Feature engineering

Reading Material

PW SKILLS

Topics Covered

1. Introduction to Feature Engineering
  1.1 Definition and Importance
  1.2 Types of Feature Engineering

2. Handling Missing Data
  2.1 Definition and Types of Missing Data
  2.2 Impact of Missing Data
  2.3 Techniques for Handling Missing Data
    2.3.1 Deletion Methods
    2.3.2 Imputation Methods
  2.4 Advanced Imputation Techniques
    2.4.1 Predictive Modeling
  2.5 Evaluating Imputation Methods
    2.5.1 Impact on Model Performance
    2.5.2 Comparison Techniques

3. Handling Imbalanced Data
  3.1 Definition and Examples
  3.2 Techniques for Handling Imbalanced Data
    3.2.1 Resampling Methods
  3.3 Advanced Techniques
    3.3.1 Algorithmic Approaches

4. Typecasting & Duplicate Handling
  4.1 Type Casting Techniques in Data Preprocessing
    4.1.1 Definition and Importance of Typecasting
    4.1.2 Common Data Types and Conversions
    4.1.3 Conversion Methods Using Pandas
  4.2 Identifying and Handling Duplicates
    4.2.1 Definition and Importance
    4.2.2 Duplicate Detection
    4.2.3 Removal Strategies

5. Handling Outliers
  5.1 Definition of Outliers
  5.2 Importance of Handling Outliers
  5.3 Identifying Outliers
  5.4 Handling Outliers

6. Feature Scaling
  6.1 Definition of Feature Scaling
  6.2 Importance of Feature Scaling
  6.3 Common Feature Scaling Techniques
  6.4 Advanced Scaling Techniques
  6.5 Choosing the Right Scaling Technique

7. Feature Encoding
  7.1 Definition and Importance of Feature Encoding
  7.2 Encoding Techniques
  7.3 Choosing the Right Encoding Technique

8. Feature Extraction
  8.1 Definition and Importance of Feature Extraction
  8.2 Feature Extraction Techniques

9. Feature Selection
  9.1 Definition and Importance of Feature Selection
  9.2 Filter Methods for Feature Selection
  9.3 Wrapper Methods for Feature Selection
  9.4 Embedded Methods for Feature Selection

10. PCA (Principal Component Analysis)
  10.1 Definition and Importance
  10.2 Steps in PCA
  10.3 Interpreting PCA Results

11. Covariance & Correlation
  11.1 Definition of Covariance
  11.2 Definition of Correlation
  11.3 Differences between Covariance and Correlation

12. Pearson Correlation Coefficient & Spearman's Rank Correlation
  12.1 Pearson Correlation Coefficient
  12.2 Spearman's Rank Correlation
  12.3 When to Use Each Method

1. Introduction to Feature Engineering

1.1 Definition and Importance

Feature engineering is the process of using domain knowledge to extract features (characteristics, properties, attributes) from raw data. These features can be used to improve the performance of machine learning algorithms. It's a crucial step in the machine learning pipeline for several reasons:

- **Improved Model Performance:** Well-engineered features can capture the underlying patterns in the data more effectively, leading to better model accuracy and generalization.
- **Reduced Complexity:** By creating informative features, we can often reduce the complexity of the model required to achieve good performance.
- **Increased Interpretability:** Thoughtfully engineered features can make the model's decisions more understandable to humans.
- **Overcoming Limitations:** Some algorithms may not be able to learn complex relationships from raw data. Feature engineering can transform the data into a format that makes these relationships more apparent.

1.2 Types of Feature Engineering

a) Feature Creation:

This involves generating new features from existing ones. Examples include:
- Combining features (e.g., BMI from height and weight)
- Extracting information from complex data types (e.g., extracting day of the week from a date)
- Creating interaction terms between features

b) Feature Transformation:

This involves changing the scale or distribution of existing features. Common techniques include:
- Normalization (scaling to a fixed range, usually 0-1)
- Standardization (scaling to zero mean and unit variance)
- Log transformation (to handle skewed distributions)
- Polynomial features (to capture non-linear relationships)

c) Feature Selection:

This process involves choosing a subset of relevant features for use in model construction. Methods include:
- Filter methods (e.g., correlation with target variable)
- Wrapper methods (e.g., recursive feature elimination)
- Embedded methods (e.g., Lasso regularization)

d) Feature Extraction:

This involves creating new features by combining the original features, often reducing the dimensionality of the data. Techniques include:
- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Autoencoder neural networks

2. Handling Missing Data

2.1 Definition and Types of Missing Data

Missing data refers to the absence of values in a dataset. Understanding the nature of missingness is crucial for choosing appropriate handling methods. There are three main types:

a) Missing Completely at Random (MCAR):
The probability of missing data is the same for all observations. The missingness is unrelated to both observed and unobserved data.

b) Missing at Random (MAR):
The probability of missing data depends only on the observed data, not on the unobserved data.

c) Missing Not at Random (MNAR):
The probability of missing data depends on unobserved data or the missing data itself.

2.2 Impact of Missing Data

Missing data can have several negative impacts on data analysis:

- **Reduced Statistical Power:** Fewer data points lead to larger standard errors and wider confidence intervals.
- **Bias in Parameter Estimates:** If the data is not MCAR, simply ignoring missing data can lead to biased estimates.
- **Complications in Data Analysis:** Many statistical methods and machine learning algorithms are designed to work with complete datasets.
- **Effects on Machine Learning Models:** Missing data can lead to overfitting, underfitting, or biased predictions.

2.3 Techniques for Handling Missing Data

2.3.1 Deletion Methods

a) Listwise Deletion (Complete Case Analysis):

This method involves removing all cases with missing data on any variable.

- **Advantage:** Simple to implement
- **Disadvantage**: Can lead to significant loss of data and potentially biased results if data is not MCAR

b) Pairwise Deletion:

This method uses all available data for each analysis, potentially using different sample sizes for different analyses.

- **Advantage:** Retains more data than listwise deletion
- **Disadvantage:** Can lead to inconsistencies across analyses

2.3.2 Imputation Methods

a) Simple Imputation:

- **Mean/Median/Mode Imputation:** Replacing missing values with the mean (for continuous variables), median (for skewed distributions), or mode (for categorical variables).
- **Constant Value Imputation:** Replacing missing values with a constant, often used to create a "missing" category.

b) Multiple Imputation:
This method involves creating multiple plausible imputed datasets, analyzing each, and pooling the results.

c) Hot-Deck Imputation:
Replacing missing values with values from similar cases in the dataset.

d) Last Observation Carried Forward (LOCF):
In time series data, missing values are replaced with the last observed value.

2.4 Advanced Imputation Techniques

2.4.1 Predictive Modeling

a) Regression Imputation:
Using other variables to predict the missing values through regression analysis.

b) K-Nearest Neighbors (KNN) Imputation:
Imputing missing values based on the values of the K most similar cases.

c) Decision Tree-Based Methods:
Using decision trees or random forests to predict missing values.

d) Machine Learning-Based Imputation:
Using advanced ML algorithms like XGBoost or neural networks for imputation.

2.5 Evaluating Imputation Methods

2.5.1 Impact on Model Performance

Evaluating the impact of imputation on model performance typically involves:

- Cross-validation techniques to assess how well the imputed data generalizes
- Sensitivity analysis to understand how different imputation methods affect the results

2.5.2 Comparison Techniques

Common metrics for comparing imputation methods include:

- **Root Mean Squared Error (RMSE):** Measures the standard deviation of the residuals
- **Mean Absolute Error (MAE):** Measures the average magnitude of the errors
- **R-squared (R^2):** For regression problems, measures the proportion of variance explained
- **Accuracy, precision, recall:** For classification problems, measure different aspects of classification performance

In practice, the choice of imputation method often depends on the specific characteristics of the dataset, the amount and pattern of missing data, and the goals of the analysis. It's important to consider multiple methods and evaluate their impact on the final results.

3. Handling Imbalanced Data

3.1 Definition and Examples

Imbalanced data refers to a situation in classification problems where the classes are not represented equally. This means that one class (the majority class) has significantly more samples than the other class(es) (the minority class(es)).

Examples of imbalanced datasets include:

- Fraud detection: Most transactions are legitimate, with only a small percentage being fraudulent.
- Medical diagnosis: In rare diseases, the number of healthy patients far outweighs the number of patients with the condition.
- Spam email detection: The majority of emails are typically not spam.

The imbalance ratio can vary greatly, from slight imbalances (e.g., 60:40) to extreme cases (e.g., 99.99:0.01).

3.2 Techniques for Handling Imbalanced Data

3.2.1 Resampling Methods

Resampling methods aim to balance the class distribution by either increasing the number of minority class samples or decreasing the number of majority class samples.

a) Oversampling:

- **Random Oversampling:** Randomly duplicate samples from the minority class.
- **Synthetic Minority Over-sampling Technique (SMOTE):** Create synthetic samples of the minority class.
- **Adaptive Synthetic (ADASYN):** Similar to SMOTE but focuses on generating samples near the decision boundary.

b) Undersampling:

- **Random Undersampling:** Randomly remove samples from the majority class.
- **Tomek Links:** Remove majority class samples that form Tomek links with minority class samples.
- **Cluster Centroids:** Use clustering to undersample the majority class.

c) Combination Methods:

- **SMOTEENN:** Combine SMOTE with Edited Nearest Neighbors.
- **SMOTETomek:** Combine SMOTE with Tomek Links removal.

3.3 Advanced Techniques

3.3.1 Algorithmic Approaches

a) Cost-sensitive Learning:
Assign higher misclassification costs to the minority class during training.

b) Ensemble Methods:

- **Balanced Random Forest:** Random Forest with balanced bootstrap samples.
- **EasyEnsemble:** Ensemble of AdaBoost learners trained on different undersampled subsets.

c) Anomaly Detection:
Treat the minority class as anomalies and use anomaly detection algorithms.

d) Focal Loss:
A modified loss function that down-weights well-classified examples.

e) One-Class Classification:
Train the model only on the majority class and treat the minority class as outliers.

4. Typecasting & Duplicate Handling

4.1 Type Casting Techniques in Data Preprocessing

4.1.1 Definition and Importance of Typecasting

Typecasting is the process of converting a variable from one data type to another. It's crucial in data preprocessing for several reasons:

- Ensures data consistency across the dataset.
- Optimizes memory usage and computational efficiency.
- Enables proper functioning of algorithms that expect specific data types.
- Facilitates correct data manipulation and analysis.

4.1.2 Common Data Types and Conversions

a) Numeric Types:

- **Integer (int):** Whole numbers
- **Float:** Decimal numbers
- **Complex:** Numbers with real and imaginary parts

b) Text Types:

- **String (str):** Textual data

c) Boolean Type:

- **Bool:** True or False values

d) Date and Time Types:

- **Datetime:** Date and time information

Common conversions include:

- String to Integer/Float (e.g., "123" to 123)
- Float to Integer (e.g., 3.14 to 3)
- Integer/Float to String (e.g., 123 to "123")
- String to Datetime (e.g., "2023-05-01" to a datetime object)

4.1.3 Conversion Methods Using Pandas

Pandas provides several methods for type conversion:

```
a) astype():
df['column'] = df['column'].astype(int)

b) to_numeric():
df['column'] = pd.to_numeric(df['column'], errors='coerce')

c) to_datetime():
df['date'] = pd.to_datetime(df['date'])

d) apply():
df['column'] = df['column'].apply(lambda x: str(x))
```

4.2 Identifying and Handling Duplicates

4.2.1 Definition and Importance

Duplicate data refers to identical or very similar records that appear multiple times in a dataset. Handling duplicates is important because:

-
- It can skew statistical analyses and machine learning models.
- It wastes storage space and computational resources.
- It can lead to incorrect conclusions or decision-making based on inflated data.

4.2.2 Duplicate Detection

Methods for detecting duplicates include:

a) Exact Duplicates:
Identify rows that are identical across all columns.

b) Partial Duplicates:
Identify rows that are identical across a subset of columns.

c) Fuzzy Matching:
Identify rows that are very similar but not exactly identical, often used for text data.

Note: In Pandas, duplicates can be detected using the duplicated() method:
duplicates = df[df.duplicated()]

4.2.3 Removal Strategies

```
a) Drop Duplicates: Remove all but the first occurrence of duplicate rows.
df = df.drop_duplicates()

b) Keep Last: Remove all but the last occurrence of duplicate rows.
df = df.drop_duplicates(keep='last')

c) Subset-based Deduplication: Remove duplicates based on specific columns.
df = df.drop_duplicates(subset=['column1', 'column2'])

d) Aggregation:Instead of removing duplicates, aggregate them using a specific function.
df = df.groupby('id').agg({'value': 'mean'})

e) Manual Review:
For critical data or complex cases, manually review and decide on each duplicate.
```

When handling duplicates, it's important to understand the nature of the duplicates and the implications of their removal on the analysis. Sometimes, duplicates might represent genuine repeated events or measurements, and their removal could lead to loss of important information.

5. Handling Outliers

5.1 Definition of Outliers

Outliers are data points that significantly differ from the majority of observations in a dataset. These values can be much higher or lower than the other data points and are often considered anomalies or errors. For example, in a dataset of human heights, if most values are between 150 cm and 190 cm, a height of 250 cm would be considered an outlier.

5.2 Importance of Handling Outliers

Handling outliers is critical because they can skew statistical analyses and predictive models. For instance, in linear regression, outliers can significantly impact the slope of the line, leading to inaccurate predictions. In clustering algorithms like K-Means, outliers can distort the centroids, leading to incorrect groupings. Therefore, identifying and addressing outliers ensures the integrity and accuracy of data analysis and modeling.

5.3 Identifying Outliers

There are several methods to identify outliers:

1. Visual Methods:

- **Box Plot:** A box plot displays the distribution of data and highlights outliers as points outside the whiskers. Outliers are typically defined as values more than 1.5 times the interquartile range (IQR) above the third quartile or below the first quartile.
- **Scatter Plot:** A scatter plot can visually highlight outliers, especially in two-dimensional datasets.

2. Statistical Methods:

- **Z-Score:** The Z-score measures how many standard deviations a data point is from the mean. Data points with a Z-score greater than a threshold (usually 3 or -3) are considered outliers.
- **IQR Method:** The interquartile range (IQR) method defines outliers as values below Q1 - 1.5*IQR or above Q3 + 1.5*IQR, where Q1 and Q3 are the first and third quartiles, respectively.

3. Model-Based Methods:

- **Isolation Forest:** This method isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. Outliers are isolated faster, making them more likely to be outliers.
- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise):** DBSCAN identifies outliers as points that do not fit well into the clusters.

5.4 Handling Outliers

Once outliers are identified, several approaches can be taken to handle them:

1. Removing Outliers:

- **Direct Removal:** Simply removing outliers from the dataset is a straightforward approach, especially if they are due to data entry errors or are irrelevant to the analysis.

- **Filtering:** Apply thresholds or rules to remove outliers (e.g., filtering out values above or below certain thresholds).

2. Transforming Outliers:

- **Log Transformation:** Applying a logarithmic transformation can reduce the impact of extreme values.
- **Winsorization:** Replace outliers with the nearest non-outlier value (e.g., replace the smallest outlier with the minimum non-outlier value).

3. Imputation:

- **Replace with Mean/Median:** Replace outliers with the mean or median of the non-outlier data.
- **Impute using Machine Learning:** Use a predictive model to estimate and replace the outlier with a more reasonable value.

4. Flagging Outliers:

Create a Binary Feature: Add a new feature that flags data points as outliers. This allows the model to learn from the outliers rather than removing them.

6. Feature Scaling

6.1 Definition of Feature Scaling

Feature scaling is the process of adjusting the scale of features in a dataset to ensure they contribute equally to the analysis or model. In many algorithms, the range of input values can significantly impact the performance, leading to biased results if one feature dominates others due to its scale.

6.2 Importance of Feature Scaling

Feature scaling is crucial for the following reasons:

1. Algorithm Sensitivity: Many machine learning algorithms, such as gradient descent-based methods (e.g., linear regression, logistic regression, neural networks) and distance-based methods (e.g., K-Means clustering, K-Nearest Neighbors), are sensitive to the scale of input features.

2. Improved Convergence: Scaling can lead to faster convergence of optimization algorithms by ensuring that all features contribute equally.

3. Model Interpretability: Scaling can make model coefficients more interpretable, especially in linear models where the magnitude of coefficients reflects the importance of each feature.

6.3 Common Feature Scaling Techniques

1. Min-Max Scaling (Normalization):

- **Formula:** $y = (x - \min(x))/(\max(x) - \min(x))$
- **Range:** Scales the data to a fixed range, usually [0, 1].
- **Use Case:** Useful when the distribution of the data is not Gaussian or when the features need to be bounded.

2. Standardization (Z-score Normalization):

- **Formula**: $x' = x - \mu/\sigma$
- **Range:** Transforms the data to have a mean of 0 and a standard deviation of 1.
- **Use Case:** Suitable when the data follows a Gaussian distribution or when the algorithm assumes

standardized data, like in PCA (Principal Component Analysis).

3. Robust Scaling:

- **Formula:** X_scaled = (X — X_median) / IQR
- **Range:** Centers and scales data based on the median and IQR (interquartile range).
- **Use Case:** Effective for data with outliers, as it reduces the impact of outliers on the scaling process.

6.4 Advanced Scaling Techniques

1. MaxAbs Scaling:

- **Formula:** $(x' = x / \max(|x|))$
- **Range:** Scales each feature by its maximum absolute value, ensuring values are within the range [-1, 1].
- **Use Case:** Useful for data that is already centered at 0 and has positive and negative values.

2. Quantile Transformation:

- **Formula:** Transforms features to follow a uniform or normal distribution.
- **Use Case:** Effective for non-linear models and when the assumption of normality is required.

3. L2 Normalization:

Formula: $||x||_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$
Range: Scales the data such that the Euclidean norm (L2 norm) of the feature vector is 1.
Use Case: Often used in text classification (e.g., TF-IDF vectors) or when the focus is on the direction rather than the magnitude of vectors.

6.5 Choosing the Right Scaling Technique

The choice of scaling technique depends on the nature of the data and the algorithm being used:

- **Min-Max Scaling:** Best for algorithms where feature values need to be bounded, such as neural networks.
- **Standardization:** Preferred for algorithms that assume normally distributed data, such as PCA or linear models.
- **Robust Scaling:** Ideal for datasets with outliers.
- **Advanced Techniques:** Use cases where specific distribution assumptions or vector norms are required.

7. Feature Encoding

7.1 Definition and Importance of Feature Encoding

Feature encoding refers to the process of converting categorical variables into numerical formats that can be used by machine learning algorithms. Many algorithms require numerical input, and without encoding, categorical data cannot be directly used in models.

7.2 Encoding Techniques

1. Label Encoding:

- **Definition:** Assigns a unique integer to each category in the variable.
- **Example:** For a "color" feature with categories ["red", "green", "blue"], the encoding might be [0, 1, 2].
- **Use Case:** Useful when the categorical variable is ordinal, meaning there is an inherent order in the categories.

2. One-Hot Encoding:

- **Definition:** Creates a binary column for each category in the variable, where a 1 indicates the presence of the category and 0 indicates its absence.
- **Example:** For a "color" feature with categories ["red", "green", "blue"], one-hot encoding would create three columns: [1, 0, 0] for "red", [0, 1, 0] for "green", and [0, 0, 1] for "blue".
- **Use Case:** Suitable for nominal (non-ordinal) categorical variables with no intrinsic order.

3. Target Encoding (Mean Encoding):

- **Definition:** Replaces categories with the mean of the target variable for each category.
- **Example:** In a classification problem, if "color" is the categorical variable and the target variable is binary, target encoding would replace each color with the average target value for that color.
- **Use Case:** Useful for high cardinality categorical variables where one-hot encoding would create too many columns.

4. Binary Encoding:

Definition: Combines label encoding and one-hot encoding by first converting categories to integers and then to binary form, creating fewer columns than one-hot encoding.
Example: For a "color" feature with categories ["red", "green", "blue"], binary encoding might assign "red" as 001,

"green" as 010, and "blue" as 011.

- **Use Case:** Suitable for variables with a large number of categories.

5. Frequency Encoding:

- **Definition:** Encodes each category based on the frequency of its occurrence in the dataset.
- **Example:** If "red" appears 50 times, "green" 30 times, and "blue" 20 times, the encoding would replace "red" with 50, "green" with 30, and "blue" with 20.
- **Use Case:** Useful when the frequency of categories is relevant to the prediction.

7.3 Choosing the Right Encoding Technique

The choice of encoding technique depends on the nature of the categorical variable and the specific use case:

- **Label Encoding:** Best for ordinal data.
- **One-Hot Encoding:** Ideal for nominal data with a small number of categories.
- **Target Encoding:** Suitable for high cardinality features in scenarios where target leakage can be controlled.
- **Binary Encoding:** Useful for high cardinality variables where one-hot encoding is impractical.
- **Frequency Encoding:** Effective when the frequency of categories is informative.

8. Feature Extraction

8.1 Definition and Importance of Feature Extraction

Feature extraction involves transforming raw data into a set of features that can be used for machine learning. It reduces the dimensionality of the data while retaining its most informative aspects, improving the efficiency and accuracy of models. By focusing on the most relevant features, feature extraction enhances model performance and interpretability.

8.2 Feature Extraction Techniques

1. Principal Component Analysis (PCA):

- **Definition:** PCA is a dimensionality reduction technique that transforms the data into a set of orthogonal (uncorrelated) components, ordered by the amount of variance they explain.
- **Use Case:** Useful when dealing with high-dimensional data where many features are correlated.

2. Linear Discriminant Analysis (LDA):

- **Definition**: LDA is a supervised method that projects the data onto a lower-dimensional space by maximizing the separation between different classes.
- **Use Case:** Effective in classification problems where the goal is to maximize class separability.

3. t-Distributed Stochastic Neighbor Embedding (t-SNE):

- **Definition:** t-SNE is a non-linear dimensionality reduction technique that visualizes high-dimensional data by preserving the local structure in a low-dimensional space.
- **Use Case:** Ideal for visualizing complex datasets in two or three dimensions, often used for exploratory data analysis.

4. Autoencoders:

Definition: Autoencoders are neural networks used to learn efficient representations of data by compressing the input into a lower-dimensional space and then reconstructing the output.
Use Case: Suitable for unsupervised learning tasks where the goal is to learn a compressed representation of the data.

5. Feature Hashing:

- **Definition:** Feature hashing is a technique that converts categorical data into a fixed-size numerical feature vector using a hash function.
- **Use Case:** Useful in natural language processing (NLP) and when dealing with large datasets with high cardinality categorical features.

Feature extraction is a critical step in the data preprocessing pipeline, particularly when working with high-dimensional data. Choosing the right technique depends on the data type, the problem being solved, and the computational resources available.

9. Feature Selection

9.1 Definition and Importance of Feature Selection

Feature selection is the process of selecting a subset of relevant features for use in model construction. It helps in reducing the dimensionality of the data, improving model performance, reducing overfitting, and speeding up the training process. By selecting only the most important features, feature selection can enhance model interpretability and generalization.

9.2 Filter Methods for Feature Selection

1. Correlation Coefficient:

- **Definition:** Measures the statistical relationship between features and the target variable.
- **Example:** If a feature has a high correlation with the target, it may be selected.
- **Use Case:** Best for linear relationships and when you want to quickly eliminate irrelevant features.

2. Chi-Square Test:

- Definition: Evaluates the association between categorical features and the target variable.
- Example: Used in feature selection for categorical data, especially in classification tasks.
- Use Case: Effective for selecting features that have a strong dependency on the target variable.

3. Mutual Information:

- **Definition**: Measures the dependency between two variables. Unlike correlation, mutual information can capture non-linear relationships.
- **Example**: A high mutual information score indicates a strong relationship between a feature and the target variable.
- **Use Case:** Suitable when you want to account for both linear and non-linear relationships.

9.3 Wrapper Methods for Feature Selection

1. Forward Selection:

- **Definition:** Starts with no features and adds them one by one based on a specific criterion (e.g., model performance).
- **Use Case:** Useful when you have a smaller dataset and can afford the computational cost of evaluating multiple models.

2. Backward Elimination:

- **Definition:** Starts with all features and removes them one by one, based on the least significant contribution to the model.
- **Use Case:** Suitable when the dataset has many irrelevant features that need to be filtered out.

3. Recursive Feature Elimination (RFE):

Definition: Uses a model to select features by recursively removing the least important features and building the model again.
Use Case: Effective in reducing the feature set and improving model performance, especially when combined with cross-validation.

9.4 Embedded Methods for Feature Selection

1. Lasso Regression (L1 Regularization):

- **Definition:** A linear model that includes an L1 penalty, which can shrink some coefficients to zero, effectively selecting a subset of features.
- **Use Case:** Useful when you want a model that performs both feature selection and prediction simultaneously.

2. Ridge Regression (L2 Regularization):

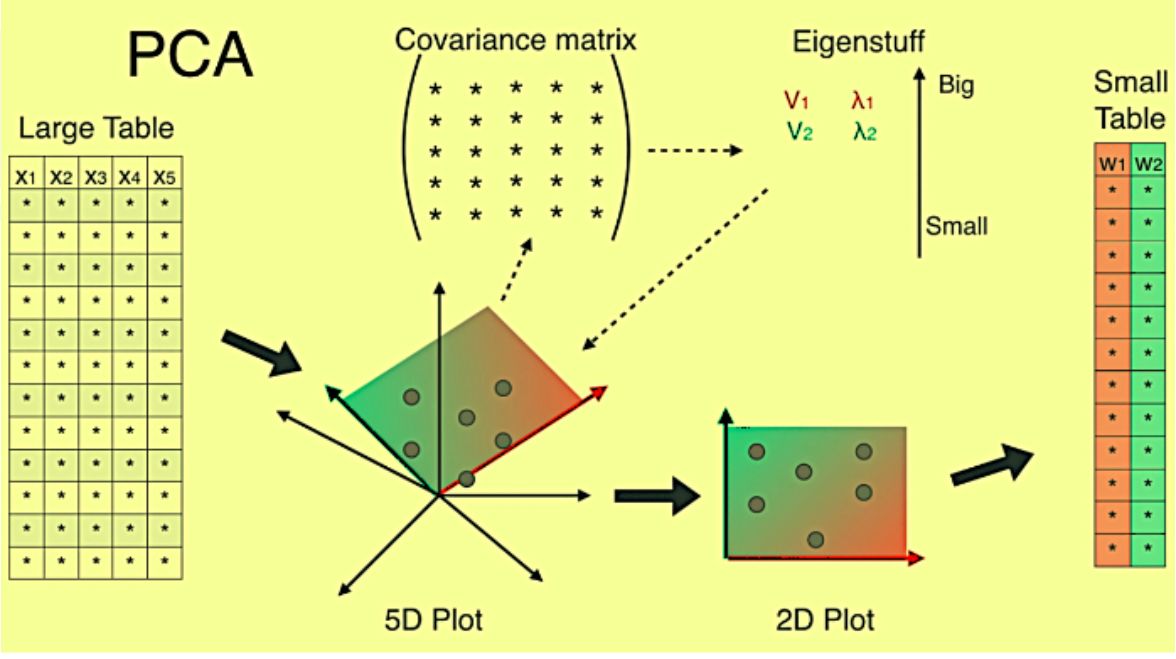
- **Definition:** Similar to Lasso but with an L2 penalty. It doesn't shrink coefficients to zero but reduces the impact of less important features.
- **Use Case:** Effective when all features are believed to be important but you want to reduce overfitting.

3. Tree-based Methods:

- **Definition:** Decision trees and ensemble methods like Random Forests and Gradient Boosting can be used to rank feature importance.
- **Use Case:** Ideal for datasets where features interact non-linearly, as these methods can capture complex relationships.

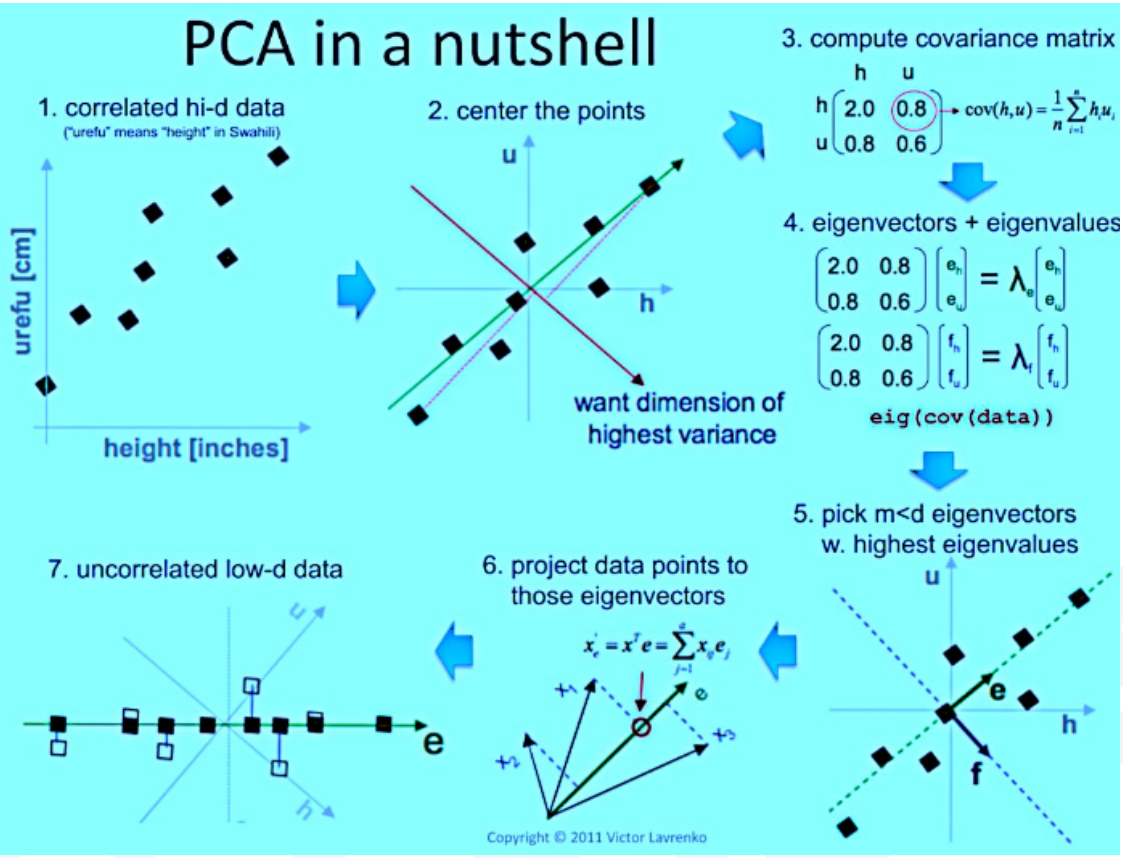
10. PCA (Principal Component Analysis)

10.1 Definition and Importance



Principal Component Analysis (PCA) is a dimensionality reduction technique that transforms data into a set of linearly uncorrelated variables called principal components. The principal components are ordered by the amount of variance they explain, with the first component capturing the most variance. PCA is important because it helps in reducing the dimensionality of large datasets while retaining the most critical information, thus improving the performance of machine learning models.

10.2 Steps in PCA



1. Standardize the Data:

Process: Convert the data into a standardized form with a mean of 0 and a standard deviation of 1. This ensures that each feature contributes equally to the analysis.

2. Compute the Covariance Matrix:

Process: Calculate the covariance matrix to understand how the features in the dataset vary together.

3. Calculate Eigenvectors and Eigenvalues:

Process: Eigenvectors determine the direction of the new feature space, and eigenvalues determine their magnitude. Together, they form the principal components.

4. Sort Eigenvectors by Eigenvalues:

Process: Rank the eigenvectors by the magnitude of their corresponding eigenvalues, selecting the top eigenvectors as the principal components.

5. Transform the Data:

Process: Project the data onto the new principal component axes to obtain a reduced-dimensional representation of the data.

10.3 Interpreting PCA Results

- **Variance Explained:** The proportion of the dataset's total variance that is captured by each principal component. Typically, the first few components capture most of the variance.
- **Scree Plot:** A plot of the eigenvalues that helps in deciding the number of components to retain. The "elbow point" in the plot indicates where adding more components yields diminishing returns.
- **Loadings:** The coefficients of the original features in the principal components. They indicate how much each feature contributes to a principal component.

PCA simplifies the data, making it easier to analyze and interpret, while also improving the performance of machine learning models by reducing noise and overfitting.

11. Covariance & Correlation

11.1 Definition of Covariance

Covariance is a statistical measure that indicates the extent to which two random variables change together. If the greater values of one variable correspond to the greater values of the other variable, and the same for lesser values, the covariance is positive. Conversely, if one variable tends to increase when the other decreases, the covariance is negative. The magnitude of covariance indicates the strength of the linear relationship between the variables.

Formula:

$$\text{Cov}(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{n-1}$$

Where:

- X_i and Y_i are the individual sample points for variables X and Y.
- $\bar{X}$ and $\bar{Y}$ are the means of variables X and Y.
- n is the number of data points.

Example:
If you're analyzing the relationship between temperature and ice cream sales, a positive covariance would suggest that as temperature increases, ice cream sales also increase.

11.2 Definition of Correlation

Correlation is a statistical measure that describes the strength and direction of a linear relationship between two variables. Unlike covariance, correlation is dimensionless and standardized, ranging from -1 to 1. A correlation of 1 implies a perfect positive linear relationship, -1 indicates a perfect negative linear relationship, and 0 suggests no linear relationship.

Formula:

$$\text{Correlation}(r) = \frac{\text{Cov}(X,Y)}{\sigma_X \sigma_Y}$$

Where:

- $\text{Cov}(X,Y)$ is the covariance of X and Y.
- σ_X and σ_Y are the standard deviations of X and Y.

Example:
Using the same example of temperature and ice cream sales, a correlation close to 1 would suggest a strong positive relationship between the two variables.

11.3 Differences between Covariance and Correlation

Scale:

- **Covariance:** The value of covariance is dependent on the units of the variables, making it difficult to interpret the strength of the relationship across different datasets.
- **Correlation:** Correlation standardized covariance, resulting in a dimensionless value between -1 and 1, which allows for easier comparison across different datasets.

Magnitude:

- **Covariance:** Provides information about the direction of the relationship but not the strength in a standardized form.
- **Correlation:** Provides both the direction and the strength of the relationship in a standardized form.

Interpretability:

- Covariance: Harder to interpret due to its dependency on units.
- Correlation: Easier to interpret as it provides a clear measure of the strength and direction of the relationship.

Sensitivity to Scale:

- **Covariance:** Sensitive to the scale of the variables.
- **Correlation:** Not affected by the scale, as it is a normalized measure.

12. Pearson Correlation Coefficient & Spearman's Rank Correlation

12.1 Pearson Correlation Coefficient

The Pearson Correlation Coefficient, denoted as (r), is a measure of the linear relationship between two continuous variables. It quantifies how strongly the variables are related and in what direction. Pearson correlation assumes that the relationship between variables is linear and that the data is normally distributed.

Formula

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

r = correlation coefficient
x_i = values of the x-variable in a sample
$\bar{x}$ = mean of the values of the x-variable
y_i = values of the y-variable in a sample
$\bar{y}$ = mean of the values of the y-variable

Example:
Consider the relationship between study hours and test scores. If the Pearson correlation coefficient is 0.85, it indicates a strong positive linear relationship, meaning that as study hours increase, test scores also tend to increase.

12.2 Spearman's Rank Correlation

Spearman's Rank Correlation is a non-parametric measure of the strength and direction of association between two ranked variables. Unlike Pearson correlation, Spearman's does not assume that the relationship between variables is linear, nor does it assume that the data is normally distributed. It assesses how well the relationship between two variables can be described using a monotonic function.

Formula:

Formula

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

ρ = Spearman's rank correlation coefficient
d_i = difference between the two ranks of each observation
n = number of observations

Example:
If you rank students based on their math and science scores, Spearman's rank correlation can show how well the ranks correspond, even if the relationship is not linear.

12.3 When to Use Each Method

Use Pearson Correlation:

- When both variables are continuous and normally distributed.
- When you assume or know that the relationship between the variables is linear.
- When you want to measure the strength and direction of a linear relationship.

Use Spearman's Rank Correlation:

- When the data is ordinal or not normally distributed.
- When the relationship between variables is monotonic but not necessarily linear.
- When you have outliers that might distort the Pearson correlation coefficient.

Example Scenario:

Pearson Correlation: Use when analyzing the relationship between height and weight in adults, assuming a linear relationship.

Spearman's Rank Correlation: Use when analyzing the relationship between customer satisfaction rankings and purchase frequency, where the relationship may not be linear.