



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

**Team Leader :** Repaka Sai Akshith

**Team member :** Rajavarapu Jaswanth Sai

**Team member :** Pemmana Visweshwar Reddy

Team ID	Team-592661
Project Name	AI-driven resource 5G optimization

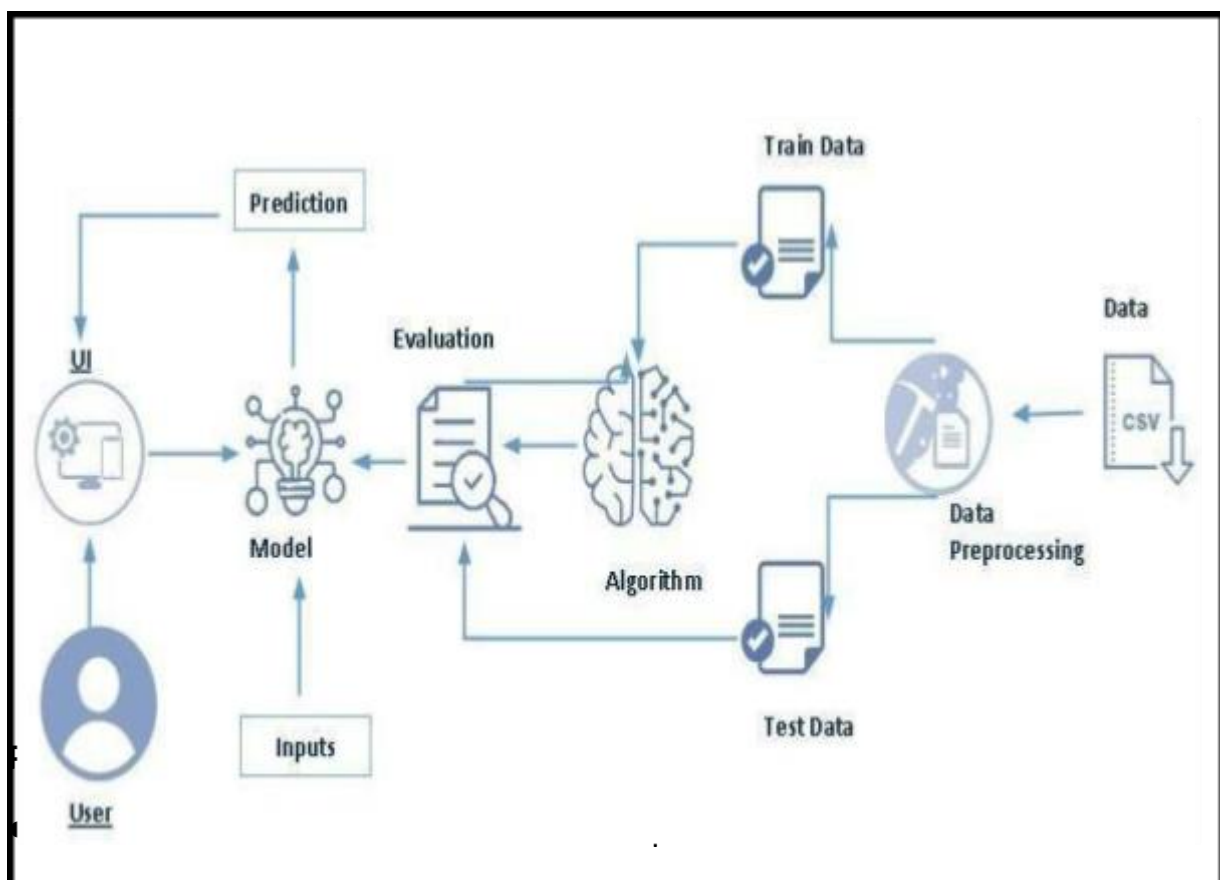
5G is divided into three frequency bands (low, mid, and high). Each band has different capabilities: the low band (less than 1GHz) has greater coverage but lower speeds, the mid band (1GHz–6GHz) offers a balance of both, and the high band (24GHz–40GHz) offers higher speeds but a smaller coverage radius.

5G is the latest evolution of cellular wireless connectivity and offers improved capacity, coverage, and lower latency. 5G offers many improvements compared to 4G but relies on the similar fundamentals to communicate with end user devices.

What makes 5G so different is the new levels of performance it offers. Similar to how 4G helped usher in the smartphone era, 5G will power new technologies across the enterprise, within smart cities, for autonomous vehicles, and ubiquitous Internet of Things (IoT) installations.

Under the hood, 5G offers significant improvements:

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



## Milestone 1: Define Problem / Problem Understanding

The rapid proliferation of data-intensive applications and the complexity of 5G networks have created a pressing need for efficient resource allocation strategies. Traditional methods struggle to adapt dynamically to changing network conditions and fail to optimize network efficiency. To address this challenge, the problem at hand is to develop AI-driven approaches that leverage advanced algorithms and techniques to optimize resource allocation in 5G networks. The objective is to maximize throughput, minimize latency, reduce energy consumption, and improve quality of service (QoS), thereby ensuring optimal network performance and meeting the ever-increasing demands of modern communication systems.

## 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.

*The business requirements for "AI-Driven Optimization of 5G Resource Allocation for Network Efficiency" are multifaceted, reflecting the complex landscape of the telecommunications industry in the 5G era :*

**Efficient Resource Allocation:** The foremost requirement is the efficient allocation of network resources. This involves optimizing the distribution of bandwidth, power, and time-frequency resources to ensure cost-effectiveness while meeting the diverse quality of service demands.

**Quality of Service (QoS):** Ensuring a high level of QoS across the network is critical. Businesses need to meet the specific and varying QoS requirements of different services and applications to ensure customer satisfaction.

**Real-time Adaptability:** Dynamic network conditions and rapidly changing user demands necessitate real-time resource allocation. The system should adapt swiftly to maintain service quality and user experience.

**Network Slicing Management:** With network slicing capabilities in 5G, operators must efficiently manage and allocate resources within these slices to cater to diverse services, industries, and applications.

**Energy Efficiency:** Achieving energy efficiency is not just a technological requirement but also an environmental and cost consideration. Optimized resource allocation can help reduce energy consumption and operational costs.

## Activity 3: Literature Survey

### **Introduction:**

The exponential growth of data traffic and the increasing complexity of 5G networks have presented significant challenges in efficiently allocating network resources. To address this, researchers have turned to artificial intelligence (AI) techniques to optimize resource allocation in 5G networks. This literature survey provides a concise overview of the state-of-the-art approaches and advancements in AI-driven optimization for 5G resource allocation, emphasizing network efficiency.

**1.Optimization Objectives:** Various optimization objectives have been explored to enhance network efficiency, including maximizing throughput, minimizing latency, reducing energy consumption, and improving quality of service (QoS). Advanced AI algorithms have been employed to achieve these objectives by dynamically allocating resources based on real-time network conditions.

**2.Machine Learning Techniques:** Machine learning (ML) techniques, such as supervised learning, unsupervised learning, and reinforcement learning, have been extensively employed in 5G resource allocation. ML models learn from historical data to predict traffic patterns, user demands, and network behaviour, enabling proactive resource allocation and efficient load balancing.

**3.Deep Learning Approaches:** Deep learning, a subset of ML, has gained significant attention due to its ability to extract complex patterns and make accurate predictions. Deep neural networks (DNNs) have been applied to optimize 5G resource allocation by leveraging their capabilities in feature extraction, traffic prediction, and resource scheduling.

**4.Genetic Algorithms and Swarm Intelligence:** Inspired by nature, optimization techniques based on genetic algorithms and swarm intelligence have been proposed. These algorithms mimic biological evolution and social behaviour to discover optimal resource allocation strategies. Genetic algorithms evolve resource allocation solutions over generations, while swarm intelligence algorithms, like particle swarm optimization, simulate the collective behaviour of individuals to find efficient solutions.

**5.Edge Computing and Network Slicing:** Edge computing and network slicing are emerging paradigms in 5G networks that offer improved resource allocation. AI-driven approaches are used to optimize resource allocation in edge servers and enable efficient network slicing, ensuring that resources are allocated dynamically and adaptively based on specific service requirements.

### **Conclusion:**

Machine learning, deep learning, genetic algorithms ,intelligence, edge computing, and network slicing are key areas of research that hold potential in improving network efficiency.

## **Activity 4: Social or Business Impact**

### **1.Improved Network Efficiency:**

AI optimization strategies enhance network efficiency by dynamically allocating resources, resulting in better utilization of available bandwidth and reduced congestion. This translates into improved service quality and customer satisfaction.

### **2. Cost Reduction:**

AI-driven resource allocation optimizes network utilization, leading to cost savings for network operators. By efficiently managing resources, operators can avoid unnecessary infrastructure investments and reduce energy consumption, resulting in significant financial benefits.

### **3. Enhanced Revenue Generation:**

Effective resource allocation ensures optimal service delivery, enabling network operators to offer premium services with improved QoS. This can lead to increased customer loyalty, higher revenue generation, and a competitive edge in the market.

### **4. Future-Proofing Networks:**

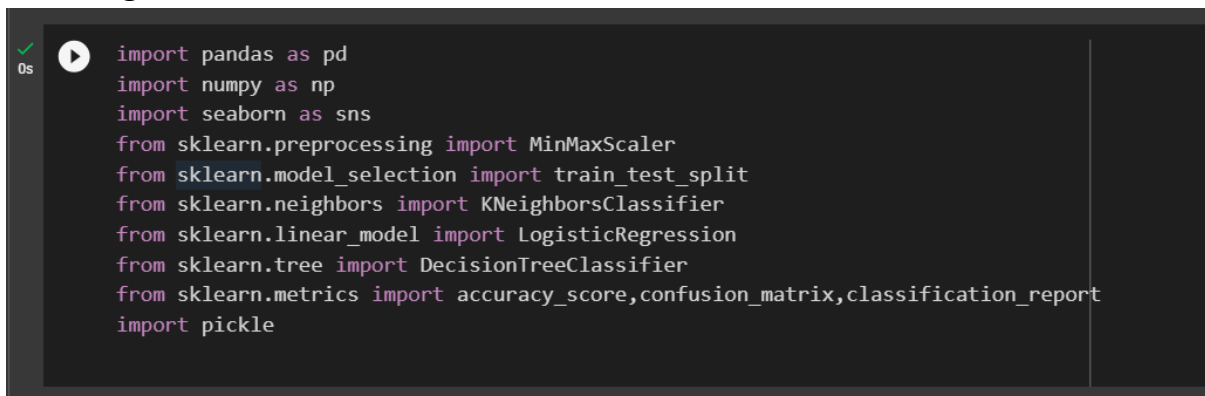
AI-driven optimization techniques enable networks to adapt to changing demands and emerging technologies. By efficiently allocating resources, operators can prepare their networks for future technologies like Internet of Things (IoT), augmented reality (AR), and virtual reality (VR), ensuring long-term business sustainability.

## Milestone 2: Data Collection & Preparation

### Activity 1: DATASET :

<https://www.kaggle.com/datasets/omarsohby14/5g-quality-of-service>

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



```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import pickle
```

### 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

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
data = pd.read_csv("Quality of Service 5G.csv")
data.head
```

	<bound method NDFrame.head of	Timestamp	User_ID	Application_Type	Signal_Strength	Latency	\
0	9/3/2023 10:00	User_1	Video_Call	-75 dBm	30 ms		
1	9/3/2023 10:00	User_2	Voice_Call	-80 dBm	20 ms		
2	9/3/2023 10:00	User_3	Streaming	-85 dBm	40 ms		
3	9/3/2023 10:00	User_4	Emergency_Service	-70 dBm	10 ms		
4	9/3/2023 10:00	User_5	Online_Gaming	-78 dBm	25 ms		
..	...	...	...	...	...		
395	9/3/2023 10:06	User_396	Streaming	-110 dBm	61 ms		
396	9/3/2023 10:06	User_397	Video_Call	-40 dBm	53 ms		
397	9/3/2023 10:06	User_398	Video_Streaming	-113 dBm	58 ms		
398	9/3/2023 10:06	User_399	Emergency_Service	-40 dBm	5 ms		
399	9/3/2023 10:06	User_400	Web_Browsing	-113 dBm	0 ms		

	Required_Bandwidth	Allocated_Bandwidth	Resource_Allocation
0	10 Mbps	15 Mbps	70%
1	100 Kbps	120 Kbps	80%
2	5 Mbps	6 Mbps	75%
3	1 Mbps	1.5 Mbps	90%
4	2 Mbps	3 Mbps	85%
..	...	...	...
395	1.3 Mbps	1.8 Mbps	85%
396	14.5 Mbps	15.8 Mbps	75%
397	1.0 Mbps	1.4 Mbps	70%
398	0.4 Mbps	0.4 Mbps	70%
399	0.1 Mbps	0.1 Mbps	70%

```
[400 rows x 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

```
0s data.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.

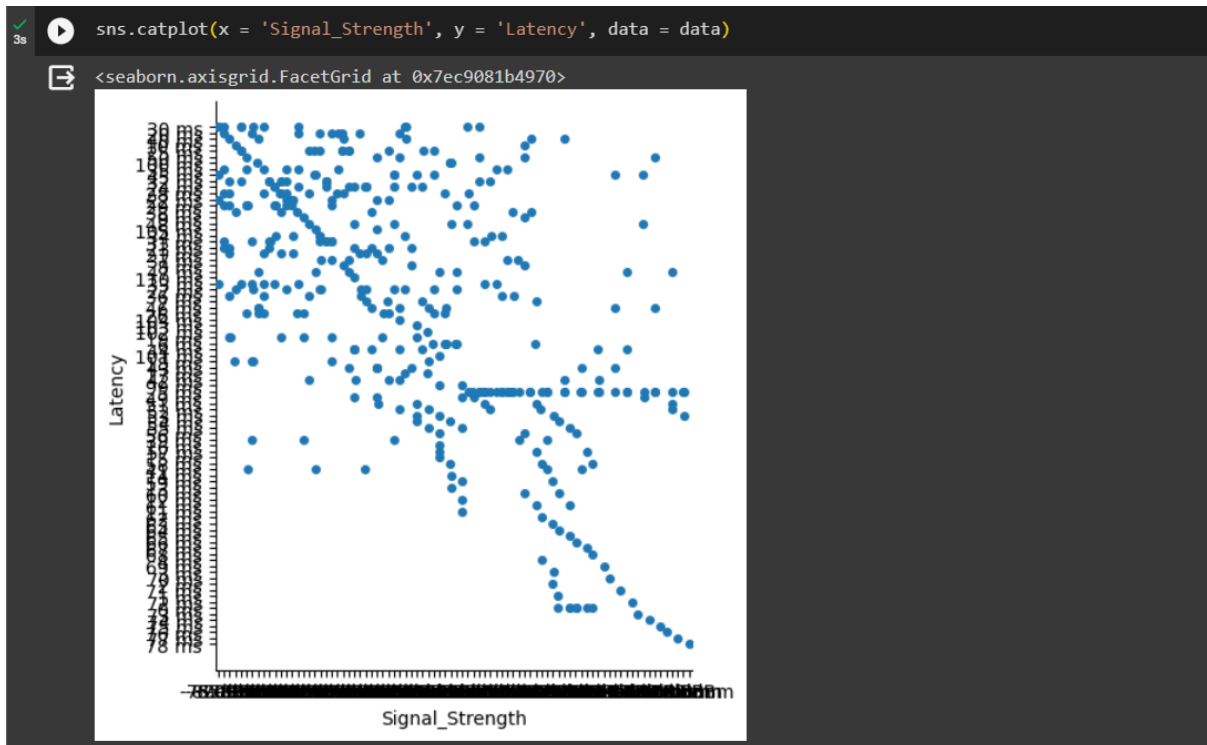
Here pandas have a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

```
[6] data.describe()
```

	Timestamp	User_ID	Application_Type	Signal_Strength	Latency	Required_Bandwidth	Allocated_Bandwidth	Resource_Allocation
count	400	400	400	400	400	400	400	400
unique	7	400	11	84	87	188	194	9
top	9/3/2023 10:01	User_1	Video_Call	-97 dBm	5 ms	0.1 Mbps	0.1 Mbps	70%
freq	60	1	58	9	35	16	16	148

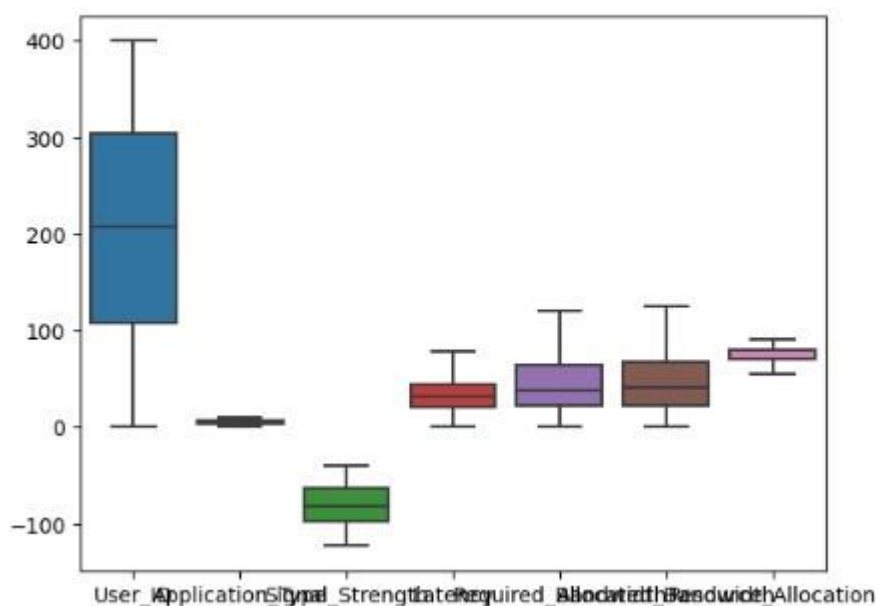
## Activity 2: Visual analysis

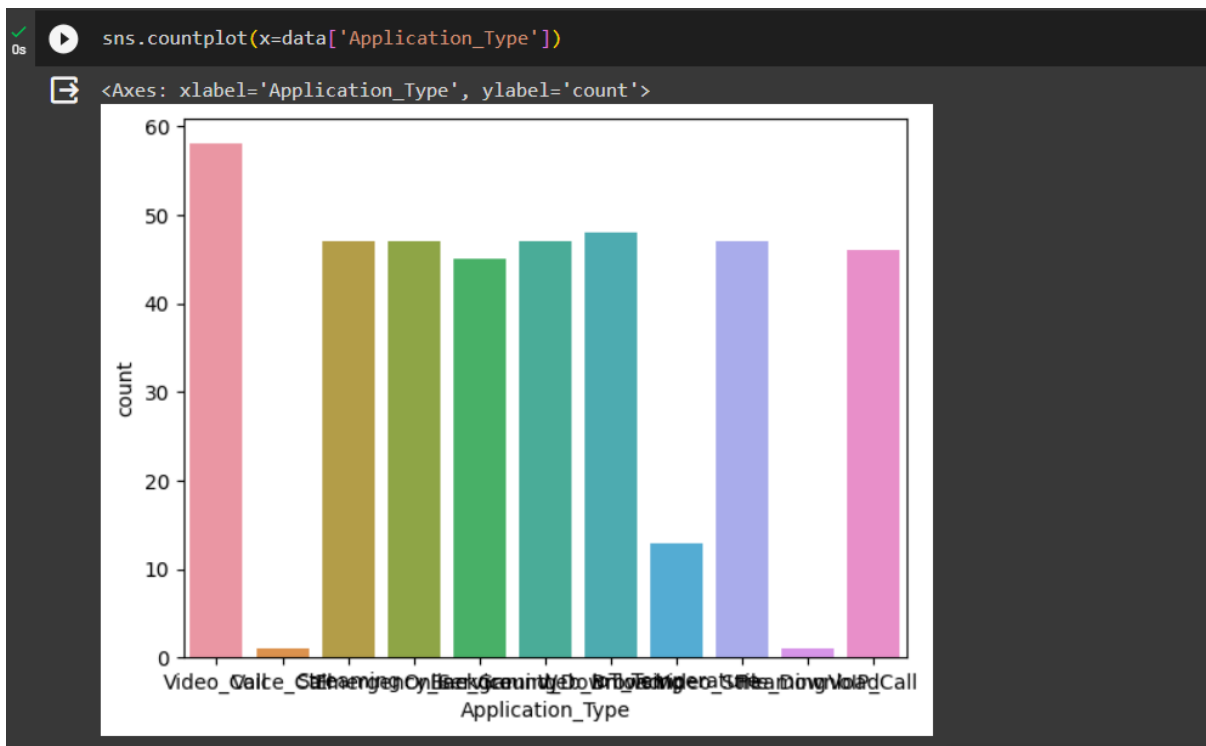
Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.



```
sns.boxplot(data = data)
```

<Axes: >





## 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 = df.drop('Resource_Allocation', axis=1)
y = df['Resource_Allocation']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
X_train
```

	Timestamp	User_ID	Application_Type	Signal_Strength	Latency	Required_Bandwidth	Allocated_Bandwidth
3	1970-01-01 00:00:00.000000000	333	1	54	2	11	16
18	1970-01-01 00:00:00.000000000	100	3	80	8	183	191
202	1970-01-01 00:00:00.000000003	116	4	49	23	108	120
250	1970-01-01 00:00:00.000000004	169	4	43	29	122	126
274	1970-01-01 00:00:00.000000004	195	4	40	33	125	143
...	...	...	...	...	...	...	...
71	1970-01-01 00:00:00.000000001	370	4	59	22	93	101
106	1970-01-01 00:00:00.000000001	9	0	81	46	177	182
270	1970-01-01 00:00:00.000000004	191	1	33	52	6	6
348	1970-01-01 00:00:00.000000005	277	6	28	49	67	69
102	1970-01-01 00:00:00.000000001	5	10	76	25	8	8

320 rows x 7 columns

## 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.

```
[23] from sklearn.ensemble import RandomForestRegressor
      from sklearn.metrics import mean_squared_error
      rf_regressor = RandomForestRegressor(random_state=42)
      rf_regressor.fit(X_train, y_train)
      rf_predictions = rf_regressor.predict(X_test)
      rf_mse = mean_squared_error(y_test, rf_predictions)
      print("Random Forest Regressor MSE:", rf_mse)
```

Random Forest Regressor MSE: 0.27952125

### 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.

```
[22] from sklearn.linear_model import LinearRegression
      linear_regressor = LinearRegression()
      linear_regressor.fit(X_train, y_train)
      linear_predictions = linear_regressor.predict(X_test)
      linear_mse = mean_squared_error(y_test, linear_predictions)
      print("Linear Regressor MSE:", linear_mse)
```

### 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 result is a tree with decision nodes and leaf nodes.

```
0s ✓ ▶ from sklearn.tree import DecisionTreeClassifier
      from sklearn.metrics import accuracy_score
      dt_classifier = DecisionTreeClassifier(random_state=42)
      dt_classifier.fit(X_train, y_train)
      dt_predictions = dt_classifier.predict(X_test)
      dt_accuracy = accuracy_score(y_test, dt_predictions)
      print("Decision Tree Classifier Accuracy:", dt_accuracy)

Decision Tree Classifier Accuracy: 0.925
```

## 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.

### Activity 1.1: Compare the model

For comparing the above four models, the compare Model function is defined.

### Linear Regression

```
0s ✓ [28] from sklearn.metrics import r2_score
      acc = r2_score(y_test, rf_predictions)
      acc

0.9202471579510498
```

### Decision Tree

```
0s ✓ [29] from sklearn.metrics import r2_score
      acc = r2_score(y_test, dt_predictions)
      acc

0.9108376799964335
```

## Random forest

```
✓ 0s ▶ from sklearn.metrics import r2_score
      acc = r2_score(y_test, linear_predictions)
      acc
0.18379814756322366
```

## 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.

```
✓ 0s [34] import pickle
✓ 0s [37] pickle.dump(dt_classifier, open('5g_allocation.pkl', 'wb'))
```

### 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, render_template, request
app = Flask(__name__)
import pickle
import joblib
```

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.

```
app = Flask(__name__)
import pickle
model = pickle.load(open("blood.pkl", "rb"))
```

### Render HTML page

```
@app.route('/user')
def user():
    return "hello user"
```

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("/") #decoratar
def index():
    return render_template("index.html")
@app.route("/about.html")
def about():
    return render_template("about.html")
@app.route("/pred.html")
def pred():
    return render_template("pred.html")

@app.route('/prediction', methods = ["POST","GET"])
def prediction():
    at = request.form["at"]
    ss = request.form["ss"]
    l = request.form["l"]
    r = request.form["r"]
    ab = request.form["ab"]
    data = [[at,ss,l,r,ab]]
    prediction = model.predict(data)
    #print(prediction)
    return render_template("pred.html",y=prediction)
if __name__ == "__main__":
    app.run(debug=False, port= 6090)
```

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 = False)
```

### 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.



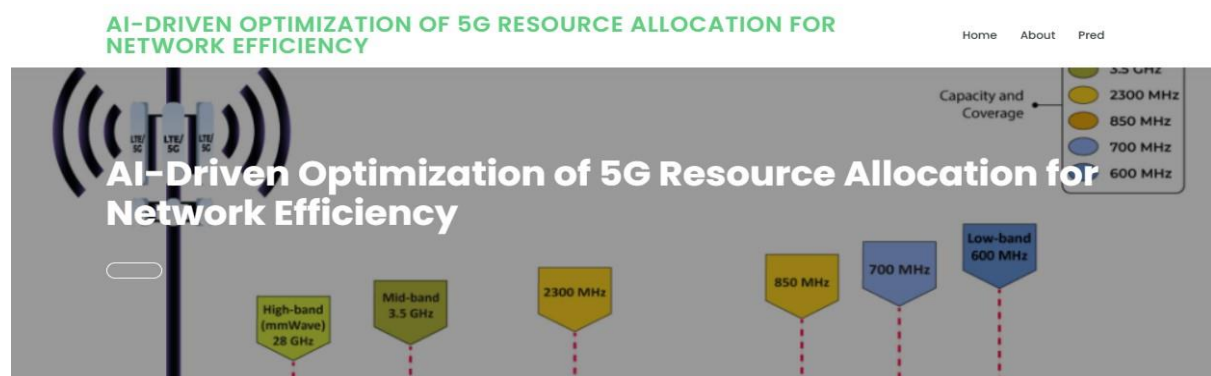
```

https://scikit-learn.org/stable/
model_persistence.html#security-maintainability-
limitations
  warnings.warn(
WARNING: This is a development server. Do not use it in a
production deployment. Use a production WSGI server
instead.
* Running on http://127.0.0.1:6090
Press CTRL+C to quit
127.0.0.1 - - [19/Sep/2023 12:14:59] "GET / HTTP/1.1" 200
-
127.0.0.1 - - [19/Sep/2023 12:14:59] "GET /static/assets/
vendor/boxicons/css/boxicons.min.css HTTP/1.1" 200 -
127.0.0.1 - - [19/Sep/2023 12:14:59] "GET /static/assets/
vendor/animate.css/animate.min.css HTTP/1.1" 200 -

```

Now, Go the web browser and write the localhost URL(  
http://127.0.0.1:6090/)to get the below result **Home**

**Page:**



**About Page:**



Welcome to the "5G Resource Allocation Dataset" – your gateway to understanding and optimizing the next-generation of wireless networks

For 5G to operate in the low-band, the n28 (700 MHz) is used, while in the mid-band of Sub-6, the n3 (1865 MHz), n7 (2655 MHz), and n78 (3500 MHz) bands are used. Moreover, for high-band mmWave, n258 (24 GHz) and n260 (37 GHz) are used. Network optimization aims to achieve several objectives in 5G networks, such as maximizing network performance, minimizing network cost, ensuring network reliability, and satisfying user expectations. These objectives are often conflicting and require trade-offs and compromises. 5G is divided into three frequency bands (low, mid, and high). Each band has different capabilities: the low band (less than 1GHz) has greater coverage but lower speeds, the mid band (1GHz–6GHz) offers a balance of both, and the high band (24GHz–40GHz) offers higher speeds but a smaller coverage radius.

**Predication Page:**

**This is a web application AI-Driven Optimization Of 5G Resource Allocation For Network Efficiency**

Application\_Type

6

Signal\_Strength

59

Latency

31

Required\_Bandwidth

23

Allocated\_Bandwidth

66

submit

{{y}}