simply because it contains larger values.

So, standardization is the way to avoid this issue by transforming features to comparable scales.

- Z-score is one of the most popular methods to standardize data, and can be done by subtracting the mean and dividing by the standard deviation for each value of each feature.

$$z = \frac{\text{value} - \text{mean}}{\text{standard deviation}}$$

- Once the standardization is done, all the features will have a mean of zero, a standard deviation of one, and thus, the same scale.

2.5 MANAGING MISSING VALUES

GQ. Explain how the missing values are handled in data preprocessing.

Imagine that you are asked to analyze a dataset. You find that there are many tuples having no recorded value for several attributes such as customer income. So, the question arising here is how to fill in the missing values for this attribute. There are several methods as discussed here.

(1) **Ignore the tuple :** When the class label is missing, this technique is used. However, unless the tuple contains numerous attributes with missing values, this approach is not particularly useful.

(2) **Fill in the missing value manually :** This approach is effective on small data set with some missing values.

(3) **Use a global constant to fill in the missing value :** You can replace all missing attribute values with global constant, such as a label like "Unknown" or $-\infty$.

(4) **Use a measure of central tendency for attribute (e.g. the mean or median) to fill in the missing value :** For example, suppose customer average income is $25000, then you can use this value to replace missing value for income.

(5) **Use the attribute mean or median for all samples belonging to the same class as the given tuple :** For example, if you are classifying customers according to their credit_score, then you can replace the missing value with the mean income value for customers in the same credit_score category as that of the given tuple. If the data distribution for a given class is skewed, then use the median value.

(6) **Use the most probable value to fill in the missing value :** This can be determined using regression, Bayesian classification or decision-tree induction.

2.6 INTRODUCTION TO DIMENSIONALITY REDUCTION

- The number of input variables or features for a dataset is referred to as its dimensionality.
- Dimensionality reduction refers to techniques that reduce the number of input variables in a dataset.
- More input features often make a predictive modeling task more challenging to model, more generally referred to as the curse of dimensionality.

- High-dimensionality statistics and dimensionality reduction techniques are often used for data visualization.
- Nevertheless, these techniques can be used in applied machine learning to simplify a classification or regression dataset in order to better fit a predictive model.
- There are a number of advantages that makes dimensionality reduction important:
  (1) The model accuracy is improved when there is less data.
  (2) When dealing with fewer dimensions, it requires a lot less computing power and also since the dater is lesser, the algorithm can train faster.
  (3) Lesser data requires lesser storage space.
  (4) Lesser dimensions can work with algorithms that cannot be used with larger dimensions.
  (5) Lesser features come with the benefit of noise and redundant variables.
- Dimensionality reduction has two main components :
  (1) **Feature selection :** This is the process where the universal set of features or variables is used to extract a subset that can be used to model the problem. Feature selection is done as Filter or Wrapper or Embedded.
  (2) **Feature extraction :** This is used to reduce data that is in a higher-dimensional space to a lower-dimensional space. For example as to how features in 3 dimensions can be reduced to two dimensions for simplicity.
- Some of the dimension reduction techniques include :
  (1) **Principal Component Analysis (PCA) :** This method is commonly used with continuous data. It works under the condition that the variance in mapped data in the lower dimensional space needs to be at the peak when the data is mapped from a higher-dimensional space. I other words it projects data where variance increases and the features with the most variance become the principal components.
  (2) **Linear Discriminant Analysis (LDA) :** This project's data in such a way that the separability of the class is maximized. Points from the same class are projected closely together while those from different classes are spaced far apart.
  (3) **Generalized Discriminant Analysis (GDA) :** The GDA is quite an effective approach when it comes to extracting non-linear features.

2.7 PRINCIPAL COMPONENT ANALYSIS (PCA)

GQ. Explain how PCA helps in dimensionality reduction.

- Principal Component Analysis (PCA) 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.
- 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.

Steps in PCA

(1) **Standardize the dataset :** First, we need to standardize the dataset and for that, we need to calculate the mean and standard deviation for each feature. We use Z-score method to standardize the dataset.

(2) **Calculate the covariance matrix for the whole dataset :** The covariance matrix for the given dataset will be calculated as below

For population

$$\text{Cov}(x, y) = \frac{\sum (x_i - \bar{x}) * (y_i - \bar{y})}{N}$$

For sample

$$\text{Cov}(x, y) = \frac{\sum (x_i - \bar{x}) * (y_i - \bar{y})}{(N - 1)}$$

Since we standardize the dataset, so the mean for each feature is 0 and the standard deviation is 1.

For example, for a 3-dimensional data set with 3 variables x, y, and z, the covariance matrix is a 3x3 matrix of this from :

$$\begin{bmatrix} \text{Cov}(x, x) & \text{Cov}(x, y) & \text{Cov}(x, z) \\ \text{Cov}(y, x) & \text{Cov}(y, y) & \text{Cov}(y, z) \\ \text{Cov}(z, x) & \text{Cov}(z, y) & \text{Cov}(z, z) \end{bmatrix}$$

(3) **Calculating Eigen values and Eigen vectors**

An eigenvector is a nonzero vector that changes at most by a scalar factor when that linear transformation is applied to it. The corresponding eigenvalue is the factor by which the eigenvector is scaled.

Let A be a square matrix (in our case the covariance matrix), v a vector and λ a scalar that satisfies $Av = \lambda v$, then λ is called eigenvalue associated with eigenvector v of A.

Rearranging the above equation, $Av - \lambda v = 0$; $(A - \lambda I)\, v = 0$

Eigen vectors can be obtained by solving the $(A - \lambda I)\, v = 0$ equation for v vector with different λ values.

(4) **Sort the eigenvectors from the highest eigenvalue to the lowest.** The eigenvector with the highest eigenvalue is the first principal component. Higher eigenvalues correspond to greater amounts of shared variance.

(5) **Select the number of principal components.** Select the top N eigenvectors (based on their eigenvalues) to become the N principal components. The optimal number of principal components is both subjective and problem-dependent. Usually, we look at the cumulative amount of shared variance explained by the combination of principal components and pick that number of components, which still significantly explains the shared variance.

(6) **Transform the original matrix :** Feature matrix * top k eigenvectors = Transformed Data

☞ **Advantages of PCA**

(1) **Easy to compute :** PCA is based on linear algebra, which is computationally easy to solve by computers.

(2) **Speeds up other machine learning algorithms :** Machine learning algorithms converge faster when trained on principal components instead of the original dataset.

(3) **Counteracts the issues of high-dimensional data :** High-dimensional data causes regression-based algorithms to overfit easily. By using PCA beforehand to lower the dimensions of the training dataset, we prevent the predictive algorithms from overfitting.

☞ **Disadvantages of PCA**

(1) **Low interpretability of principal components :** Principal components are linear combinations of the features from the original data, but they are not as easy to interpret. For example, it is difficult to tell which are the most important features in the dataset after computing principal components.

(2) **The trade-off between information loss and dimensionality reduction :** Although dimensionality reduction is useful, it comes at a cost. Information loss is a necessary part of PCA. Balancing the trade-off between dimensionality reduction and information loss is unfortunately a necessary compromise that we have to make when using PCA.

2.8 FEATURE EXTRACTION

GQ. Explain how kernel PCA and Local Binary Pattern helps in dimensionality reduction.

- Feature extraction is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing.
- A characteristic of these large data sets is a large number of variables that require a lot of computing resources to process.
- Feature extraction is the name for methods that select and /or combine variables into features, effectively reducing the amount of data that must be processed, while still accurately and completely describing the original data set.
- The feature extraction technique gives us new features which are a linear combination of the existing features. The new set of features will have different values as compared to the original feature values.
- **The main aim is that fewer features will be required to capture the same information.**

- We might think that choosing fewer features might lead to underfitting but in the case of the Feature Extraction technique, the extra data is generally noise.

2.8.1 Kernel PCA

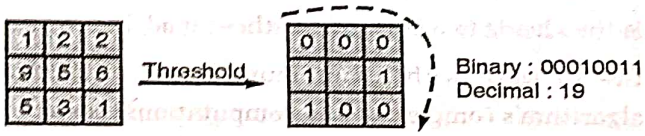
- Kernel PCA was developed in an effort to help with the classification of data whose decision boundaries are described by non-linear function.
- The idea is to go to a higher dimension space in which the decision boundary becomes linear.
- Here's an easy argument to understand the process. Suppose the decision boundary is described by a third order polynomial $= a + bx + cx^2 + dx^3$. Now, plotting this function in the usual $x - y$ plane will produce a wavy line, something similar to the made-up decision boundary in the right-hand-side picture above.
- Suppose instead we go to a higher dimensionality space in which the axes are x, x^2, x^3 and y. In this 4D space the third order polynomial becomes a linear function and the decision boundary becomes a hyperplane.
- So, the trick is to find a suitable transformation (up-scaling) of the dimensions to try and recover the linearity of the boundary. In this way the usual PCA decomposition is again suitable.
- This is all good but, as always, there's a catch. A generic non-linear combination of the original variables will have a huge number of new variables which rapidly blows up the computational complexity of the problem.
- However, we won't know the exact combination of non-linear terms we need, hence the large number of combinations that are in principle is required.
- Let's try and explain this issue with another simple example. Suppose we have only two wavelengths, call them λ_1 and λ_2. Now suppose we want to take a generic combination up to the second order of these two variables. The new variable set will then contain the following: $[\lambda_1,\lambda_2,\lambda_1\lambda_2,\lambda_1\lambda_1,\lambda_2\lambda_2]$. So, we went from 2 variables to 5, just by seeking a quadratic combination!
- Since one in general has tens or hundreds of wavelengths, and would like to consider higher order polynomials, you can get the idea of the large number of variables that would be required.
- Now fortunately there is a solution to this problem, which is commonly referred to as the **kernel trick.**
- Ok, let's call x the original set of n variables, let's call ϕ (x) the non-linear combination (mapping) of these variables into a m > nm > n dataset.
- Now we can compute the kernel function κ (x) = ϕ (x) ϕ^T(x). Note that the kernel function in practice is an array even though we are using a function (continuous) notation.
- Now, it turns out that the kernel function plays the same role as the covariance matrix did in linear PCA.
- This means that we can calculate the eigenvalues and eigenvectors of the kernel matrix and these are the new principal components of the m-dimensional space where we mapped our original variables into.
- The kernel trick is called this way because the kernel function (matrix) enables us to get to the

eigenvalues and eigenvector without actually calculating $\phi(x)$ explicitly. This is the step that would blow up the number of variables and we can circumvent it using the kernel trick.

- There are of course different choices for the kernel matrix. Common ones are the Gaussian kernel or the polynomial kernel.
- A polynomial kernel would be the right choice for decision boundaries that are polynomial in shape.
- A Gaussian kernel is a good choice whenever one wants to distinguish data points based on the distance from a common centre.
- Once we have the kernel, we follow the same procedure as for conventional PCA. Remember the kernel plays the same role as the covariance matrix in linear PCA, therefore we can calculate its eigenvalues and eigenvectors and stack them up to the selected number of components we want to keep.

2.8.2 Local Binary Pattern (LBP)

- *Local Binary Pattern (LBP)* is a very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.
- The LBP feature vector, in its simplest form, is created in the following manner :



(1B1)

(i) Divide the examined window into cells (e.g. 16x16 pixels for each cell).

(ii) For each pixel in a cell, compare the pixel to each of its 8 neighbors (on its left-top, left-middle, left-bottom, right-top, etc.). Follow the pixels along a circle, i.e. clockwise or counterclockwise.

(iii) In the above step, the neighbours considered can be changed by varying the radius of the circle around the pixel, R and the quantization of the angular space P.

(iv) Where the center pixel's value is greater than the neighbor's value, write "0". Otherwise, write "1". This gives an 8-digit binary number (which is usually converted to decimal for convenience).

(v) Compute the histogram, over the cell, of the frequency of each "number" occurring (i.e., each combination of which pixels are smaller and which are greater than the centre). This histogram can be seen as a 256-dimensional feature vector.

(vi) Optionally normalize the histogram.

(vii) Concatenate (normalized) histograms of all cells. This gives a feature vector for the entire window.

- The feature vector can now then be processed using some machine-learning algorithm to classify images. Such classifiers are often used for face recognition or texture analysis.

2.9 INTRODUCTION TO VARIOUS FEATURE SELECTION TECHNIQUES

GQ. What is feature selection? Explain different feature selection algorithms.
GQ. Explain forward and backward feature selection process.

- Feature Selection is the method of reducing the input variable to your model by using only relevant data and getting rid of noise in data.
- It is the process of automatically choosing relevant features for your machine learning model based on the type of problem you are trying to solve.
- We do this by including or excluding important features without changing them.
- It helps in cutting down the noise in our data and reducing the size of our input data.

Benefits of Feature Selection

(1) Using unnecessary feature variables for the prediction can deteriorate the performance of a predictive model. Thus, feature selection helps in improving the model performance.

(2) Algorithms like linear regression and logistic regression must avoid using correlated features. Using feature selection methods thus leads to a better fit of these models.

(3) It is an excellent practice to work with a minimum set of predictive modeling features as they significantly reduce the algorithm's complexity and computational costs.

Feature Selection Algorithms

Feature selection algorithms can be classified into three major categories :

(1) Filter methods (2) Wrapper Methods (3) Intrinsic Methods

(1) Filter methods

- Filter methods for feature selection are usually pre-processing techniques that independently consider each feature in the dataset.
- It implements its model on each feature and then evaluates which can then be used to analyze its impact on a predictive model.
- Such methods include information gain, entropy, consistency-based feature selection, correlation matrix, etc.
- For basic guidance, we can refer to the following table for defining correlation coefficients.

Feature/Response	Continuous	Categorical
Continuous	Pearson's correlation	LDA
Categorical	Anova	Chi-Square

- **Pearson's Correlation :** It is used as a measure for quantifying linear dependence between two continuous variables X and Y. Its value varies from – 1 to + 1. Pearson's correlation is given as :

$$\rho_{X,Y} = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y}$$

- **LDA :** Linear discriminant analysis is used to find a linear combination of features that characterizes or separates two or more classes (or levels) of a categorical variable.
- **ANOVA :** ANOVA stands for Analysis of variance. It is similar to LDA except for the fact that it is operated using one or more categorical independent features and one continuous dependent feature. It provides a statistical test of whether the means of several groups are equal or not.
- **Chi-Square :** It is a is a statistical test applied to the groups of categorical features to evaluate the likelihood of correlation or association between them using their frequency distribution.
- One thing that should be kept in mind is that filter methods do not remove multicollinearity. So, we must deal with multicollinearity of features as well before training models for your data.
- The filter methods are popular for feature selection methods because of their generic behavior.
- However, they are proven disadvantageous as they do not consider the nature of a predictive model and typically reduce its accuracy.

(2) Wrapper Methods

- The wrapper methods aim to create a subset of features from the given dataset that results in the best performance of a predictive model.
- In other words, it tests every subset of the available variables for the model's accuracy.
- There are two kinds of wrapper methods for feature selection, greedy and non-greedy.
- The greedy search approach involves following a path that heads towards achieving the best results at the given time. This approach results in locally best results. An example of a greedy search method is the Recursive Feature Elimination (RFE) method.
- On the other hand, the non-greedy approach involves assessing all the previous feature subsets and can lead to a path that results in the overall best performance. Genetic Algorithms (GA) and Simulated Annealing (SA) are examples of non-greedy wrapper methods.

(3) Intrinsic Methods

- This method combines the qualities of both the Filter and Wrapper method to create the best subset.
- In these methods, the feature selection algorithm is blended as part of the learning algorithm, thus having its own built-in feature selection methods.
- It means if you are using these algorithms, you don't need to worry about using a feature selection method explicitly.
- These methods are fast and easy to implement as no external algorithm is required to filter features.

- Examples of intrinsic methods for feature selection are :

  (i) **Rule-and-Tree-based algorithms :** The basic idea behind the mathematical structure of these algorithms is to split the dataset into different sets based on a feature variable in a manner that results in a homogenous spread in the resulting subsets. Thus, a feature variable that didn't lead to a split is automatically considered redundant by the model.

  (ii) **Multivariate adaptive regression spline (MARS) models :** The MARS algorithms create new feature variables from the existing ones in the dataset. These features are then added to a linear model in sequence. If the algorithm does not use a few features to create the MARS features, they are considered irrelevant and automatically ignored.

  (iii) **Regularization models :** These models assign weights to features in a model to improve the fit quality. The lasso regularization method implements consequences that can be narrowed down to absolute zero, indicating you should remove the feature from the predictive model's equation.

2.9.1 Forward Feature Selection

- Forward feature 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.

2.9.2 Backward Feature Selection

- In backward feature selection, also called 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.
- Below are some main steps which are used to apply backward elimination process:

  Step 1 : Firstly, we need to select a significance level to stay in the model. (SL=0.05)

  Step 2 : Fit the complete model with all possible predictors/independent variables.

  Step 3 : Choose the predictor which has the highest P-value, such that.

  If P-value > SL, go to step 4.

  Else Finish, and Our model is ready.

  Step 4 : Remove that predictor.

  Step 5 : Rebuild and fit the model with the remaining variables.

2.10 STATISTICAL FEATURE ENGINEERING

GQ. Explain different statistical measures in feature engineering with suitable examples.

- It is essential to have an overall picture of the data, if data preprocessing is to be made successful.
- Statistical description of data is useful in identifying the properties of the data and highlight which data value should be treated at noise or outliers.

- Following are the basic statistical description of data :

2.10.1 Measures of Central Tendency

- A measure of central tendency is a number used to represent the center or middle of a set of data values.
- The mean, median, mode and midrange are commonly used measures of central tendency.

(i) Mean

- The mean, or average, of n numbers is the sum of the numbers divided by n.
- The mean is denoted by $\bar{x}$ and is read as "x-bar".
- For the data set $x_1, x_2, \ldots, x_n$, the mean is

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{x_1 + x_2 + \ldots + x_n}{n}$$

- Sometimes, weights are associated with the value x_i. The weights reflect the importance, significance or frequency of occurrence to their respective values. The weighted arithmetic mean or the weighted average is computed as

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} = \frac{w_1 x_1 + w_2 x_2 + \ldots + w_n x_n}{w_1 + w_2 + \ldots + w_n}$$

- Mean has one limitation; it is highly sensitive to outliers. Under such condition, median would be a better measure of central tendency.

(ii) Median

- The median of n numbers is the middle number when numbers are written in order.
- If n is even, the median is the mean of the two middle numbers.
- When we have large number of observations, the median is expensive to compute.
- In such case, we can approximate the median of the entire data set by interpolation using the formula :

$$\text{median} = L_1 + \left(\frac{\frac{n}{2} - (\Sigma \text{freq})_1}{\text{freq}_{\text{median}}} \right) \text{width}$$

where,

L_1 is the lower boundary of the median interval,

n is the number of values in the entire data set,

$(\Sigma \text{ freq})_1$ is the sum of frequencies of all of the intervals that are lower than the median interval

$Freq_{median}$ is the frequency of the median interval and

width is the width of the median interval.

(iii) Mode

The mode of n numbers is the number or numbers that occur most frequently.

There may be one mode, no mode or more than one mode.

For unimodal numeric data that are asymmetrical, we have the following empirical relation :

mean – median ≈ 3 × (mean – median)

(iv) Midrange

It is the average of the largest and smallest values in the set.

(v) Size, Count

Size or count is the number of data points in a data set.

Size = n = count(x_i) where i = 1...n.

(vi) Length

Length defines the number of rows present in the dataset. In Python, you can **use the built-in len() method.**

Ex. 2.10.1 : The data set below gives the waiting time (in minutes) of several people having the oil changed in their car at an auto mechanics shop. 22, 18, 25, 21, 28, 26, 20, 28, 20. Find the mean, median, mode and the midrange of the data set.

☑ **Soln. : Data set :** 22, 18, 25, 21, 28, 26, 20, 28, 20

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{x_1 + x_2 + \ldots + x_n}{n}$$

$$= \frac{22 + 18 + 25 + 21 + 28 + 26 + 20 + 28 + 20}{9} = 23.11$$

To find median, arrange the values in order.

18, 20, 20, 21, 22, 25, 26, 28, 28

There are total 9 values i.e. n is odd. Thus, the median is the middle value.

$$Median = 22$$

Mode is the number or numbers that occur most frequently. Here, 20 and 28 are repeated twice. Thus, data set is bimodal with values 20 and 28.

$$Midrange = \frac{largest\ value + smallest\ value}{2} = \frac{28 + 18}{2} = 23$$

2.10.2 Dispersion of Data

- A measure of dispersion is a statistic that tells you how dispersed, or spread out, data values are.
- The measures include range, quantiles, quartiles, percentiles, the interquartile range, and the five-number summary displayed as a boxplot, variance, and standard deviation.

(i) Quartiles

- Quartiles are values that divide your data into quarters.
- However, quartiles are not shaped like pizza slices; instead they divide your data into four segments according to which the numbers fall on the number line.
- The four quarters that divide a data set into quartiles are:
  (a) The lowest 25% of numbers. Also called the 1st quartile (Q_1) or 25th percentile.
  (b) The next lowest 25% of numbers (up to the median). Also called the 2nd quartile (Q_2) or 50th percentile.
  (c) The second highest 25% of numbers (above the median). Also called the 3rd quartile (Q_3) or 75th percentile.
  (d) The highest 25% of numbers. Also called the 4th quartile (Q_4) or 100th percentile.
- As quartiles divide numbers up according to where their position is on the number line, you have to put the numbers in order before you can figure out where the quartiles are.

(ii) Interquartile Range (IQR)

- Interquartile range is defined asthe difference between the upper and lower quartile values in a set of data.
- It is commonly referred to as IQR and is used as a measure of spread and variability in a data set.

$$IQR = Q_3 - Q_1$$

(iii) Five Number Summary

- The five number summary gives you a rough idea about what your data set looks like.
- It includes five items: the minimum value, the first quartile (Q_1), the median, the third quartile (Q_3), the maximum value.
- In order for the five numbers to exist, your data set must meet these two requirements :
  (a) Your data must be **univariate**. In other words, the data must be a single variable. For example, this list of weights is one variable: 120, 100, 130, 145. If you have a list of ages and you want to compare the ages to weights, it becomes bivariate data (two variables). For example: age 1 (25 pounds), 5 (60 pounds), 15 (129 pounds). The matching pairs makes it impossible to find a five number summary.
  (b) Your data must be **ordinal, interval, or ratio**.

Steps to Find a Five-Number Summary

Step 1 : Put your numbers in ascending order (from smallest to largest).

For example, consider the data set in order as: 1, 2, 5, 6, 7, 9, 12, 15, 18, 19, 27.

Step 2 : Find the minimum and maximum for your data set.

In the example in step 1, the minimum (the smallest number) is 1 and the maximum (the largest number) is 27.

Step 3 : Find the median. The median is the middle number.

Step 4 : Place parentheses around the numbers above and below the median. (This is not technically necessary, but it makes Q_1 and Q_3 easier to find).

(1, 2, 5, 6, 7), 9, (12, 15, 18, 19, 27).

Step 5 : Find Q_1 and Q_3. Q_1 can be thought of as a median in the lower half of the data, and Q_3 can be thought of as a median for the upper half of data.

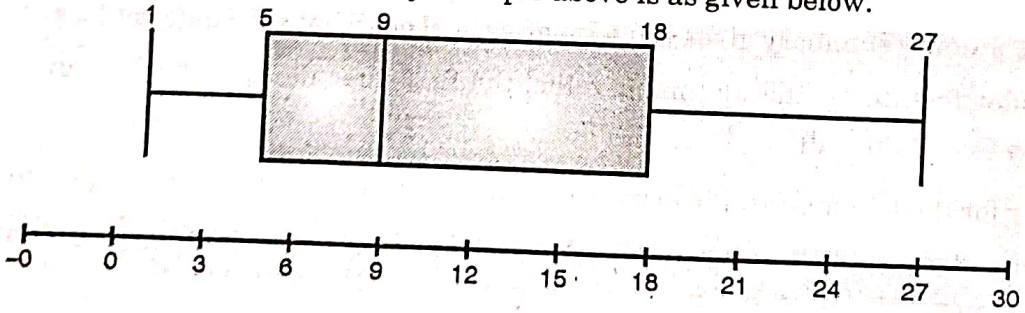
(1, 2, 5, 6, 7), 9, (12,15,18,19,27).

Step 6 : Write down your summary found in the above steps.

minimum = 1, Q_1= 5, median = 9, Q_3 = 18 and maximum = 27.

(iv) Boxplot

- A boxplot (or whisker plot) is defined as a graphical method of displaying variation in a set of data.
- A boxplot incorporates the five-summary as follows:
  (a) The ends of the box are at the quartiles and the box length is the interquartile range.
  (b) The median is marked by a line within a box.
  (c) Two lines (called whiskers) outside the box extend to the minimum and maximum values in the data set.
- The boxplot for five-number summary example above is as given below.



(184)**Fig. 2.10.1 : Boxplot Example**

Boxplots can be computed in *O(n logn)* time.

(v) Outlier

It is a value higher or lower than 1.5 × IQR **(Inter-Quartile Range)**

(vi) Variance and Standard Deviation

- Variance and standard deviation are measures of data dispersion. They indicate how spread out a data distribution is.
- For the data set $x_1, x_2, \ldots, x_n$, the variance is calculated as

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \left(\frac{1}{n}\sum_{i=1}^{n}x_i^2\right) - (\bar{x})^2$$

where $\bar{x}$ is the mean value of the observation

- The standard deviation, σ, of the observations is the square root of the variance σ^2.
- A low standard deviation indicates that the data observations tend to be very close to the mean, while a high standard deviation indicates that the data observations are spread out over a large range of values.
- When all observations have the same value, $\sigma = 0$. Otherwise, $\sigma > 0$.

2.11 MULTIDIMENSIONAL SCALING

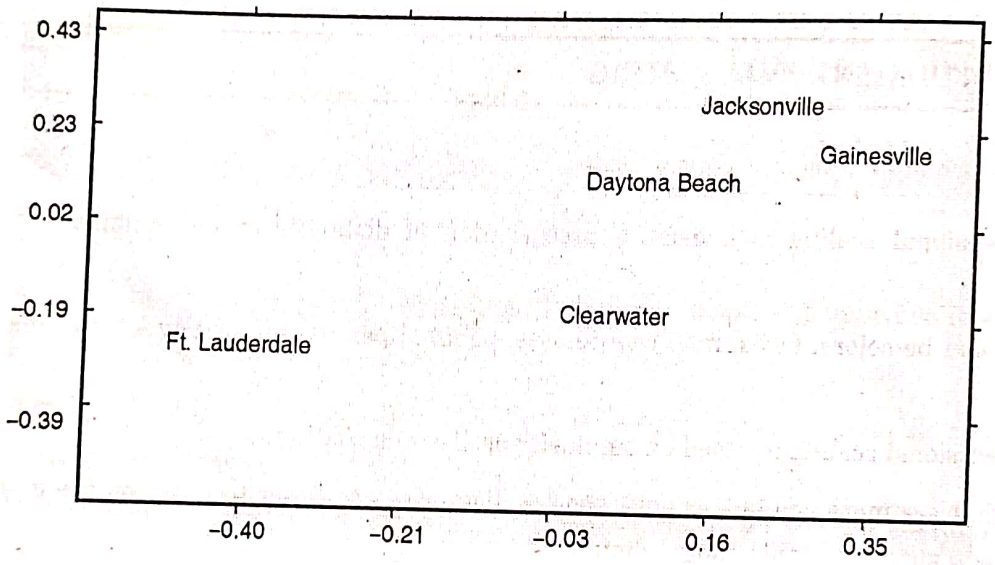
GQ. Explain the concept of multidimensional scaling.

- Multidimensional scaling is a visual representation of distances or dissimilarities between sets of objects.
- "Objects" can be colors, faces, map coordinates, political persuasion, or any kind of real or conceptual stimuli.
- Multidimensional scaling is based on similarity or dissimilarity data.
- Objects that are more similar (or have shorter distances) are closer together on the graph than objects that are less similar (or have longer distances).
- As well as interpreting dissimilarities as distances on a graph, MDS can also serve as a dimension reduction technique for high-dimensional data.
- The term scaling comes from psychometrics, where abstract concepts ("objects") are assigned numbers according to a rule. For example, you may want to quantify a person's attitude to global warming. You could assign a "1" to "doesn't believe in global warming", a 10 to "firmly believes in global warming" and a scale of 2 to 9 for attitudes in between. You can also think of "scaling" as the fact that you're essentially scaling down the data (i.e., making it simpler by creating lower-dimensional data).
- Data that is scaled down in dimension keeps similar properties. For example, two data points that are close together in high-dimensional space will also be close together in low-dimensional space.
- The "multidimensional" part is due to the fact that you aren't limited to two dimensional graphs or data.
- Three-dimensional, four-dimensional and higher plots are possible.

- Multidimensional scaling uses a square, symmetric matrix for input. The matrix shows relationship between items.
- For a simple example, let's say you had a set of cities in Florida and their distances :

CITY	Clearwater	Daytona Beach	Ft. Lauderdale	Gainesville	Jacksonville
Clearwater	0	159	247	131	197
Daytona Beach	159	0	230	97	89
Ft. Lauderdale	247	230	0	309	317
Gainesville	131	97	309	0	68
Jacksonville	197	89	317	68	0

- The scaling produces a graph like the one below.



- The very simple example above shows cities and distances, which are easy to visualize as a map. However, multidimensional scaling can work on "theoretically" mapped data as well.

Basic steps

(1) **Assign a number of points to coordinates in N-dimensional space :** N-dimensional space could be 2-dimensional, 3-dimensional, or higher spaces (at least, theoretically, because 4-dimensional spaces and above are difficult to model). The orientation of the coordinate axes is arbitrary and is mostly the researcher's choice. For maps like the one in the simple example above, axes that represent north/south and east/west make the most sense.

(2) **Calculate Euclidean distances for all pairs of points :** The Euclidean distance is the straight-line distance between two points x and y in Euclidean space. It's calculated using the Pythagorean theorem ($c^2 = a^2 + b^2$), although it becomes somewhat more complicated for N-dimensional space. This results in the similarity matrix.

(3) **Compare the similarity matrix with the original input matrix by evaluating the stress function :** Stress is a goodness-of-fit measure, based on differences between predicted and actual distances. Fits close to zero are excellent, while anything over 0.2 should be considered "poor".

(4) **Adjust coordinates, if necessary, to minimize stress.**

Types of MDS

(1) **Metric MDS :** Metric MDS already has the input matrix in the form of distances (i.e. actual distances between cities) and therefore the distances have meaning in the input matrix and create a map of actual physical locations from those distances.

(2) **Non-metric MDS :** In non-metric MDS, the distances are just a representation of the rankings (i.e., high as in 7 or low as in 1) and they do not have any meaning on their own but they are needed to create the map using Euclidean geometry and the map then just shows the similarity in rankings represented by distances between coordinates on the map.

2.12 MATRIX FACTORIZATION TECHNIQUE

GQ. Explain the concept of matrix factorization in recommender system.

- Matrix factorization is a class of collaborative filtering algorithms used in recommender systems.
- Matrix factorization algorithms work by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices.
- The idea behind matrix factorization is to represent users and items in a lower dimensional latent space.
- Let $R \in \mathbb{R}^{m \times n}$ denote the interaction matrix with m users and n items, and the values of R represent explicit ratings.
- The user-item interaction will be factorized into a user latent matrix $P \in \mathbb{R}^{m \times k}$ and an item latent matrix $Q \in \mathbb{R}^{n \times k}$, where $k << m, n$ is the latent factor size.
- Let p_u denote the u^{th} row of P and q_i denote the i^{th} row of Q. For a given item i, the elements of q_i measure the extent to which the item possesses those characteristics such as the genres and languages of a movie. For a given user u, the elements of p_u measure the extent of interest the user has in items' corresponding characteristics.
- These latent factors might measure obvious dimensions as mentioned in those examples or are completely uninterpretable. The predicted ratings can be estimated by

$$\hat{R} = PQ^T$$

Where $\hat{R} \in \mathbb{R}^{m \times n}$ is the predicted rating matrix which has the same shape as R.

- One major problem of this prediction rule is that users/items biases cannot be modelled. For example, some users tend to give higher ratings or some items always get lower ratings due to poorer quality.

These biases are commonplace in real-world applications. To capture these biases, user specific and item specific bias terms are introduced. Specifically, the predicted rating user gives to item is calculated by

$$\hat{R}_{ui} = p_u q_i + b_u + b_i$$

- Then, we train the matrix factorization model by minimizing the mean squared error between predicted rating scores and real rating scores. The objective function is defined as follows :

$$\underset{p, Q, b}{\text{argmin}} \sum_{(u, i,) \in k} ||R_{ui} - \hat{R}_{ui}||^2 + \lambda (||P||_F^2 + ||Q||_F^2 + b_u^2 + b_i^2)$$

where λ denotes the regularization rate. The regularizing term

$$\lambda (||P||_F^2 + ||Q||_F^2 + b_u^2 + b_i^2)$$

is used to avoid over-fitting by penalizing the magnitude of the parameters. The (u, i) pairs for which R_{ui} is known are stored in the set $k = \{(u, i) | R_{ui} \text{ is known}\}$. The model parameters can be learned with an optimization algorithm, such as Stochastic Gradient Descent and Adam.

- An intuitive illustration of the matrix factorization model is shown below:

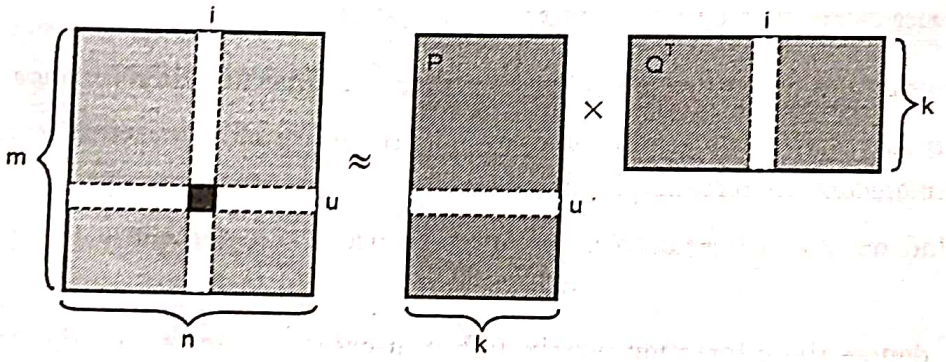


Fig. 2.12.1 : Illustration of matrix factorization model

Descriptive Questions

Q. 1 What is feature engineering ? Explain the four different processes in feature engineering.

Q. 2 What is data preprocessing ? Explain the steps involved in data preprocessing.

Q. 3 How do you handle missing data in dataset ?

Q. 4 Explain different types of feature selection methods.

Q. 5 Explain different statistical measures in feature engineering with suitable examples.

Q. 6 Write a short note on : Principal Component Analysis (PCA)

Q. 7 Write a short note on : Multidimensional Scaling

Q. 8 Explain the concept of Matrix Factorization.

Chapter Ends ...

□□□