SMARTBRIDGE
Let's Bridge the Gap

Smart
Internz

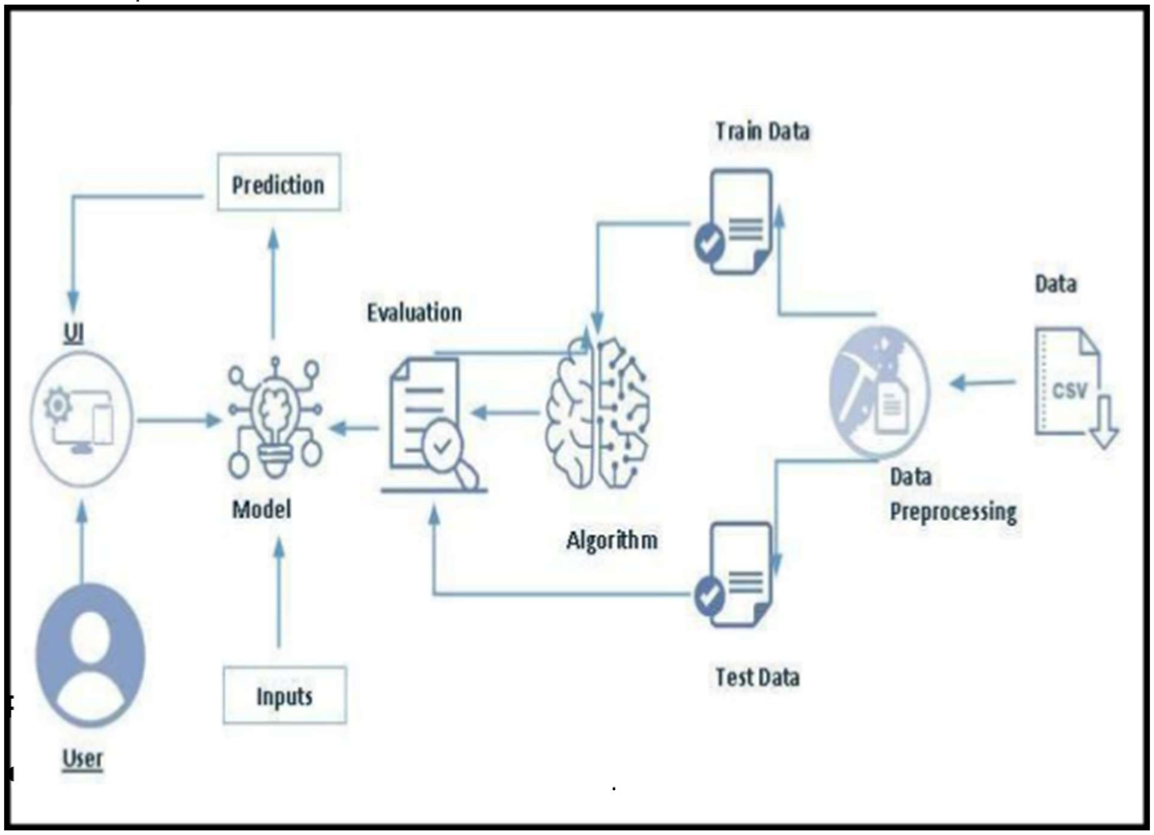
AI-Driven Optimization of 5G Resource Allocation for Network Efficiency

Three unique frequency bands low, mid, and high define 5G technology. Every band has different capabilities: the low band (less than 1GHz) covers a large area but operates at slower speeds; the mid band (1GHz–6GHz) balances speed and coverage; and the high band (24GHz–40GHz) offers better speeds but a smaller coverage area.

5G, the most recent development in cellular wireless connectivity, is a major step forward, providing more capacity, wider coverage, and lower latency. Although it is based on the same basic concepts as 4G, its exceptional performance levels set it apart.

Under the hood, 5G offers significant improvements:

- Wired like reliability
- Ultra-low latency <20ms
- Gbps data rate



- **Define Problem / Problem Understanding**

○ Specify the business problem

○ Business requirement

○ Literature Survey

○ Social or Business Impact.

● **Data Collection & Preparation**

○ Collect the dataset ○ Data Preparation

● **Exploratory Data Analysis**

○ Descriptive statistical ○ Visual

● **Model Building**

○ Training the model in multiple algorithms

○ Testing the model

● **Performance Testing & Hyperparameter Tuning**

○ Testing model with multiple evaluation metrics

○ Comparing model accuracy before & after applying hyperparameter tuning ●

Model Deployment

○ Save the best model

○ Integrate with Web Framework

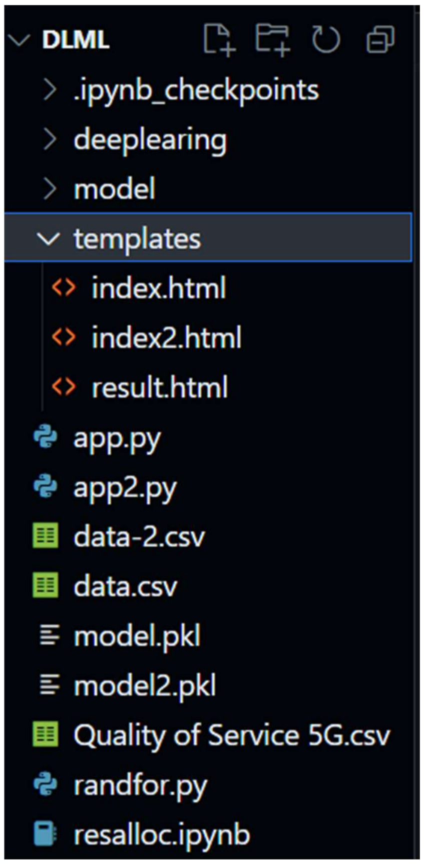
● **Project Demonstration & Documentation**

○ Record explanation Video for project end to end solution

○ Project Documentation-Step by step project development procedure

Project Structure:

Create the Project folder which contains files as shown below



```
DLML
  > .ipynb_checkpoints
  > deeplearing
  > model
  v templates
      index.html
      index2.html
      result.html
    app.py
    app2.py
    data-2.csv
    data.csv
    model.pkl
    model2.pkl
    Quality of Service 5G.csv
    randfor.py
    resalloc.ipynb
```

- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- model.pkl is our saved model. Further we will use this model for flask integration.
- Training folder contains a model training file.

Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the business problem

Refer Project Description

Activity 2: Business requirement.

There are essentially three types of 5G bands supported in India — low-band, mid-band, and high-band (mm Wave) – based on frequency. Simply put, the higher the frequency, the better the speed and shorter the range of the network. The company want to find the resource allocation by it's Application Type, Signal Strength, Latency, Required Bandwidth, Allocated Bandwidth.

Activity 3: Literature Survey

The advent of fifth-generation (5G) communication technology brings forth a myriad of promises, including higher data rates, enhanced user experiences, lower power consumption, and remarkably short latency. To realize these objectives, 5G networks employ a complex multi-layer model that encompasses device-to-device networks, macro-cells, and various categories of small cells. This intricate network architecture aims to cater to users' quality-of-service (QoS) requirements, prompting numerous studies to explore interference management and resource allocation challenges within the 5G ecosystem.

Resource Allocation Challenges:

With the escalating demand for cellular services and the constraints of limited resources, effective management of network traffic and operations emerges as a critical issue in resource distribution. The primary concern revolves around mitigating network congestion to ensure an optimal QoS for end-users. While resource distribution in the context of 5G has been addressed in a limited number of review papers, a comprehensive overview specifically focusing on 5G resource allocation is notably absent in the literature.

Activity 4: Social or Business Impact.

Social Impact: By providing resource allocation percentage by its Application Type, Signal Strength, Latency, Required Bandwidth, Allocated Bandwidth. It will helpful to the organization for how much resource should be allocated by their preferences.

Milestone 2: Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset

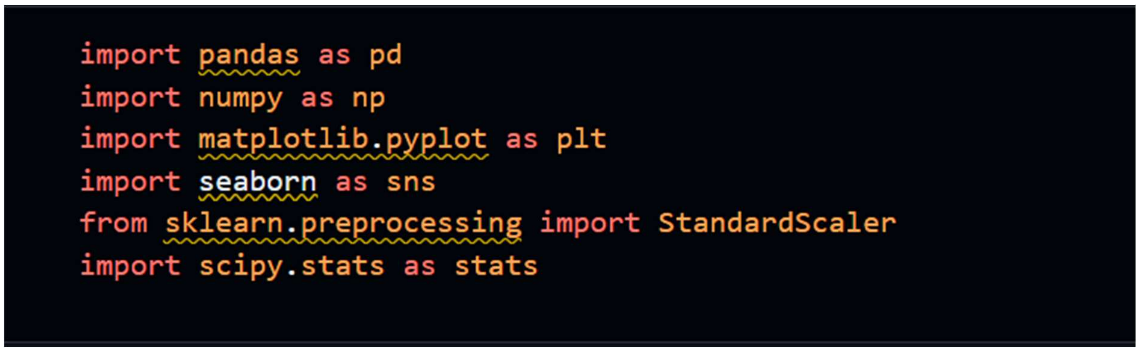
Activity 1: Collect the dataset There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset **link:** **https://www.kaggle.com/datasets/omarsobhy14/5g-quality-of-service**

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.

Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1.1: Importing the libraries Import the necessary libraries as shown in the image.



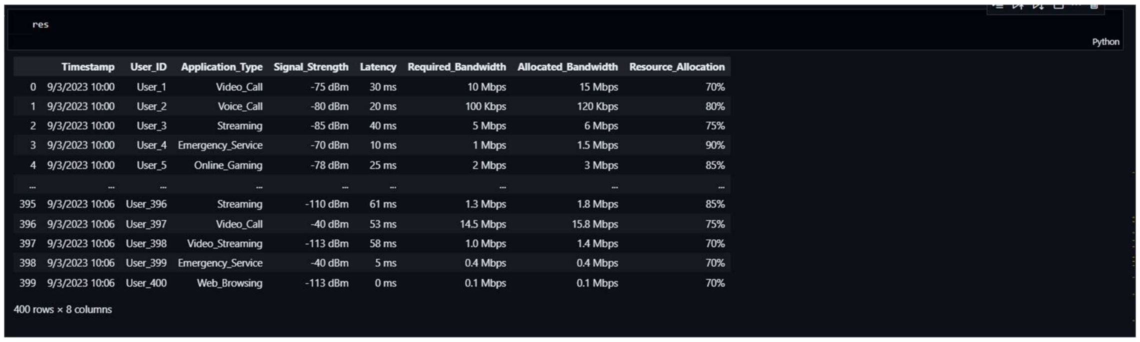
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
import scipy.stats as stats
```

Activity 1.2: 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.

Activity 2: Data Preparation



```
res
```

Python

	Timestamp	User_ID	Application_Type	Signal_Strength	Latency	Required_Bandwidth	Allocated_Bandwidth	Resource_Allocation
0	9/3/2023 10:00	User_1	Video_Call	-75 dBm	30 ms	10 Mbps	15 Mbps	70%
1	9/3/2023 10:00	User_2	Voice_Call	-80 dBm	20 ms	100 Kbps	120 Kbps	80%
2	9/3/2023 10:00	User_3	Streaming	-85 dBm	40 ms	5 Mbps	6 Mbps	75%
3	9/3/2023 10:00	User_4	Emergency_Service	-70 dBm	10 ms	1 Mbps	1.5 Mbps	90%
4	9/3/2023 10:00	User_5	Online_Gaming	-78 dBm	25 ms	2 Mbps	3 Mbps	85%
...	...	...	...	...	...	...	...	...
395	9/3/2023 10:06	User_396	Streaming	-110 dBm	61 ms	1.3 Mbps	1.8 Mbps	85%
396	9/3/2023 10:06	User_397	Video_Call	-40 dBm	53 ms	14.5 Mbps	15.8 Mbps	75%
397	9/3/2023 10:06	User_398	Video_Streaming	-113 dBm	58 ms	1.0 Mbps	1.4 Mbps	70%
398	9/3/2023 10:06	User_399	Emergency_Service	-40 dBm	5 ms	0.4 Mbps	0.4 Mbps	70%
399	9/3/2023 10:06	User_400	Web_Browsing	-113 dBm	0 ms	0.1 Mbps	0.1 Mbps	70%

400 rows × 8 columns

As we have understood how the data is, let's pre-process the collected data. The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results.

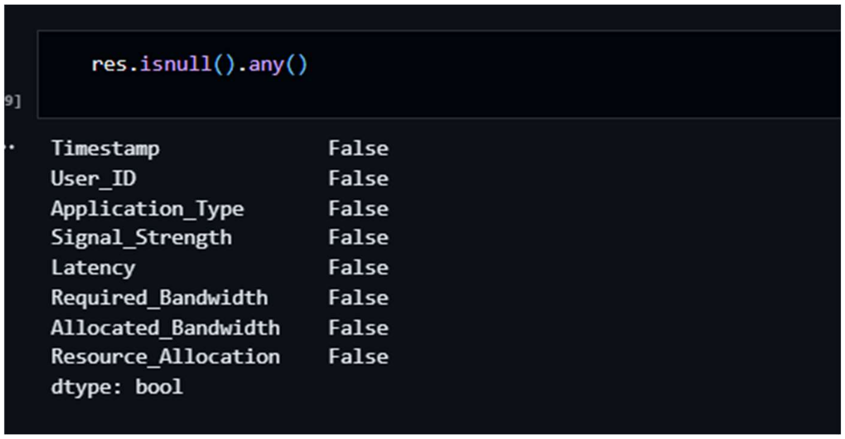
This activity includes the following steps.

● Handling missing values

● Handling categorical data

● Handling Outliers

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps



```
res.isnull().any()
```

```
Timestamp              False
User_ID                False
Application_Type       False
Signal_Strength        False
Latency                False
Required_Bandwidth     False
Allocated_Bandwidth    False
Resource_Allocation    False
dtype: bool
```

Activity 2.2: Handling Categorical Values

As we can see our dataset has categorical data, we must convert the categorical data to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding techniques. There are several techniques but, in our project, we are using manual encoding with the help of list comprehension.

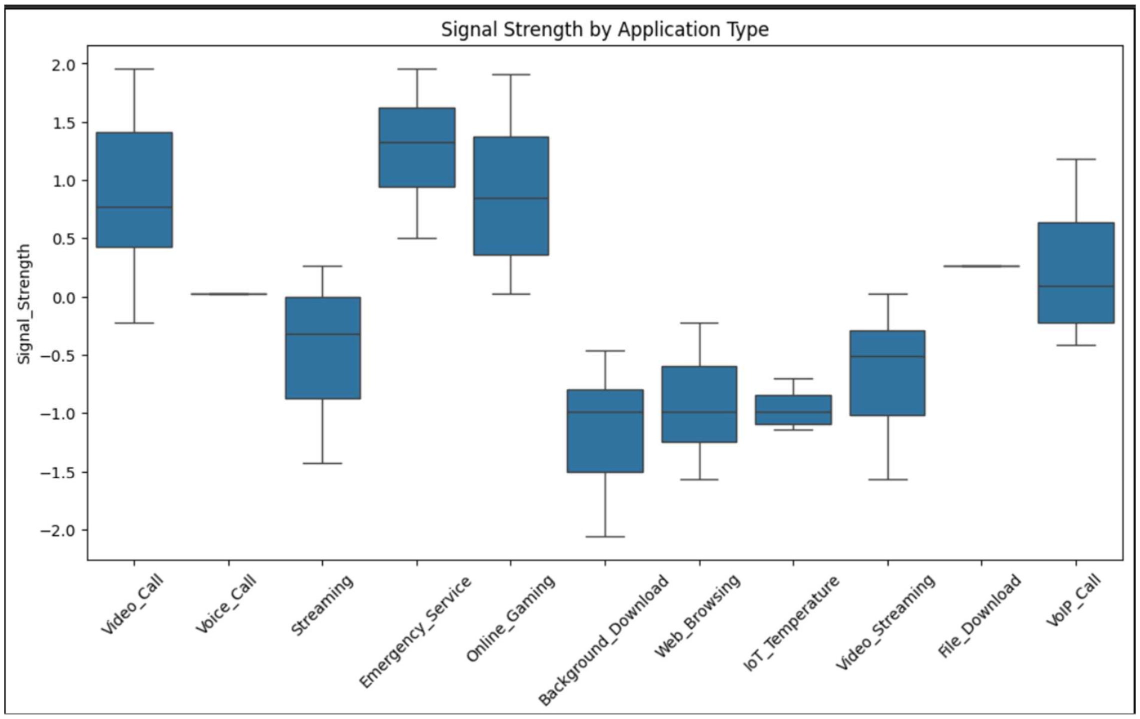
Milestone 3: Exploratory Data Analysis

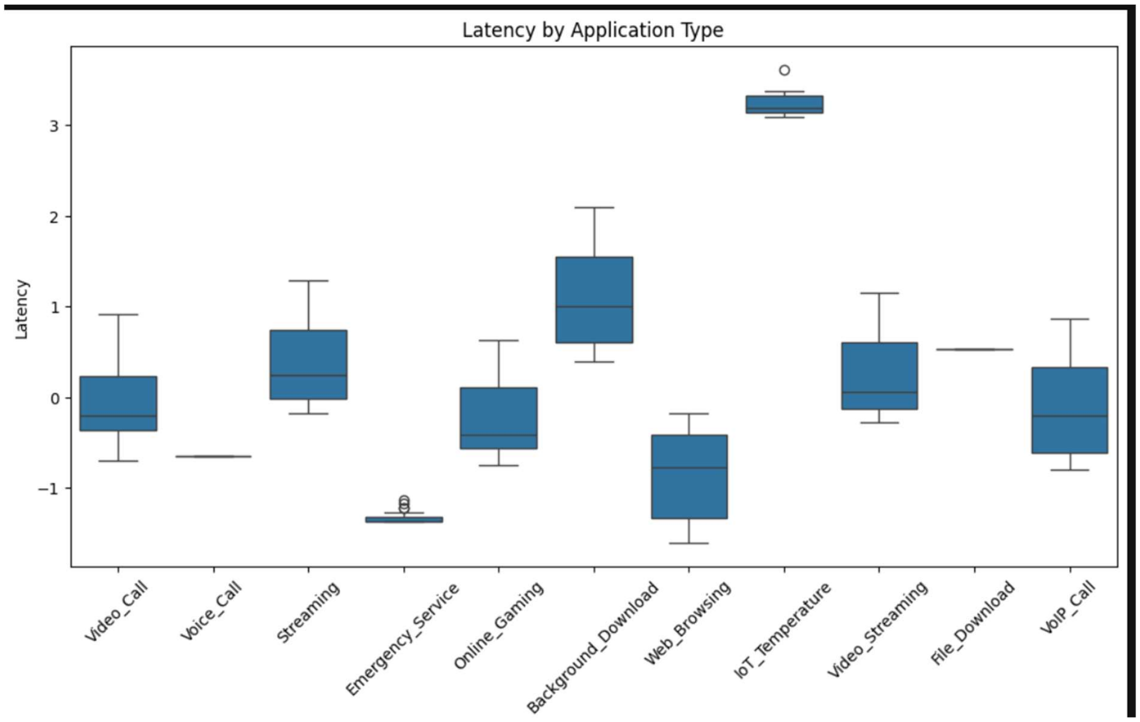
Activity 1: Descriptive statistical Descriptive analysis is to study the basic features of data with the statistical process.

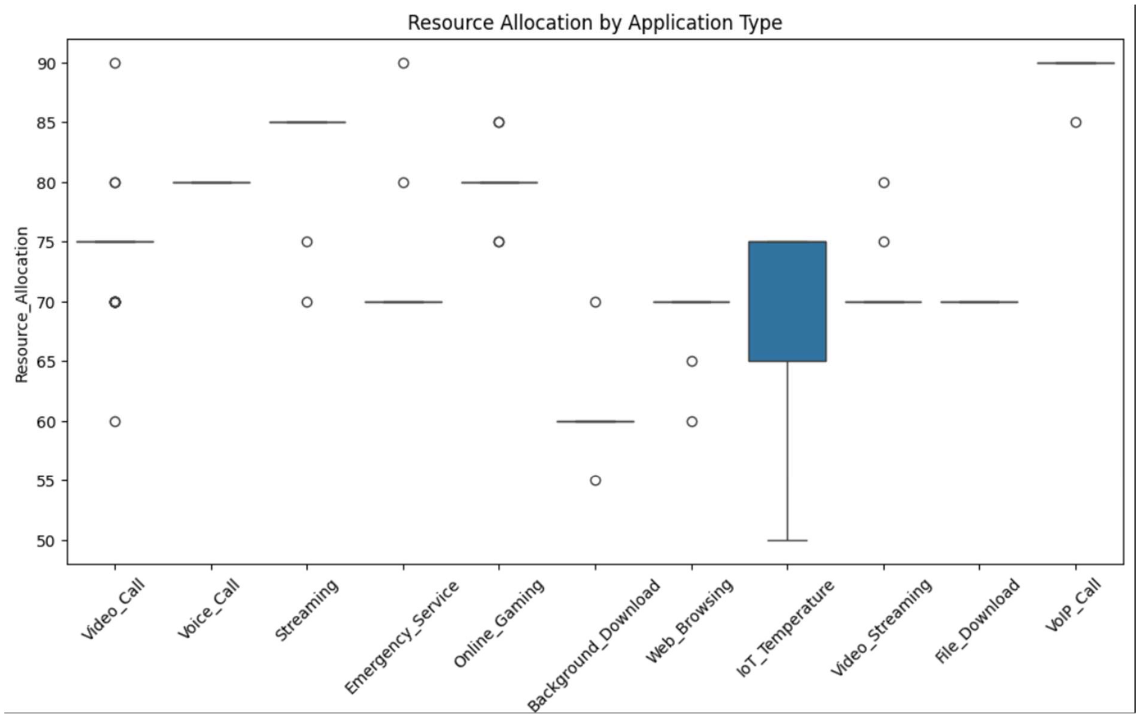
Here pandas have a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

Activity 2: 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.


Signal Strength by Application Type
Signal_Strength
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
Video_Call
Voice_Call
Streaming
Emergency_Service
Online_Gaming
Background_Download
Web_Browsing
IoT_Temperature
Video_Streaming
File_Download
VoIP_Call

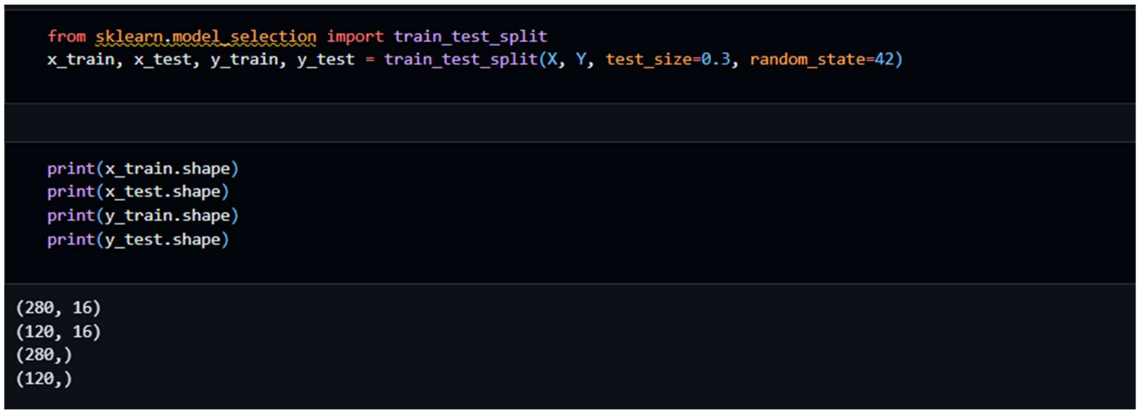

Latency by Application Type
Latency
3
2
1
0
−1
Video_Call
Voice_Call
Streaming
Emergency_Service
Online_Gaming
Background_Download
Web_Browsing
IoT_Temperature
Video_Streaming
File_Download
VoIP_Call



Splitting data into train and test

Now let's 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 x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using train test split () function from sklearn. As parameters, we are passing x, y, test size, random state.



```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=42)
```

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

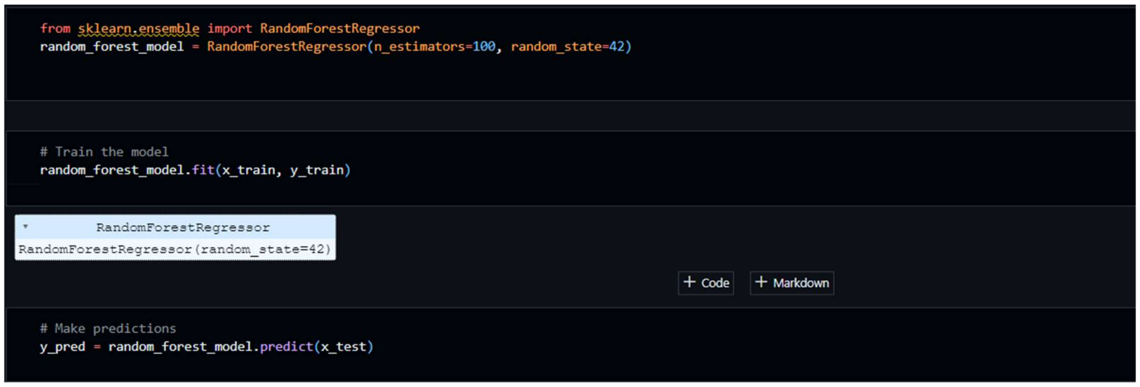
```
(280, 16)
(120, 16)
(280,)
(120,)
```

Milestone 4: Model Building

Activity 1: Training the model in 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 classification algorithms. The best model is saved based on its performance.

Activity 1.1: Random Forest regressor model

A function named random Forest is created and train and test data are passed as the parameters. Inside the function, Random Forest Classifier algorithm is initialised and training data is passed to the model with. Fit () function. Test data is predicted with. predict () function and saved in a new variable. For evaluating this analysis r2Score is used.

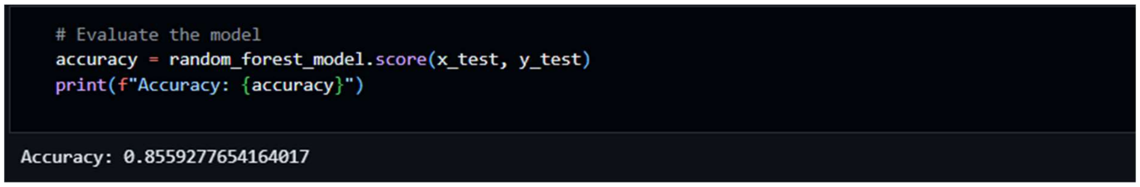


```
from sklearn.ensemble import RandomForestRegressor
random_forest_model = RandomForestRegressor(n_estimators=100, random_state=42)
```

```
# Train the model
random_forest_model.fit(x_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

```
# Make predictions
y_pred = random_forest_model.predict(x_test)
```

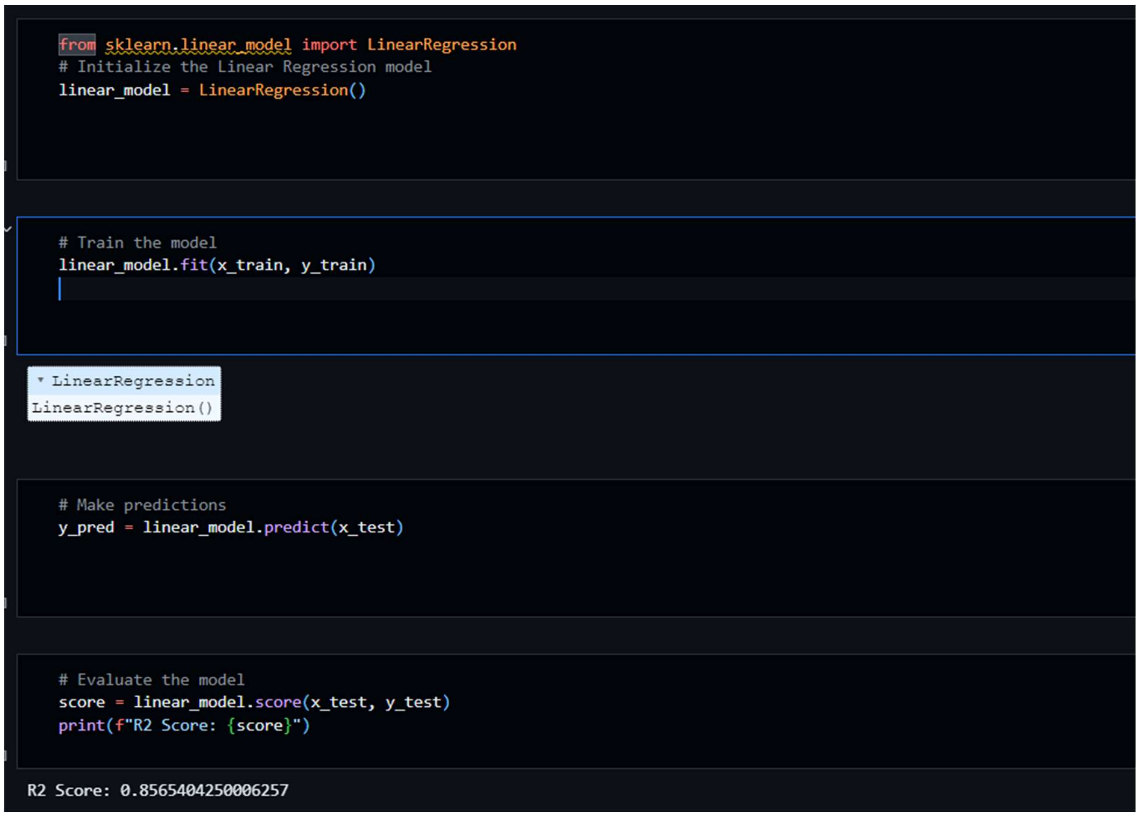


```
# Evaluate the model
accuracy = random_forest_model.score(x_test, y_test)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.8559277654164017
```

Activity 1.2 Linear regressor model

Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.



```
from sklearn.linear_model import LinearRegression
# Initialize the Linear Regression model
linear_model = LinearRegression()
```

```
# Train the model
linear_model.fit(x_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
# Make predictions
y_pred = linear_model.predict(x_test)
```

```
# Evaluate the model
score = linear_model.score(x_test, y_test)
print(f"R2 Score: {score}")
```

```
R2 Score: 0.8565404250006257
```

Activity 1.3: Decision Tree Classifier model

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final reult is a tree with decision nodes and leaf nodes.

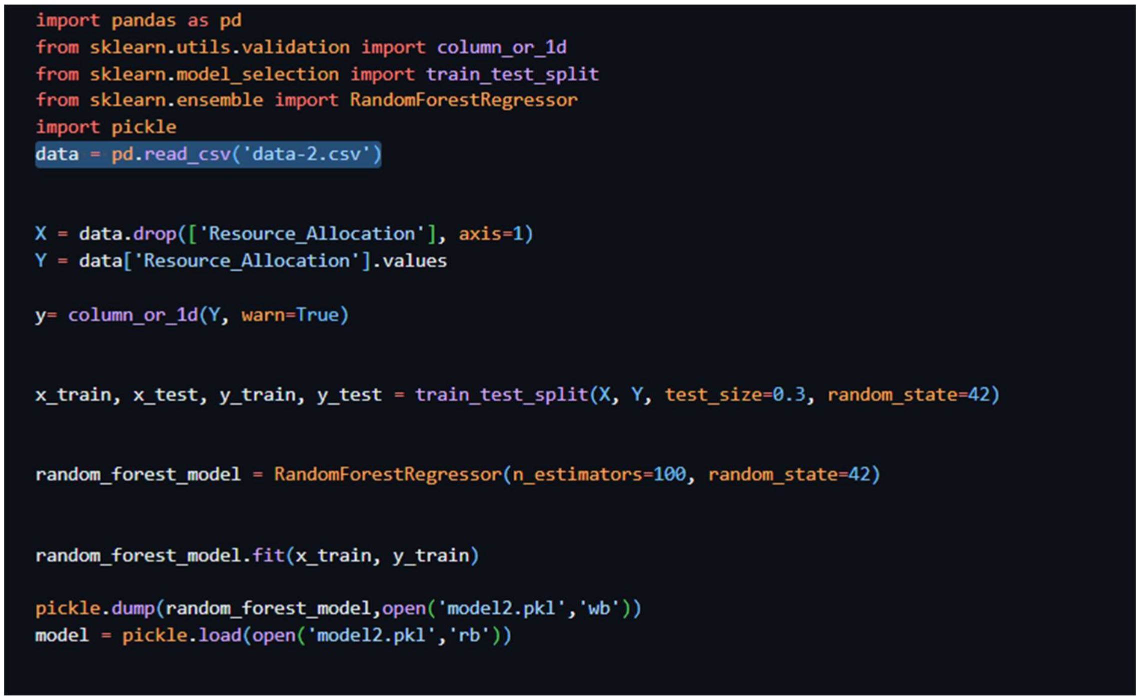
Milestone 5: Performance Testing & Hyperparameter Tuning

Activity 1: 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 classification tasks including accuracy, precision, recall, support and F1-score.

Milestone 6: Model Deployment

Activity 1: 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.



```
import pandas as pd
from sklearn.utils.validation import column_or_1d
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
import pickle
data = pd.read_csv('data-2.csv')


X = data.drop(['Resource_Allocation'], axis=1)
Y = data['Resource_Allocation'].values

y= column_or_1d(Y, warn=True)


x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=42)


random_forest_model = RandomForestRegressor(n_estimators=100, random_state=42)


random_forest_model.fit(x_train, y_train)

pickle.dump(random_forest_model,open('model2.pkl','wb'))
model = pickle.load(open('model2.pkl','rb'))
```

Activity 2: Integrate with Web Framework

In this section, we will be building a web application that is integrated to 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

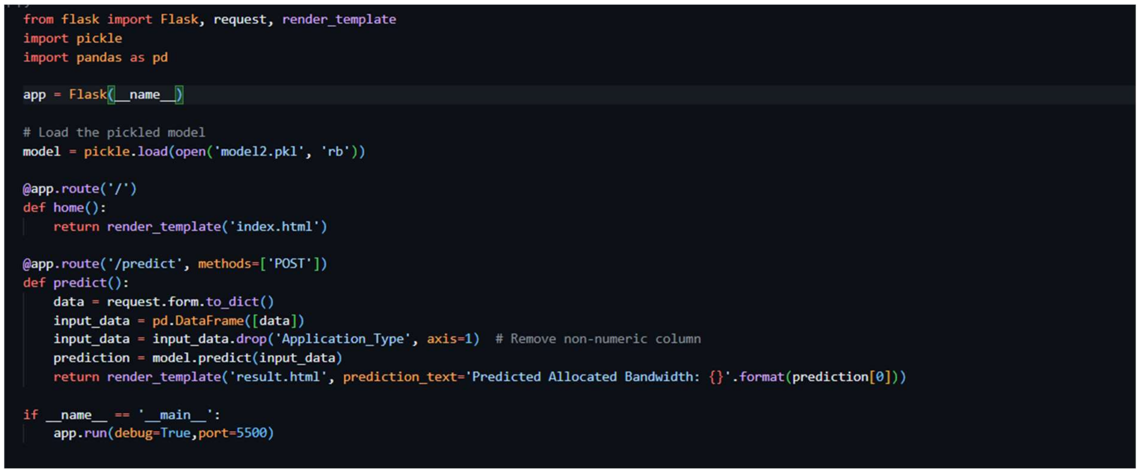
Activity 2.1: Building Html Pages: For this project create two HTML files namely

• home.html

• predict.html • submit.html and save them in the templates folder. Refer this link for templates.

Activity 2.2:

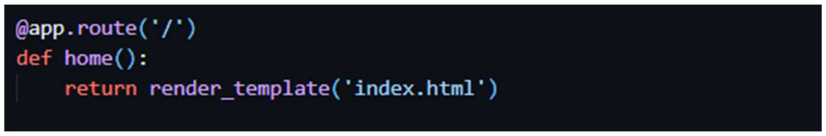
Build Python code:



```
from flask import Flask, request, render_template
import pickle
import pandas as pd

app = Flask(__name__)

# Load the pickled model
model = pickle.load(open('model2.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    data = request.form.to_dict()
    input_data = pd.DataFrame([data])
    input_data = input_data.drop('Application_Type', axis=1)  # Remove non-numeric column
    prediction = model.predict(input_data)
    return render_template('result.html', prediction_text='Predicted Allocated Bandwidth: {}'.format(prediction[0]))

if __name__ == '__main__':
    app.run(debug=True,port=5500)
```

Import the libraries Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.

Render HTML page

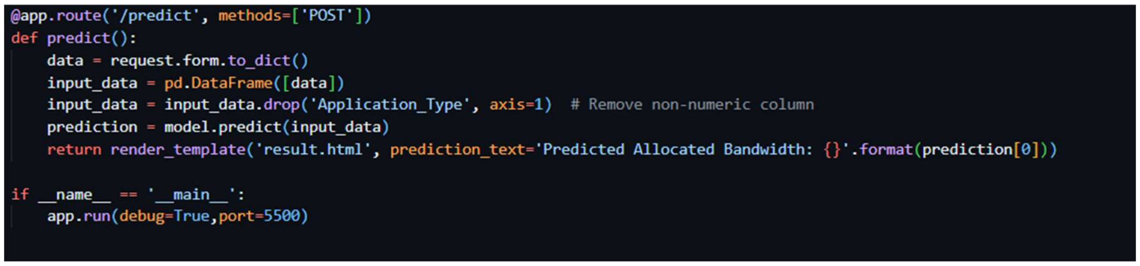


```
@app.route('/')
def home():
    return render_template('index.html')
```

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

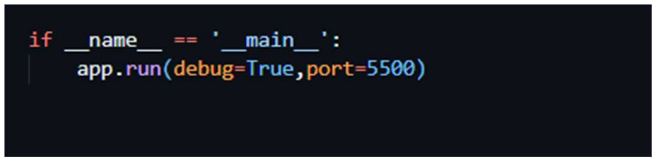
In the above example, '/' URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI



```
@app.route('/predict', methods=['POST'])
def predict():
    data = request.form.to_dict()
    input_data = pd.DataFrame([data])
    input_data = input_data.drop('Application_Type', axis=1)  # Remove non-numeric column
    prediction = model.predict(input_data)
    return render_template('result.html', prediction_text='Predicted Allocated Bandwidth: {}'.format(prediction[0]))

if __name__ == '__main__':
    app.run(debug=True,port=5500)
```

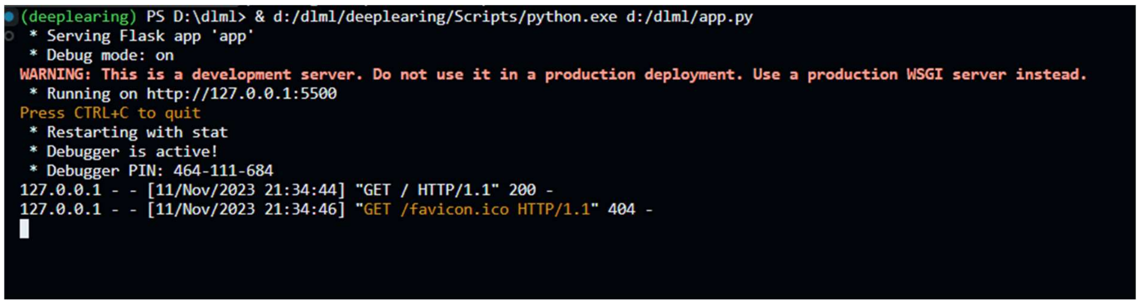
Here we are routing our app to predict () function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model. Predict () function. This function returns the prediction. And this prediction value will be rendered to the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



```
if __name__ == '__main__':
    app.run(debug=True,port=5500)
```

Activity 2.3: Run the web app application

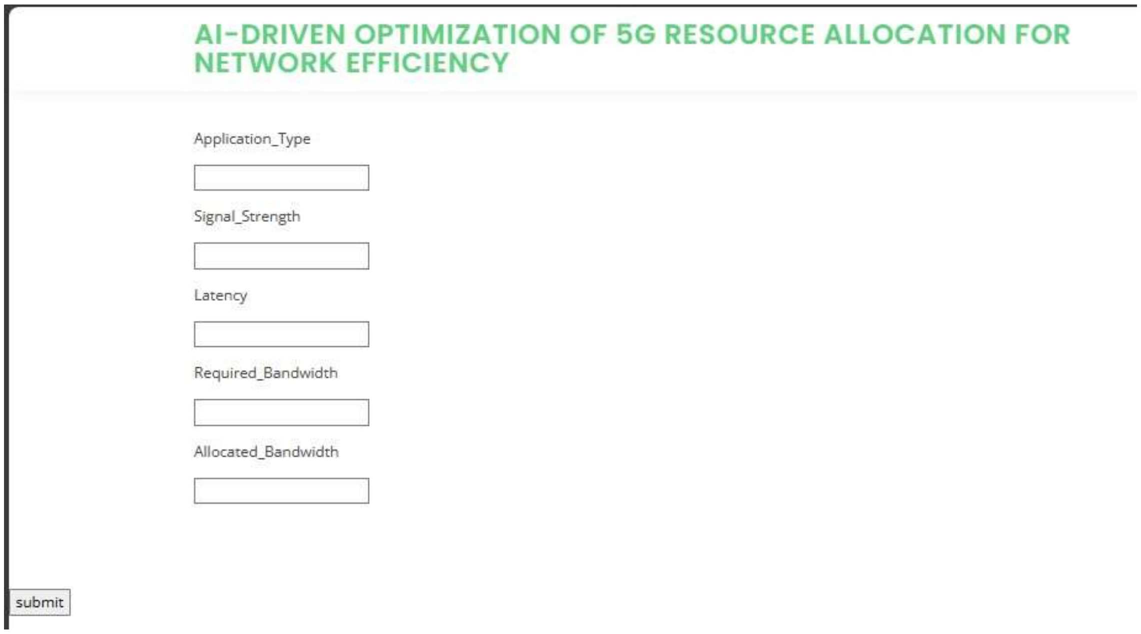
- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type "python app.py" command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.



```
(deeplearing) PS D:\dlml> & d:/dlml/deeplearing/Scripts/python.exe d:/dlml/app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5500
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 464-111-684
127.0.0.1 - - [11/Nov/2023 21:34:44] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [11/Nov/2023 21:34:46] "GET /favicon.ico HTTP/1.1" 404 -
```

Now, Go the web browser and write the localhost URL(http://127.0.0.1:5500/)to get the below result **Home**

About Page:



AI-DRIVEN OPTIMIZATION OF 5G RESOURCE ALLOCATION FOR NETWORK EFFICIENCY

Application_Type

Signal_Strength

Latency

Required_Bandwidth

Allocated_Bandwidth