ML Unit 2:

PYQs.-2022

Q3)a) Consider a vector x = (23, 29, 52, 31, 45, 19, 18, 27) Apply feature
scaling and find out min-max scaled values as well as z-score values.
[8]
b) Explain the process of Principal Component Analysis (PCA) in brief.
[7]

Q4) a) How to handle missing values in a dataset that will be used for training the ML model? [5]
b) Explain the types of wrapper methods for feature selection. [5]
c) Explain Local Binary Pattern (LBP) feature extraction technique with
suitable example.

UNIT TEST Questions

Q.3 a) What do you mean by regression? Explain with suitable example.
b) Write a short note on : [6]
i) MAE
ii) RMSE
iii) R2

c) What is gradient descent? Compare batch gradient and stochastic gradient descent.

OR

Q4 a) Explain the process of Principal Component Analysis (PCA) in brief.

b) How to handle missing values in a dataset that will be used for training the ML model?

c) Explain the types of wrapper methods for feature selection.

Explain Concept of Feature Engineering.

Feature engineering is the process of using domain knowledge to create new features or modify existing ones to improve the performance of a machine learning model. It involves several steps:

1.Feature Creation/Construction

Domain Knowledge: Leveraging understanding of the problem domain to generate new features. For instance, in a dataset with date information, you might create features like "day of the week" or "month" that could be relevant to the model.

Mathematical Operations: Combining or transforming existing features. For example, creating a feature "age" from "date of birth" or deriving "body mass index (BMI)" from "weight" and "height."

2.Feature Transformation:
Modifying existing features to make them more suitable for modeling. This includes operations like scaling, normalization, and encoding categorical variables.

Normalization: Scaling features to a standard range, often [0, 1]. This is particularly useful when features have different units or scales. Normalization helps in speeding up the convergence of gradient descent algorithms.
Standardization: Transforming features to have a mean of 0 and a standard deviation of 1. This is useful for algorithms that assume data is normally distributed, like Support Vector Machines (SVMs) and Principal Component Analysis (PCA).
Encoding Categorical Variables: Converting categorical data into numerical form. Common techniques include:
One-Hot Encoding: Creating binary columns for each category.
Label Encoding: Assigning each category a unique integer.
Target Encoding: Replacing categories with the mean of the target variable for each category.

3.Feature Selection:
Choosing the most important features that contribute to the predictive power of the model while removing irrelevant or redundant features. This helps in reducing overfitting and improving model performance.

Filter Methods: Using statistical tests to select features that have the strongest relationship with the target variable. Examples include Chi-square test, ANOVA, and correlation coefficient.

Wrapper Methods: Iteratively selecting features based on model performance. Techniques include:
Sequential Forward Selection (SFS): Starting with no features and adding one feature at a time that improves performance.
Sequential Backward Selection (SBS): Starting with all features and removing one feature at a time that has the least impact on performance.

Embedded Methods: Feature selection integrated into the training process of the model, such as Lasso regression which performs both feature selection and regularization.

4.Feature Extraction:
Creating new features from the original ones using techniques such as Principal Component Analysis (PCA) or Local Binary Pattern (LBP). This helps in reducing dimensionality and capturing important patterns in the data.

Principal Component Analysis (PCA): Reducing dimensionality by transforming features into a set of orthogonal components that capture the most variance in the data. PCA is useful for visualizing high-dimensional data and improving model efficiency.

Kernel PCA: Extending PCA to handle non-linearly separable data by using kernel functions to map data to higher dimensions where linear PCA can be applied.

Local Binary Pattern (LBP): A texture descriptor used in image processing to represent local spatial patterns, useful for tasks like facial recognition.

Explain Feature Extraction.

technical-
Feature extraction is a process that extracts a set of new features from the original features through some functional mapping. Feature extraction method creates a new feature set.

Feature extraction increases the accuracy of learned models by extracting features from the input data. This phase of the general framework reduces the dimensionality of data by removing the redundant data.

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 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 process of feature extraction is useful when you need to reduce the number of resources needed for processing without losing important or relevant information.

Feature extraction can also reduce the amount of redundant data for a given analysis. Also, the reduction of the data and the machine's efforts in building variable combinations (features) facilitate the speed of learning and generalization steps in the machine learning process.

gpt-
Feature extraction is a technique used to transform raw data into a set of new features that can be more useful for modeling.

1. Principal Component Analysis (PCA)
Purpose: PCA reduces the dimensionality of the data while preserving as much variance as possible. It transforms the original features into a new set of features called principal components, which are uncorrelated and ordered by the amount of variance they capture.
How It Works: PCA identifies the directions (principal components) along which the variance of the data is maximized. The first principal component captures the most variance, the second captures the next most variance orthogonally to the first, and so on.

Applications: PCA is useful for visualizing high-dimensional data in 2D or 3D, and for reducing the number of features while retaining essential information.

2. Kernel PCA

Purpose: Kernel PCA extends PCA to handle non-linear data by using kernel functions. This allows it to find principal components in higher-dimensional spaces that are not possible with linear PCA.

How It Works: Kernel PCA applies a kernel function (e.g., polynomial or radial basis function) to map data into a higher-dimensional space where linear PCA can be performed. This enables capturing complex, non-linear relationships.

Applications: Used for non-linear dimensionality reduction and feature extraction in tasks where data is not linearly separable.

3. Local Binary Pattern (LBP)

Purpose: LBP is used in image processing to describe texture by encoding local spatial patterns. It's effective for tasks like facial recognition and texture classification.

How It Works: For each pixel in an image, LBP compares the pixel's intensity with its neighboring pixels. It generates a binary pattern based on whether the neighboring pixels are greater than or less than the central pixel. These patterns are then aggregated to form a feature vector representing the texture.

Applications: Used for texture classification, facial recognition, and object detection.

What is the goal of feature extraction.

The goal of feature extraction is to transform raw data into a set of features that are more suitable for modeling and analysis. The main objectives are:

Dimensionality Reduction: To reduce the number of features while retaining the most important information. This helps in making models more efficient and reducing computational costs.

Improving Model Performance: By extracting meaningful features, you can enhance the performance of machine learning models. The new features often capture essential patterns and structures in the data better than the original features.

Noise Reduction: To reduce the impact of irrelevant or redundant features, which can lead to overfitting and decreased model performance. Feature extraction can help in isolating the most informative features.

Simplification: To simplify the data representation, making it easier to visualize and interpret. For example, reducing high-dimensional data to a few principal components can aid in understanding the underlying structure of the data.

Enhanced Interpretability: To create features that are more interpretable and relevant to the problem at hand. For instance, features derived from domain knowledge

can provide insights that are more aligned with the objectives of the analysis.

Explain the types of wrapper methods for feature selection.

Wrapper methods for feature selection are techniques that evaluate subsets of features to find the best ones for a model. They do this by training the model with different sets of features and choosing the set that performs best. Here are the main types:

Sequential Forward Selection (SFS):
How It Works: Starts with no features and adds one feature at a time. At each step, it adds the feature that improves the model's performance the most.
Goal: Build a feature set by sequentially adding the most useful features.
Example: If you start with a basic feature and add features like age, income, and education one by one, SFS will pick the combination that gives the best results.

Sequential Backward Selection (SBS):
How It Works: Starts with all features and removes one feature at a time. At each step, it removes the feature that has the least impact on the model's performance.
Goal: Reduce the feature set by removing the least useful features.
Example: If you start with all features and remove them one by one, SBS will keep removing features until you have the best set remaining.

Recursive Feature Elimination (RFE):
How It Works: Trains the model and removes the least important feature based on model weights or coefficients. It repeats this process until only the most important features remain.
Goal: Eliminate less important features while retaining the most useful ones.
Example: If you use a linear model, RFE will look at which features have the smallest coefficients and remove them, focusing on those that contribute the most.

Explain Preprocessing of data in brief.

Preprocessing of data is a crucial step in preparing raw data for machine learning models. It involves several steps to ensure the data is clean, well-formatted, and suitable for analysis. Here's a brief overview:

1.Data Cleaning:
Handling Missing Values: Filling in missing data with mean, median, or a specific value, or removing records with missing data altogether.
Removing Noise: Filtering out irrelevant or inconsistent data that could negatively affect the model's performance.
2.Data Transformation:
Normalization: Scaling data to a standard range, like [0, 1], to ensure that no single feature dominates the model.

Standardization: Adjusting data to have a mean of 0 and a standard deviation of 1, making it easier for some algorithms to converge and perform better.
Encoding Categorical Data: Converting categorical variables (like "yes" or "no") into numerical format so that models can process them. Techniques include one-hot encoding and label encoding.

3.Data Integration:
Combining Data from Multiple Sources: Merging data from different databases or files into a single dataset for analysis.

4.Data Reduction:
Dimensionality Reduction: Reducing the number of features while preserving important information, often using techniques like PCA.
Feature Selection: Identifying and keeping only the most relevant features for the model.

5.Data Splitting:
Train-Test Split: Dividing the data into training and testing sets to evaluate the model's performance on unseen data.

Preprocessing is essential because it improves the quality of the data, which in turn enhances the performance and accuracy of machine learning models.

How to handle missing values in a dataset that will be used for training the ML model?

Handling missing values in a dataset is important to ensure that your machine learning model performs well. Here are some common methods to handle missing values:

1. Remove Missing Data:

Remove Rows: If a few rows have missing values, you can remove those rows from the dataset. This is only advisable if the number of missing values is small compared to the size of the dataset.

Remove Columns: If an entire column has a large number of missing values, you might consider dropping that column, especially if it's not crucial to the model.

2. Impute Missing Values:

Mean/Median Imputation: Replace missing values with the mean or median of the non-missing values in the same column. This is common for numerical data.

Example: If the column "age" has some missing values, you could fill them in with the average or median age of the other entries.

Mode Imputation: For categorical data, replace missing values with the most frequent category (mode).

Example: If the column "gender" has missing values, you can fill them in with the most common gender in the dataset.

Forward/Backward Fill: For time-series data, use the previous (forward fill) or next (backward fill) value to fill in the missing data.

3. Predict Missing Values:
Use Algorithms: You can use machine learning algorithms to predict the missing values based on other features in the dataset. This approach is more complex but can be effective if the missing values are significant.

4. Flag and Fill:
Add a New Feature: Create a new column to flag where the data was missing (e.g., 1 for missing, 0 for not missing), and then fill the missing values using any of the above methods. This helps the model know that there was missing data and how it was handled.

5. Use Models That Handle Missing Data:
Some Algorithms: Certain machine learning algorithms, like decision trees or Random Forests, can handle missing data internally without requiring explicit imputation.

Explain the process of Principal Component Analysis (PCA) in brief.

technical-
This method was introduced by Karl Pearson. It works on a condition that while the data in a higher dimensional space is mapped to data in a lower dimension space, the variance of the data in the lower dimensional space should be maximum.

Principal Component Analysis (PCA) is to reduce the dimensionality of a data set by finding a new set of

variables, smaller than the original set of variables, retains most of the sample's information and useful for the compression and classification of data.

In PCA, it is assumed that the information is carried in the variance of the features, that is, the higher the variation in a feature, the more information that feature carries.

Hence, PCA employs a linear transformation that is based on preserving the most variance in the data using the least number of dimensions.

gpt-
Here's a brief overview of the process:

1. Standardize the Data:
Before applying PCA, it's important to standardize the data so that each feature has a mean of 0 and a standard deviation of 1. This ensures that all features contribute equally to the analysis.

2. Compute the Covariance Matrix:
Calculate the covariance matrix of the data to understand how the features vary with respect to each other. The covariance matrix shows the relationships between the features.

3. Calculate Eigenvalues and Eigenvectors:
From the covariance matrix, compute the eigenvalues and eigenvectors.
Eigenvalues tell you the amount of variance captured by each principal component.

Eigenvectors represent the directions of these principal components.

4. Sort Eigenvalues and Select Principal Components:
Sort the eigenvalues in descending order and choose the top eigenvectors (principal components) that correspond to the largest eigenvalues. These components capture the most variance in the data.

5. Transform the Data:
Multiply the original data by the selected eigenvectors to project the data onto the new principal component space. This gives you the new set of features (principal components) that are uncorrelated and capture the maximum variance.

6. Reduce Dimensions:
Depending on the problem, you can now reduce the number of dimensions by selecting only the top principal components, which simplifies the data while retaining the most important information.

7. Interpret the Results:
The resulting principal components can be used for further analysis, such as visualization or as input features for machine learning models.

Kernel PCA

Kernel PCA is a powerful technique that extends the capabilities of standard PCA to handle non-linear data. It allows for the extraction of non-linear principal components that can capture more complex patterns in high-dimensional datasets.

Key Points:

Nonlinear Form of PCA: Unlike standard PCA, which works well with linear data, Kernel PCA is designed to exploit the complex spatial structures in non-linear, high-dimensional data.

Principal Components: Kernel PCA can extract up to n non-linear principal components, where n is the number of samples in the dataset. This is done without requiring extremely expensive computations, making it efficient for high-dimensional data.

Standard Steps of Kernel PCA:

1.Construct the Kernel Matrix (K):

This matrix is built by applying a kernel function (e.g., polynomial, Gaussian) to the data. The kernel function transforms the original data into a higher-dimensional space where it becomes easier to find the principal components.

2.Compute the Gram Matrix:

The Gram matrix is a modified version of the kernel matrix, adjusted to center the data in the new space. Centering is crucial for accurately capturing the variance in the data.

3.Solve for Eigenvectors:

An N-dimensional column vector of eigenvectors is computed, where N is the number of samples. These eigenvectors correspond to the non-linear principal components, capturing the directions of maximum variance in the transformed space.

4.Compute the Kernel Principal Components:

The non-linear principal components are derived from the eigenvectors and can be used to reduce the dimensionality of the data while retaining its essential structure.

Kernel PCA supports both transform and inverse_transform.

Transform: Kernel PCA can transform the original data into the new feature space defined by the kernel principal components. This is useful for reducing dimensionality or for preparing data for further analysis.

Inverse Transform: Kernel PCA also supports an inverse transform, allowing you to map the transformed data back to the original feature space. This is useful for interpreting the results or reconstructing the data after dimensionality reduction.

Local Binary Pattern

Local Binary Pattern (LBP) is a simple yet powerful feature extraction technique mainly used in image processing, particularly for texture classification. It works by summarizing the local structure around each pixel in an image.

How Local Binary Pattern (LBP) Works:
Neighborhood Comparison:

For each pixel in the image, LBP compares the pixel's intensity with the intensities of its surrounding neighbors. The neighbors are typically the 8 pixels around the central pixel.
Binary Pattern Creation:

A binary code is generated by setting the value to 1 if the neighbor's intensity is greater than or equal to the central pixel's intensity, and 0 otherwise. This forms an 8-bit binary number (for 8 neighbors).
Decimal Conversion:

The binary code is then converted into a decimal number, which is assigned to the central pixel. This number represents the local texture around that pixel.
Histogram Creation:

After processing the entire image, a histogram of the LBP values is created. This histogram captures the distribution of local textures across the image.
Feature Vector:

The histogram acts as a feature vector representing the texture of the image. This vector can then be used as input for machine learning algorithms to classify or analyze the texture.

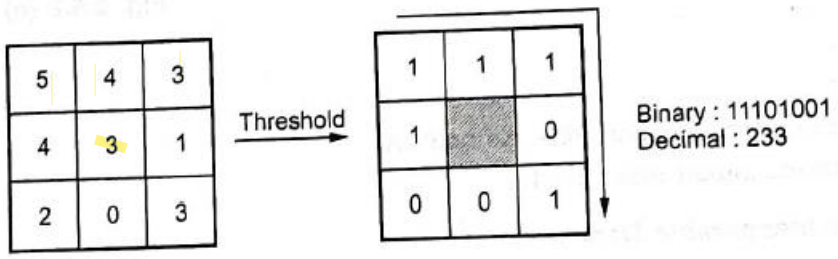

Fig. 2.6.1 LBP code

- Simple LBP feature vector is created in the following manner :

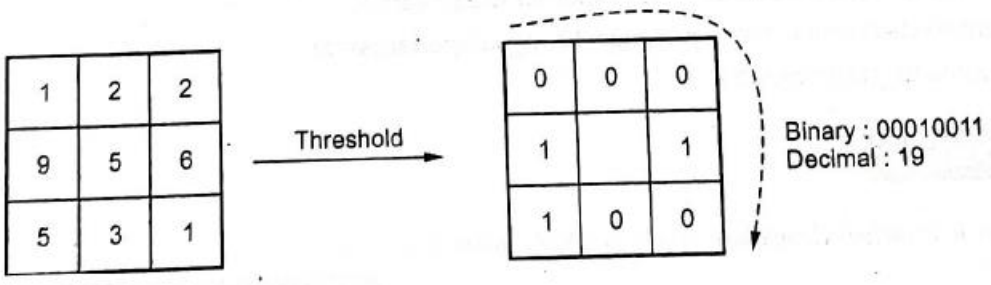

Fig. 2.6.2 LBP feature vector

Explain Multidimensional Scaling.

technical+gpt
Multidimensional Scaling (MDS) Overview
Multidimensional Scaling (MDS) is a technique used to map high-dimensional data into a lower-dimensional space while preserving the pairwise distances between points as closely as possible. This helps in visualizing and interpreting complex data structures.

Key Points:
Objective: The goal of MDS is to find a lower-dimensional representation of the data where the distances between points reflect their original pairwise distances in the high-dimensional space.

Use Case: MDS is useful when the data does not have a clear linear relationship or when it's not known whether such a relationship exists. It is typically used for dimensionality reduction and can serve as a preprocessing step in classification and regression tasks.

Types of MDS:

1. Metric MDS/Classical MDS: This version of MDS aims to preserve the pairwise distance/dissimilarity measure as much as possible.

2. Non-Metric MDS: This method is applicable when only the ranks of a dissimilarity metric are known. MDS then maps the objects so that the ranks are preserved as much as possible

Steps in MDS:

1.Input Data: Start with a matrix representing pairwise dissimilarities or distances between objects.

2.Initialization: Place the objects in a low-dimensional space (2D or 3D) initially, often randomly.

3.Optimization: Iteratively adjust the positions of objects in the lower-dimensional space to minimize the difference between the original pairwise distances and those in the reduced space. This is done using optimization algorithms that aim to minimize the stress function.

4.Output: The final configuration represents the objects in a lower-dimensional space where their distances closely reflect the original dissimilarities.

Example:
Imagine you have survey data where each respondent's answers are compared pairwise. MDS can be used to reduce this high-dimensional data to a 2D plot. In this plot, respondents who are similar in their answers will be placed close to each other, helping to visualize patterns and clusters in the data.

Explain Matrix Factorization Techniques.

technical+gpt
Matrix Factorization Techniques in Recommender Systems:

1.Dominant in Collaborative Filtering:
Matrix factorization is widely used in collaborative filtering recommenders, offering superior accuracy compared to classic nearest-neighbor methods.

2.Recommender System Inputs:
User-item interactions are represented in a matrix, with users as rows and items (e.g., movies) as columns.

3.Content vs. Collaborative Filtering:
Content Filtering: Builds profiles for users/products based on their characteristics.
Collaborative Filtering: Analyzes user-item interactions to find relationships and make predictions.

4.Latent Features:

Matrix factorization generates latent features by decomposing the user-item interaction matrix into lower-dimensional matrices.

5.Prediction and Recommendations:

By understanding the relationships between users and items, matrix factorization helps predict user preferences and make recommendations.

6.Matrix Decomposition:

Matrix factorization decomposes a large matrix (user-item interactions) into smaller matrices representing user and item factors.

7.Advantages:

Superior to nearest-neighbor techniques.

Allows incorporation of additional information like implicit feedback, temporal effects, and confidence levels.

8.Sparsity Handling:

The user-item matrix is often sparse (many missing ratings), and matrix factorization efficiently handles this sparsity to make accurate predictions.

9.Applications:

Used in platforms like Amazon, YouTube, and Netflix to provide personalized recommendations based on user behavior.

10.High Correspondence:

A high match between user and item factors results in strong recommendations.

Feature engineering is a crucial step in the machine learning process that involves preparing and selecting the most relevant features (variables) from raw data to improve the model's performance. It involves four stages:

Feature Creation:
In this stage, new features are generated from the existing data. The goal is to create features that capture additional information that the original features do not explicitly provide. This might involve combining existing features, creating interaction terms, or deriving new metrics that could be useful for the model.

Example: If you have a dataset with height and weight, you might create a new feature called BMI (Body Mass Index) using the formula BMI = weight / (height^2).

Feature Transformation:
Feature transformation involves modifying existing features to make them more suitable for the model. This can include normalization (scaling features to a similar range), encoding categorical variables (e.g., one-hot encoding), and applying mathematical transformations (e.g., logarithmic transformations) to handle skewed distributions.

Example: Converting a categorical feature like "City" into numerical values using one-hot encoding, where each unique city gets its binary column.

Feature Extraction:
Feature extraction focuses on reducing the dimensionality of the dataset by extracting relevant information from the existing features. This stage is often used when dealing with large datasets with many features. Techniques like Principal Component Analysis (PCA) or Singular Value Decomposition (SVD) are common for feature extraction.

Example: Using PCA to reduce a high-dimensional dataset to a smaller set of components that capture most of the variance in the data.

Feature Selection:
Feature selection is the process of identifying and selecting the most important features that contribute to the model's prediction power. It helps in reducing overfitting, improving model accuracy, and speeding up the training process. Methods for feature selection include filter methods (e.g., correlation analysis), wrapper methods (e.g., recursive feature elimination), and embedded methods (e.g., feature importance in tree-based models).

Example: Using a feature importance ranking from a Random Forest model to select the top 10 most important features for training a final model.