PS 643 - August 16, 2024

Supervised ML and Introduction to Python

Annotations in Supervised ML:

- **Human Annotations**: Can be unreliable even with inter-annotator agreement (e.g., detection of a suspicious person).
- **Natural Annotations**: Derived from the natural world or the nature of the data itself.

Determining the Best Algorithm:

- Best algorithm isn't always the most powerful.
- Depends on application needs (e.g., resource efficiency, explainability).

Bias-Variance Tradeoff:

- **Variance**: Degree of data dispersion.
- Balance between bias and variance is important.

Model Complexity:

- Consideration of computational costs of algorithms.

K-Nearest Neighbors (KNN):

- Simple and high-performing.
- No training time; uses existing data.
- Example: Comparing photos to find the closest match.

Decision Trees:

- Finds the feature which divides the dataset such that the remaining dataset is more homogenous.
- Repeatedly finds the best feature split.

Bayesian Classifier:

- Uses Naive Bayes conditional probability.
- Assumes that the feature does not affect other features. This assumption is not true but it makes the maths very simple and it works.

Assumptions in ML:

- **Linearity**: Many models assume linear relationships, but non-linear classifiers are needed when this assumption doesn't hold.

Neural Networks:

- Requires at least three layers to model both linear and non-linear relationships.

Support Vector Machine (SVM):

- Binary algorithm using kernels to classify data.
- Disadvantage: Only handles binary problems; multi-class requires multiple binary classifiers, working in a sequence.

Regression vs. Classification:

- **Regression**: Continuous outputs.
- **Classification**: Discrete outputs.

Unsupervised ML:

- **Clustering**: Finds internal groupings in unannotated data.
- **Parametric Algorithms**: User provides the number of clusters (hyperparameters).
- **Non-Parametric Algorithms**: No predefined parameters (e.g., Bayesian distribution, chinese restaurants and indian buffets).

Topic Modeling:

- Can I find the thematic clusters from 1000 question papers of IITB.
- Identifies thematic clusters in large datasets.
- Determines the ratio of each theme in documents.

Anomaly Detection:

- Identifies data points that don't fit established patterns.
- Example: Detecting suspicious activity in a classroom.

GANs (Generative Adversarial Networks):

- **Generator and Discriminator**: Compete to improve each other.
- Generator creates data, discriminator identifies real vs. fake.

Algorithm

- An algorithm is a step-by-step procedure for solving a problem or performing a task, typically executable by a machine.

Machine Languages

- Very close to the hardware. Not used today.

Assembly Language

- Converts machine language into human-understandable commands (e.g., `ADD`, `SUB`).
- Still used in embedded systems.

High-Level Languages

- Fully human-readable, built on top of assembly language.
- Example: Python.

Python

- **Easy and Inefficient**: Python is known for its simplicity but is less efficient than other high-level languages.

Compilers vs. Interpreters

- **Compilers**: Convert entire code at once (batch processing).
- **Interpreters**: Execute code line by line (sequential processing).

Python Basics

- You can run Python scripts using IDLE or the command line.
- **Comments**: For human readability.

```
# Single-line comment

'''
```

Multiline comment

'''

Keywords and Identifiers: Keywords are reserved words and cannot be used as variable names.

```
and = 10  # This will cause an error
```

Printing and Input: `print()` outputs text, and `input()` takes user input.

```
print("Hello, World!")  # Outputs "Hello, World!"
name = input("Enter your name: ")  # Prompts user to enter a name
```

Basic Math Operators:

- `+, -, /, *`
- `**` for exponentiation
- `%` for remainder

```
result = 10 + 5  # Adds 10 and 5
power = 2 ** 3  # 2 raised to the power of 3
remainder = 10 % 3  # Remainder when 10 is divided by 3
```

Data Types:

- `int` for integers
- `float` for decimal numbers
- `str` for strings (text)
- `list` for collections of items

```
x = 5      # Integer (int)
y = 3.14    # Floating-point number (float)
name = "John"  # String (str)
lst = [1, 2, "apple", 4.5]  # List with mixed data types
```

String Indexing: Access specific characters in a string using indices.

```
text = "Hello, World!"
char = text[5]  # Returns the 6th character (indexing starts at 0)
substring = text[2:8]  # Returns characters from index 2 to 7
```

```
start_to_10 = text[:10]  # Returns first 10 characters
from_10_to_end = text[10:]  # Returns characters from index 10 to the end
last_5_chars = text[-5:]  # Returns the last 5 characters
```

Lists: Lists are collections of items that can contain mixed data types.

```
lst = ["apple", "banana", "cherry"]  # A list with 3 items
lst.append("potato")  # Adds 'potato' to the end of the list
lst.remove("banana")  # Removes the first occurrence of 'banana'
combined_list = lst + ["orange", "grape"]  # Concatenates two lists
length = len(lst)  # Returns the number of items in the list
```

String and List Operations:

- `split()` breaks a string into a list based on a delimiter (default is space).
- `" ".join()` combines list elements into a single string with a specified separator.

```
sentence = "This is a sentence"
words = sentence.split()  # Splits sentence into a list of words
joined_sentence = " ".join(words)  # Joins the list back into a string
```

Boolean Variables and Relational Operators:

- `True` and `False` are boolean values.
- `>` (greater than), `<` (less than), `==` (equal to), `!=` (not equal to) are relational operators.
- `and`, `or`, `not` are logical operators.

```
flag = True  # Boolean value
x = 30

if x > 25 and not flag:
    print("Hi")
```

Shorthand Operations: Use shorthand for arithmetic operations.

- `+=`, `-=`, `*=`, `/=` are shorthands for updating a variable's value.

```
x = 10
x += 5  # Adds 5 to x (x = x + 5)
```

Conditional Statements: Execute code based on conditions.

```
x = 150
if x < 100:
    print("Here")  # Executed if x is less than 100
elif x < 200:
    print("Where")  # Executed if x is between 100 and 199
else:
    print("There")  # Executed if x is 200 or more
```