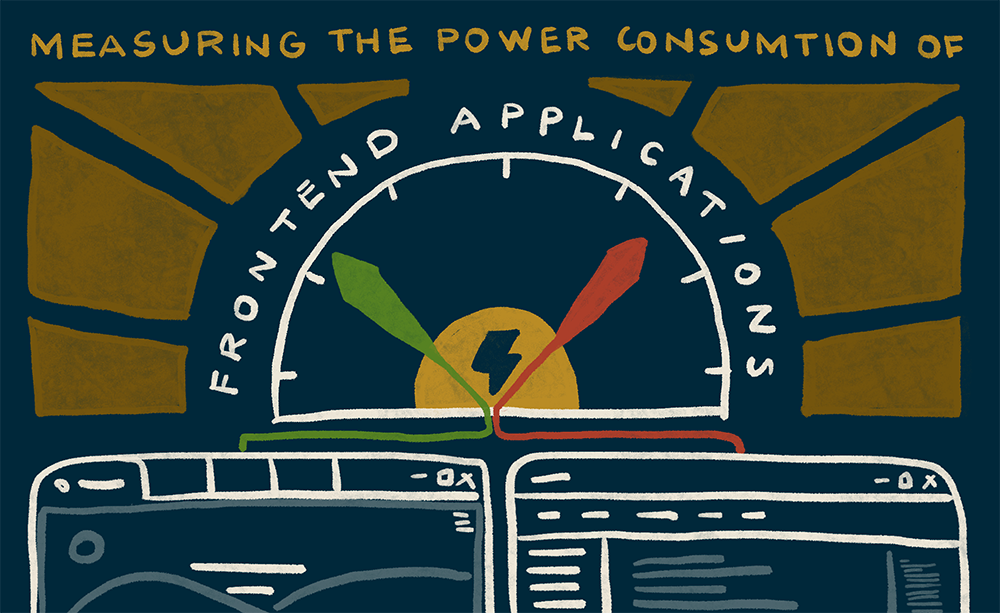
|  |  |
| --- | --- |
| NAME | MAHESWARAN.S |
| REG.NO. | 420121104031 |
| DEPARTMENT | CSE |
| YEAR | III |
| COLLEGE NAME | AKT MCET |
| GROUP | IBM GROUP-5 |

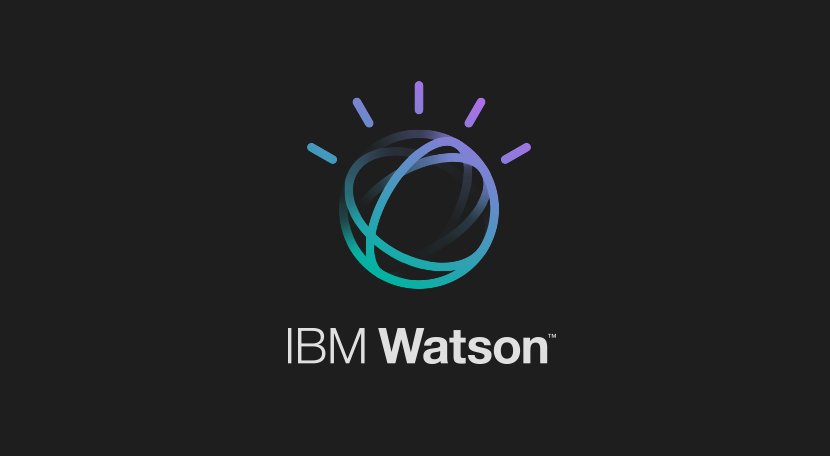
**MEASURE ENERGY CONSUMPTION**

****

* There are quite a few innovative approaches to measure energy consumption. One interesting method involves using smart meters and IoT (Internet of Things) devices.
* These smart meters can provide real-time data on energy usage and send it to a centralized system. With the help of machine learning algorithms, patterns and trends in energy consumption can be analyzed, allowing for more efficient energy management.
* Another cool idea is leveraging blockchain technology for energy tracking. By creating a transparent and decentralized ledger of energy transactions, you can ensure accurate and tamper-proof records of energy consumption.
* This not only enhances trust but also facilitates the development of a more efficient and accountable energy ecosystem.

**Module 11: IBM CLOUDS AND WATSON AI SERVICES**

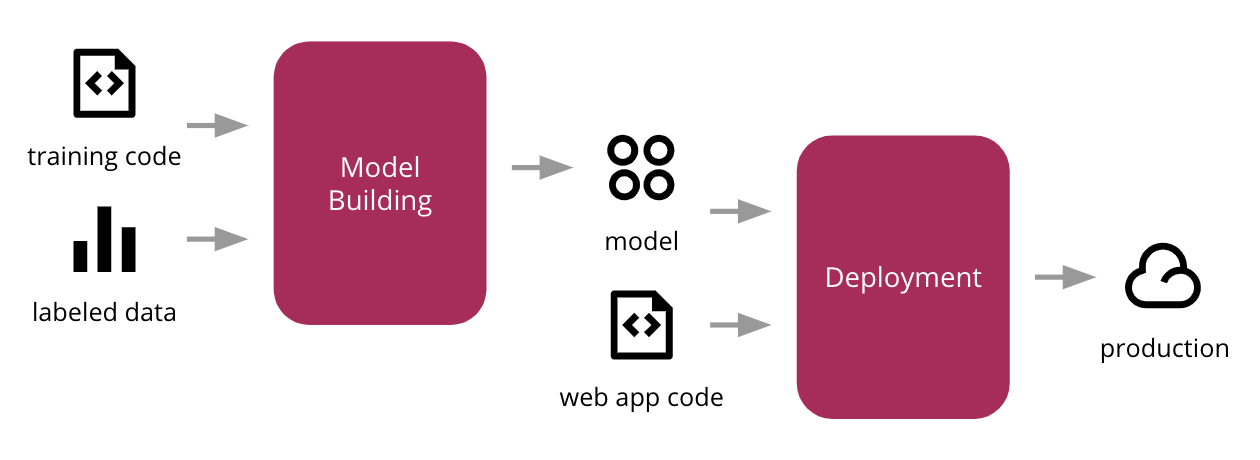
IBM Cloud and Watson AI services offer a robust suite of tools and capabilities for businesses looking to harness the power of cloud computing and artificial intelligence. The IBM Cloud provides a flexible and scalable infrastructure for deploying applications and services. Here are some key aspects:



1. **Infrastructure as a Service (IaaS**): IBM Cloud offers virtual servers, storage, and networking resources, allowing businesses to deploy and scale applications without the need to invest in physical hardware.
2. **Platform as a Service (PaaS):** Developers can take advantage of platform services for building, deploying, and managing applications more efficiently. This includes databases, development frameworks, and middleware.
3. **Watson AI Services:** IBM Watson is a suite of AI services that leverages machine learning and natural language processing. It includes tools for language translation, chatbots, image recognition, and more. Watson allows businesses to infuse AI into their applications without the need for extensive expertise in machine learning.
4. **Blockchain Services:** IBM Cloud offers blockchain services for developing, testing, and deploying blockchain networks. This can be valuable for industries such as finance, supply chain, and healthcare.
5. **Data and Analytics:** IBM Cloud provides various data and analytics services, including data warehousing, analytics, and machine learning tools. This enables businesses to derive insights from their data and make informed decisions.
6. **Security and Compliance:** IBM Cloud emphasizes security and compliance, offering tools and services to help businesses secure their applications and data in the cloud environment.
7. **Hybrid and Multi-Cloud Solutions:** IBM recognizes the importance of hybrid and multi-cloud strategies. Businesses can integrate on-premises infrastructure with IBM Cloud or leverage multiple cloud providers for a more flexible and resilient architecture.

**Programming code**

**Module 12: BUILD AND DEPLOY ML APPLICATION**

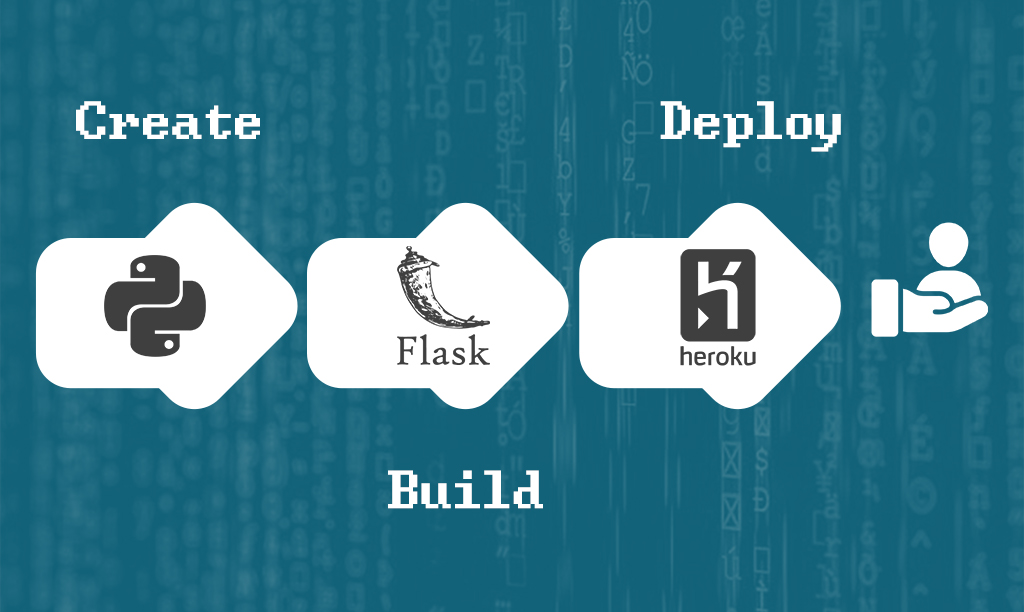
****

1. **Define Objectives**: Clearly define the objectives of your ML application. What problem are you trying to solve? What outcomes are you expecting?
2. **Data Collection** and Preparation: Gather relevant data for training your model. Clean and preprocess the data to handle missing values, outliers, and ensure it's suitable for training.
3. **Feature Engineering**: Identify relevant features and transform them to enhance the performance of your model. This step can significantly impact the success of your ML application.
4. **Model Selection and Training**: Choose a suitable ML algorithm based on your objectives and data characteristics. Split your data into training and testing sets, and train your model using the training data.
5. **Evaluation**: Evaluate the performance of your model using the testing data. Metrics such as accuracy, precision, recall, and F1 score can help assess the model's effectiveness.
6. **Fine-Tuning**: Adjust hyperparameters and refine your model to improve its performance. This may involve multiple iterations to achieve optimal results.
7. **Deployment:** Once you have a trained and validated model, deploy it for use in a production environment. Consider using cloud platforms or containerization for scalability and ease of management.
8. **Integration:** Integrate your ML model into the broader application architecture. Ensure that it interacts seamlessly with other components and services.
9. **Monitoring and Maintenance:** Implement monitoring mechanisms to track the performance of your deployed model. Regularly update the model with new data and retrain it as needed to maintain accuracy.

10. **Scalability:** Design your ML application to handle increased loads and scale as the demand grows. Cloud services and container orchestration can assist in scaling your application.

11**. Security:** Implement security measures to protect both the model and the data. This includes encryption, access controls, and secures communication protocols.

12***. Documentation***: Document your ML application thoroughly, including the model architecture, data sources, and any other relevant information. This documentation is valuable for future maintenance and updates.



**Programming Code**

import Flask, request, render\_template

from sklearn.model\_selection

import train\_test\_split from sklearn.ensemble

import RandomForest Classifier from sklearn.metrics

import accuracy score

import pandas as pd

iris = pd.read\_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data',

names=['sepal\_length', 'sepal\_width', 'petal\_length', 'petal\_width', 'species'])

# Features and target variable

X = iris.drop('species', axis=1)

y = iris['species']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

Print(f'Model Accuracy: {accuracy:.2f}')

app = Flask(\_\_name\_\_)

@app. route('/')

def home():

return render\_template('index.html')

@app.route('/predict', methods=['POST'])

def predict():

features = [float(request.form['sepal\_length']),

float(request.form['sepal\_width']),

float(request.form['petal\_length']),

float(request.form['petal\_width'])]

prediction = model.predict([features])[0]

return render\_template('index.html', prediction=prediction)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)