Modelling and Evaluation

Evaluating Performance of a Model

1. **Importance of Model Evaluation**

- Ensures the model meets the desired accuracy and performance standards.

- Helps in understanding how well the model generalizes to new, unseen data.

- Provides insights into the strengths and weaknesses of the model.

2. Evaluation Metrics

- **Accuracy:** The ratio of correctly predicted observations to the total observations. Useful for balanced datasets.

  - Problem: In cases of imbalanced datasets, accuracy can be misleading.

  - Precision: The ratio of true positive predictions to the total predicted positives. Indicates the accuracy of the positive predictions.

  - Problem: High precision with low recall means the model misses many positives.

  - Recall (Sensitivity): The ratio of true positive predictions to the total actual positives. Indicates the model's ability to identify all relevant instances.

  - Problem: High recall with low precision means the model has many false positives.

  - F1 Score: The harmonic mean of precision and recall. Useful for imbalanced datasets.

  - Problem: Does not differentiate between precision and recall.

  - **ROC-AUC (Receiver Operating Characteristic - Area Under Curve):** Measures the model's ability to distinguish between classes. Higher AUC indicates a better model.

  - Problem: AUC can be over-optimistic if there is a significant class imbalance.

3. Confusion Matrix

- A table used to describe the performance of a classification model.

  - True Positives (TP): Correctly predicted positive instances.

  - True Negatives (TN): Correctly predicted negative instances.

  - False Positives (FP): Incorrectly predicted positive instances (Type I error).

  - False Negatives (FN): Incorrectly predicted negative instances (Type II error).

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Improving Performance of a Model

1. Data Preprocessing

- Handling Missing Values: Use imputation methods or remove missing data points.

- Feature Scaling: Standardize or normalize features to improve model performance.

- Feature Engineering: Create new features or transform existing ones to capture more information.

2. Model Tuning

- **Hyperparameter Tuning:** Use techniques like Grid Search, Random Search, or Bayesian Optimization to find the best hyperparameters.

- Cross-Validation: Use k-fold cross-validation to ensure the model generalizes well to unseen data.

3. Advanced Techniques

- Ensemble Methods: Combine multiple models to improve performance (e.g., Bagging, Boosting, Stacking).

- Regularization: Add penalties to the model to avoid overfitting (e.g., L1, L2 regularization).

- Neural Networks: Use deep learning techniques for complex tasks where traditional models fall short.

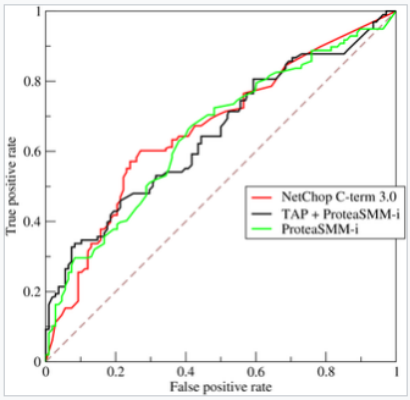
4. Addressing Overfitting and Underfitting

- Overfitting: When the model performs well on training data but poorly on test data.

  - Solutions: Simplify the model, use more training data, apply regularization.

- Underfitting: When the model performs poorly on both training and test data.

- Solutions: Increase model complexity, improve feature selection, remove noise from the data.



ROC curve of three predictors of peptide cleaving in the proteasome.

Basics of Feature Engineering

Introduction

Feature Engineering is the process of using domain knowledge to select, modify, or create features that make machine learning algorithms work better. It is a crucial step in the data preprocessing phase of machine learning and can significantly influence the performance of models.

Key Steps:

1. Understanding Data: In-depth understanding of the dataset, including its structure, types of variables, and underlying patterns.

2. Cleaning Data: Handling missing values, outliers, and inconsistencies in the data.

3. Creating New Features: Generating new features that capture hidden patterns and interactions in the data.

Feature Transformation

1. Definition:

- Transforming existing features into new forms that are more suitable for machine learning models.

2. Techniques:

- Scaling: Normalizing features to a standard range (e.g., 0-1) or standard deviation. Common methods include Min-Max scaling and Standardization.

  - Min-Max Scaling: $X' = \frac{X - X_{min}}{X_{max} - X_{min}}$
  - Standardization: $X' = \frac{X - \mu}{\sigma}$

-

- Log Transformation: Applying logarithmic transformation to reduce skewness in data.

  - Useful for features with a long tail or exponential growth.

    - Useful for features with a long tail or exponential growth.
    - Formula: $X' = \log(X + 1)$

-

- Box-Cox Transformation: A family of power transformations to stabilize variance and make the data more normally distributed.

- Encoding Categorical Variables:
  - One-Hot Encoding: Converts categorical variables into a binary matrix.
  - Label Encoding: Assigns each unique category a numerical label.

- Polynomial Features: Generating new features by combining existing features (e.g., squared terms, interaction terms).
  - Formula: $X_1^2, X_1 \cdot X_2, \ldots$

- Binning: Grouping continuous variables into discrete bins.
  - Useful for reducing the impact of outliers.

3. Practical Example:

- Original Feature: `Income`

- Transformed Feature (Log Transformation): `Log_Income = \log(Income + 1)`

Feature Subset Selection

1. Definition:

- Selecting a subset of relevant features to use in model construction.

2. Importance:

- Reduces the complexity of the model.

- Improves model performance by removing irrelevant or redundant features.

- Enhances generalization by reducing overfitting.

3. Techniques:

- Filter Methods:
  - Select features based on their statistical properties.

- Examples: Correlation, Chi-Squared test, ANOVA.

- Wrapper Methods:

  - Use a predictive model to evaluate the combination of features and select the best subset.

  - Examples: Recursive Feature Elimination (RFE), Forward Selection, Backward Elimination.

- Embedded Methods:

  - Perform feature selection as part of the model training process.

  - Examples: Lasso (L1 regularization), Ridge (L2 regularization), Tree-based methods (e.g., Random Forest, Gradient Boosting).

4. Practical Example:

- Using Recursive Feature Elimination (RFE) with a logistic regression model to select the top 10 features.

1. One-Hot Encoding Example:

Original Feature	One-Hot Encoded Features
Color	Color_Red
Red	1
Blue	0
Green	0

2. Polynomial Features Example:

Original Features	Polynomial Features
X1	X2
2	3
1	4

Summary:

Feature Engineering is a fundamental aspect of the machine learning pipeline that involves transforming and selecting features to improve model performance. It includes techniques like scaling, encoding, polynomial features, and selection methods such as filter, wrapper, and embedded methods. Proper feature engineering can lead to more accurate, efficient, and interpretable models.

Learning Problems and Concept Learning

Introduction

Concept Learning is the task of inferring a Boolean-valued function from training examples. In simpler terms, it involves identifying a general concept or rule from specific examples. This is fundamental in machine learning as it forms the basis for classification tasks.

Find-S Algorithm

1. Definition:

- Find-S (Find-Specific) is an algorithm used to find the most specific hypothesis that fits all positive examples in the training data.

2. Process:

- Initialize the hypothesis `h` to the most specific hypothesis (i.e., all attributes are set to the most specific value).

- For each positive example in the training data, update `h` to be the most specific hypothesis that covers the example.

- Ignore negative examples.

3. Steps:

- Start with the most specific hypothesis: h = `⟨Φ, Φ, ..., Φ⟩`.

- For each positive training instance x:

  - For each attribute constraint a_i in h:

    - If the constraint a_i in h is satisfied by x_i, do nothing.

    - If the constraint a_i in h is not satisfied by x_i, replace a_i by the next more general constraint that is satisfied by x_i.

- Output the hypothesis `h`.

4. Example:

Instance	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySpor
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

Initial Hypothesis:

`h = ⟨Φ, Φ, Φ, Φ, Φ, Φ⟩`

After Instance 1:

`h = ⟨Sunny, Warm, Normal, Strong, Warm, Same⟩`

After Instance 2:

`h = ⟨Sunny, Warm, ?, Strong, Warm, Same⟩`

After Instance 4:

`h = ⟨Sunny, Warm, ?, Strong, ?, ?⟩`

Version Space and Candidate Elimination Algorithm

1. Definition:

- The Version Space represents the set of all hypotheses that are consistent with the training examples.

- The Candidate Elimination Algorithm systematically narrows down the Version Space by considering both positive and negative examples.

2. Version Space Representation:

- The Version Space is represented by its boundary sets:

  - G (General boundary): The set of the most general hypotheses.

- S (Specific boundary): The set of the most specific hypotheses.

3. Candidate Elimination Algorithm Steps:

- Initialize `S` to the most specific hypothesis.

- Initialize `G` to the most general hypothesis.

- For each training example:

  - If the example is positive:

    - Remove from `G` any hypothesis that does not cover the example.

    - For each hypothesis `s` in `S` that does not cover the example, remove `s` from `S`. Add to `S` all minimal generalizations `h` of `s` such that `h` covers the example and some member of `G` is more general than `h`. Remove from `S` any hypothesis that is more general than another hypothesis in `S`.

  - If the example is negative:

    - Remove from `S` any hypothesis that covers the example.

    - For each hypothesis `g` in `G` that covers the example, remove `g` from `G`. Add to `G` all minimal specializations `h` of `g` such that `h` does not cover the example and some member of `S` is more specific than `h`. Remove from `G` any hypothesis that is less general than another hypothesis in `G`.

4. Example:

Instance	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySpor
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

Initial S and G:

- `S = ⟨Φ, Φ, Φ, Φ, Φ, Φ⟩`

- `G = ⟨?, ?, ?, ?, ?, ?⟩`

After Instance 1 (Positive):

- `S = ⟨Sunny, Warm, Normal, Strong, Warm, Same⟩`
- `G` remains unchanged.

After Instance 2 (Positive):

- `S = ⟨Sunny, Warm, ?, Strong, Warm, Same⟩`
- `G` remains unchanged.

After Instance 3 (Negative):

- `S` remains unchanged.
- `G` is updated to remove hypotheses that cover negative instance:
  - `G = ⟨Sunny, ?, ?, ?, ?, ?⟩`, `⟨?, Warm, ?, ?, ?, ?⟩`, ...

After Instance 4 (Positive):

- `S = ⟨Sunny, Warm, ?, Strong, ?, ?⟩`
- `G` is updated to include more general hypotheses that are consistent with the new positive instance:
  - `G = ⟨Sunny, ?, ?, Strong, ?, ?⟩`, ...

Summary

- Find-S Algorithm: Finds the most specific hypothesis that fits all positive examples.
- Version Space: The set of all hypotheses consistent with the training examples, bounded by the most general (G) and the most specific (S) hypotheses.
- Candidate Elimination Algorithm: Systematically narrows the Version Space by considering both positive and negative examples to maintain consistency with all observed data.

These methods provide a foundational understanding of how machine learning models can learn concepts from examples and refine their hypotheses based on new data.

Bayesian Concept Learning

Why Bayesian Methods are Important

1. Foundations of Probability:

- Bayesian methods provide a formal framework for incorporating prior knowledge and updating beliefs in light of new evidence.

- They offer a principled way to handle uncertainty and make probabilistic inferences.

2. Flexibility:

- Can be applied to various types of data and models, making them versatile across different domains and applications.

3. Interpretability:

- Bayesian methods produce probabilistic outputs, which can be interpreted as degrees of belief or confidence levels.

- They help in understanding the uncertainty associated with predictions.

4. Incorporation of Prior Knowledge:

- Ability to incorporate prior information about the problem domain, which can be crucial when data is scarce or noisy.

5. Robustness:

- Bayesian methods are less prone to overfitting compared to frequentist methods, especially when dealing with small datasets.

Bayes' Theorem

1. Definition:

- Bayes' Theorem describes the probability of an event, based on prior knowledge of conditions that might be related to the event.

2. Formula:

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)}$$

Where:

- $P(H|E)$ is the posterior probability: the probability of hypothesis H given evidence E.
- $P(E|H)$ is the likelihood: the probability of evidence E given that H is true.
- $P(H)$ is the prior probability: the initial probability of hypothesis H.
- $P(E)$ is the marginal likelihood: the total probability of evidence E.

Bayes' Theorem and Concept Learning

1. Bayesian Concept Learning:

- In Bayesian concept learning, we aim to find the hypothesis \(H \) that maximizes the posterior probability given the training data \(D \).

- This approach evaluates hypotheses probabilistically and updates their probabilities as more data becomes available.

2. Posterior Probability Calculation:

$$P(H|D) = \frac{P(D|H) \cdot P(H)}{P(D)}$$

- $P(H|D)$: Posterior probability of hypothesis H given data D.
- $P(D|H)$: Likelihood of data D given hypothesis H.
- $P(H)$: Prior probability of hypothesis H.
- $P(D)$: Evidence or marginal likelihood of data D.

3. MAP (Maximum A Posteriori) Hypothesis:

$$H_{MAP} = \arg\max_H P(H|D)$$

-

4. Applications:

- Bayesian methods are applied in various fields such as spam filtering, medical diagnosis, and machine learning due to their ability to update beliefs and handle uncertainty.

Applications of Naïve Bayes Classifier

1. Definition:

- The Naïve Bayes classifier is a simple probabilistic classifier based on Bayes' Theorem, assuming strong (naïve) independence between the features.

2. Types:

Multinomial Naïve Bayes: Used for discrete data, like text classification.

Gaussian Naïve Bayes: Used for continuous data, assuming Gaussian distribution.

Bernoulli Naïve Bayes: Used for binary/boolean features.

3. Advantages:

Scalability: Efficient in terms of both storage space and computational time.

Simplicity: Easy to implement and interpret.

Performance: Performs well even with small amounts of data and can handle high-dimensional data.

4. Steps in Naïve Bayes Classification:

Training Phase:

1. Calculate the prior probability for each class.
2. Calculate the likelihood of each feature given each class.

Prediction Phase:

1. Use Bayes' Theorem to compute the posterior probability for each class.
2. Assign the class with the highest posterior probability to the instance.

5. Formula:

$$P(C_k|x) = \frac{P(x|C_k) \cdot P(C_k)}{P(x)}$$

Where:

- $P(C_k|x)$: Posterior probability of class C_k given instance x.
- $P(x|C_k)$: Likelihood of instance x given class C_k.
- $P(C_k)$: Prior probability of class C_k.
- $P(x)$: Evidence or marginal likelihood of instance x.

6. Applications:

Text Classification: Spam detection, sentiment analysis, document categorization.

Medical Diagnosis: Predicting diseases based on symptoms.

Market Basket Analysis: Understanding consumer buying patterns.

Example:

Text Classification with Multinomial Naïve Bayes:

- Training Data: A collection of emails labeled as spam or not spam.

- Features: Words occurring in the emails.

- Objective: Classify a new email as spam or not spam based on the words it contains.

Steps:

1. Calculate the prior probabilities \(P(spam) \) and \(P(not_spam) \).

2. For each word, calculate the likelihood \(P(word|spam) \) and \(P(word|not_spam) \).

3. For a new email, use these probabilities to calculate the posterior probability of the email being spam or not spam.

4. Classify the email based on the higher posterior probability.

Summary

Bayesian Methods: Provide a probabilistic framework for learning and inference, incorporating prior knowledge and updating beliefs.

Bayes' Theorem: Fundamental to Bayesian methods, allowing the computation of posterior probabilities.

Bayesian Concept Learning: Uses Bayes' Theorem to evaluate and update hypotheses.

Naïve Bayes Classifier: A simple yet powerful probabilistic classifier with applications in text classification, medical diagnosis, and more.

Bayesian methods, with their ability to handle uncertainty and incorporate prior knowledge, are essential tools in the machine learning toolbox, providing robust and interpretable models.