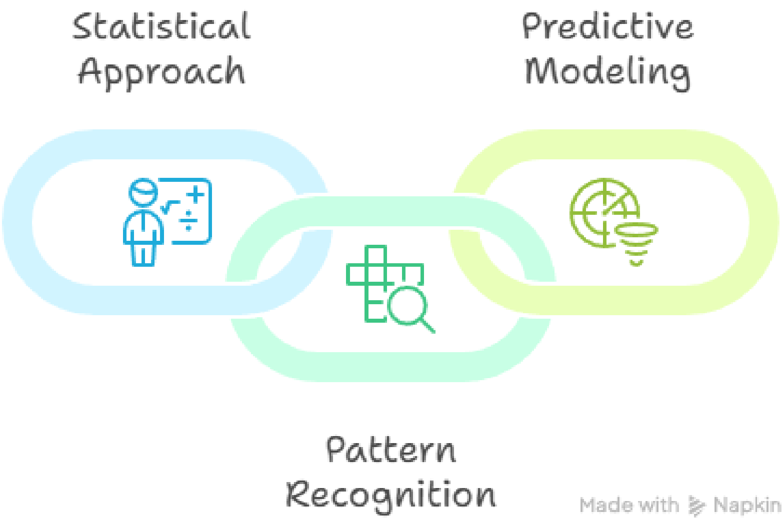
Machine Learning Project Pipeline

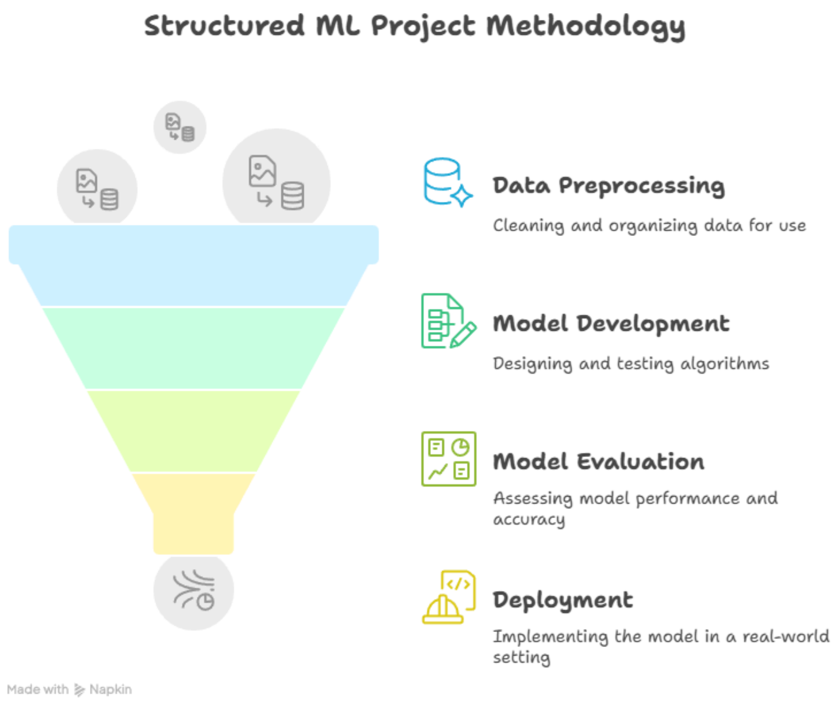
What is Machine Learning (ML)?

Machine Learning is a **statistical approach** that enables systems to **learn patterns from data** and predict outcomes on new, unseen data. It's the brain behind recommendation systems, fraud detection, and even self-driving cars!



Why a Project Methodology?

Building a successful ML model isn't just about algorithms—it's about following a **structured process** that ensures reliability, accuracy, and business value.



Popular Methodologies in ML Projects:

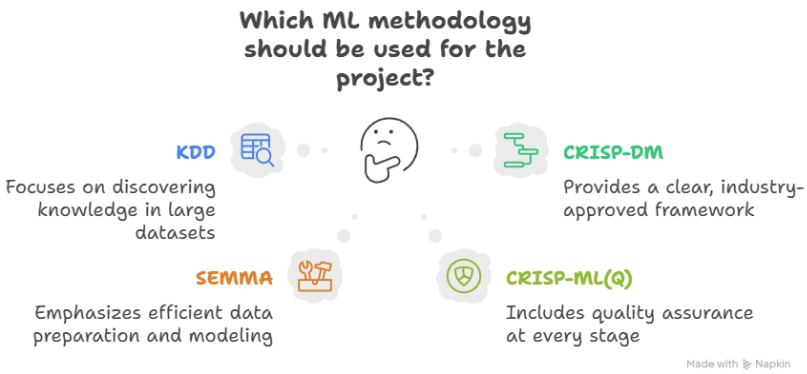
KDD (Knowledge Discovery in Databases)
Focuses on identifying useful knowledge and patterns hidden in large data sets.

CRISP-DM (Cross-Industry Standard Process for Data Mining)
The most commonly used methodology that provides a clear, industry-approved framework.

SEMMA (Sample, Explore, Modify, Model, Assess)
A process designed by SAS that focuses on preparing and modeling data efficiently.

CRISP-ML(Q): Cross-Industry Standard Process for Machine Learning with Quality Assurance

An **enhanced version of CRISP-DM,** this method introduces **quality checks** at every stage to ensure the solution is production-ready and robust.



Six Steps in CRISP-ML(Q):

1. Business & Data Understanding
Understand the business goal and gather the relevant data.

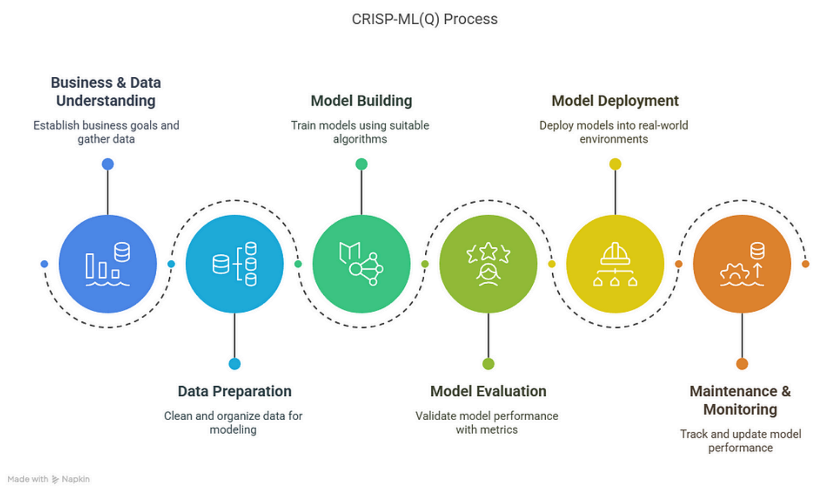
2 . Data Preparation / Preprocessing
Clean and organize data to make it ready for modeling.

3. Model Building
Train machine learning models using suitable algorithms.

4. Model Evaluation
Validate model performance using appropriate metrics (accuracy, precision, recall, etc.).

5. Model Deployment
Deploy the model into a real-world environment for use.

6. Maintenance & Monitoring
Track model performance over time and make updates when needed.



1. Business & Data Understanding

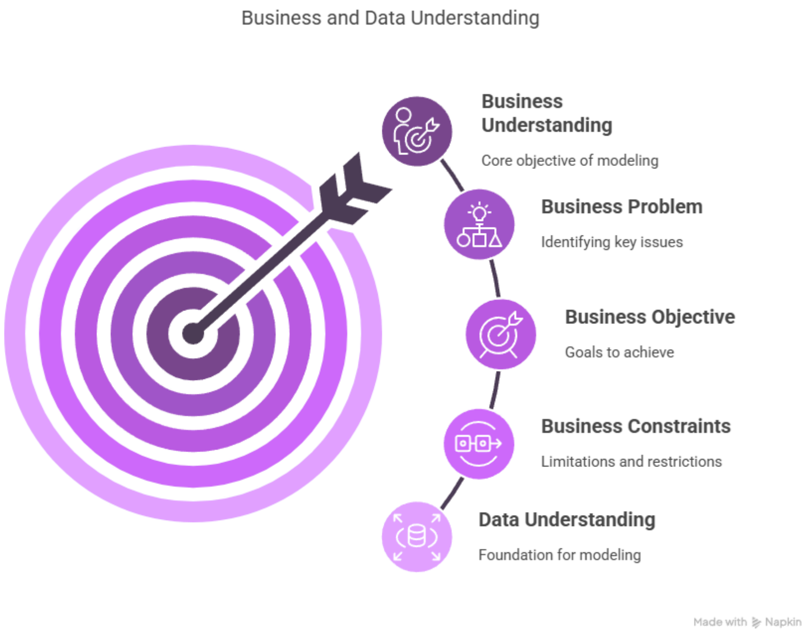
i) Business Understanding
First, we need to know *why* we're building a model.

- **Business Problem**: Are sales dropping? Is customer churn rising?
  👉 *Example*: A telecom company wants to *predict which customers are likely to leave (churn).*
- **Business Objective**: Reduce churn rate and increase customer loyalty.
- **Business Constraints**: Budget, time, legal rules, or data access limitations.

ii) Data Understanding
Know your data before modeling!

- **Data Collection:** From databases, APIs, CSV/Excel files, or web scraping.
  👉 *Example*: Fetch customer details from a SQL database + recent interactions via API.
- **Data Description**:
- Shape (e.g., 10,000 rows × 15 columns)
- Data types: numerical (e.g., age), categorical (e.g., gender)
- **Input vs Output:**
  Inputs = customer features

Output = churn (yes/no)

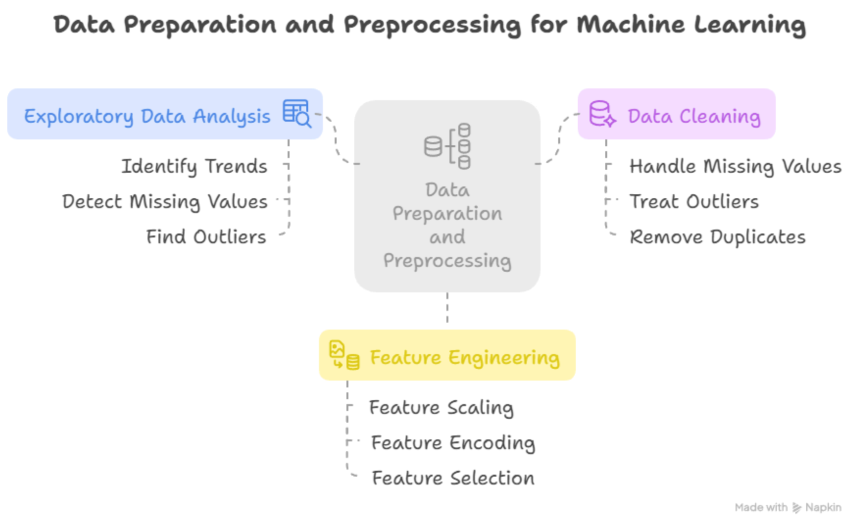


2. Data Preparation / Preprocessing

Data should be **clean and machine-readable**!
📌 *ML needs numerical, structured input!*

- **EDA (Exploratory Data Analysis)**
  Identify trends, patterns, missing values, outliers, duplicates.
  👉 *Example*: Finding that customers with high monthly charges churn more.
- **Data Cleaning:**
- Handling **missing values** using mean, median, forward-fill, or advanced imputation like KNN.
- **Outlier treatment:** Use algorithms like Linear Regression or SVM.
- Remove duplicates & fix type mismatches.
- **Feature Engineering:**
- **Feature Scaling:** Standardization, normalization
- **Feature Encoding:** One-hot encoding for categories

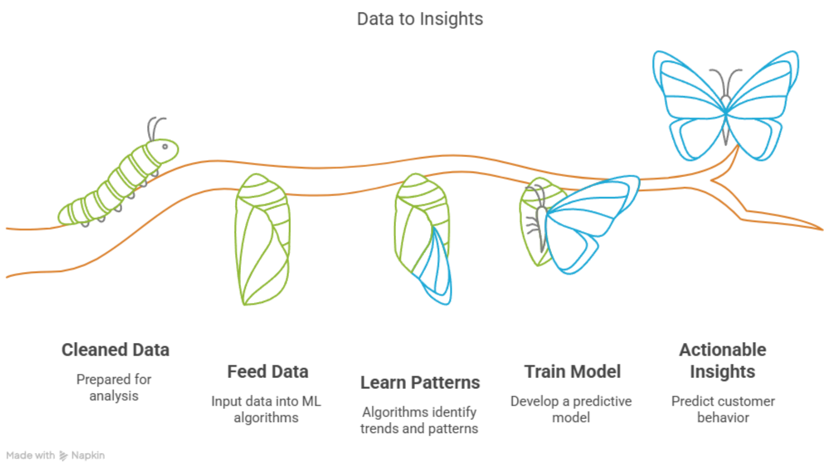
- **Feature Selection:** Keep only the most useful features.



3. Model Building

Feed cleaned data into ML algorithms to **learn patterns.**

👉 *Example*: Train a Random Forest model to classify whether a customer will churn or not.



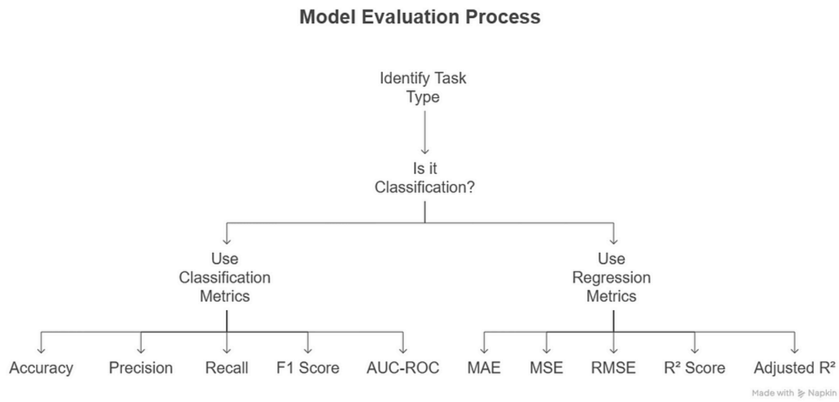
4. Model Evaluation

Check how well the model is performing using metrics:

- **Classification Tasks** (like churn prediction):

- Accuracy, Precision, Recall, F1 Score, AUC-ROC
- **Regression Tasks** (like predicting house price):
- MAE, MSE, RMSE, R^2 Score, Adjusted R^2

👉 *Example*: Your churn model has 89% accuracy and high recall—great for capturing churners!

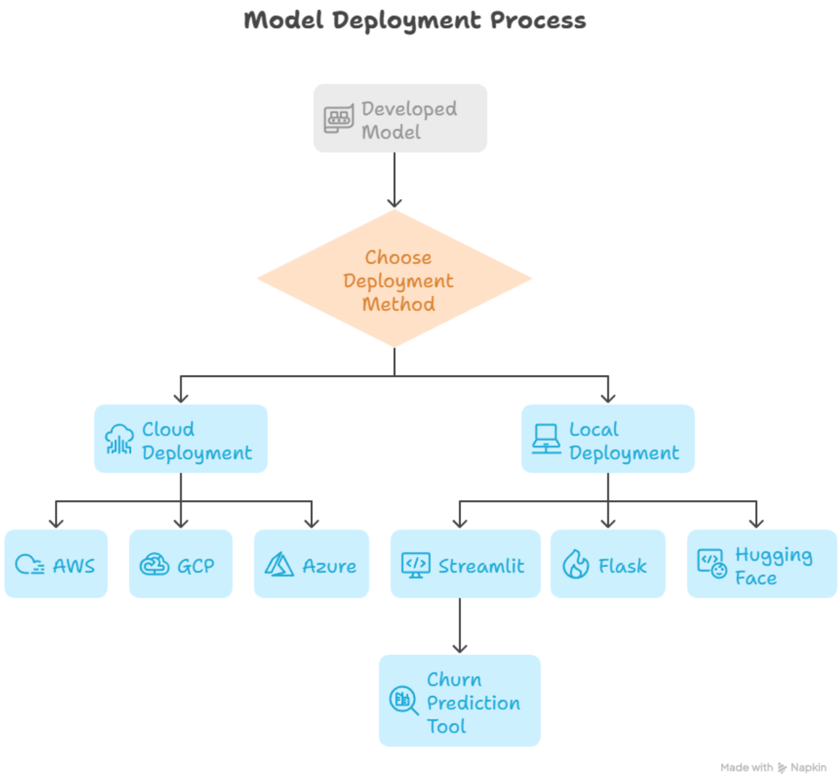


5. Model Deployment

Make your model available to users.

- **Cloud Deployment**: AWS, GCP, Azure
- **Local Deployment:** Streamlit, Flask, Hugging Face

👉 *Example*: Host the churn prediction tool on **Streamlit**, where sales staff can upload customer info and get predictions in real-time!

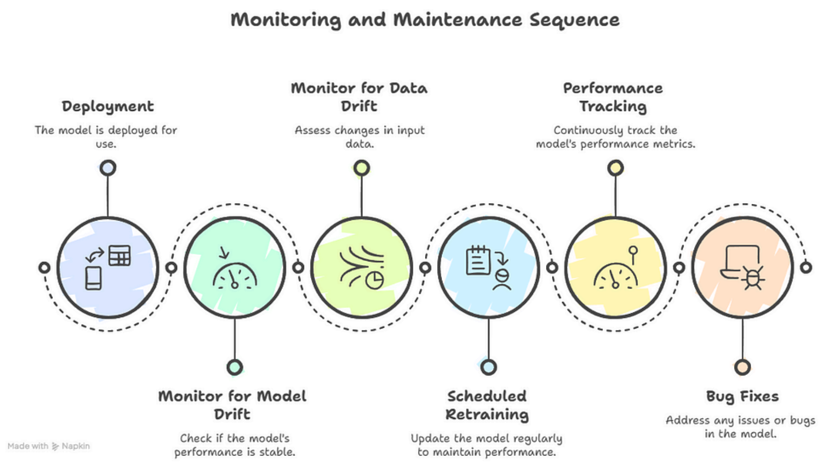


6. Monitoring & Maintenance

Once deployed, continuously monitor for:

- **Model Drift**: Is the model still performing well?
- **Data Drift**: Is the input data changing?
- **Scheduled retraining**, performance tracking, bug fixes.

👉 *Example*: Over time, customer behavior changes. So the model is retrained every 3 months using updated data.



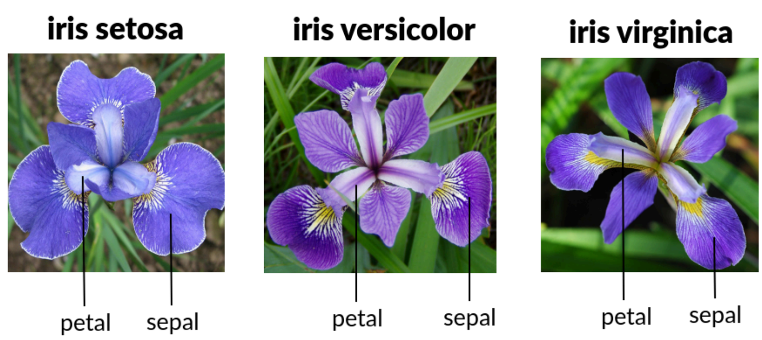
Example: Predicting Iris Flower Species

We'll use the **Iris Dataset** for this classification task.

1. Business & Data Understanding

Business Understanding:

Goal: Predict the species of Iris flowers based on features like sepal length, sepal width, petal length, and petal width.
Objective: Classify the Iris flower species into one of the three classes: *Iris-setosa*, *Iris-versicolor*, or *Iris-virginica*.
Constraint: Handle missing values in the dataset and ensure accurate classification within a limited timeframe.



Data Understanding:

```
import pandas as pd

# Load dataset
df = pd.read_csv(r"D:\IND 2\Downloads\Datasets\Datasets\Iris.csv")

# Overview
print(df.shape)
print(df.dtypes)
df.head()
```

```
(166, 6)
Id               float64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1.0	5.1	3.5	1.4	0.2	Iris-setosa
1	2.0	4.9	3.0	1.4	0.2	Iris-setosa
2	3.0	NaN	3.2	1.3	0.2	Iris-setosa
3	4.0	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	5.0	3.6	1.4	0.2	Iris-setosa

📌 **Key Notes:**

Target: Species 🌸

Inputs: SepalLengthCm 📏, SepalWidthCm 📏, PetalLengthCm 🌸, PetalWidthCm 🌸

2. Data Preprocessing / Preparation

EDA + Cleaning:

```
# Drop the 'Id' column
df = df.drop('Id', axis=1)

# Fill missing values in numerical columns using mean
df['SepalLengthCm'] = df['SepalLengthCm'].fillna(df['SepalLengthCm'].mean())
df['SepalWidthCm']  = df['SepalWidthCm'].fillna(df['SepalWidthCm'].mean())
df['PetalLengthCm'] = df['PetalLengthCm'].fillna(df['PetalLengthCm'].mean())
df['PetalWidthCm']  = df['PetalWidthCm'].fillna(df['PetalWidthCm'].mean())

# Fill missing values in categorical column 'Species' using mode
df['Species'] = df['Species'].fillna(df['Species'].mode()[0])

# Check for missing values
print(df.isnull().sum())
print()
# Drop duplicate rows
df = df.drop_duplicates()

print(df.duplicated().sum())
```

```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64

0
```

```
# Display the updated DataFrame
df
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	6.6	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

146 rows × 5 columns

3. Model Building

```
from sklearn.model_selection import train_test_split

# Features (independent variables)
X = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]

# Target (dependent variable)
Y = df['Species']

# Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=29)
```

```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()
lr.fit(X_train,Y_train)
```

```
LogisticRegression
LogisticRegression()
```

4. Model Evaluation

```
from sklearn.metrics import f1_score, precision_score, recall_score, accuracy_score

# Calculate F1 Score
f1 = f1_score(Y_test, Y_pred, average='weighted')
print(f'F1 Score: {f1:.2f}')

# Calculate Precision
precision = precision_score(Y_test, Y_pred, average='weighted')
print(f'Precision: {precision:.2f}')

# Calculate Recall
recall = recall_score(Y_test, Y_pred, average='weighted')
print(f'Recall: {recall:.2f}')

# Calculate Accuracy
accuracy = accuracy_score(Y_test, Y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

```
F1 Score: 0.93
Precision: 0.93
Recall: 0.93
Accuracy: 93.33%
```

Metrics for Iris Species Prediction:

- **Accuracy:** Around **93.33%**
- **F1 Score: 0.93**
- **Precision: 0.93**
- **Recall: 0.93**

5. Model Deployment

```
SepalLengthCm = 5.5
SepalWidthCm = 4.1
PetalLengthCm = 4.0
PetalWidthCm = 1.2
lr.predict([[SepalLengthCm,SepalWidthCm,PetalLengthCm,PetalWidthCm]])
```

```
array(['Iris-versicolor'], dtype=object)
```

6. Monitoring & Maintenance

```
import pickle
with open(r"D:\Machine learning\model.pkl",'wb') as file:
    pickle.dump(lr,file)      # Serializing/saving the model
```

```
SepalLengthCm = 5.5
SepalWidthCm = 4.1
PetalLengthCm = 4.0
PetalWidthCm = 1.2
```

```
with open(r"D:\Machine learning\model.pkl",'rb') as file:
    model = pickle.load(file)

model.predict([[SepalLengthCm,SepalWidthCm,PetalLengthCm,PetalWidthCm]])
```

```
array(['Iris-versicolor'], dtype=object)
```

Conclusion:

Track accuracy and recall monthly to detect any performance drift for predicting Iris species.
Set up alerts if the model's performance falls below the desired threshold (e.g., accuracy < 80% or recall < 0.80).
Retrain quarterly with updated Iris data to ensure the model remains accurate and adaptable to any new data patterns.

Pickle simplifies the process of saving and loading your trained model. **Deploy the saved model** for real-time predictions, ensuring you can classify Iris species efficiently with new input data.