Tematy które pojawiły się na kolokwium polskiej infy:

1. Bayes

Bayes' Theorem:

- Bayes' Theorem provides a probabilistic framework to update beliefs based on new evidence.
  - $P(A\mid B) = \frac{P(B\mid A)P(A)}{P(B)}$
    - P(A|B): Probability of A given B (posterior probability).
    - P(B|A): Probability of B given A.
    - P(A): Prior probability of A.
    - P(B): Total probability of B.

Importance in Data Analysis:

- **Predictive Modeling**: Used to compute probabilities for classification problems.
- **Dynamic Updates**: Allows for updating probabilities dynamically as new data becomes available.
- **Foundation for Naïve Bayes**: Forms the core of Naïve Bayes classifiers, which assume independence between predictors.

Applications:

- Spam filtering: Classify emails as spam or not based on word occurrences.
- Medical diagnosis: Determine the likelihood of diseases based on observed symptoms.
- Sentiment analysis: Identify sentiments (positive, negative, neutral) in text data.

Naïve Bayes Classifier:

- Assumes independence between features.
- Simplifies computation by treating each feature's contribution as independent.
- Particularly effective for large datasets.

Steps in Bayesian Inference:

- Start with a **prior** probability.
- Collect evidence (data) to calculate the **likelihood**.
- Compute the **posterior** probability using Bayes' Theorem.
- Update the model iteratively as new data is collected.

Advantages:

- Computationally efficient.
- Easy to interpret results.
- Works well even with small datasets.

Limitations:

- Assumes feature independence (may not hold true in many real-world cases).
- Performance can degrade if the assumption of independence is heavily violated.

---

2. XAI - eXplainable Artificial Intelligence

Explainable Artificial Intelligence (XAI) refers to techniques and methods that make the outcomes and operations of AI models understandable to humans. It aims to address the "black-box" nature of many machine learning models, particularly complex ones like deep learning.

Key Objectives of XAI:

1. **Transparency**: Provide insights into how a model processes data and makes decisions.
2. **Trust**: Enhance user confidence in AI systems by explaining their predictions.
3. **Accountability**: Ensure AI decisions can be traced and justified, critical for ethical and regulatory compliance.
4. **Improved Model Performance**: Help developers identify biases and errors in models.

Methods of XAI:

1. **Feature Importance**:
   - Identifying which input features most influence the model's predictions.
   - Tools like SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-Agnostic Explanations) are commonly used.
2. **Visualization Techniques**:
   - Heatmaps, decision trees, or attention mechanisms highlight how the model interprets data.
3. **Simplified Models**:
   - Using interpretable models (e.g., decision trees, linear regression) alongside complex models to approximate and explain their behavior.
4. **Counterfactual Explanations**:
   - Providing scenarios that show how changes to inputs could alter the output.

Importance of XAI in Applications:

- **Healthcare**: Justify diagnoses or treatment recommendations.
- **Finance**: Explain credit scoring or loan approval decisions.
- **Legal and Regulatory Compliance**: Meet requirements for AI accountability and ethical use.

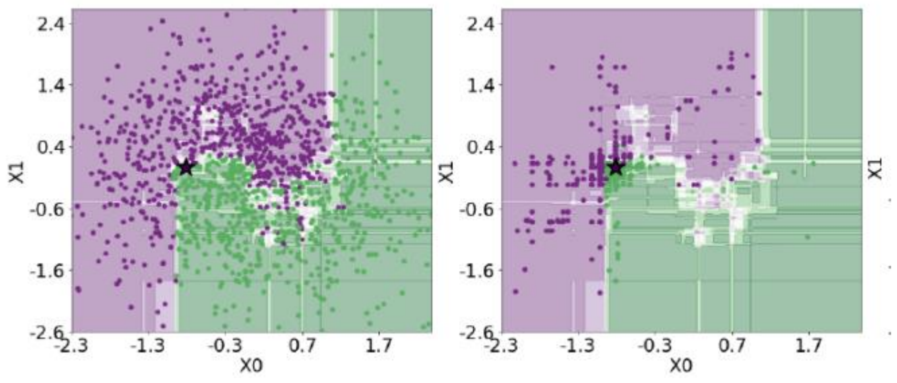
---

3. Lore i lime w python

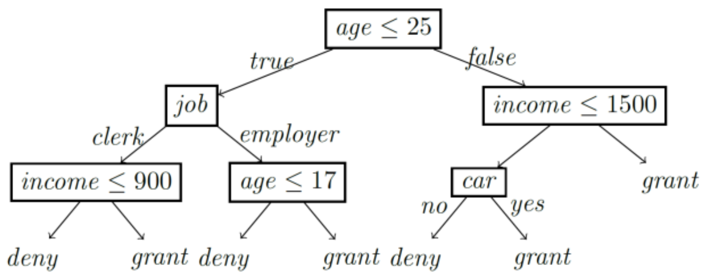
Both **LIME** and **LORE** are critical tools in Explainable AI (XAI), enhancing the interpretability of complex models by focusing on specific predictions. They help build trust in AI systems by making decisions more transparent and understandable.

LORE - (LOcal Rule-based Explanation) – counterfactual explanation

- **Purpose**: Generates rule-based explanations for predictions, providing interpretable and concise insights.
- **How It Works**:
    - Creates synthetic data around the instance to understand local decision boundaries.
    - Constructs rules that approximate the behavior of the model in that locality.
- **Advantages**:
    - Provides rule-based explanations, which are easier to understand for non-expert users.
    - Useful for understanding decisions in classification tasks.
- **Applications**:
    - Common in scenarios requiring interpretable decision-making, such as fraud detection or compliance monitoring.
- Generation of examples from the neighbourhood
    - Application of genetic algorithm
    - Considerably denser neighborhood comparing to uniform random generation



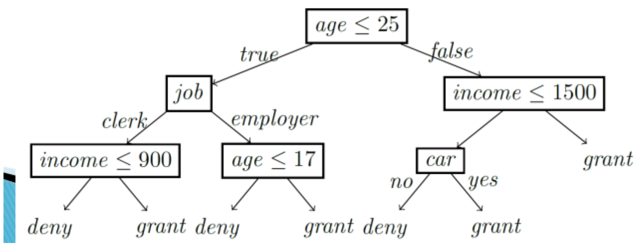
- Local classifier generation
    - Generation of a decision tree
    - Generated example set used for training
    - Local approximation of the black box classifier



- Explanation extraction
    - Extraction of rules from the decision tree

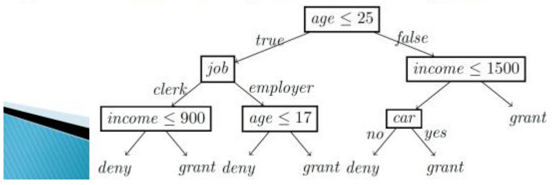
- Factual rule is derived from the path corresponding to the decision

age $\leq$ 25 and job = clerk and income $\leq$ 900 $\Rightarrow$ denny



- Counterfactual rules are derived from the paths leading to the opposit decision

age $\leq$ 25 and job = clerk and income > 900 $\Rightarrow$ grant
17 < age $\leq$ 25 and job = employer $\Rightarrow$ grant
age > 25 and income $\leq$ 1500 and car = yes $\Rightarrow$ grant
age > 25 and income > 1500 $\Rightarrow$ grant



LIME (Local Interpretable Model-Agnostic Explanations):

- **Purpose**: Provides local explanations for individual predictions of any machine learning model.
- **How It Works**:
  - Perturbs the input data by generating synthetic samples around the data point of interest.
  - Fits an interpretable surrogate model (e.g., linear regression) to approximate the local behavior of the complex model.
- **Advantages**:
  - Model-agnostic: Can be applied to any black-box model.
  - Explains specific predictions, making it user-friendly and interpretable.
- **Applications**:
  - Used in domains like healthcare, finance, and natural language processing to explain individual decisions.

---

4. Reguła asocjacyjna A => B jak to inaczej zapisać i czego wymaga

An association rule A⇒B is used in data mining to identify relationships between items in a dataset. The rule suggests that when itemset A occurs in a transaction, itemset B is likely to occur as well.

Formal Representation:

- $A, B \subseteq I$ (both A and B are subsets of the set of all items I).
- $A \cap B = \emptyset$ (the sets A and B are disjoint).
- $A \cup B \subseteq T$ (the union of A and B is a subset of transaction T).

Requirements:

- **Support** (s): Measures how often the itemsets A∪B occur together in the dataset. It is expressed as $s(A \Rightarrow B) = P(A \cup B)$, where P is the probability.
- **Confidence** (c): Reflects the likelihood that B occurs when A is present. It is calculated as $c(A \Rightarrow B) = P(B|A) = \frac{f(A \cup B)}{f(A)}$, where f denotes the frequency of an itemset.

Alternate Notation: The rule A ⇒ B can also be written using logical expressions, such as:

- buys(A)∧buys(B)
- Conditional probability P(B|A)

Conditions for Use:

- The rule must meet **minimum support** and **confidence thresholds** set by the user.
- The itemsets involved should be **frequent**, i.e., appear sufficiently often in the dataset.

---

5. Apriori

- Classic algorithm for searching frequent itemsets
- One-dimensional, single-level, Boolean association rules are generated basing on the found itemsets
- The algorithm performs iterative search using prior knowledge (apriori)

Apriori property:

- Each non-empty subset of a frequent set is a frequent set
- Adding a new element A to set /, which is not frequent, will not make set / ∪A more frequent than /

Basic steps of the algorithm:

- Iteratively perform merging and pruning
- Merging:
  - A set of frequent itmsets L_{k-1} is given
  - In order to generate the set L_k , the itemsets differing in one element are merged, e.g. $\{a, b, c\} \bowtie \{a, b, d\} \rightarrow \{a,b,c,d\}$
  - A set of candidates Ck is being created
- Pruning
  - The set C_k may contain itemsets that are not frequent ($L_k \subseteq C_k$)
  - Checking if the candidate is a frequent itemset consists in checking whether its k-1 element subsets are in L_{k-1}
  - After deleting the sets that are not frequent, L_k is received

6. Tp/tp+fp – co to za wzór

Assessment of exploration results – prognostic models

- Binary classification
- Nomenclature (medically inspired)
  - TP (true positive)
  - FN (false negative)
  - TN (true negative)
  - FP (false positive)
- Additional measures of assessment of the selected class recognition
  - sensitivity = TP / (TP+FN)
  - specificity = TN / (FP+TN)
- Other measures
  - true-positive rate
    - TPR = TP / (TP+FN) = sensitivity
  - false-positive rate
    - FPR = FP / (FP+TN) = 1 – specificity
- More measures
  - Positive predictive values
    - PPV = TP / (TP+FP)
  - Negative predive values
    - NPV = TN / (FN+TN)
  - Overall classification accuracy
    - $Acc = \frac{TP+TN}{TP+TN+FP+FN}$
  - Sum of sensitivity and specificity (class gain, Youden's statistics)
    - SSS = sens + spec
  - G-mean
    - $Gmean = \sqrt{sens * spec}$
  - F-measure
    - $Fmeasure = \frac{2*PPV*sens}{PPV+sens}$

---

7. Libraries for data analysis

NumPy:

- A foundational library for numerical computing in Python.
- Key functionalities:
  - Handling vectors, arrays, and multidimensional arrays (tensors).
  - Performing fast arithmetic operations and linear algebra.
- Often used as the basis for other libraries, such as pandas and SciPy.

Pandas:

- Designed for working with tabular data (DataFrame) and one-dimensional series (Series).

- Provides powerful tools for data manipulation and analysis.

SciPy:

- Focused on scientific computing tasks such as statistics, optimization, and signal processing.
- Built on top of NumPy.

Scikit-learn:

- A library for machine learning, covering tasks such as classification, regression, clustering, and dimensionality reduction.

TensorFlow, Keras, and PyTorch:

- Libraries for building and training neural networks:
  - **TensorFlow**: A comprehensive framework for deep learning.
  - **Keras**: A high-level API built on TensorFlow for simpler model building.
  - **PyTorch**: A flexible, low-level API offering more control but requiring more code.

Matplotlib:

- One of the oldest libraries for data visualization.
- Provides extensive customization options for creating various types of charts (line, bar, histogram, etc.).

Seaborn:

- Built on top of Matplotlib, offering more aesthetically pleasing and advanced visualizations.
- Simplifies the creation of composite charts like heatmaps and pair plots.

Plotly:

- A library for creating interactive visualizations, often used for web-based applications and dashboards.

NOTATKI Z WYKŁADÓW

LECTURE 1 Introduction to the subject of data analysis and mining

Definitions:

- **Data Mining**: The process of automatically discovering non-trivial, unknown, potentially useful regularities (rules, patterns, trends) in data.
- **Goal**: To better understand the data and the fragment of reality it describes.
- **Data Sources**: Databases, data warehouses, distributed repositories, and stream data sources.
- **Applications**: Banking, medicine, industry, retail, telecom, bioinformatics, etc.

Big Data vs. Classic Data:

- **Big Data**: Large-scale datasets such as millions of transactions in hypermarkets or bioinformatics datasets.
- **Classic Data**: Smaller datasets used for moderate-sized problems like disease diagnostics or targeted marketing.

What Data Mining Is and Isn’t:

- **Not Data Mining**: Queries like "How many packages of product X were sold?"
- **Data Mining**: Complex tasks such as predicting machine failures or customer behavior trends.

Reasoning in Data Mining:

- **Deductive Reasoning**: Based on known knowledge; does not generate new knowledge.
- **Inductive Reasoning**: Generalizes facts to create new knowledge; used in learning algorithms.

2. Machine Learning and Computational Intelligence

Machine Learning (ML):

- **Definition**: Constructing and refining the representation of facts based on experience.
- **Key Concept**: External data is used to improve performance on similar data in the future.

Computational Intelligence (CI):

- **Definition**: A branch of computer science dealing with complex problems not easily modeled by traditional algorithms.
- **Applications**: Image analysis, sentiment analysis, autonomous vehicles, etc.

3. Methodologies for Data Mining

General Process:

1. **Data Preparation**: 70-80% of the time, including cleaning, standardization, and transformation.
2. **Modeling**: Building AI/ML models.
3. **Evaluation and Interpretation**: Assessing model quality and insights.
4. **Deployment**: Applying results iteratively or cyclically.

SEMMA Methodology:

1. **Sample**: Data selection.
2. **Explore**: Discover relationships and visualize data.
3. **Modify**: Improve data quality (handling outliers, normalization).
4. **Model**: Build models.
5. **Assess**: Evaluate results.

CRISP-DM Methodology:

1. **Business Understanding**: Define goals and success criteria.
    - Identify key objectives and desired outputs.
    - Establish clear success metrics for the overall project.
2. **Data Understanding**: Examine data quality and features.
    - Identify outliers, missing data, and imbalanced datasets.
    - Use exploratory visualizations like histograms or box plots.
3. **Data Preparation**: Deduplicate, transform, and visualize data.
    - Standardize and normalize features.
    - Handle missing values using techniques such as imputation or removal.
    - Aggregate or reduce dimensions if necessary.
4. **Modeling**: Choose and apply modeling techniques.
    - Select models tailored to classification, regression, or clustering tasks.
    - Iterate with parameter tuning and optimization.
5. **Evaluation**: Assess model quality.
    - Use metrics like accuracy, precision, recall, or RMSE based on task type.
    - Validate using train-test splits or cross-validation.
    - Interpret results with respect to business goals.
6. **Deployment**: Determine how results are used.
    - Plan for ongoing monitoring and maintenance.

  - Establish workflows for model updates and retraining.

4. Association Rules in Data Mining

Definitions:

- **Association Rule**: A ⇒ B, where A and B are itemsets with no overlap.
- **Support**: s(A ⇒ B) = P(A ∪ B) – the frequency of itemsets.
- **Confidence**: c(A ⇒ B) = P(B|A) = s(A ∪ B) / s(A).

Types of Rules:

1. **By Value**:
   - Boolean: Buys(bread) ∧ Buys(butter) ⇒ Buys(milk).
   - Quantitative: Age[30,39] ∧ Income[30,45] ⇒ Buys(TV).
2. **By Data Dimension**:
   - One-dimensional or Multi-dimensional.
3. **By Abstraction Level**:
   - Single-level or Multi-level.

Frequent Pattern Mining:

- **Frequent Itemsets**: Sets of items that appear together in a dataset frequently enough to meet a minimum support threshold.
- **Strong Association Rules**: Association rules that meet both minimum support and confidence thresholds.
- **Correlation and Pattern Analysis**: Extends association rule mining to detect correlations and maximal patterns.
- **Closed Itemsets**: Itemsets where no superset has the same support.

Apriori Algorithm:

- **Goal**: Find frequent itemsets and generate strong rules.
- **Principle**: If an itemset is frequent, all its subsets are frequent.

Steps:

1. **Merging**: Combine itemsets differing by one element.
2. **Pruning**: Remove non-frequent itemsets.
3. **Repeat** until no new frequent itemsets are found.

Enhancements to Apriori:

- **Hash Tables**: To efficiently count itemset frequencies.

- **Transaction Reduction**: Remove transactions that do not contain frequent items to reduce computation.
- **Sampling**: Analyze smaller data subsets to estimate patterns.

Example:

- **Dataset**: Transactions with items like bread, milk, and butter.
- **Support Count**: Calculate frequency of each itemset.
- **Generate Rules**: For example, {bread, milk} ⇒ {butter} with confidence calculated as f(bread ∧ milk ∧ butter) / f(bread ∧ milk).

LECTURE 2 Introduction to the Python language for data analysis

1. Overview of Python

Key Features:

- **General-Purpose**: Suitable for various applications including data analysis, web development, and automation.
- **Open Source**: Free and widely supported.
- **Cross-Platform**: Compatible with multiple operating systems.
- **Object-Oriented and Functional**: Supports both paradigms.
- **Dynamic Typing**: Variables do not require explicit declaration.
- **Interpreted Language**: Executes code line-by-line, aiding debugging.
- **Extensive Libraries**: Includes libraries for data analysis, machine learning, and visualization.

Popularity:

- Python ranks highly on the TIOBE index due to its versatility and robust community support.

Development Environments:

- **Local**:
  - Python.org
  - Anaconda
- **Cloud**:
  - Google Colab
  - DataCamp DataLab
- **Integrated Development Environments (IDEs)**:
  - VS Code
  - Jupyter Notebook
  - PyCharm

2. Python Basics

Comments:

- **Single-Line**: # This is a comment
- **Multi-Line**: """ This is a multi-line comment """

Basic Data Types:

- **Numeric**: int, float, complex.
- **Boolean**: True, False.
- **Strings**: "Hello" or 'World'.
- **None**: Represents null values.

Operators:

- **Mathematical**: +, -, *, /, //, %, **.
- **Logical**: and, or, not.
- **Comparison**: ==, !=, >, <, >=, <=.
- **Identity**: is, is not.

Key Constructs:

- **Variables**: Defined using = (e.g., x = 10).
- **Strings**: Support indexing, slicing, and formatting (e.g., f"Hello {name}").

3. Data Structures

Lists:

- **Definition**: Mutable sequence of elements.
- **Common Operations**:
    - Add: li.append(4)
    - Remove: li.pop()
    - Slicing: li[1:3]

Tuples:

- **Definition**: Immutable sequence of elements.
- **Unpacking**: a, b, c = (1, 2, 3).

Dictionaries:

- **Definition**: Key-value pairs.
- **Access**: dict["key"].
- **Methods**: keys(), values().

Sets:

- **Definition**: Collection of unique elements.
- **Operations**: Union (|), Intersection (&), Difference (-).

4. Control Flow

Conditional Statements:

```
if x > 10:
    print("Greater")
elif x == 10:
    print("Equal")
else:
    print("Smaller")
```

Loops:

- **For Loop**:

```
for i in range(5):
    print(i)
```

- **While Loop**:

```
x = 0
while x < 5:
    x += 1
    print(x)
```

5. Functions

Definition:

```
def add(a, b):
    return a + b
```

Calling:

```
result = add(5, 3)
```

Importing Modules:

- Entire module: import math
- Specific functions: from math import sqrt
- Aliases: import numpy as np

6. Libraries for Data Analysis

NumPy:

- **Purpose**: Fundamental library for numerical computing. Supports operations on arrays, matrices, and tensors. Basis for other libraries.
- **Capabilities**: Fast arithmetic operations, linear algebra, and mathematical functions.
- **Example**:

```
import numpy as np
a = np.array([1, 2, 3])
print(a * 2)
```

pandas:

- **Purpose**: Data manipulation and analysis using tabular data structures like DataFrames and Series.
- **Capabilities**: Reading/writing CSVs, filtering data, handling missing values, and data aggregation.
- **Example**:

```
import pandas as pd
df = pd.read_csv('data.csv')
print(df.head())
```

Matplotlib:

- **Purpose**: Visualization library for creating static, animated, and interactive plots.
- **Capabilities**: Line plots, bar charts, histograms, scatter plots, and full customization of visual elements.
- **Example**:

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3], [4, 5, 6])
plt.show()
```

Seaborn:

- **Purpose**: Statistical data visualization built on top of Matplotlib.
- **Capabilities**: Heatmaps, pair plots, violin plots, and integration with pandas.
- **Example**:

```
import seaborn as sns
sns.scatterplot(x="total_bill", y="tip", data=df)
plt.show()
```

SciPy:

- **Purpose**: Extends NumPy with scientific and technical computing capabilities.
- **Capabilities**: Optimization, signal processing, statistics, and numerical integration.
- **Example**:

```
from scipy import stats
mean, var = stats.norm.fit(data)
```

Scikit-learn:

- **Purpose**: Comprehensive library for machine learning.
- **Capabilities**: Supports classification, regression, clustering, and dimensionality reduction.
- **Example**:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
```

TensorFlow and Keras:

- **Purpose**: Libraries for building and training machine learning and deep learning models.
- **Capabilities**: TensorFlow handles low-level operations, while Keras offers a high-level API for neural networks.
- **Example**:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
model = Sequential()
model.add(Dense(64, activation='relu', input_dim=10))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy')
model.fit(X_train, y_train, epochs=10)
```

PyTorch:

- **Purpose**: Deep learning framework providing flexibility and control.
- **Capabilities**: Dynamic computation graphs and lower-level API for neural network customization.
- **Example**:

```
import torch
x = torch.tensor([1.0, 2.0], requires_grad=True)
y = x**2
z = y.sum()
z.backward()
print(x.grad)
```

Plotly:

- **Purpose**: Interactive and web-ready visualizations.
- **Capabilities**: Dashboards, interactive scatter plots, line charts, and 3D plots.
- **Example**:

```
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length", color="species")
fig.show()
```

LECTURE 3 Initial data quality assessment. Pre-processing of data

1. Why Data Requires Preparation

Challenges with Real Data:

- **Format Issues**: Text, audio, video, databases, etc.
- **Incompleteness**: Missing attribute values or overly general values.
- **Noise**: Errors or outliers in the dataset.
- **Inconsistency**: Variability in names, timestamps, or formatting.

Importance:

- Many algorithms require specific formats (e.g., tabular, numerical).
- Data quality impacts the effectiveness of machine learning algorithms and results.

2. Data Formats

Tabular Data (Classic ML):

- **Columns**: Features or attributes.
- **Rows**: Examples or records.

Specialized Data Formats:

- **Audio**: Represented by amplitude over time; often requires spectrograms or mel spectrograms for analysis.
- **Images**: Converted to vectors or tensors; deep methods involve neural networks for feature extraction.
- **Text**: Requires vectorization using techniques like Bag of Words (BoW), TF-IDF, or embeddings.
- **Time Series**: Analyzed for trends, seasonality, segmented using overlapping time windows.

3. Text Data Processing

Tokenization:

- Splits text into smaller units (tokens), such as words or phrases.

Vectorization:

- **Bag of Words (BoW)**: Represents text as a frequency matrix.
- **TF-IDF**: Weighs terms by importance across documents.
- **Embeddings**: Dense representations (e.g., Word2Vec, GloVe) capturing semantic meaning.

4. Data Quality Criteria

- **Accuracy**: Reflects real-world values.
- **Completeness**: Ensures all required data is present.
- **Uniqueness**: Removes duplicates.
- **Timeliness**: Data is up-to-date.
- **Consistency**: Uniform across datasets.
- **Validity**: Adheres to predefined rules.

5. Data Profiling

Objectives:

- Provides insights into data quality and structure.
- Identifies missing values, duplicates, and outliers.

Statistical Measures:

- **Minimum and Maximum**: Range of values.
- **Mean**: Average of values.
- **Median**: Middle value of sorted data.
- **Variance**: Measure of spread in data.
- **Standard Deviation**: Square root of variance.
- **Mode**: Most frequent value.
- **Percentiles and Quartiles**: Divide data into equal-sized intervals.
- **Correlation**: Relationship strength between variables.
- **Empty Values**: Identifying where data is missing and its extent.
- **Extreme Values**: Checking maximum/minimum values against expectations.
- **Duplicates**: Identifying repeated records.
- **Visualization**: Using histograms, scatter plots, and box plots to assess distributions and patterns.

6. Visualization Techniques

Common Plots:

- **Scatter Plots**: Show relationships between two variables. Enhance with color and size for additional dimensions.

- **Box Plots**: Highlight medians, quartiles, and outliers.
- **Histograms**: Visualize frequency distributions.
- **Heatmaps**: Show correlations or aggregated data.
- **Radar Charts**: Represent multidimensional data in a radial layout.
- **Parallel Coordinates Plot**: Visualize high-dimensional data as lines crossing parallel axes.
- **Scatter Plot Matrices**: Pairwise scatter plots for multiple variables.
- **Alternative Plots for Dense Data**:
    - Heatmaps for density.
    - Hexagonal binning to group data points.
    - Semi-transparent points for overlapping values.

7. Data Cleaning

Parsing:

- Extract structured data from raw text.
- **Methods**: Specify tokens to extract (e.g., names, postal codes) and ensure format adherence.

Standardization:

- Normalize representation (e.g., "W-wa" to "Warsaw").
- Techniques include removing special characters, reordering, and dictionary-based corrections.

Nominal Values Conversion:

- **One-Hot Encoding**: Converts categories into binary vectors.
- **Target Encoding**: Maps categories to mean target values.
- **Grouping**: Consolidate nominal values with many levels.

Deduplication Methods:

- String comparison (e.g., edit distance).
- Phonetic similarity measures.
- Handling abbreviations (e.g., "st." to "street").

Types of Unknown Values:

- **Missing**: Value not known.
- **Not Applicable**: Attribute irrelevant (e.g., pregnancy for males).

- **Irrelevant**: Attribute does not affect classification.

Outliers and Detection:

- **Definition**: Values significantly different from others.
- **Detection Methods**:
  - Graphical: Histograms, scatter plots, box plots.
  - Statistical: Z-score, three-sigma rule.
  - Algorithms: DBSCAN, Isolation Forest.

Three-Sigma Rule:

- In normal distribution, 99.7% of data lies within three standard deviations.
- Values beyond this range are considered outliers.

8. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- Groups points based on density.
- **Core Points**: Have at least MinPts neighbors within distance ε.
- **Border Points**: Near core points but with fewer neighbors.
- **Outliers**: Points not in any cluster.
- **Distance Methods**: Calculate proximity of points to determine anomalies.

9. Isolation Forest

- Identifies anomalies by isolating data points using random partitioning.
- Outliers require fewer splits to isolate compared to normal data.

10. Normalization

Purpose:

- Adjust scales of variables to ensure fair contributions to analysis.

Techniques:

- **Min-Max Scaling**: Transform values to [0, 1] range.
- **Z-Score Normalization**: Standardize data by subtracting the mean and dividing by the standard deviation.

11. Feature Selection

Importance:

- Reduces noise, enhances interpretability, and improves performance.

Techniques:

- **Expert Knowledge**: Based on domain understanding.
- **Statistical Metrics**:
  - **Removal Criterion - Proportion of Missing Values**: Remove columns exceeding a threshold.
  - **Low Variance**: Drop features with constant or near-constant values.
  - **High Correlation**: Remove one of the strongly correlated variables.
- **Model-Based**: Random Forest importance scores, forward/backward selection.

12. Attribute Selection

Methods:

- **Forward Selection**: Start with an empty set; iteratively add attributes improving model performance.
- **Backward Elimination**: Start with all attributes; remove those with minimal impact.

13. Dimensionality Reduction

Principal Component Analysis (PCA):

- Orthogonally transforms data into uncorrelated components.
- Focuses on variance, allowing dimensionality reduction.
- **Notes**: PCA results may be hard to interpret and require normalization.

Linear Discriminant Analysis (LDA):

- Projects data to maximize class separability.
- Supervised alternative to PCA.

Autoencoders:

- Neural networks that learn compressed representations.
- Reconstruct inputs via an encoder-decoder structure.

Benefits:

- Simplifies models, improves computational efficiency, and reduces overfitting.

14. Instance Selection

Purpose:

- Reduce dataset size while preserving representativeness.

Techniques:

- **Random Sampling**: Stratified or with replacement.
- **Oversampling/Undersampling**: Balances class distribution.
- **Distance-Based**: Clustering to select representative examples.

15. Discretization

Purpose:

- Converts numerical attributes into discrete intervals.
- Simplifies data, reduces noise, and prevents overfitting.

Methods:

- **Supervised**: Considers class labels while dividing intervals.
- **Unsupervised**: Ignores class labels.
- **Global vs. Local**: Global treats attributes independently; local depends on other attributes.

LECTURE 4 Clustering data and assessing the quality of division

1. Introduction to Data Clustering

Definition:

- Clustering involves identifying groups of similar objects, where objects in a group are more similar to each other than to objects in other groups.

Applications:

- **Segmentation**: Market segmentation or dividing customer bases.
- **Pre-processing**: Reducing dataset complexity.
- **Generalized Data Description**: Summarizing data through clusters.
- **Information Granulation**: Grouping similar information.
- **Personalization**: Recommender systems and targeted advertising.

2. Data Similarity

Types:

- **Numerical Data**: Examples include age, geographical location, transaction value, stock quotes, and gene expression values.
- **Non-Numerical Data**: Includes symbolic, text, or complex structures.

Metric Properties:

- **Identity of Indiscernibles**: $d(x, y) = 0 \iff x = y$
- **Symmetry**: $d(x, y) = d(y, x)$
- **Triangle Inequality**: $d(x, z) \leq d(x, y) + d(y, z)$

Common Distance Metrics:

- **Euclidean Distance**: Measures straight-line distance in n-dimensional space.
- **Manhattan Distance**: Measures distance as the sum of absolute differences between coordinates.
- **Cosine Similarity**: Captures similarity based on vector directions rather than magnitude.
- **Pearson Correlation Coefficient**: Measures linear correlation between variables.
- **Jaccard Coefficient** (Binary Data): Measures the similarity between two sets.
- **Hamming Distance**: Counts differing elements in two sequences.
- **Minkowski Distance** (Generalization): Includes Euclidean () and Manhattan () as special cases.

3. Clustering Methods

Idea Perspective:

- **Hierarchical**: Forms a tree-like structure (dendrogram).
- **Divisive (Optimization-Based)**: Splits data iteratively (e.g., k-means).
- **Density-Based**: Focuses on dense regions in the dataset.

Result Perspective:

- **Hard Clustering**: Each object belongs to one cluster.
- **Fuzzy Clustering**: Objects have degrees of membership in multiple clusters.

4. Hierarchical Clustering

Process:

1. Initialize: Treat each data object as a separate cluster.
2. Merge: Combine the closest clusters based on a linkage criterion.
3. Stop: Terminate when the desired number of clusters is achieved or maximum cluster distance is exceeded.

Linkage Criteria:

- **Single Link**: Distance between the nearest points in clusters.
- **Complete Link**: Distance between the farthest points in clusters.
- **Average Link**: Average pairwise distance between all points in clusters.
- **Minimum Variance**: Minimizes overall variance.

Output:

- **Dendrogram**: Visual representation of hierarchical clusters, showing merges and splits.

Advantages:

- Captures nested relationships.
- Does not require specifying the number of clusters in advance.

Limitations:

- Computational complexity: $O(n^2)$
- Sensitive to noise and outliers.

5. k-Means Clustering

Process:

1. Initialize: Randomly select k centroids.
2. Assignment: Assign each data point to the closest centroid.
3. Update: Recalculate centroids based on assigned points.
4. Iterate: Repeat until convergence or a stop condition is met.

Algorithm Details:

- Criterion Function: Minimized to achieve tight clusters.
- Stop Conditions:
    - Centroids do not change significantly.
    - Maximum number of iterations reached.

Issues:

- Sensitive to initial centroids.
- May converge to local minima.
- Requires predefined k value.
- Struggles with varying cluster sizes, densities, and shapes.

Complexity:

- $O(n)$ where n is the number of iterations.

6. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

Characteristics:

- Identifies clusters of arbitrary shapes.
- Detects noise points that do not belong to any cluster.

Parameters:

- **ε**: Radius defining the neighborhood.
- **m**: Minimum number of points required in a neighborhood to form a dense region.

Algorithm:

1. For each unvisited point:
    - Check if it is a core point (has at least m neighbors within ε).
    - If true, form a new cluster and expand it by recursively including density-reachable points.
    - Mark points not density-reachable as noise.

Advantages:

- Does not require specifying the number of clusters.
- Handles noise effectively.

Limitations:

- Sensitive to parameter selection (ε, m).

Complexity:

- $O(n * \log n)$ with spatial indexing - when a spatial indexing structure like a k-d tree or R-tree is used, the algorithm can efficiently retrieve neighborhood points, reducing complexity.
- $O(n^2)$ otherwise - Without such structures, the algorithm must compare each point to all others to find neighbors, which results in quadratic complexity.

7. OPTICS Algorithm

- **Order-based Clustering**: Similar to DBSCAN but computes a reachability distance for each point.
- **Advantages**:
    - Handles clusters with varying densities.
    - Visualizes clustering structure as a reachability plot.
- **Output**: Cluster orderings and reachability distances.

8. Fuzzy Clustering (e.g., Fuzzy c-Means)

Fuzzy Sets:

- Membership values range from 0 to 1, indicating degrees of belonging to a set.
- Example: "It is warm outside" can have partial truth (e.g., 0.7).

Algorithm:

1. Initialize: Membership matrix with degrees of belonging.
2. Assign: Calculate membership values using:
3. Update: Recalculate cluster centroids:
4. Iterate: Repeat until convergence.

Output:

- **Fuzzy Partition Matrix**: Degrees of membership.
- **Prototype Matrix**: Cluster centroids.

9. Clustering Quality Evaluation

Types of Evaluation:

- **External**: Compares clustering results with a ground truth.
- **Internal**: Evaluates cohesiveness and separation of clusters.
- **Relative**: Compares results across different parameter settings.

Metrics:

- **Cluster Cohesiveness (Δ)**:
  - **Complete Diameter**: Maximum distance between points in a cluster:
  - **Average Diameter**: Mean pairwise distance:
  - **Centroid Diameter**: Mean distance to the cluster centroid:
- **Cluster Separation (δ)**:
  - **Single Linkage**: Distance between closest points in two clusters.
  - **Complete Linkage**: Distance between farthest points.
  - **Average Linkage**: Average distance between points in two clusters:

Indices:

- **Davies-Bouldin Index**: Ratio of cluster scatter to separation.
- **Dunn Index**: Ratio of minimum separation to maximum diameter.

LECTURE 5 Predictive methods 1

1. Experimental Evaluation of Predictive Models

Importance:

- Inductive reasoning is fallible; results must be validated on independent datasets.
- Datasets:
  - **Training Set (Tr)**: Used for model creation.
  - **Test Set (Ts)**: Used for validation.
  - **Verification Set (V)**: Optimizes parameters during analysis.
- Larger sets yield better hypotheses and approximations of inference validity.

Issues:

- Inference can be unreliable due to:
  - Overfitting to training data.
  - Limited generalization to new data.
  - High variance in predictions.
  - Dataset biases, leading to poor real-world performance.

2. Experimental Evaluation Methods

Train-and-Test (Hold-Out):

- Divide dataset into training (e.g., 70%) and test (e.g., 30%) subsets.

Cross-Validation (CV):

- Split data into k folds; train on k-1 folds, test on the remaining fold.
- Popular: 5-fold, 10-fold CV.
- **Stratified CV** ensures class proportions are consistent across folds.

Leave-One-Out CV:

- Train on examples; test on the remaining example.
- Repeated for all examples (suitable for small datasets).

Bootstrapping:

- Draw examples with replacement to form a training set.
- Test on examples not drawn.
- Repeat times.
- Final evaluation:

3. Classification Quality Measures

Binary Classification Metrics:

- **True Positives (TP)**: Correctly predicted positive instances.
- **False Negatives (FN)**: Positive instances predicted as negative.
- **True Negatives (TN)**: Correctly predicted negative instances.
- **False Positives (FP)**: Negative instances predicted as positive.

Derived Metrics:

- **Sensitivity (True Positive Rate)**: Indicates the ability to detect positive cases.
  $\frac{TP}{(TP+FN)}$ $\quad \mathrm{TPR} = \mathrm{TP} / (\mathrm{TP} + \mathrm{FN}) = \text{sensitivity}$
- **Specificity (True Negative Rate)**: Measures the ability to detect negative cases.
  $\frac{TN}{(FP+TN)}$ $\quad \mathrm{FPR} = \mathrm{FP} / (\mathrm{FP} + \mathrm{TN}) = 1 - \text{specificity}$
- **Positive Predictive Value (PPV):** Indicates precision of positive predictions.
  $\mathrm{PPV} = \mathrm{TP} / (\mathrm{TP} + \mathrm{FP})$
- **Negative Predictive Value (NPV):** Reflects precision of negative predictions.
  $\mathrm{NPV} = \mathrm{TN} / (\mathrm{FN} + \mathrm{TN})$
- **Overall Accuracy (Acc):** Measures overall correctness of predictions.
  $$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$
- **Balanced Accuracy (BAcc):** Accounts for imbalanced datasets.
  $$BAcc = \frac{1}{2}\sum_{i=0}^{k} TPR_i$$
- **F-Measure:** Balances precision and recall.
  $$Gmean = \sqrt{sens * spec}$$
- **G-Mean:** Captures balance between sensitivity and specificity.
  $$Fmeasure = \frac{2 * PPV * sens}{PPV + sens}$$

ROC Curve:

- **Definition**: Plots True Positive Rate (TPR) vs. False Positive Rate (FPR) at various thresholds.
- **Area Under Curve (AUC)**:
  - A higher AUC indicates better classifier performance.
  - indicates random guessing.
  - indicates perfect classification.

- ROC curves help evaluate model thresholds and compare classifiers.

Lift Chart:

- **Purpose**: Evaluates model's effectiveness in concentrating positive cases in top predictions.
- **How**: Plots cumulative percentage of positives captured against sample size percentage.

4. Assessment of Forecasting Models

What Else Can Be Assessed:

- **Model Robustness**: Stability across different datasets.
- **Complexity**: Simplicity in interpreting model outputs.
- **Scalability**: Feasibility for larger datasets.

Evaluation of Regression Models:

- **Mean Absolute Error (MAE)** - Measures the average magnitude of errors in predictions without considering their direction.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |f(x_i) - d(x_i)|$$

- **Mean Squared Error (MSE)** - Penalizes larger errors more heavily by squaring them, providing a sensitive measure of prediction accuracy.

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- **Root Mean Squared Error (RMSE)** - Gives the error in the same units as the original data, derived from the MSE.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

- **R-Squared (Coefficient of Determination)** - Represents the proportion of variance in the dependent variable that is predictable from the independent variables.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y}_i)^2}$$

5. Verification of Significance Between Predictive Models

- **Statistical Tests**:
    - Paired t-tests for comparing means of error distributions.
    - Wilcoxon signed-rank test for non-parametric analysis.
- **Randomization Tests**: Evaluate differences by randomizing class labels.

- **Confidence Intervals**: Ensure significance of differences between model scores.

6. Decision Trees

Inductive Reasoning:

- Generalization from examples to infer patterns.
- **Positive Examples**: Guide hypothesis creation.
- **Counterexamples**: Refine or adjust hypotheses.
- Decision trees explicitly represent hypotheses as interpretable structures.

Hypotheses Representation:

- Decision trees, classification rules, parameter vectors (e.g., Bayesian), or mathematical equations (e.g., linear discriminants).

Structure:

- **Nodes:** Contain tests determining the next branch.
- **Branches:** Identified by conditions met at the parent node.
- **Leaves:** Represent class labels or predictions.
- Regression trees may contain continuous values or equations.

Example:

- A tree predicts decisions based on attributes like weather (e.g., "If Outlook=sunny AND Humidity=normal THEN decision=1").

7. Decision Tree Induction

Top-Down Induction of Decision Trees (TDIDT):

1. Check stop criterion; if met, create a leaf.
2. Choose a test for the current node.
3. Split the dataset based on the test.
4. Recursively induce subtrees for each subset.

General Information About Tests:

- Tests determine the complexity and efficiency of the decision tree.
- A good test minimizes differentiation among examples within subsets, leading to homogenous leaves quickly.
- Simpler tests generally result in more interpretable trees.

Variants:

- Algorithms: CLS, ID3, C4.5, C5.
- Tests:
  - **Nominal Attributes:**
    - Identity: The test result is the value of the attribute.
    - Equality: The test result is 1 if the attribute matches a specific value, 0 otherwise.
    - Membership: The test result is 1 if the attribute value belongs to a specified set, 0 otherwise.
  - **Ordinal/Continuous Attributes:**
    - Membership: The test result is 1 if the attribute value is within a specified range, 0 otherwise.
    - Inequality: The test result is 1 if the attribute value is smaller/larger than a threshold, 0 otherwise.

8. Decision Trees: Tests and Measures

Entropy:

- Entropy measures the level of impurity or disorder in a dataset:

$$Ent(S) = -\sum_{i=1}^{k} p_i log_2(p_i)$$

  - p_i: Probability of an instance belonging to class i.
  - Higher entropy implies more disorder, while lower entropy indicates homogeneity in the dataset.

Conditional Entropy:

- Measures the entropy after splitting the dataset on an attribute a:

$$ENt(S|a) = -\sum_{j=1}^{p} \frac{n_{S_j}}{n} Ent(S_j)$$

  - Considers probabilities of subsets formed by the attribute split.
  - This evaluates how the split affects the disorder. Lower conditional entropy suggests that the split is effective at reducing uncertainty.

Information Gain:

- Reduction in entropy due to attribute a:

$$Gain(S, a) = Ent(S) - Ent(S|a)$$

  - High information gain indicates an attribute is effective for classification.

Chi-Square () Test:

- Measures the dependence between two variables (e.g., the class label and an attribute):

$$\chi^2 = \sum_{i=1}^{k}\sum_{j=1}^{l}\frac{(f_{ij} - e_{ij})^2}{e_{ij}}$$

  - f_{ij}: Observed frequency in the i-th row and j-th column of a contingency table.
  - e_{ij}: Expected frequency under independence assumption.
  - Larger values indicate stronger relationships between the attribute and the class.

9. Decision Tree Pruning

Purpose:

- Prevent overfitting by replacing subtrees with leaves.
- Assign leaves the most frequent class label in the subtree.

Techniques:

- **Pre-Pruning**: Stop growth early based on a threshold (e.g., minimum samples per node).
- **Post-Pruning**: Remove branches after full growth by comparing error rates on validation data.
- **Cost-Complexity Pruning**: Balances model complexity with error reduction using a penalty term.
- **Evaluation**: Use validation data to assess whether pruning improves performance.

LECTURE 6 Predictive methods 2

1. Introduction to Ensemble Learning

Overview:

- Ensemble learning combines multiple models to improve prediction accuracy.
- Used for both classification and regression tasks.

Error Components:

- **Bias**: Error introduced by the assumptions of the model.
- **Variance**: Error due to incomplete representation of the domain.

Error Equation:

- Total Error = Bias + Variance

2. Ensemble Learning Process

Steps:

1. **Ensemble Generation**:
   - Create a set of models.
2. **Ensemble Reduction**:
   - Remove redundant or less effective models.
3. **Integration of Ensemble Decisions**:
   - Combine outputs of individual models for final predictions.

Types of Ensembles:

- **Homogeneous**: Same algorithm generates different models.
- **Heterogeneous**: Different algorithms generate the models.

3. Ensemble Generation Methods

Data Manipulation:

- **Sampling**: Random sampling with replacement (bagging).
- **Feature Selection**:
  - Random feature selection.
  - Genetic algorithms.
- **Decision Attribute Manipulation**: Modify target variables.

Model Generation Manipulation:

- **Algorithm Parameters**: Vary parameter settings.
- **Sequential Generation**: Train models iteratively (boosting).
- **Parallel Generation**: Train models independently (bagging).

Model Selection:

- **Clustering**: Group similar models and select representatives.
- **Search Methods**:
  - Exhaustive search.
  - Heuristic approaches (e.g., genetic algorithms).
  - Sequential search (forward, backward).

4. Integration of Ensemble Decisions

For Classification:

- **Plurality Vote**:
  - Simple majority.
  - Constitutional majority.
- **Weighted Voting**: Assign weights to models based on performance.

For Regression:

- **Mean Value**: Average predictions of models.
- **Weighted Mean**: Weighted average based on model accuracy.
- **Median**: Use the middle prediction value.

5. Bagging (Bootstrap Aggregating)

Overview:

- Bagging stands for Bootstrap Aggregating.
- It generates multiple models by training them on bootstrapped (randomly resampled with replacement) subsets of the training dataset.
- Reduces variance and improves prediction stability by combining outputs.

Process:

1. Create multiple datasets by randomly sampling with replacement from the original dataset.
2. Train a model (e.g., decision tree) on each of these datasets.
3. Combine predictions using:

    - **Majority Voting** (for classification).
    - **Averaging** (for regression).

Key Features:

- Each model is trained on a slightly different dataset, making them diverse.
- Aggregation reduces the risk of overfitting compared to using a single model.

6. Random Forest

Description:

- Random Forest is an extension of bagging that builds an ensemble of decision trees with added randomness.
- It introduces feature-level randomness to further diversify the models.

Key Steps:

1. Generate multiple bootstrapped datasets.
2. For each decision tree:
    - At each split, randomly select a subset of features instead of considering all features.
    - Use the selected features to determine the best split.
3. Combine predictions using majority voting (classification) or averaging (regression).

Benefits:

- Reduces overfitting by averaging predictions across many trees.
- Handles large datasets and works well with high-dimensional data.
- Robust to noise and irrelevant features.

7. Boosting

Overview:

- Boosting is an iterative technique that builds models sequentially, where each model attempts to correct the errors of the previous one.
- Focuses on improving weak learners by giving more weight to misclassified examples.

Key Features:

1. Models are trained sequentially, with each model focusing on examples where the previous ones performed poorly.
2. Final predictions are made by combining the outputs of all models, often using weighted sums.

3. The goal is to reduce bias and variance simultaneously.

Types:

- **AdaBoost (Adaptive Boosting):**
  - Focuses on misclassified examples by assigning higher weights.
  - Models are trained on reweighted datasets.
- **Gradient Boosting:**
  - Uses gradient descent to minimize a loss function.
  - Adjusts predictions by adding models that optimize the residual errors.

Benefits:

- Can produce highly accurate models.
- Works well with weak learners (e.g., shallow decision trees).

Limitations:

- Sensitive to noisy data and outliers.
- Computationally expensive due to sequential training.

8. Gradient Boosting

Description:

- Gradient Boosting is a sequential ensemble method that builds models iteratively by focusing on minimizing the errors of the previous models through optimization of a loss function.

Process:

1. Initialize the model with a constant prediction (e.g., mean of the target values):
2. **Iteratively improve the model**: For m=1to M (number of boosting iterations):
   - Compute pseudo-residuals
   - Train a base learner (weak model) to fit the pseudo-residuals
   - Update the ensemble model
3. Output the final model

Applications:

- Widely used for structured data tasks (classification and regression).
- Common implementations include XGBoost, LightGBM, and CatBoost.

9. Stacking (Stacked Generalization)

Description:

- Stacking combines the predictions of multiple models (level-0 models) using a meta-model (level-1 model).
- Unlike bagging and boosting, stacking allows for the use of different types of models.

Process:

1. Train several base models (level-0 models) on the training data.
2. Use the predictions of these models to form a new dataset.
   - The features of the new dataset are the predictions from the base models.
   - The target remains the same as the original dataset.
3. Train a meta-model (level-1 model) on this new dataset.
   - The meta-model learns how to combine the predictions of the base models.

Example:

- Base models could be decision trees, logistic regression, and SVMs.
- The meta-model could be a linear regression model or another ensemble method.

Benefits:

- Can achieve higher accuracy by leveraging the strengths of different base models.
- Reduces the risk of overfitting when properly tuned.

Limitations:

- Computationally expensive due to training multiple models.
- Requires careful design and tuning of the meta-model.

LECTURE 7 Time-series analysis

1. Time Series Overview

Definition:

- A **time series** is a sequence of values recorded over time.
- Typically recorded at regular intervals but can be irregular (adjusted through resampling).

Resampling Methods:

- **Downsampling**: Aggregating values when reducing frequency.
- **Upsampling**: Filling in values using interpolation or forward/backward filling.

Goals of Analysis:

- Discovering patterns.
- Forecasting future values.

Predictors and Forecast Horizon:

- **Predictors**: Variables used to predict future values in the time series (e.g., lagged values, external variables).
- **Forecast Horizon**: The period into the future for which predictions are made (short-term, medium-term, long-term).

Types of Time Series Modeling:

1. **Univariate (1:1 Single-step)**: Involves only one time series variable.
2. **Multivariate (1:1 Multi-step)**: Includes additional variables or time series as predictors.
3. **Deterministic Models (N:1 Multi-step)**: Rely on known patterns and rules.
4. **Stochastic Models (N:M Multi-step)**: Incorporate randomness and probabilistic elements.

Examples of Time Series:

- Weather data.
- Financial indicators (e.g., stock prices, exchange rates).
- Electricity/gas consumption.
- Product sales.
- IoT metrics (e.g., CPU performance).

2. Key Components of Time Series

Patterns in Time Series:

- **Trend**: Long-term increase or decrease in the data.

    - Not always linear, can change direction.
- **Seasonality**: Repeating patterns at fixed intervals (e.g., yearly, weekly).
- **Cyclicality**: Non-fixed intervals influenced by external factors (e.g., business cycles).

Differentiating Seasonality and Cyclicality:

- **Seasonality**: Constant frequency (e.g., daily temperature changes).
- **Cyclicality**: Variable frequency (e.g., economic cycles).

Visualization Techniques:

- **Line Charts**: Show trends, spikes, and seasonality.
- **Seasonal Plots**: Highlight seasonal variations.
- **Lag Plots**: Reveal relationships between consecutive points.

3. Autocorrelation

Definition:

- Measures the relationship between values at different time lags.
- **ACF (Autocorrelation Function)**: Describes autocorrelation coefficients across lags.

Characteristics:

- Data with a **trend**: High autocorrelation for small lags, decreasing over time.
- **Seasonal data**: Peaks at seasonal intervals.
- **White noise**: Random data with no discernible patterns, mean zero, and constant variance. Autocorrelation is approximately zero for all lags.

4. Handling Time Series Data

Time Stamps:

- Represent the time context of each observation, critical for alignment and analysis.
- **Best Practices**:
    - Use UTC or a consistent timezone to avoid mismatches.
    - Ensure uniform granularity (e.g., hourly, daily) across datasets.
    - Address daylight saving changes for regional data.
    - Validate the completeness and correctness of timestamp intervals.

Missing Values:

- **Imputation Methods**:

    - **Forward fill**: Use the last available value.
    - **Backward fill**: Use the next available value.
    - **Moving average**: Smooth data by averaging values over a rolling window.
    - **Interpolation**: Estimate missing values using linear or spline methods.
    - **Model-Based Imputation**: Predict missing values using machine learning or time-series models.
- **Removal**:
    - Drop rows or periods with extensive missing data if they introduce significant noise.

Changing Sampling Frequency:

- **Upsampling**:
    - Add timestamps for higher granularity.
    - Use interpolation methods to estimate missing values between samples.
- **Downsampling**:
    - Aggregate data using statistical measures (mean, sum, median).
    - Retain essential patterns while reducing resolution.

Correction and Transformation:

- **Calendar Adjustments**:
    - Normalize metrics (e.g., monthly sales divided by the number of days in the month).
- **Inflation Adjustments**:
    - Convert monetary values to constant currency for fair comparison.
- **Mathematical Transformations**:
    - Logarithmic transformation for reducing skewness.
    - Box-Cox transformation for stabilizing variance.
    - Detrending to isolate residual patterns.

5. Data Smoothing

Purpose:

- Eliminate noise to highlight patterns.

Techniques:

- **Moving Average**:

    - Calculation of the average value within a window of neighboring points. Variants include median and weighted averages.
- **Exponential Smoothing**:
    - Smoothing factor (0 to 1).

6. Decomposition of Time Series

Types:

- **Additive**:
    - Suitable for constant variability.
- **Multiplicative**:
    - Used for proportional seasonal variability.

Methods:

- **STL (Seasonal and Trend decomposition using Loess)**:
    - Combines trend, seasonal, and residual components flexibly.
    - Handles non-linear trends and changing seasonality.
    - Works well with both short-term and long-term series.

7. Simple Forecasting Methods

Benchmarks:

- **Mean**: Forecast as the average of past values.
- **Naive**: Use the last observed value as the forecast.
- **Seasonal Naive**: Repeat the value from the same season in the past.
- **Drift**: Linear extrapolation based on the first and last observations.

8. Residual Analysis

Residuals:

- Difference between observed and predicted values:

Assumptions:

- No autocorrelation.
- Zero mean.
- Constant variance.

- Residuals should resemble white noise (no discernible patterns).

Diagnostic Tools:

- **Histogram of Residuals**: Tests for normality.
- **ACF Plot of Residuals**: Detects remaining autocorrelation.
- **QQ Plot**: Compares residual distribution to a normal distribution.
- **Ljung-Box Test**: Assesses randomness.

9. Measures of Forecast Accuracy

Definitions:

- **Mean Absolute Error (MAE)**:
  - Measures the average magnitude of errors.
- **Mean Squared Error (MSE)**:
  - Penalizes larger errors more heavily.
- **Root Mean Squared Error (RMSE)**:
  - Provides error in the same units as the data.
- **Mean Absolute Percentage Error (MAPE)**:
  - Scales errors relative to observed values.

Limitations:

- Scale-dependent metrics (MAE, MSE, RMSE) are not comparable across datasets with different units.
- MAPE can produce undefined or extreme values when .

10. Advanced Forecasting Techniques

Stationarity and Differencing:

- **Stationarity**: Time series with constant mean, variance, and autocorrelation over time.
- **Differencing**: Subtracting consecutive values to stabilize trends and remove seasonality.

Auto ARIMA:

- Automatically selects optimal ARIMA parameters () using AIC/BIC criteria.
- Handles both trend and seasonality efficiently.

Exponential Smoothing (ETS Models):

- Incorporates trends and seasonality:

    - **Trend components**: None (N), Additive (A), Damped (Ad).
    - **Seasonal components**: None (N), Additive (A), Multiplicative (M).

Prophet:

- Developed by Facebook for strong seasonality and long histories:
    - g(t): Trend.
    - s(t): Seasonality.
    - h(t): Holiday effects.
    - Random error.
    - Robust to missing data and outliers.

11. Machine Learning for Time Series

Adapting ML Methods:

- **Recursive Approach**: Predict one step at a time, using previous predictions for subsequent steps.
- **Direct Approach**: Train separate models for each forecast horizon.

Feature Engineering:

- Transform raw time series into tabular data:
    - Statistical features (e.g., mean, variance).
    - Domain-specific features (e.g., peaks, periodicity).

Classical ML Algorithms:

- Linear regression, decision trees, random forests, boosting.

12. Deep Learning for Time Series

Neural Networks:

- Eliminate the need for feature engineering.
- Struggle with trends and seasonality if not explicitly modeled.

Architectures:

- **Feedforward Neural Networks (FFN)**: Simplest, not time-aware.
- **Recurrent Neural Networks (RNN)**: Sequential processing, uses hidden states.
- **LSTM/GRU**:
    - Handle long-term dependencies via gates (forget, input, output).

    - Effective for capturing both short and long-term patterns.
- **Attention Mechanisms**:
    - Focus on important time steps.
    - Improve model interpretability.
- **DeepAR**: Amazon’s RNN-based model for probabilistic multi-series forecasting.
- **Transformer Models**: Advanced architectures for handling large-scale time-series data.

LECTURE 8 Language models

1. Introduction to Language Models

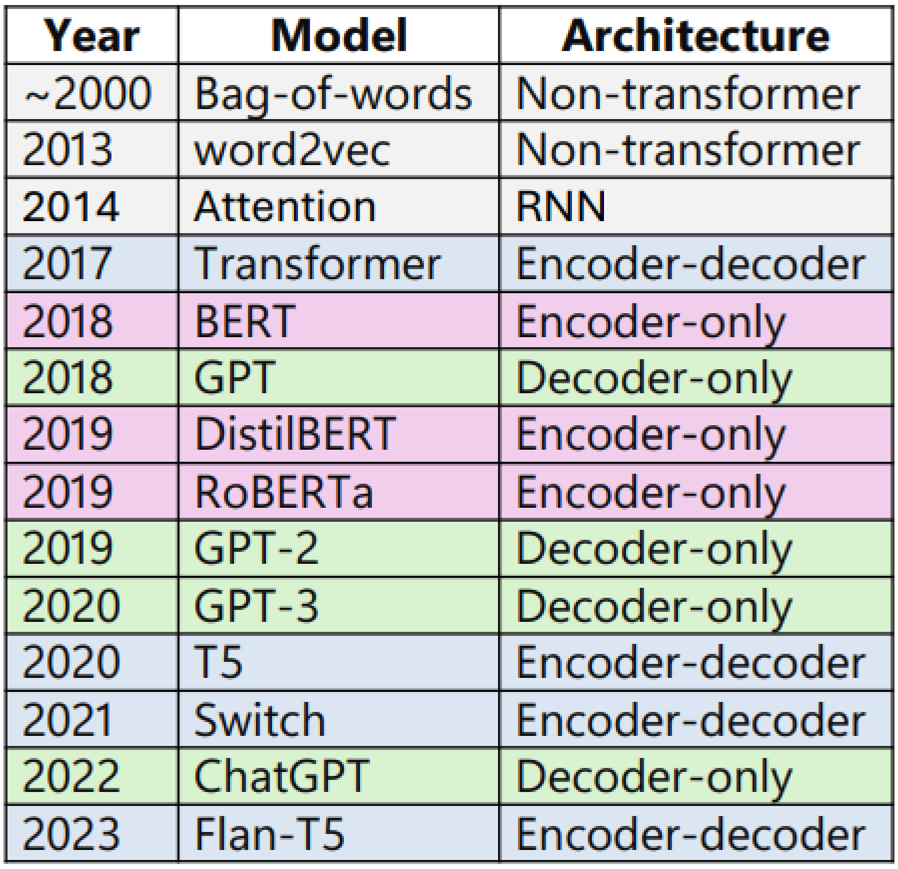
Definitions:

- **Artificial Intelligence (AI):**
  - Computer systems performing tasks resembling human intelligence (e.g., speech recognition, language translation).
  - Often overused to describe systems that may not exhibit true intelligence.
- **Natural Language Processing (NLP):**
  - A subfield of AI focused on enabling analysis, processing, and generation of human language.
- **Language Model:**
  - A model analyzing, understanding, and generating text based on patterns in data.

Applications:

- Text generation, summarization, emotion detection, translation, spell checking, chatbots, etc.

2. Historical context



Year	Model	Architecture
~2000	Bag-of-words	Non-transformer
2013	word2vec	Non-transformer
2014	Attention	RNN
2017	Transformer	Encoder-decoder
2018	BERT	Encoder-only
2018	GPT	Decoder-only
2019	DistilBERT	Encoder-only
2019	RoBERTa	Encoder-only
2019	GPT-2	Decoder-only
2020	GPT-3	Decoder-only
2020	T5	Encoder-decoder
2021	Switch	Encoder-decoder
2022	ChatGPT	Decoder-only
2023	Flan-T5	Encoder-decoder

3. Text Vectorization

Purpose:

- Convert text into numerical form (vectors) for ML/AI tasks.

Methods:

1. **Bag of Words (BoW):**
    - Splits text into tokens, creates a vocabulary, and counts token occurrences.
    - Ignores semantics or word order.
    - Strengths: Simple and interpretable.
    - Weaknesses: High dimensionality and lack of semantic information.
    - Example:
        - Texts: ["The cat runs fast", "The dog runs slowly"].
        - Vocabulary: [the, cat, runs, fast, dog, slowly].
        - Vector: [1, 1, 1, 1, 0, 0] for "The cat runs fast".
2. **TF-IDF (Term Frequency-Inverse Document Frequency):**
    - Enhances BoW by weighting terms based on their importance:
        - **TF**: Frequency of a term in the document.
        - **IDF**: How rare a term is across all documents.
    - Formula contains term frequency in document and number of documents containing term.
3. **Word Embeddings (e.g., word2vec, GloVe):**
    - Represent words as dense vectors in a continuous space.
    - Embeddings capture semantic relationships: similar words are close in vector space.
    - Applications: Sentiment analysis, document classification, entity recognition.

Types of Embeddings:

- **Static Embeddings:** Fixed representations for each word (e.g., word2vec, GloVe).
- **Contextual Embeddings:** Dynamic representations considering context (e.g., BERT, GPT).

Applications of Embeddings:

- Text similarity, machine translation, information retrieval, and clustering.

4. Tokenization

Definition:

- Divides text into smaller components (tokens) such as words, phrases, punctuation marks, or symbols.
- Example: "Hello! How are you?" becomes ["Hello", "!", "How", "are", "you", "?"]

Types:

- **Word-Level Tokenization:** Splits text into individual words.
- **Subword Tokenization:** Splits text into subword units, useful for handling unknown words (e.g., Byte Pair Encoding).
- **Character-Level Tokenization:** Splits text into individual characters, useful for character-based models.

5. Word Embeddings

Advantages:

- Capture semantic meaning and relationships between words.
- Efficient representation compared to sparse matrices in BoW or TF-IDF.

Contextual Properties:

- Similar words are close in vector space.
- Example embeddings (illustrative):

Word	Vector
cat	[0.6, 0.9]
kitten	[0.5, 0.8]

6. Attention Mechanisms

Problems with RNNs:

- Context fades over long sentences.
- Sequential processing limits scalability.

Solution:

- **Attention Mechanism:**
    - Focuses on relevant parts of input sequences, amplifying their importance.
    - Enables capturing long-range dependencies.

    - Computes attention weights to assign importance to each input token.

7. Transformer Architecture

Key Innovations:

- Replaces RNNs with self-attention mechanisms for parallel training.
- Composed of encoder and decoder blocks.

Encoder:

- Processes input sequences and generates contextual embeddings.
- Consists of self-attention layers and feedforward layers.

Decoder:

- Generates output sequences, incorporating encoder outputs.
- Uses masked self-attention to ensure predictions are made sequentially.

Components:

1. **Self-Attention Mechanism:** Captures relationships between all tokens.
2. **Feedforward Neural Network:** Adds non-linearity to representations.
3. **Positional Encoding:** Encodes token positions to retain sequence information.

8. Language Model Types

Encoder-Only Models:

- Focus on understanding the input sequence without generating new content.
- Example: **BERT (Bidirectional Encoder Representations from Transformers):**
    - Trained using Masked Language Modeling (MLM), predicting masked words in a sentence.
    - Applications: Classification, sentiment analysis, named entity recognition.

Decoder-Only Models:

- Focus on text generation and work autoregressively by predicting the next token based on previous ones.
- Example: **GPT (Generative Pre-trained Transformer):**
    - Trained using Causal Language Modeling (CLM), predicting the next word in a sequence.
    - Applications: Text completion, dialogue systems, content generation.

Encoder-Decoder Models:

- Combine the strengths of encoders and decoders for tasks requiring input-to-output mappings.
- Example: **T5 (Text-to-Text Transfer Transformer):**
  - Converts all NLP tasks into a text-to-text format.
  - Applications: Machine translation, summarization, question answering.

9. Large Language Models (LLMs)

Definition:

- Advanced language models trained on massive datasets with billions of parameters.

Characteristics:

- Capable of understanding context and generating coherent, human-like text.
- Examples: GPT-3, PaLM, ChatGPT.

Applications:

- Conversational AI, creative writing, code generation, question answering.

10. Training Language Models

Steps:

1. **Pretraining:**
   - On large text corpora, learns grammar, language patterns, and factual knowledge.
   - Common objective functions:
     - **Masked Language Modeling (MLM):** Predict masked tokens (e.g., BERT).
     - **Causal Language Modeling (CLM):** Predict the next token (e.g., GPT).
2. **Fine-tuning:**
   - Adapts the model to specific tasks or datasets.
   - Requires smaller datasets and is faster than pretraining.

Modes of Training:

- **Unsupervised Pretraining:** Utilizes unannotated text to learn general patterns.
- **Supervised Fine-tuning:** Tailors the model for specific tasks using labeled data.
- **Reinforcement Learning with Human Feedback (RLHF):** Optimizes responses using human preferences.

Challenges:

- Computationally expensive.
- Ethical concerns: bias and misinformation in training data.

11. Prompt Engineering

Purpose:

- Designing inputs to guide model responses effectively.

Techniques:

1. **Basic Prompts:** Short and direct.
2. **Structured Prompts:** Add instructions or context.
3. **Chain-of-Thought Prompts:** Guide reasoning step by step to enhance problem-solving.
    - Example: "To solve this problem, first do X, then Y, and finally Z."

Components:

- Persona, instruction, context, format, audience, tone, and data.

12. Controlling Response Randomness

Parameters:

- **Temperature:**
    - Higher: Produces more creative and diverse responses.
    - Lower: Generates more deterministic and focused responses.
- **Top P (Nucleus Sampling):**
    - Limits token selection to a cumulative probability threshold, focusing on the most likely tokens while avoiding outliers.
- **Top-K Sampling:**
    - Restricts token selection to the top most probable tokens.

13. Advanced Prompting

Chain Prompting:

- **Description:** Decomposes complex problems into smaller, manageable steps over multiple prompts.
- **Advantages:** Improves clarity, accuracy, and reasoning.
- **Example:**
    - Prompt 1: "What are the key elements of X?"

    - Prompt 2: "Explain how element Y influences Z."

Few-shot Learning:

- **Description:** Provides a few examples within the prompt to guide the model's behavior.
- **Advantages:** Enhances model performance on tasks with limited data.
- **Example:**
    - "Translate the following sentences:
        1. English: 'Hello' → French: 'Bonjour'
        2. English: 'Goodbye' → French: 'Au revoir'
        3. English: 'Thank you' → French:"

14. Popular Tools

OpenAI:

- API for GPT-based models.

Hugging Face:

- Provides tools for pre-trained models and fine-tuning.

LangChain:

- Framework for building LLM-based applications.

LECTURE 9 XAI and AutoML

1. Introduction to XAI and AutoML

XAI (Explainable Artificial Intelligence):

- Focuses on providing clarity and understanding of AI model decisions.
- Ensures that models are interpretable and align with ethical and societal standards.

AutoML (Automated Machine Learning):

- Aims to automate the creation, training, and optimization of machine learning models.
- Enhances accessibility and reduces the expertise needed to build effective ML systems.

2. XAI (Explainable Artificial Intelligence)

Definitions:

- **Explainability:** Actively provides reasons or details for decisions made by the model.
- **Interpretability:** Passively allows users to deduce how a model works.
- **Responsible AI:** Considers ethical, moral, and societal implications.

Target Audiences and Goals:

1. **Domain Experts:**
    - o Goal: Trust the model and gain scientific insights.
2. **Model Users:**
    - o Goal: Understand the model's decisions for fairness and transparency.
3. **Regulatory Entities:**
    - o Goal: Ensure compliance with legislation (e.g., GDPR's "Right to Explanation").
4. **Corporate Stakeholders:**
    - o Goal: Assess corporate AI usage and compliance.
5. **Developers and Data Scientists:**
    - o Goal: Improve model efficiency and explore new functionality.

Challenges:

- Balancing interpretability and accuracy.
- Addressing the "black-box" nature of advanced AI models.

3. XAI: Model Types and Techniques

Interpretable Models:

- Models with inherent transparency and simplicity.
- Examples:
    - **Linear/Logistic Regression:**
        - Directly interpretable coefficients showing feature importance.
    - **Decision Trees:**
        - Tree-structured models that make decisions based on feature splits.
        - Visualizable and explainable through clear decision rules.
    - **Rule-Based Systems:**
        - Use explicit IF-THEN rules derived from data or domain knowledge.
        - Example: "IF income > 50k AND age < 40, THEN approve loan."
    - **k-Nearest Neighbors (kNN):**
        - Predicts by finding the closest data points (neighbors) to the input.
        - Explanation based on similarity to known examples.
    - **General Additive Models (GAM):**
        - Extends linear models with non-linear terms while remaining interpretable.
    - **Bayesian Models:**
        - Provide probabilistic interpretations and uncertainty estimates.

Post-Hoc Explainability:

- Provides explanations for complex "black-box" models.
- Techniques:
    1. **Model-Agnostic:** Independent of the model architecture.
    2. **Model-Specific:** Tailored to particular models.

Explanation Types:

- **Global Explanations:** Insight into the overall behavior of the model.
- **Local Explanations:** Insight into specific predictions.
- **Explanation Type Perspective:**
    - **Feature Importance:** Which features matter the most for predictions.
    - **Feature Interaction:** How features work together to influence decisions.
    - **Counterfactual Reasoning:** What changes would lead to a different outcome.

4. XAI Techniques

Global Explanations:

1. **Partial Dependence Plot (PDP):**
   - **Description:**
     - Visualizes the average effect of a feature on predictions while marginalizing over other features.
     - Helps identify linear and non-linear relationships between features and outcomes.
   - **Strengths:**
     - Easy to interpret.
     - Provides insights into feature influence at a global level.
   - **Limitations:**
     - Assumes feature independence, which may not hold in real-world datasets.
2. **Permutation Feature Importance:**
   - **Description:**
     - Measures the importance of a feature by randomly shuffling its values and observing the impact on model performance.
   - **Formula:**
     - e_{perm}: Error with permuted values.
     - e_{orig}: Original error.
   - **Strengths:**
     - Model-agnostic.
     - Easy to implement.
   - **Limitations:**
     - May overestimate importance if features are correlated.
3. **Global Surrogate Models:**
   - **Description:**
     - Approximate a complex black-box model with an interpretable one (e.g., decision tree).
     - The surrogate model is trained on the same inputs and outputs as the original model.
   - **Strengths:**

    - Provides a simplified overview of the model behavior.
  - **Limitations:**
    - Accuracy of the surrogate depends on its ability to approximate the black-box model.

Local Explanations:

1. **Ceteris Paribus:**
   - **Description:**
     - Latin for "all else unchanged."
     - Analyzes the effect of changing a single feature on a prediction while keeping all other features constant.
   - **Strengths:**
     - Offers intuitive insights for individual predictions.
   - **Limitations:**
     - Assumes feature independence and ignores interactions.
2. **LIME (Local Interpretable Model-Agnostic Explanations):**
   - **Description:**
     - Creates a locally linear approximation of the model by perturbing the input data.
     - Assigns weights to perturbed samples based on proximity to the original input.
   - **Applications:**
     - Tabular, text, and image data.
   - **Strengths:**
     - Versatile and model-agnostic.
   - **Limitations:**
     - Approximation may not be faithful for highly non-linear models.
3. **SHAP (Shapley Additive ExPlanations):**
   - **Description:**
     - Based on cooperative game theory, assigns a fair contribution value to each feature for a prediction.
     - Considers all possible subsets of features.
   - **Strengths:**
     - Provides both global and local explanations.

        - Guarantees consistency and fairness in feature attribution.
    - **Limitations:**
        - Computationally expensive for models with many features.
4. **Counterfactual Explanations:**
    - **Description:**
        - Identifies the minimal changes needed to achieve a different prediction outcome.
        - Example:
            - Factual Rule: "Age 25 and income 900 Deny."
            - Counterfactual Rule: "Income > 900 Approve."
    - **Strengths:**
        - Offers actionable insights.
        - Highlights decision boundaries.
    - **Limitations:**
        - May require unrealistic changes for certain predictions.
5. **LORE (Local Rule-Based Explanations):**
    - **Description:**
        - Generates decision rules that explain individual predictions.
        - Constructs a synthetic neighborhood around the instance and extracts rules.
    - **Strengths:**
        - Provides interpretable and compact rules.
        - Highlights contrasts between the prediction and alternatives.
    - **Limitations:**
        - Relies on the quality of synthetic data generation.

5. XAI Applications

- Healthcare: Enhancing trust in diagnostic models.
- Finance: Ensuring fairness in credit scoring.
- Legal: Verifying model compliance with regulations.
- Corporate AI Governance: Explaining AI decisions to stakeholders.

6. Automated Machine Learning (AutoML)

Purpose:

- Streamlines model creation by automating data preprocessing, feature engineering, model selection, and hyperparameter tuning.

Challenges Addressed:

- Handling different data distributions, tasks (classification, regression), and metrics (accuracy, F1-score, AUC).
- Balancing class imbalance, sparsity, missing data, and irrelevant features.

Tasks Supported:

1. **Classification:** Binary, multi-class, and multi-label.
2. **Regression:** Predicting continuous outcomes.
3. **Feature Engineering:** Automatically identifying relevant features.

Methods Used:

- Decision Trees, Rule-Based Systems, Random Forests, Boosting, kNN, SVM, Deep Learning.

7. Key AutoML Techniques

Hyperparameter Optimization:

- Automates the tuning of model parameters.
- Techniques:
    1. Grid Search.
    2. Random Search.
    3. Bayesian Optimization.

Meta-Learning:

- "Learning to Learn":
    - Utilizes prior knowledge from related tasks to optimize new ones.
- Process:
    1. Extract meta-features (e.g., dataset size, feature count).
    2. Use these features to recommend optimal algorithms or settings.

8. Meta-Learning: Advanced Concepts

Goals:

- Automate the learning process.
- Improve model quality and understanding of performance factors.

Meta-Features:

- **Definition:** Meta-features describe the characteristics of a dataset, helping to inform machine learning processes in AutoML and meta-learning systems.
- **Types of Meta-Features:**
  1. **Data-Based:** Basic properties of the dataset, such as:
     - Number of instances, features, and missing values.
     - Statistical metrics (e.g., mean, variance, skewness).
  2. **Concept-Based:** Class-specific information:
     - Class imbalance ratios.
     - Intra-class variance or overlap of class distributions.
  3. **Model-Based:** Characteristics derived from trained models, like:
     - Depth and node counts of decision trees.
     - Support vector counts in SVM
  4. **Landmarking:** Performance of simple models as benchmarks:
     - Accuracy of k-Nearest Neighbors (k=1).
     - Runtime or performance of Naive Bayes.

Applications:

- Algorithm selection and ranking.
- Performance prediction for new datasets.

9. Future Directions

- Integrating XAI with AutoML for both transparency and automation.
- Developing more robust meta-learning frameworks.
- Enhancing trustworthiness in AI through explainability and automation.

LECTURE 10 Neuro-fuzzy systems

1. Explainable Artificial Intelligence

Definition: Explainable artificial intelligence (XAI) refers to AI systems where humans can understand the reasoning behind decisions. This involves transparency and interpretability.

Key Characteristics:

- **Transparency**: Users can see how inputs lead to outputs.
- **Interpretability**: Models must be comprehensible.
- **Applications**:
    - Medicine: AI for diagnostics.
    - Defense: Decision support systems.
    - Finance: Risk analysis and fraud detection.
    - Law: Legal document processing.
    - Autonomous vehicles: Decision transparency for navigation.

Regulatory Context: The General Data Protection Regulation (GDPR) mandates that individuals have the right to explanation (Article 22), prohibiting decisions based solely on automated processing that significantly impact individuals.

2. Intelligence and Artificial Intelligence

Definition of Intelligence:

1. The ability to correctly answer questions never asked before.
2. The ability to acquire and apply skills.
3. The ability to achieve complex goals.

Definition of Artificial Intelligence (AI): Intelligence that operates on non-biological substrates, enabling systems to interpret data, build knowledge, and apply it to achieve specific goals.

Types of AI:

1. **Weak (Narrow) AI**: Focused on solving a specific problem (e.g., chess, autonomous cars).
2. **Strong (General) AI**: Capable of solving any problem with human-like intelligence.
3. **Superintelligence**: Exceeds human intelligence, potentially leading to singularity.

Model Types:

- **White Box Models**:
    - Transparent and interpretable by humans.
    - Clear decision flow with known contributions of features.

- **Black Box Models**:
    - Complex and difficult to interpret.
    - Known inputs and outputs, but unknown decision-making processes.

3. Fuzzy Sets and Fuzzy Logic

Core Concepts:

- A fuzzy set represents elements with degrees of membership rather than binary (0 or 1) inclusion.
    - Membership Function (μ): Maps elements in a space to a value between 0 and 1, indicating degree of membership.

Definitions:

1. **Set**: A collection of distinct elements.
    - Example: A = {a, b, c}.
2. **Temperature**:
    - Categories: cold, warm, and hot.
    - Each category has varying degrees of membership.
3. **Fuzzy Set**:
    - A set where membership of elements is gradual, not binary.
    - Defined explicitly with a membership function or as ordered pairs (x, μA(x)).
4. **Core**: The set of elements with membership value equal to 1.
5. **Support**: The set of elements with membership values greater than 0.

Examples of Fuzzy Sets:

- **Triangular Membership Function**: Defined by three points (a, b, c), forming a triangle.
- **Trapezoidal Membership Function**: Defined by four points (a, b, c, d).
- **Gaussian Membership Function**: Smooth, bell-shaped curve.
- **Sigmoid Membership Function**: S-shaped curve.

Operations on Fuzzy Sets:

1. **T-norms (Intersection)**: Models "AND" relationships (e.g., μA ∩ B(x) = min(μA(x), μB(x))).
2. **S-norms (Union)**: Models "OR" relationships (e.g., μA ∪ B(x) = max(μA(x), μB(x))).
3. **Negation**: Models "NOT" (μ¯A(x) = 1 - μA(x)).

Fuzzy Implications:

- Rules of inference expressed as "If-Then" statements.

    - **Zadeh Implication**: Considers the minimum or maximum of values.
    - **Łukasiewicz Implication**: Considers bounded sums of values.
    - **Reichenbach Implication**: Incorporates probabilistic elements.

Extension Principle:

- Extends crisp functions to fuzzy domains.
    - Given , fuzzy set A in maps to fuzzy set B in :

Type-2 Fuzzy Sets:

- Handle uncertainty in membership functions.
    - **Primary membership**: Degree of membership.
    - **Secondary membership**: Degree of uncertainty.

4. Fuzzy Systems

Components:

1. **Fuzzification Block**:
    - Converts crisp inputs into fuzzy values.
    - Techniques:
        - Fuzzy singletons.
        - Membership functions (triangular, Gaussian).
2. **Fuzzy Rule Base**:
    - Consists of linguistically interpretable "If-Then" rules.
        - Example: *If temperature is high, then fan speed is fast*.
        - Premises: Inputs represented by fuzzy sets.
        - Consequences: Outputs represented by fuzzy sets.
        - **Premise Definition**: The input condition (e.g., "temperature is high") evaluated by membership functions.
3. **Defuzzification Block**:
    - Converts fuzzy outputs into crisp values.
    - Methods:
        - **Center of Gravity (COG)**: Calculates weighted average of membership values.
        - **Maximum Methods**: Uses maxima of the membership function (e.g., Small of Maxima, Median of Maxima).

System with Parameterised Consequences:

- These systems utilize parameterized rules for their consequences, often integrating artificial neural network (ANN) principles.
- **Features**:
  1. Premises are represented using Gaussian membership functions.
  2. Premises are aggregated using T-norm (product).
  3. Consequences are modeled with triangular fuzzy sets.
  4. Core localization of fuzzy sets in the consequences is derived from a combination of input attribute values.
  5. Aggregation operators like Modified Indexed Center of Gravity (MICOG) are employed.
- **Examples**:
  - Systems such as ANNBFIS (Artificial Neural Network-Based Fuzzy Inference System) and ANLIR (Artificial Neural Network Logical Interpretation of Rules).
  - These systems allow logical interpretation of rules using methods like Reichenbach implication or other logical operators.

Number of Rules:

- The number of rules in a fuzzy system depends on the partitioning of the input domain and the complexity of the system.
- Common methods for determining the number of rules include:
  - **Grid-Based Partitioning**: Dividing the input domain into equal regions.
  - **Scatter-Based Partitioning**: Using clustering techniques to identify rule regions.
  - **Hierarchical Partitioning**: Using multi-level structures to generate rules dynamically.
- Some systems (e.g., hierarchical systems) adaptively determine the number of rules.

Elaboration of Consequences:

- **Mamdani Systems**:
  - Use clustering techniques to derive the consequences for rules.
- **TSK Systems**:
  - Utilize linear regression to model consequences as functions of inputs.
- **Goal**:
  - Ensure the consequences align with observed data while maintaining interpretability.

Tuning of System Parameters:

- Tuning is critical for optimizing fuzzy systems and involves:
    - **Optimization Scope**:
        - Local: Adjusting parameters of specific rules or regions.
        - Global: Optimizing the entire system.
        - Combined: Employing both local and global techniques.
    - **Optimization Techniques**:
        - Gradient descent.
        - Simulated annealing.
        - Genetic algorithms.
        - Least squares.
    - **Data Presentation**:
        - Batch: Using all data at once.
        - Online: Incremental learning from data streams.

Types of Fuzzy Systems:

- **Mamdani-Assilan (MA)**:
    - Premises use triangular fuzzy sets with T-norms (e.g., min).
    - Consequences use triangular fuzzy sets with S-norms (e.g., max).
- **Takagi-Sugeno-Kang (TSK)**:
    - Premises use triangular fuzzy sets.
    - Consequences are linear functions or fuzzy singletons.

Comparison:

- MA: High interpretability, lower precision.
- TSK: High precision, lower interpretability.

5. Interpretability in Fuzzy Systems

Two Principal Objectives of Fuzzy Models:

1. **Precision**: Capturing and modeling complex phenomena accurately. This is exemplified by precise fuzzy modeling (e.g., TSK models).
2. **Interpretability**: Presenting information in a way that humans can easily understand (e.g., Mamdani models).

Definition of Interpretability: Interpretability refers to the ability of a fuzzy system to present its rules and decision-making process in a way that is comprehensible to humans. It is a crucial characteristic of fuzzy models, enabling trust and validation by human experts.

High-Level Interpretability:

- Features:
  - Few rules.
  - Simple and clear rules.
  - Transparent rule base.
  - Cohesion (similar premises lead to similar consequences).
- Objective: Prioritizes understandability, often at the expense of precision.

Low-Level Interpretability:

- Features:
  - Moderate number of membership functions.
  - Full coverage of the input domain.
  - Normalized membership functions (each reaches 1).
  - Complementarity (sum of all membership functions is 1 at any input value).
- Objective: Ensures detailed and comprehensive modeling of input-output relationships.

6. Neuro-Fuzzy Systems

Definition:

- Combines **fuzzy systems** (to model imprecision) and **artificial neural networks (ANNs)** (to generalize and train from data).

Key Features:

- Learns fuzzy rules from data.
- Identifies system parameters such as the number of rules, premises, and consequences.
- Optimizes system parameters using techniques like gradient descent and genetic algorithms.

Partitioning Input Domains:

- **Grid-Based**: Uniform partition.
- **Scatter-Based**: Data clustering.
- **Hierarchical**: Multi-level partitioning.

Examples:

- **ANFIS (Adaptive Network-Based Fuzzy Inference System)**:
    - TSK model with Gaussian membership functions.
    - Premises optimized using gradient descent.
    - Consequences optimized using least squares.

Advantages:

- Models data imprecision.
- Combines interpretability and generalization.
- Incorporates expert (human) knowledge.

7. Summary

1. **Explainable AI**:
    - Emphasizes transparency and interpretability.
    - Crucial for applications in sensitive domains (e.g., medicine, law).
2. **Fuzzy Sets and Logic**:
    - Handle imprecision through membership functions.
    - Operations like T-norms, S-norms, and negation allow flexible modeling.
    - Type-2 fuzzy sets handle uncertainties in membership functions.
3. **Fuzzy Systems**:
    - Comprise fuzzification, rule base, and defuzzification blocks.
    - Employ models like Mamdani (interpretable) and TSK (precise).
    - System parameters (e.g., number of rules, premises, consequences) are tuned for performance.
4. **Interpretability**:
    - High-level interpretability prioritizes simple, comprehensible rules.
    - Low-level interpretability ensures detailed and precise modeling.
5. **Neuro-Fuzzy Systems**:
    - Combine fuzzy logic and neural networks.
    - Learn and generalize from data while retaining interpretability.
    - Applications include ANFIS and other hybrid models.
6. **Optimization Techniques**:
    - Global and local optimization methods enhance system tuning.

- Methods like gradient descent, genetic algorithms, and least squares refine model performance.

7. **General Importance**:
    - Fuzzy systems handle complexity and imprecision effectively.
    - Neuro-fuzzy systems adapt and generalize efficiently, bridging expert knowledge and data-driven insights.
