**Predictive Pulse: Harnessing Machine Learning for Blood Pressure Analysis**

**INTRODUCTION:**

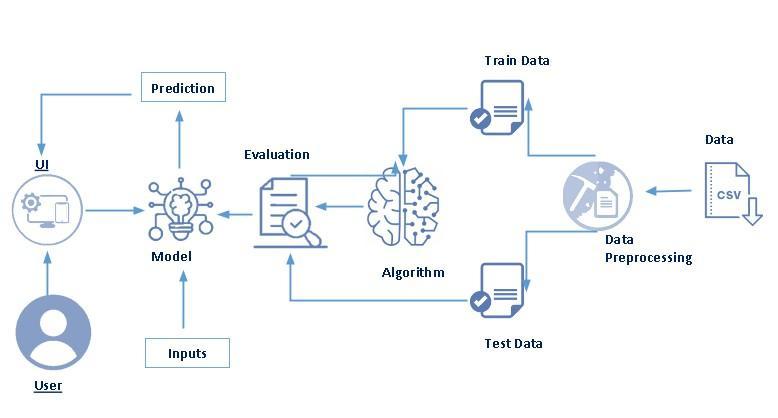
**Overview:**

Predictive Pulse is an innovative project harnessing machine learning algorithms to analyze and predict blood pressure fluctuations. This cutting-edge technology integrates seamlessly with wearable devices or health monitoring systems, continuously collecting real-time physiological data like heart rate, activity levels, and other pertinent biometrics. This data fuels advanced machine learning models, facilitating the analysis of patterns and trends to forecast changes in blood pressure.

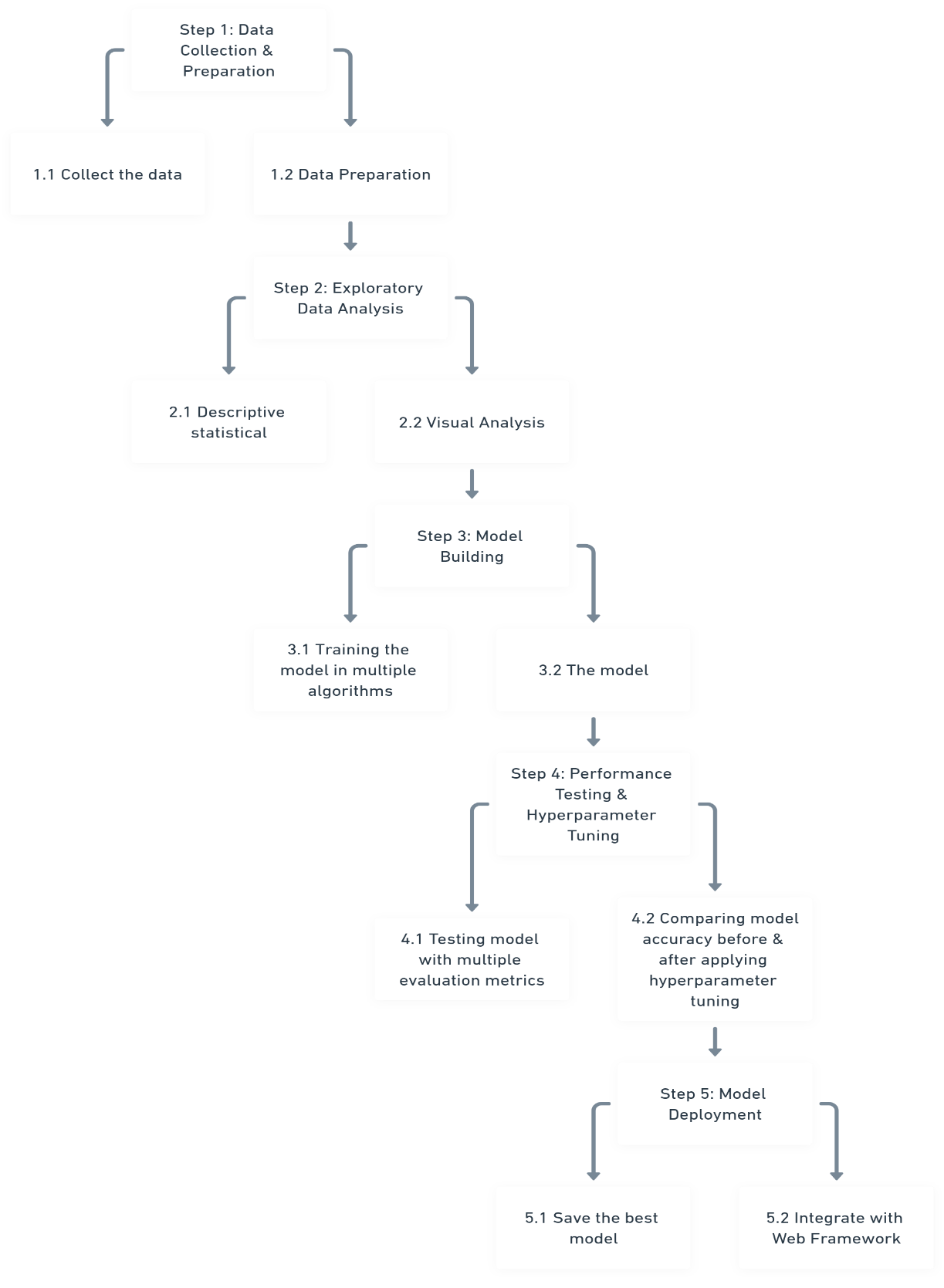
Scenario 1: A patient managing hypertension wears a compatible wearable device featuring Predictive Pulse technology. Throughout the day, the device monitors their vital signs and transmits data securely. If the machine learning model identifies a potential spike in blood pressure based on observed patterns, it promptly alerts the patient and their healthcare providers. This real-time notification enables swift intervention or medication adjustments, preventing potential complications.

Scenario 2: A fitness enthusiast relies on a smartwatch equipped with Predictive Pulse capabilities to track their health and performance. The machine learning model analyzes their blood pressure trends over time, offering personalized insights and recommendations. These insights help optimize their workouts and lifestyle choices, promoting cardiovascular health and minimizing potential health risks.

Scenario 3: A healthcare provider oversees a population health initiative focused on preventing cardiovascular diseases among at-risk individuals. Leveraging Predictive Pulse technology, they remotely monitor patients and identify those at higher risk of developing hypertension or experiencing blood pressure fluctuations. This data-driven approach enables targeted interventions such as lifestyle modifications, medication adherence reminders, or telehealth consultations, effectively managing and preventing complications.

**Technical Architecture:**  


**PROJECT FLOW:**



**Data Collection & Preparation**

Data collection is fundamental to machine learning, providing the raw material for training algorithms and making predictions. This process involves gathering relevant information from various sources such as databases, surveys, sensors, and web scraping. The quality, quantity, and diversity of collected data significantly impact the performance and accuracy of ML models.

patient\_data.csv - <https://drive.google.com/file/d/1qYvKqg4w_w4blizSVqmLvwY25m7V7N3_/view?usp=sharing>

**Importing the libraries**

Import the necessary libraries like numpy, pandas, matplotlib,seaborn

**Read the Dataset**

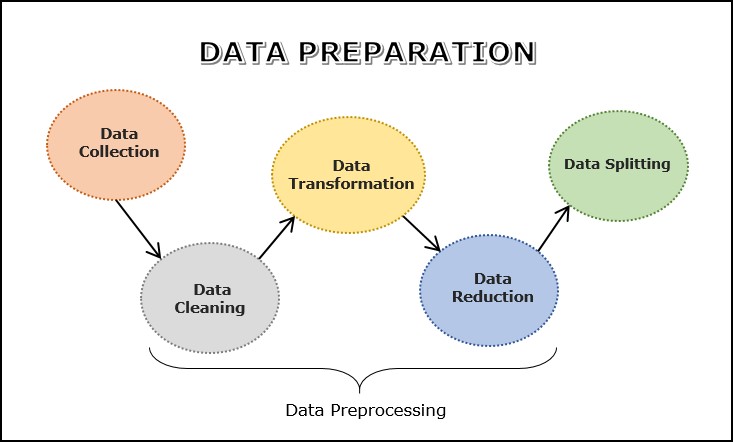
Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.   
In pandas, we have a function called read\_csv() to read the dataset. As a parameter, we have to give the directory of the CSV file.

**Data Preparation**

Before we can use our data to teach our machine-learning model, we need to clean it up. That means we have to deal with missing information, like when there's no data for some entries. We also have to figure out what to do with categories, like types of stages and outliers, which are unusual data points. This activity includes the following steps. 

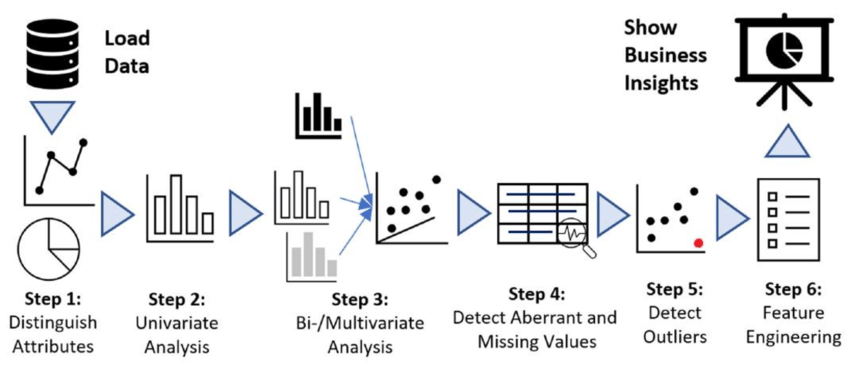
Handling missing values

 Handling categorical data



**Exploratory Data Analysis**

Exploratory Data Analysis (EDA) is the process of analyzing datasets to summarize their main characteristics, often using visualizations and descriptive statistics. It helps in understanding the data's structure, distribution, and any patterns or anomalies before applying machine learning algorithms.



**Descriptive statistical**

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas have a worthy function called describe. With this described function we can understand the unique, top, and frequent values of categorical features. And we can find mean, std, min, max, and percentile values of continuous features.

**Visual analysis**

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

**Univariate analysis**

In simple words, univariate analysis is understanding the data with a single feature. Here we have displayed two different graphs such as distplot and countplot.  
We check the number of females and males present in our dataset. Using matplotlib library and using pie chart we check the count.

**Bivariate analysis**

To find the relation between two features we use bivariate analysis. Here we are visualizing the relationship between two different Features.

The relationship between 'Take Medication' and 'Severity' suggests a potential correlation between medication adherence and the severity of the medical condition. Analyzing the data might reveal whether patients who take their medication regularly tend to experience lower severity levels compared to those who do not. Understanding this relationship could inform healthcare professionals about the effectiveness of prescribed medications in managing the condition and the importance of adherence to treatment protocols for better outcomes.

**Multivariate analysis**

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used pairplot from the seaborn package.

Splitting data into train and test

split the Dataset into train and test sets. First, split the dataset into x and y and then split the data set. Here x and y variables are created. On the x variable, df is passed by dropping the target variable. And on y target variable is passed.

**Model Building**

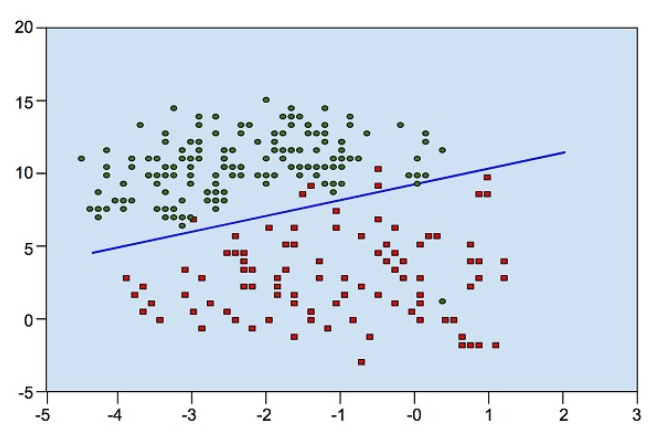
Model building in machine learning refers to the process of creating a mathematical representation of a real-world process using algorithms and data. This involves training a machine learning algorithm on a dataset to identify patterns, relationships, and predictions based on input data.



**Training and testing the models using multiple algorithms**

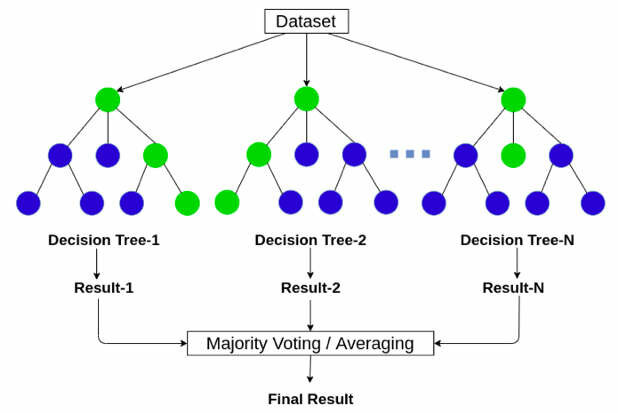
Now our data is cleaned and it’s time to build the model. We can train our data on different algorithms. For this project, we are applying three Regression algorithms. The best model is saved based on its performance

**Logistic Regression Model**

A variable named logistic\_regression is created and train and test data are passed as the parameters. Inside the function, the Linear Regression algorithm is initialized and training data is passed to the model with.fit() function. Test data is predicted with. predict() function and save it in a new variable. For evaluating the model, an accuracy score and classification report are used. 

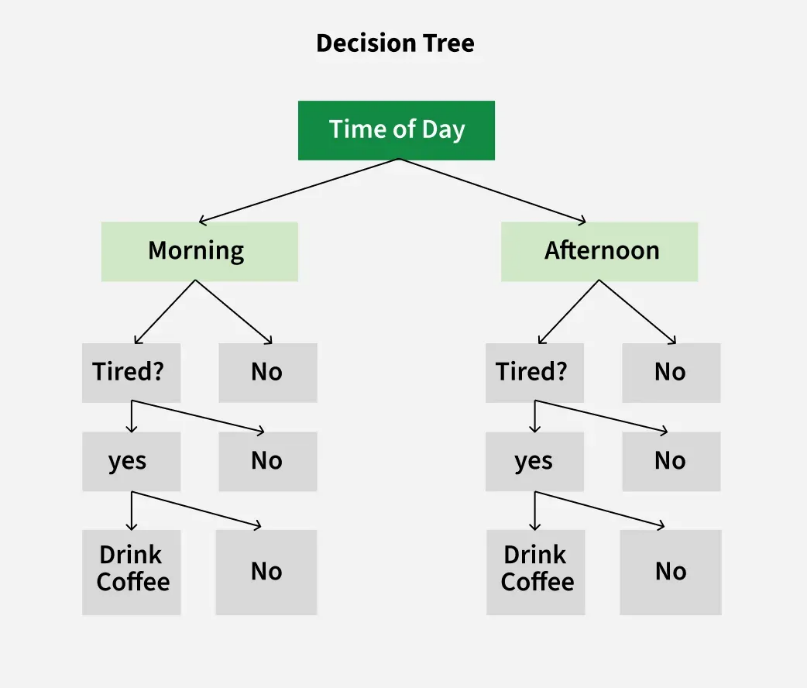
**Random Forest Regressor**

A variable named random\_forest is created and train and test data are passed as the parameters. Inside the function, the RandomForestClassifier algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, an accuracy score and classification report are used.



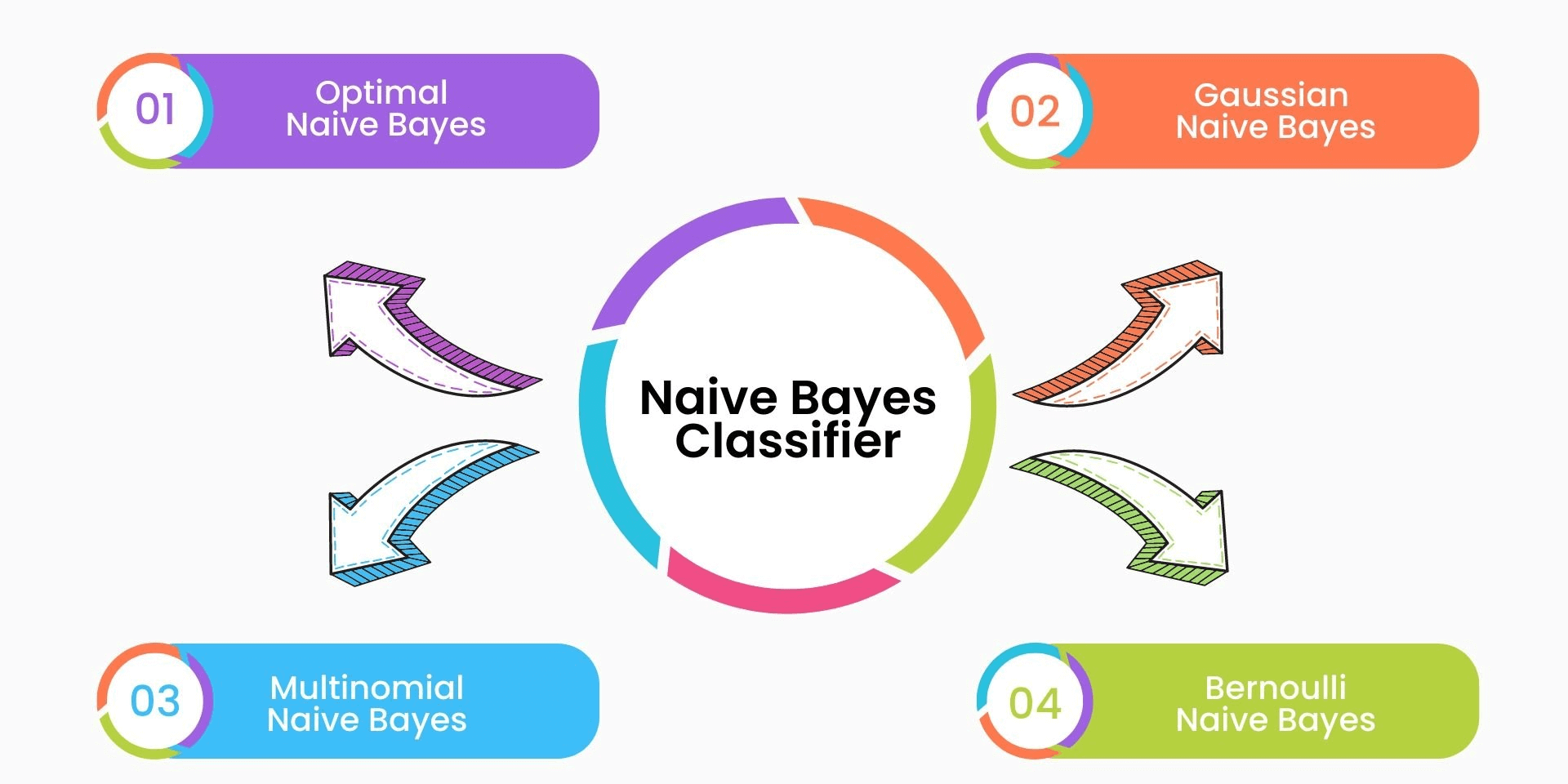
**Decision Tree Model**

A function named decision\_tree\_model is created and train and test data are passed as the parameters. Inside the function, the Decision Classifier algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, an accuracy score and classification report are used.



**Gaussian Navies Bayes**

A variable named NB is created and train and test data are passed as the parameters. Inside the function, the Gaussian Navies Bayes algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with the .predict() function and saved in the new variable. For evaluating the model, accuracy score and classification report are used.



**Multinomial Navies Bayes**

A function named ada\_model is created and train and test data are passed as the parameters. Inside the function, the Multinomial Navies Bayes algorithm is initialized and training data is passed to the model.the fit() function. Test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, an accuracy score and classification report are used.

**Testing the model**

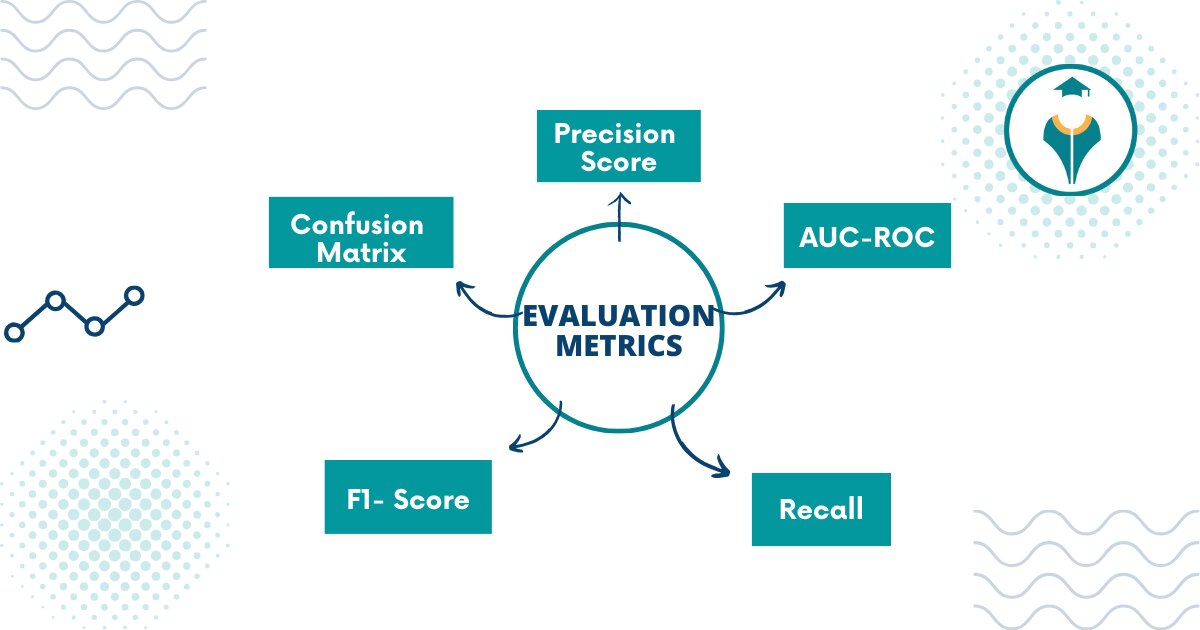
Here we have tested with the Lasso model algorithm. You can test with all algorithms. With the help of the predict() function.

**Performance testing and Hyper Parameter Tunning**

Performance testing in machine learning evaluates how well a model performs based on various metrics (e.g., accuracy, precision, recall, F1-score) using a test dataset. It involves assessing the model's ability to make predictions or classifications accurately. Hyperparameter tuning refers to the process of finding the best set of hyperparameters (e.g., learning rate, regularization strength, number of trees in a random forest) that optimize model performance.

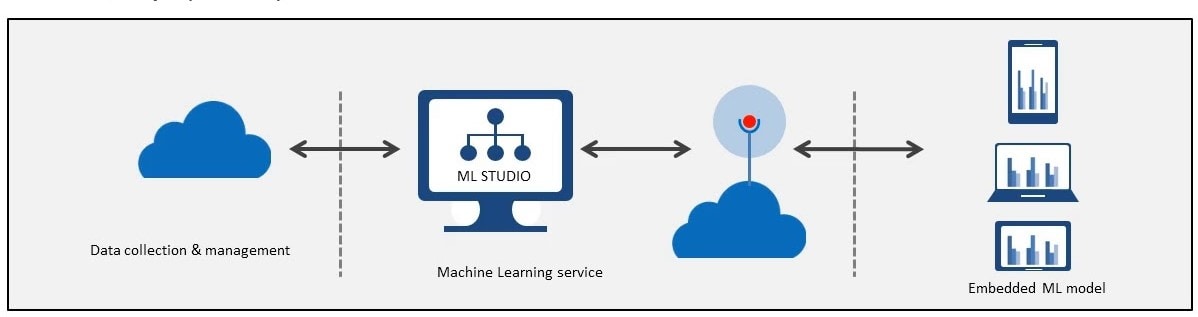
**Testing model with multiple evaluation metrics**

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for regression tasks including Accuracy scores and classification reports.  
Compare the model  
For comparing the above models, we create a data frame with the accuracy of the models   
We can see the accuracy of all models and based on accuracy and Test Accuracy Lasso regression model is the highest



**Model Deployment**

Model Deployment in Machine Learning refers to the process of integrating a trained machine learning model into a production environment where it can make predictions on new, unseen data. This process enables the model to be used by end-users or other systems to generate actionable insights or perform automated tasks.



**Save the best model**

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

**Integrate with Web Framework**

In this section, we will be building a web application that is integrated into the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.   
  
This section has the following tasks 

Building HTML Pages

Building server-side script

Run the web application

**Building Html Pages**

For this project create three HTML files namely

Index.html

details.html

prediction.html



and save them in the templates folder.

**Build Python code**

Import the libraries

Load the saved model. Importing the Flask module in the project is mandatory. An object of the Flask class is our WSGI application. The Flask constructor takes the name of the current module (\_\_name\_\_) as an argument.

Render HTML page:

Here we will be using a declared constructor to route to the HTML page that we have created earlier.

In the above example, the ‘/’ URL is bound with the index.html function. Hence, when the home page of the web server is opened in the browser, the HTML page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:Here we are routing our app to predict the () function. This function retrieves all the values from the HTML page using a Post request. That is stored in an array. This array is passed to the model. predict() function. This function returns the prediction. This prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

**Run the web application**

Open the Anaconda prompt from the start menu

Navigate to the folder where your Python script is.

Now type the “python app.py” command

Navigate to the localhost where you can view your web page.

Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.



**Output:**

