<u>**Project Report:**</u>

Building Machine Learning Solutions for O-RAN Use Cases in 5G NR Networks

Submitted by:
Spoorthi M [CSE21193] & *Richa Vivek Savant* [CSE21171]

Under the supervision of:
Mr. Anoop Babu Thevara and *Mrs. Manjula Sathish*
[Team Mentors, NMS Group, Center for Development of Telematics]

On

August 3rd 2023

Department of Computer Science and Engineering

Amrita Vishwa Vidyapeetham

Bangalore, Karnataka, India

Individual Acknowledgements

I would like to extend my heartfelt gratitude to two individuals who have been instrumental in the successful completion of this project: First and foremost, I am deeply thankful to Anoop Babu Thevara, my esteemed mentor, who generously provided me with the valuable opportunity to work on this project. His guidance, expertise, and unwavering support throughout the entire journey were instrumental in shaping my understanding of the project's intricacies. His patience and willingness to share insights whenever I encountered challenges were remarkable.

Next, I extend my sincere appreciation to Manjula S., whose unwavering dedication and guidance were invaluable to me throughout the project. She patiently walked me through every step, ensuring I thoroughly comprehended each detail. Her ability to help me navigate and overcome obstacles with grace and kindness was truly commendable. Her input, suggestions and unwavering guidance have contributed significantly to the project's success. I could not have accomplished this without her constant encouragement and unwavering belief in my abilities.

Index

Introduction to O-RAN (Open Radio Access Network)

The telecommunications industry is undergoing a significant transformation with the evolution of 5G (Fifth Generation) technology. One of the key innovations driving this transformation is the concept of an Open Radio Access Network (O-RAN). O-RAN is a revolutionary approach to designing and deploying cellular network infrastructure that aims to improve network flexibility, efficiency, and innovation while reducing costs.

In an O-RAN architecture, the network components, such as baseband units, remote radio units, and control functions, are decoupled and can be provided by different vendors. This disaggregation promotes interoperability and encourages competition, leading to increased innovation and more cost-effective solutions.

O-RAN Alliance, a global industry consortium, plays a central role in driving the development and adoption of O-RAN principles and specifications.

In this context, we have developed machine-learning solutions for a few of the O-RAN use cases for further optimization.

Technology Used

The project focused on building machine learning solutions for O-RAN (Open Radio Access Network) use cases in 5G NR (New Radio) networks.

The goal was to address and implement solutions for various challenges including:

1. Context-based handover management in V2X communication
2. QoE optimization
3. QoS-based resource optimization
4. Local Indoor Positioning in RAN
5. Energy Saving

The Python programming language was chosen as the primary tool for implementing these solutions due to its versatility, rich libraries, and active developer community.

Machine Learning Libraries:

1. **scikit-learn**: This open-source machine learning library provides various tools for classification, regression, clustering, dimensionality reduction, and more. It offers a consistent interface for implementing various algorithms and model evaluation techniques.
2. **TensorFlow and PyTorch**: These deep learning frameworks enable the creation of complex neural network architectures. They offer automatic differentiation, GPU acceleration, and pre-built layers for building advanced models like convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

Documentation and Reporting:

‘Google Colab Notebooks’ was used for interactive development and documentation. It allowed the integration of code, visualizations, and explanatory text, making it easy to present the project's progress and results.

Methodology:

Building a machine learning (ML) model involves a series of steps that take you from data collection to verifying error rates.

A comprehensive methodology for building an ML model is as follows:

1. **Problem Definition and Data Collection:**

   Defining the problem to be solved with the ML model.
   Identify the variables (features) that are relevant to the given problem.
   Collect relevant and representative data that includes both the features and the target variable.
2. **Data Preprocessing:**

   Handle missing data by imputation or removal.
   Handle outliers that might adversely affect the model's performance.
   Normalize or scale features if necessary to ensure they're on similar scales.
   Encode categorical variables into numerical format (one-hot encoding, label encoding, etc.).
   Split the data into training, validation, and testing sets.
3. **Feature Engineering:**

   Create new features that might provide additional insights.
   Select or transform features based on domain knowledge.
4. **Model Selection:**

   Choose appropriate algorithms based on the problem type (classification, regression, etc.).
   Research and select algorithms that are well-suited for the dataset's size and complexity.
5. **Model Training:**

   Train the selected algorithms on the training data:
   1. Regression models
   2. Classification models
   To ensure the robustness of the machine learning models and prevent overfitting, techniques such as k-fold cross-validation were used.
   Hyperparameter tuning was performed using methods like GridSearchCV or RandomizedSearchCV to find optimal model configurations.
6. **Model Evaluation:**

   Choose appropriate evaluation metrics (accuracy, precision, recall, F1-score, etc.).

Adjust the model or hyperparameters if necessary based on validation results.

7. **Model Testing and Error Analysis:**

   Use the testing set to assess the final model's performance.
   Analyze the errors made by the model to gain insights into its strengths and weaknesses.

8. **Error Rates Verification:**

   Calculate the relevant error metrics on the testing set, such as accuracy, mean squared error, etc.
   Compare the error rates to the baseline or to other models to assess the model's effectiveness.

9. **Data Visualization:**

   The Python libraries Matplotlib and Seaborn were used for data visualization.
   Visualizing the data and model results helps in understanding the trends, patterns, and effectiveness of the developed solutions.

10. **Iterate and Improve:**

    Machine learning is an iterative process. Continuously gather feedback, monitor the model's performance, and make improvements as new data becomes available.

Use case 1: Context-Based Dynamic HO Management for V2X

Motivation:

V2X stands for vehicle-to-anything communication. As vehicles traverse along a highway, due to their high speed and the heterogeneous natural environment V2X UE-s are handed over frequently, at times in a suboptimal way, which may cause handover (HO) anomalies: e.g., short stay, ping-pong, and remote cell. Such suboptimal HO sequences substantially impair the functionality of V2X applications. Since HO sequences are mainly determined by the Neighbour Relation Tables (NRTs), maintained by the xNBs, there is hardly room for UE-level customization. This UC aims to present a method to avoid and/or resolve problematic HO scenarios by using past navigation and radio statistics in order to customize HO sequences on a UE level using AI/ML solutions.

Problem Statement:

For efficient handover optimization, the throughput has to be optimal. For ensuring this, we build a regression model for throughput prediction in V2X communication.

Reference Paper Used:

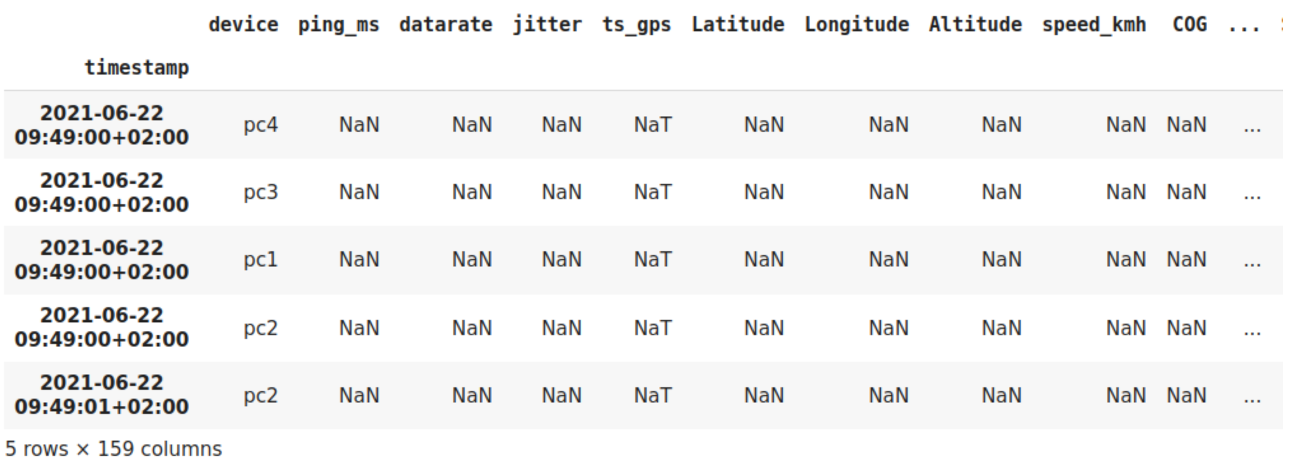
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9381854

Dataset Source and Particularities:

https://ieee-dataport.org/open-access/berlin-v2x

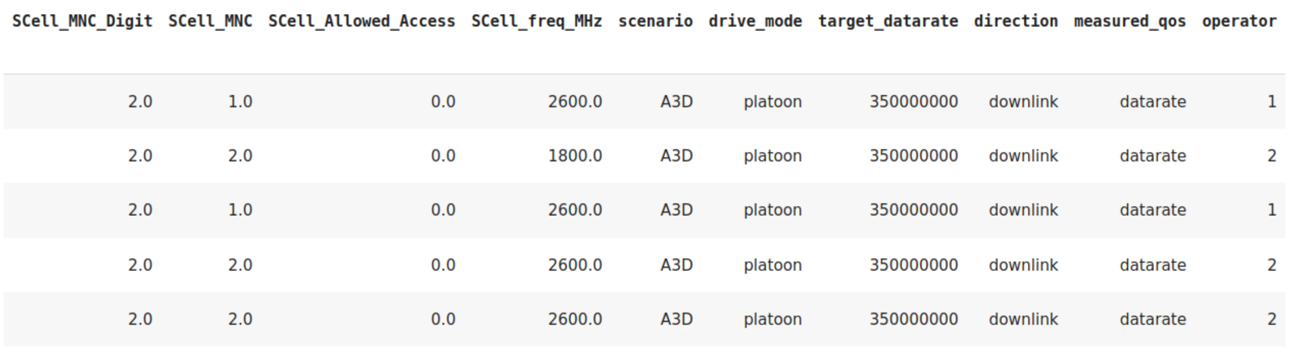
The Berlin V2X dataset offers high-resolution GPS-located wireless measurements across diverse urban environments in the city of Berlin for both cellular and sidelink radio access technologies, acquired with up to 4 cars over 3 days. The data enables thus a variety of different machine learning (ML) studies towards vehicle-to-anything (V2X) communication.

dataframe.head():



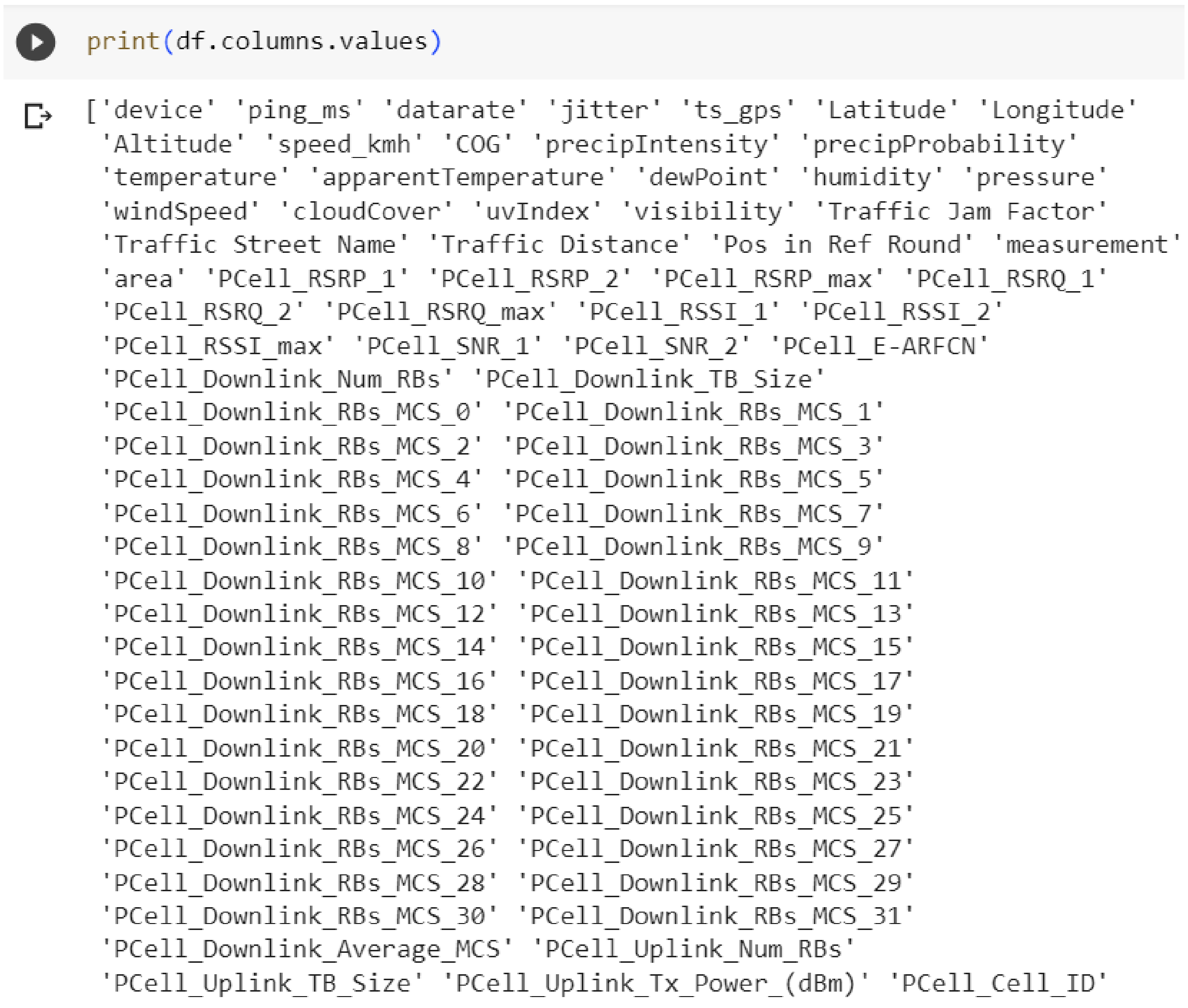
	device	ping_ms	datarate	jitter	ts_gps	Latitude	Longitude	Altitude	speed_kmh	COG	...
timestamp											
2021-06-22 09:49:00+02:00	pc4	NaN	NaN	NaN	NaT	NaN	NaN	NaN	NaN	NaN	...
2021-06-22 09:49:00+02:00	pc3	NaN	NaN	NaN	NaT	NaN	NaN	NaN	NaN	NaN	...
2021-06-22 09:49:00+02:00	pc1	NaN	NaN	NaN	NaT	NaN	NaN	NaN	NaN	NaN	...
2021-06-22 09:49:00+02:00	pc2	NaN	NaN	NaN	NaT	NaN	NaN	NaN	NaN	NaN	...
2021-06-22 09:49:01+02:00	pc2	NaN	NaN	NaN	NaT	NaN	NaN	NaN	NaN	NaN	...

5 rows × 159 columns



SCell_MNC_Digit	SCell_MNC	SCell_Allowed_Access	SCell_freq_MHz	scenario	drive_mode	target_datarate	direction	measured_qos	operator
2.0	1.0	0.0	2600.0	A3D	platoon	350000000	downlink	datarate	1
2.0	2.0	0.0	1800.0	A3D	platoon	350000000	downlink	datarate	2
2.0	1.0	0.0	2600.0	A3D	platoon	350000000	downlink	datarate	1
2.0	2.0	0.0	2600.0	A3D	platoon	350000000	downlink	datarate	2
2.0	2.0	0.0	2600.0	A3D	platoon	350000000	downlink	datarate	2

dataframe.columns:



```
print(df.columns.values)
```

```
['device' 'ping_ms' 'datarate' 'jitter' 'ts_gps' 'Latitude' 'Longitude'
 'Altitude' 'speed_kmh' 'COG' 'precipIntensity' 'precipProbability'
 'temperature' 'apparentTemperature' 'dewPoint' 'humidity' 'pressure'
 'windSpeed' 'cloudCover' 'uvIndex' 'visibility' 'Traffic Jam Factor'
 'Traffic Street Name' 'Traffic Distance' 'Pos in Ref Round' 'measurement'
 'area' 'PCell_RSRP_1' 'PCell_RSRP_2' 'PCell_RSRP_max' 'PCell_RSRQ_1'
 'PCell_RSRQ_2' 'PCell_RSRQ_max' 'PCell_RSSI_1' 'PCell_RSSI_2'
 'PCell_RSSI_max' 'PCell_SNR_1' 'PCell_SNR_2' 'PCell_E-ARFCN'
 'PCell_Downlink_Num_RBs' 'PCell_Downlink_TB_Size'
 'PCell_Downlink_RBs_MCS_0' 'PCell_Downlink_RBs_MCS_1'
 'PCell_Downlink_RBs_MCS_2' 'PCell_Downlink_RBs_MCS_3'
 'PCell_Downlink_RBs_MCS_4' 'PCell_Downlink_RBs_MCS_5'
 'PCell_Downlink_RBs_MCS_6' 'PCell_Downlink_RBs_MCS_7'
 'PCell_Downlink_RBs_MCS_8' 'PCell_Downlink_RBs_MCS_9'
 'PCell_Downlink_RBs_MCS_10' 'PCell_Downlink_RBs_MCS_11'
 'PCell_Downlink_RBs_MCS_12' 'PCell_Downlink_RBs_MCS_13'
 'PCell_Downlink_RBs_MCS_14' 'PCell_Downlink_RBs_MCS_15'
 'PCell_Downlink_RBs_MCS_16' 'PCell_Downlink_RBs_MCS_17'
 'PCell_Downlink_RBs_MCS_18' 'PCell_Downlink_RBs_MCS_19'
 'PCell_Downlink_RBs_MCS_20' 'PCell_Downlink_RBs_MCS_21'
 'PCell_Downlink_RBs_MCS_22' 'PCell_Downlink_RBs_MCS_23'
 'PCell_Downlink_RBs_MCS_24' 'PCell_Downlink_RBs_MCS_25'
 'PCell_Downlink_RBs_MCS_26' 'PCell_Downlink_RBs_MCS_27'
 'PCell_Downlink_RBs_MCS_28' 'PCell_Downlink_RBs_MCS_29'
 'PCell_Downlink_RBs_MCS_30' 'PCell_Downlink_RBs_MCS_31'
 'PCell_Downlink_Average_MCS' 'PCell_Uplink_Num_RBs'
 'PCell_Uplink_TB_Size' 'PCell_Uplink_Tx_Power_(dBm)' 'PCell_Cell_ID'
'PCell_Downlink_frequency' 'PCell_Uplink_frequency'
'PCell_Downlink_bandwidth_MHz' 'PCell_Uplink_bandwidth_MHz'
'PCell_Cell_Identity' 'PCell_TAC' 'PCell_Band_Indicator' 'PCell_MCC'
'PCell_MNC_Digit' 'PCell_MNC' 'PCell_Allowed_Access' 'PCell_freq_MHz'
'SCell_RSRP_1' 'SCell_RSRP_2' 'SCell_RSRP_max' 'SCell_RSRQ_1'
'SCell_RSRQ_2' 'SCell_RSRQ_max' 'SCell_RSSI_1' 'SCell_RSSI_2'
'SCell_RSSI_max' 'SCell_SNR_1' 'SCell_SNR_2' 'SCell_E-ARFCN'
'SCell_Downlink_Num_RBs' 'SCell_Downlink_TB_Size'
'SCell_Downlink_RBs_MCS_0' 'SCell_Downlink_RBs_MCS_1'
'SCell_Downlink_RBs_MCS_2' 'SCell_Downlink_RBs_MCS_3'
'SCell_Downlink_RBs_MCS_4' 'SCell_Downlink_RBs_MCS_5'
'SCell_Downlink_RBs_MCS_6' 'SCell_Downlink_RBs_MCS_7'
'SCell_Downlink_RBs_MCS_8' 'SCell_Downlink_RBs_MCS_9'
'SCell_Downlink_RBs_MCS_10' 'SCell_Downlink_RBs_MCS_11'
'SCell_Downlink_RBs_MCS_12' 'SCell_Downlink_RBs_MCS_13'
'SCell_Downlink_RBs_MCS_14' 'SCell_Downlink_RBs_MCS_15'
'SCell_Downlink_RBs_MCS_16' 'SCell_Downlink_RBs_MCS_17'
'SCell_Downlink_RBs_MCS_18' 'SCell_Downlink_RBs_MCS_19'
'SCell_Downlink_RBs_MCS_20' 'SCell_Downlink_RBs_MCS_21'
'SCell_Downlink_RBs_MCS_22' 'SCell_Downlink_RBs_MCS_23'
'SCell_Downlink_RBs_MCS_24' 'SCell_Downlink_RBs_MCS_25'
'SCell_Downlink_RBs_MCS_26' 'SCell_Downlink_RBs_MCS_27'
'SCell_Downlink_RBs_MCS_28' 'SCell_Downlink_RBs_MCS_29'
'SCell_Downlink_RBs_MCS_30' 'SCell_Downlink_RBs_MCS_31'
'SCell_Downlink_Average_MCS' 'SCell_Uplink_Num_RBs'
'SCell_Uplink_TB_Size' 'SCell_Uplink_Tx_Power_(dBm)' 'SCell_Cell_ID'
'SCell_Downlink_frequency' 'SCell_Uplink_frequency'
'SCell_Downlink_bandwidth_MHz' 'SCell_Uplink_bandwidth_MHz'
'SCell_Cell_Identity' 'SCell_TAC' 'SCell_Band_Indicator' 'SCell_MCC'
'SCell_MNC_Digit' 'SCell_MNC' 'SCell_Allowed_Access' 'SCell_freq_MHz'
'scenario' 'drive_mode' 'target_datarate' 'direction' 'measured_qos'
'operator']
```

The dataset is further subdivided into 3 data frames to serve 3 different application sets:

1) File transfer, TCP session throughput

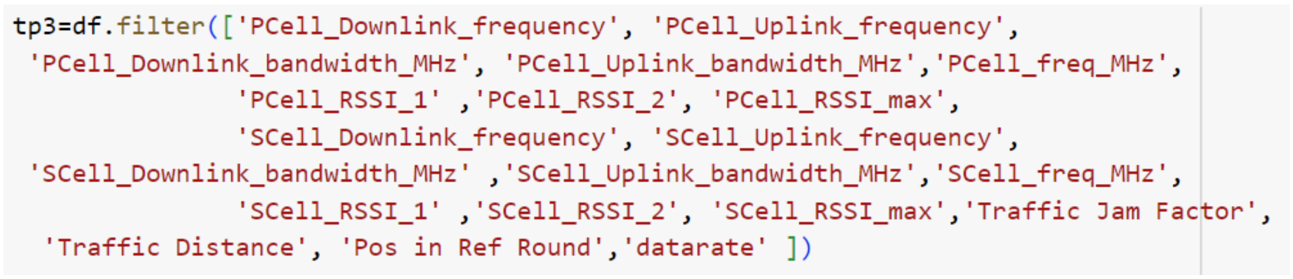
```
tp1 = df.filter(['Latitude','Longitude',
                    'PCell_RSRP_1', 'PCell_RSRP_2' ,'PCell_RSRP_max', 'PCell_RSRQ_1',
 'PCell_RSRQ_2' ,'PCell_RSRQ_max', 'PCell_RSSI_1' ,'PCell_RSSI_2',
 'PCell_RSSI_max', 'PCell_SNR_1' ,'PCell_SNR_2',
                    'SCell_RSRP_1' ,'SCell_RSRP_2', 'SCell_RSRP_max' ,'SCell_RSRQ_1',
 'SCell_RSRQ_2' ,'SCell_RSRQ_max', 'SCell_RSSI_1' ,'SCell_RSSI_2',
 'SCell_RSSI_max', 'SCell_SNR_1', 'SCell_SNR_2', 'target_datarate','datarate'], axis=1)
```

2) Frame Size Optimization

```
tp2 = df.filter([ 'PCell_Downlink_Num_RBs' ,'PCell_Downlink_TB_Size','PCell_Downlink_Average_MCS' 'PCell_Uplink_Num_RBs'
 'PCell_Uplink_TB_Size' 'PCell_Uplink_Tx_Power_(dBm)'
 'PCell_SNR_1' ,'PCell_SNR_2',
               'SCell_Downlink_Num_RBs', 'SCell_Downlink_TB_Size' ,'SCell_SNR_1', 'SCell_SNR_2',
                   'SCell_Downlink_Average_MCS',

 'SCell_Downlink_frequency', 'SCell_Uplink_frequency',
 'SCell_Downlink_bandwidth_MHz' ,'SCell_Uplink_bandwidth_MHz','datarate'], axis=1)
```

3) Interference classification and Channel selection



```
tp3=df.filter(['PCell_Downlink_frequency', 'PCell_Uplink_frequency',
 'PCell_Downlink_bandwidth_MHz', 'PCell_Uplink_bandwidth_MHz','PCell_freq_MHz',
               'PCell_RSSI_1' ,'PCell_RSSI_2', 'PCell_RSSI_max',
               'SCell_Downlink_frequency', 'SCell_Uplink_frequency',
 'SCell_Downlink_bandwidth_MHz' ,'SCell_Uplink_bandwidth_MHz','SCell_freq_MHz',
               'SCell_RSSI_1' ,'SCell_RSSI_2', 'SCell_RSSI_max','Traffic Jam Factor',
  'Traffic Distance', 'Pos in Ref Round','datarate' ])
```

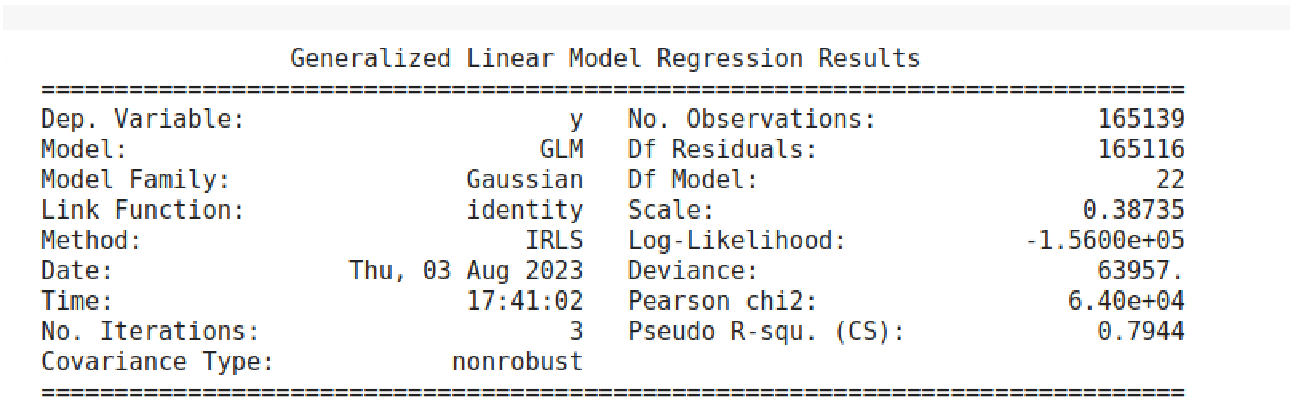
ML Algorithms Used:

1. File transfer, TCP session throughput (tp1):
    a. Generalized Linear Model
    b. KNN Regression
    c. Neural Network
2. Frame Size Optimization (tp2):
    a. Multi-Layer Perceptron Regressor
3. Interference classification and Channel selection (tp3):
    a. Linear Regression

Output:

Error ratios - Mean Square Error

1. Generalized Linear Model



```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:               165139
Model:                            GLM   Df Residuals:                   165116
Model Family:                Gaussian   Df Model:                           22
Link Function:               identity   Scale:                         0.38735
Method:                          IRLS   Log-Likelihood:            -1.5600e+05
Date:                Thu, 03 Aug 2023   Deviance:                       63957.
Time:                        17:41:02   Pearson chi2:                 6.40e+04
No. Iterations:                     3   Pseudo R-squ. (CS):             0.7944
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const       2.682e-18   2.06e-19     13.035      0.000    2.28e-18    3.09e-18
x1             0.0104      0.002      5.573      0.000       0.007       0.014
x2             0.0339      0.002     18.171      0.000       0.030       0.038
x3            -0.4819      0.024    -20.207      0.000      -0.529      -0.435
x4             0.1207      0.016      7.641      0.000       0.090       0.152
x5             0.6762      0.016     41.016      0.000       0.644       0.709
x6            -0.0102      0.010     -1.044      0.296      -0.029       0.009
x7             0.0211      0.003      7.014      0.000       0.015       0.027
x8            -0.2165      0.009    -25.072      0.000      -0.233      -0.200
x9            -0.1666      0.015    -11.048      0.000      -0.196      -0.137
x10           -0.0069      0.007     -0.984      0.325      -0.021       0.007
x11           -0.0069      0.007     -0.984      0.325      -0.021       0.007
x12            0.2897      0.006     50.860      0.000       0.278       0.301
x13           -0.0895      0.006    -13.965      0.000      -0.102      -0.077
x14            0.0987      0.024      4.199      0.000       0.053       0.145
x15           -0.6301      0.026    -24.486      0.000      -0.681      -0.580
x16           -0.0211      0.012     -1.814      0.070      -0.044       0.002
x17           -1.6163      0.052    -31.146      0.000      -1.718      -1.515
x18            0.2486      0.038      6.575      0.000       0.174       0.323
x19            1.7910      0.054     33.274      0.000       1.686       1.897
x20           -0.0879      0.019     -4.526      0.000      -0.126      -0.050
x21            0.3401      0.011     31.050      0.000       0.319       0.362
x22            0.3401      0.011     31.050      0.000       0.319       0.362
x23            0.2732      0.006     49.238      0.000       0.262       0.284
x24            0.0177      0.007      2.660      0.008       0.005       0.031
x25            0.2555      0.002    130.415      0.000       0.252       0.259
==============================================================================
```

2. KNN Regression

```
k = 5
knn_model = KNeighborsRegressor(n_neighbors=k)
```

```
knn_model.fit(X_train, y_train)
```

```
▾ KNeighborsRegressor
KNeighborsRegressor()
```

The Mean squared error and R-squared error are as follows:

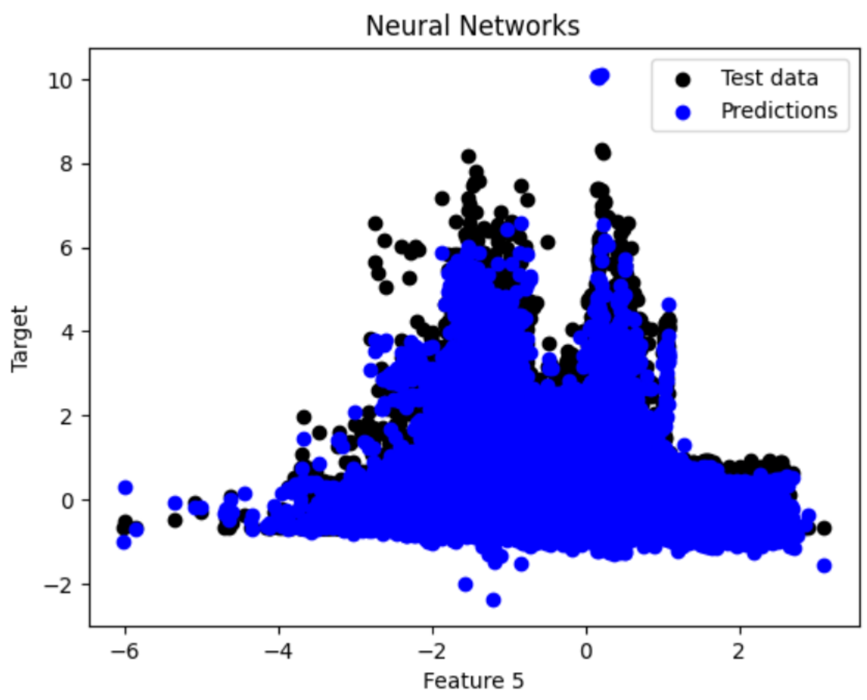
```
Mean Squared Error: 0.0857
R-squared: 0.9151
```

3. Neural Network

```
model = keras.Sequential([
    keras.layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(1)  # Output layer with a single neuron for regression
])
```

```
model.fit(X_train, y_train, epochs=10, batch_size=32, verbose=1)
```

```
Epoch 1/10
5161/5161 [==============================] - 12s 2ms/step - loss: 0.2383
Epoch 2/10
5161/5161 [==============================] - 11s 2ms/step - loss: 0.1928
Epoch 3/10
5161/5161 [==============================] - 17s 3ms/step - loss: 0.1775
Epoch 4/10
5161/5161 [==============================] - 22s 4ms/step - loss: 0.1663
Epoch 5/10
5161/5161 [==============================] - 15s 3ms/step - loss: 0.1586
Epoch 6/10
5161/5161 [==============================] - 11s 2ms/step - loss: 0.1525
Epoch 7/10
5161/5161 [==============================] - 10s 2ms/step - loss: 0.1481
Epoch 8/10
5161/5161 [==============================] - 9s 2ms/step - loss: 0.1447
Epoch 9/10
5161/5161 [==============================] - 10s 2ms/step - loss: 0.1429
Epoch 10/10
5161/5161 [==============================] - 10s 2ms/step - loss: 0.1389
<keras.callbacks.History at 0x7e42ef17a590>
```



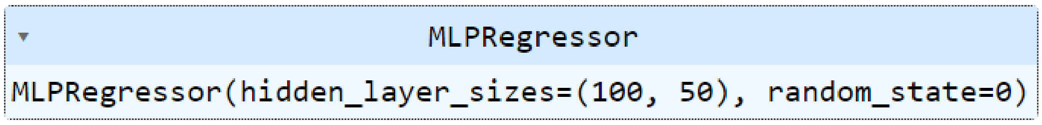
The Mean absolute error is as follows:

```
0.21094070942789594
```

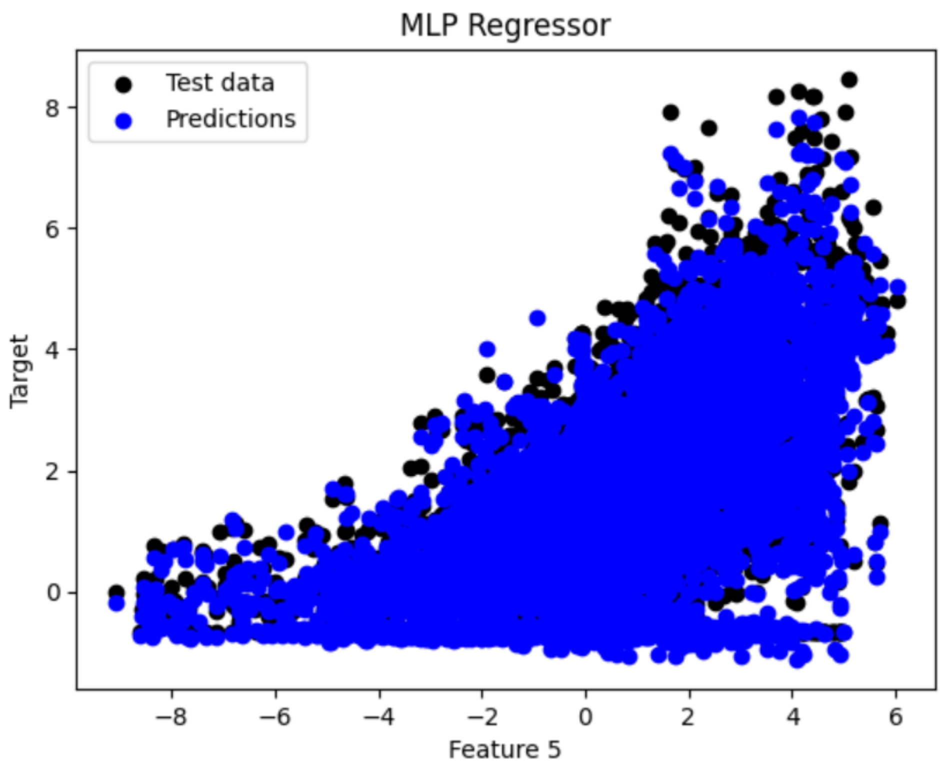
4. Multi-Layer Perceptron Regressor

```
hidden_layer_sizes = (100, 50)  # Number of neurons in each hidden layer
activation = 'relu'             # Activation function
alpha = 0.0001                  # L2 regularization parameter

model = MLPRegressor(hidden_layer_sizes=hidden_layer_sizes, activation=activation, alpha=alpha, random_state=0)
```



```
MLPRegressor
MLPRegressor(hidden_layer_sizes=(100, 50), random_state=0)
```



The Mean squared error is as follows:

```
mse = mean_squared_error(y_test_scaled, y_pred)
print(f"Mean Squared Error: {mse}")
```

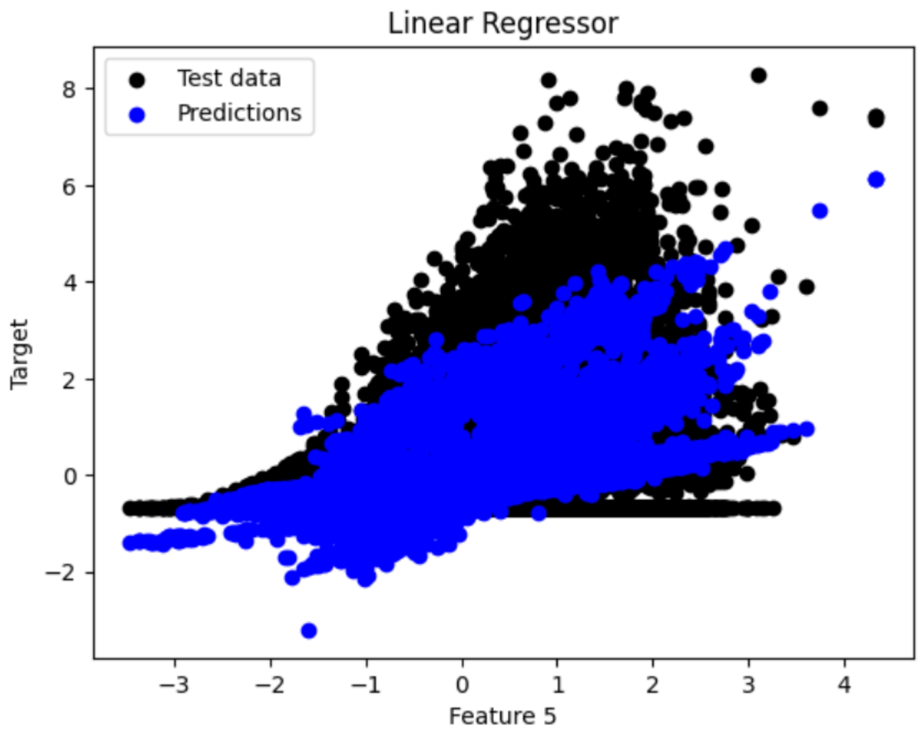
```
Mean Squared Error: 0.1566944037647099
```

5. Linear Regression

```
model = LinearRegression()

# Train the model on the training data
model.fit(X_train_scaled, y_train_scaled)

# Make predictions on the test data
y_pred = model.predict(X_test_scaled)
```



The Mean squared error is as follows:

```
mse = mean_squared_error(y_test_scaled, y_pred)
print(f"Mean Squared Error: {mse}")
```

```
Mean Squared Error: 0.7193982167890219
```

Colab Notebook Link:

https://colab.research.google.com/drive/16YcXrV1uWOwoNB4P0RMO7yVRBEvmzqkN?usp=sharing

Use case 2:QoE optimization

Motivation:

In the context of ORAN (Open Radio Access Network), QoE stands for "Quality of Experience." QoE refers to the overall perception of users or subscribers regarding the quality and performance of the services they receive over the wireless network. It takes into account various factors, including but not limited to throughput, latency, packet loss and availability of service.

Improving QoE is a critical objective for network operators as it directly impacts user satisfaction and, in turn, customer retention and loyalty. ORAN aims to provide a more open and flexible RAN architecture that can help enhance QoE by enabling interoperability between equipment from different vendors and promoting innovation in radio access network technologies. This can lead to more efficient and effective network deployment and management, ultimately contributing to a better user experience.

Problem Statement:

Efficiently managing the complex and dynamic ORAN environment to provide seamless, high-quality services to users across diverse use cases can be challenging. As the number of connected devices and data-intensive applications increases, ensuring optimal QoE is vital to meet user expectations and maintain customer loyalty. We use classification algorithms to predict an optimal Mean opinion Score.

Reference Papers Used:

https://ieeexplore.ieee.org/document/8930519

https://arxiv.org/pdf/2202.02454.pdf

Dataset Source and Particularities:

https://github.com/Lamyne/Poqemon-QoE-Dataset/tree/master

This work is proposed to describe a new dataset that contains many QoE Influence Factors (QoE IFs) and subjective Mean Opinion Scores(MOS).

--> Number of Instances :

-class 1 (MOS = 1): 92

-class 2 (MOS = 2): 119

-class 3 (MOS = 3): 244

-class 4 (MOS = 4): 787

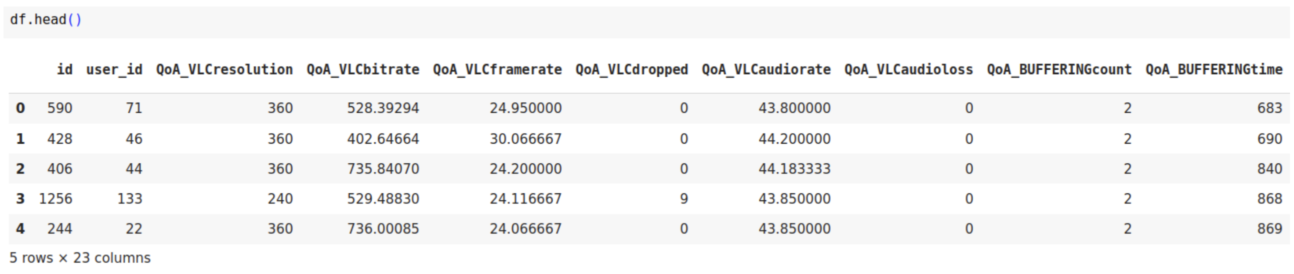
-class 5 (MOS = 5): 300

For calculating the mean opinion score many factors were taken into consideration and accordingly a number was assigned to each of the classifications like:

(MOS) Mean Opinion Score that a tester will give at the end of each video view.

- 5 -> Excellent
- 4 -> Good
- 3 -> Fair
- 2 -> Poor
- 1 -> Bad

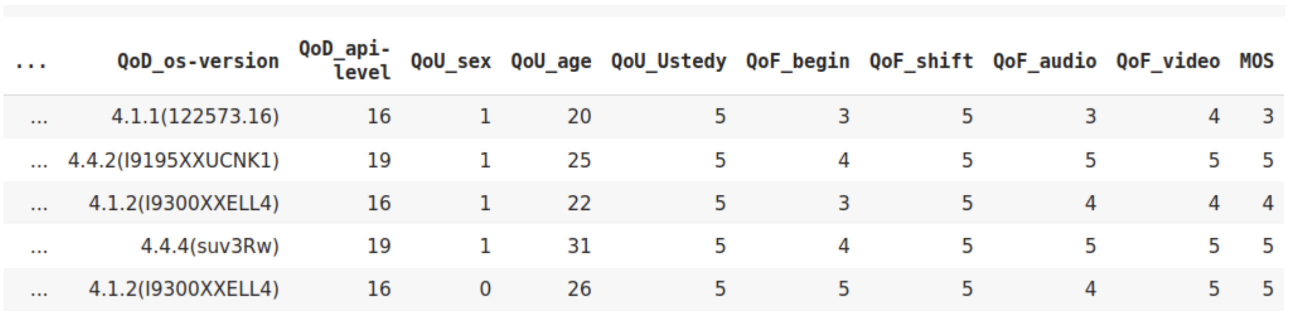
dataframe.head():



```
df.head()
```

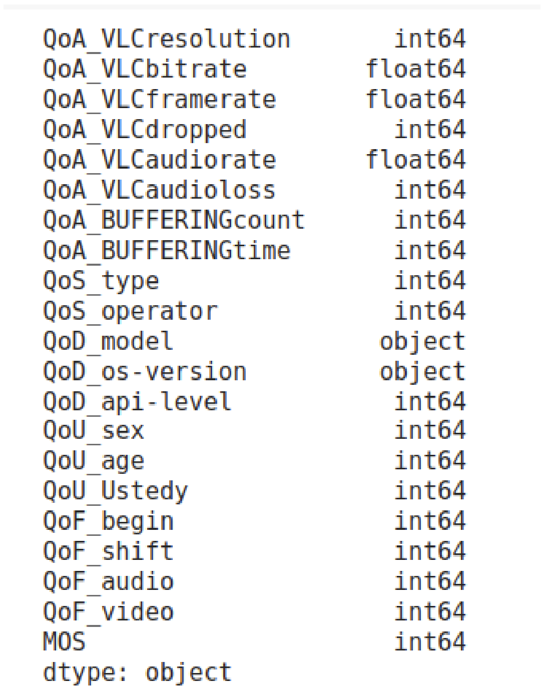
	id	user_id	QoA_VLCresolution	QoA_VLCbitrate	QoA_VLCframerate	QoA_VLCdropped	QoA_VLCaudiorate	QoA_VLCaudioloss	QoA_BUFFERINGcount	QoA_BUFFERINGtime
0	590	71	360	528.39294	24.950000	0	43.800000	0	2	683
1	428	46	360	402.64664	30.066667	0	44.200000	0	2	690
2	406	44	360	735.84070	24.200000	0	44.183333	0	2	840
3	1256	133	240	529.48830	24.116667	9	43.850000	0	2	868
4	244	22	360	736.00085	24.066667	0	43.850000	0	2	869

5 rows × 23 columns



...	QoD_os-version	QoD_api-level	QoU_sex	QoU_age	QoU_Ustedy	QoF_begin	QoF_shift	QoF_audio	QoF_video	MOS
...	4.1.1(122573.16)	16	1	20	5	3	5	3	4	3
...	4.4.2(I9195XXUCNK1)	19	1	25	5	4	5	5	5	5
...	4.1.2(I9300XXELL4)	16	1	22	5	3	5	4	4	4
...	4.4.4(suv3Rw)	19	1	31	5	4	5	5	5	5
...	4.1.2(I9300XXELL4)	16	0	26	5	5	5	4	5	5

dataframe.columns:



```
QoA_VLCresolution       int64
QoA_VLCbitrate        float64
QoA_VLCframerate      float64
QoA_VLCdropped          int64
QoA_VLCaudiorate      float64
QoA_VLCaudioloss        int64
QoA_BUFFERINGcount      int64
QoA_BUFFERINGtime       int64
QoS_type                int64
QoS_operator            int64
QoD_model              object
QoD_os-version         object
QoD_api-level           int64
QoU_sex                 int64
QoU_age                 int64
QoU_Ustedy              int64
QoF_begin               int64
QoF_shift               int64
QoF_audio               int64
QoF_video               int64
MOS                     int64
dtype: object
```

ML Algorithms Used:

1. Support Vector machine (SVM) classifier
2. K-Nearest Neighbour (K-NN) classifier
3. Random Forest (RF) classifier
4. Gradient tree Boosting (GB)
5. Multi-Layer Perceptron (MLP) based Neural Network (NN)
6. Stochastic Gradient Descent (SGD) classifier
7. Decision Tree (DT) classifier

Output:

Error ratios - Mean Square Error [Optimal Algorithm → Gradient Tree Boosting Classifier]

1. Support Vector Machine (SVM) classifier:

```
[1 4 4 4 4 4 1 4 4 4 4 4 2 4 4 4 4 4 4 4 4 4 4 1 4 4 4 4 4 4 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 1 1 4 4 4 4 4
 4 4 4 4 4 4 1 4 4 4 4 4 4 4 4 4 4 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 4 4 2 4 1 4 4 4 4 4 4 4 4 4 4 4 4 4 1 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 1 4 4 2 4 4 4 2 4 4 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 4 4 4 4 4 4 2 2 4 4 4 4 4 4 4 4 4 4 4 1 1 4 4 4 4 4 4 4 4 4 4 4 4 1 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 1 4 4
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 1 4 4 4 4 1 4 4 4 4 2 4 4 4 4 4 4 4 1 4 4 4
 4 4 4 4 4 4 4 2 4 4 4 4 4]
[[ 17   0   1   0   0]
 [  1   6   0   0   1]
 [  0   0   0   0   0]
 [  2  13  44 162  62]
 [  0   0   0   0   0]]
```

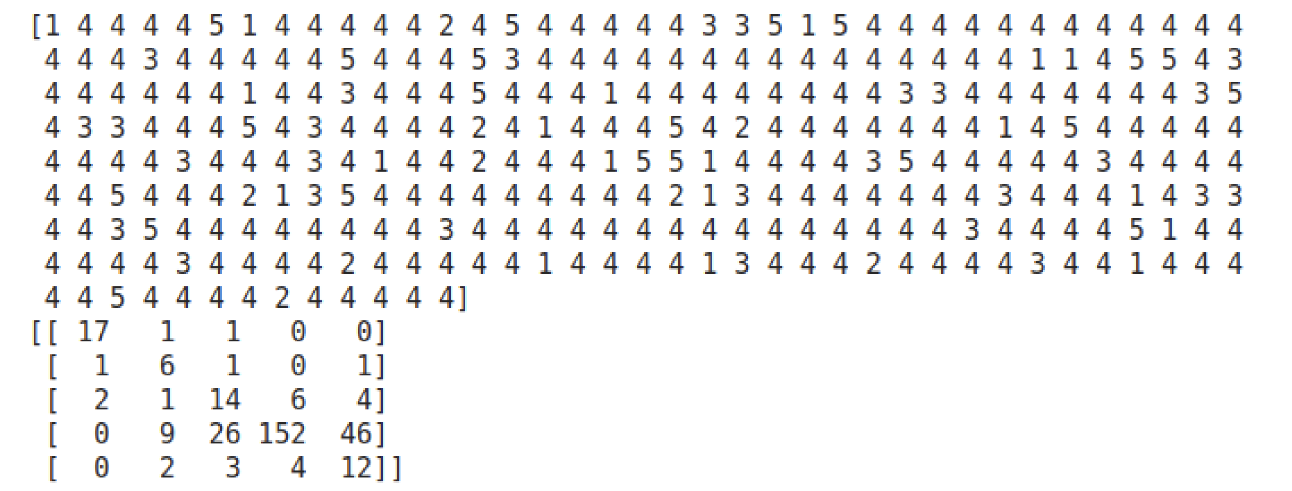
The Mean square error, Root mean square error and mean absolute error are as follows:

```
0.6148867313915858
The difference between actual and predicted values 0.7841471363153638
```

```
mean_absolute_error(y_test, y_pred)
```

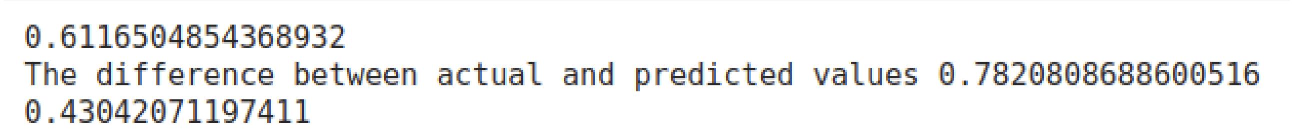
```
0.46601941747572817
```

2. K-Nearest Neighbour (K-NN) classifier:



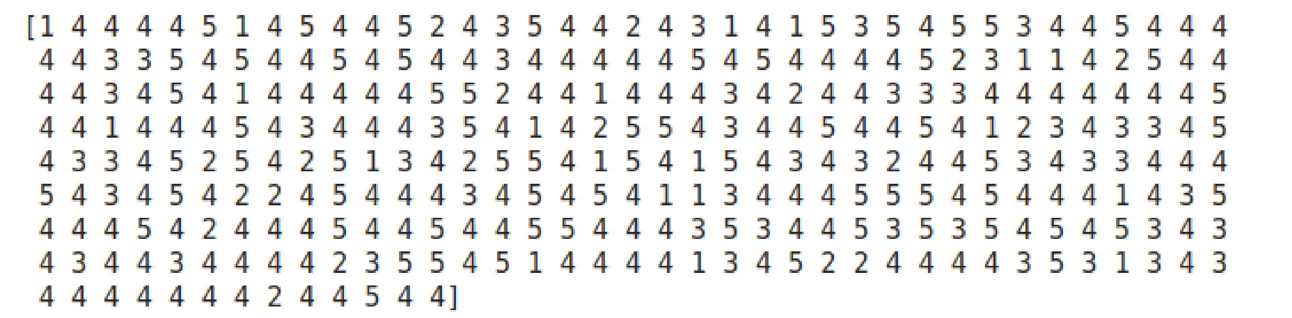
```
[1 4 4 4 4 5 1 4 4 4 4 4 2 4 5 4 4 4 4 4 3 3 5 1 5 4 4 4 4 4 4 4 4 4 4 4 4
 4 4 4 3 4 4 4 4 4 5 4 4 4 5 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 1 1 4 5 5 4 3
 4 4 4 4 4 4 1 4 4 3 4 4 4 5 4 4 4 1 4 4 4 4 4 4 4 4 3 3 4 4 4 4 4 4 4 3 5
 4 3 3 4 4 4 5 4 3 4 4 4 4 2 4 1 4 4 4 5 4 2 4 4 4 4 4 4 4 1 4 5 4 4 4 4 4
 4 4 4 4 3 4 4 4 3 4 1 4 4 2 4 4 4 1 5 5 1 4 4 4 4 3 5 4 4 4 4 4 3 4 4 4 4
 4 4 5 4 4 4 2 1 3 5 4 4 4 4 4 4 4 4 4 2 1 3 4 4 4 4 4 4 4 3 4 4 4 1 4 3 3
 4 4 3 5 4 4 4 4 4 4 4 4 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 4 4 4 4 5 1 4 4
 4 4 4 4 3 4 4 4 4 2 4 4 4 4 4 1 4 4 4 4 1 3 4 4 4 2 4 4 4 4 3 4 4 1 4 4 4
 4 4 5 4 4 4 4 2 4 4 4 4 4]
[[ 17   1   1   0   0]
 [  1   6   1   0   1]
 [  2   1  14   6   4]
 [  0   9  26 152  46]
 [  0   2   3   4  12]]
```

The Mean square error, Root mean square error and mean absolute error are as follows:



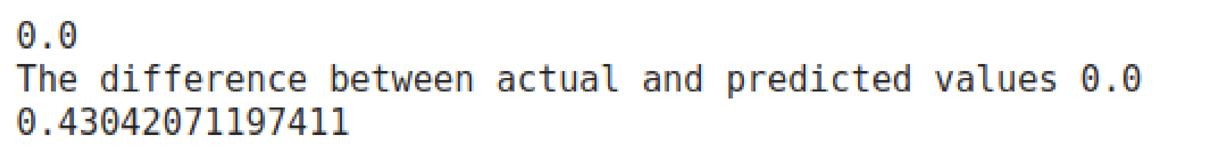
```
0.6116504854368932
The difference between actual and predicted values 0.7820808688600516
0.43042071197411
```

3. Random Forest (RF) classifier:



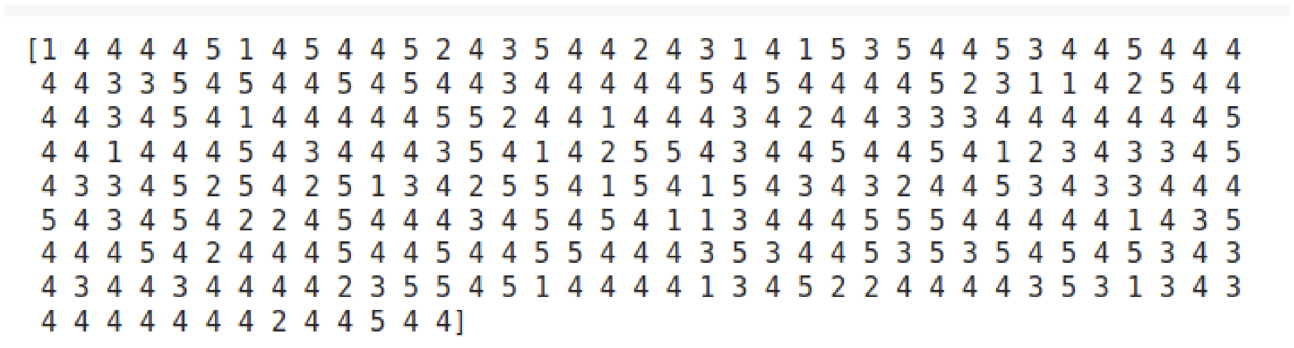
```
[1 4 4 4 4 5 1 4 5 4 4 5 2 4 3 5 4 4 2 4 3 1 4 1 5 3 5 4 5 5 3 4 4 5 4 4 4
 4 4 3 3 5 4 5 4 4 5 4 5 4 4 3 4 4 4 4 4 5 4 5 4 4 4 4 5 2 3 1 1 4 2 5 4 4
 4 4 3 4 5 4 1 4 4 4 4 4 5 5 2 4 4 1 4 4 4 3 4 2 4 4 3 3 3 4 4 4 4 4 4 4 5
 4 4 1 4 4 4 5 4 3 4 4 4 3 5 4 1 4 2 5 5 4 3 4 4 5 4 4 5 4 1 2 3 4 3 3 4 5
 4 3 3 4 5 2 5 4 2 5 1 3 4 2 5 5 4 1 5 4 1 5 4 3 4 3 2 4 4 5 3 4 3 3 4 4 4
 5 4 3 4 5 4 2 2 4 5 4 4 4 3 4 5 4 5 4 1 1 3 4 4 4 5 5 5 4 5 4 4 4 1 4 3 5
 4 4 4 5 4 2 4 4 4 5 4 4 5 4 4 5 5 4 4 4 3 5 3 4 4 5 3 5 3 5 4 5 4 5 3 4 3
 4 3 4 4 3 4 4 4 4 2 3 5 5 4 5 1 4 4 4 4 1 3 4 5 2 2 4 4 4 4 3 5 3 1 3 4 3
 4 4 4 4 4 4 4 2 4 4 5 4 4]
```

The Mean square error, Root mean square error and mean absolute error are as follows:



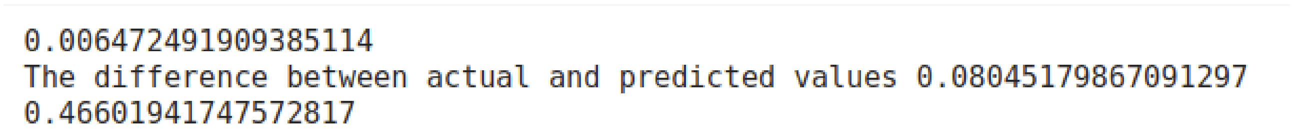
```
0.0
The difference between actual and predicted values 0.0
0.43042071197411
```

4. Gradient tree Boosting (GB):



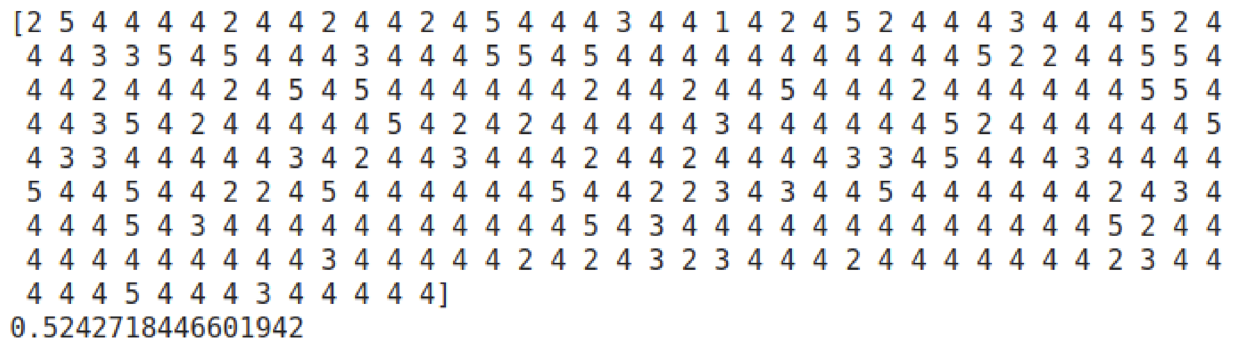
```
[1 4 4 4 4 5 1 4 5 4 4 5 2 4 3 5 4 4 2 4 3 1 4 1 5 3 5 4 4 5 3 4 4 5 4 4 4
 4 4 3 3 5 4 5 4 4 5 4 5 4 4 3 4 4 4 4 4 5 4 5 4 4 4 4 5 2 3 1 1 4 2 5 4 4
 4 4 3 4 5 4 1 4 4 4 4 4 5 5 2 4 4 1 4 4 4 3 4 2 4 4 3 3 3 4 4 4 4 4 4 4 5
 4 4 1 4 4 4 5 4 3 4 4 4 3 5 4 1 4 2 5 5 4 3 4 4 5 4 4 5 4 1 2 3 4 3 3 4 5
 4 3 3 4 5 2 5 4 2 5 1 3 4 2 5 5 4 1 5 4 1 5 4 3 4 3 2 4 4 5 3 4 3 3 4 4 4
 5 4 3 4 5 4 2 2 4 5 4 4 4 3 4 5 4 5 4 1 1 3 4 4 4 5 5 5 4 4 4 4 4 1 4 3 5
 4 4 4 5 4 2 4 4 4 5 4 4 5 4 4 5 5 4 4 4 3 5 3 4 4 5 3 5 3 5 4 5 4 5 3 4 3
 4 3 4 4 3 4 4 4 4 2 3 5 5 4 5 1 4 4 4 4 1 3 4 5 2 2 4 4 4 4 3 5 3 1 3 4 3
 4 4 4 4 4 4 4 2 4 4 5 4 4]
```

The Mean square error, Root mean square error and mean absolute error are as follows:



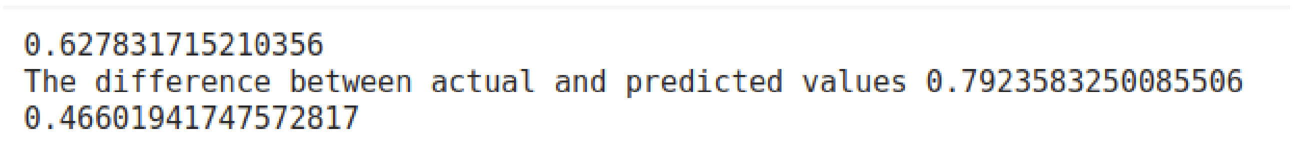
```
0.006472491909385114
The difference between actual and predicted values 0.08045179867091297
0.46601941747572817
```

5. Multi-Layer Perceptron (MLP) based Neural Network (NN) :



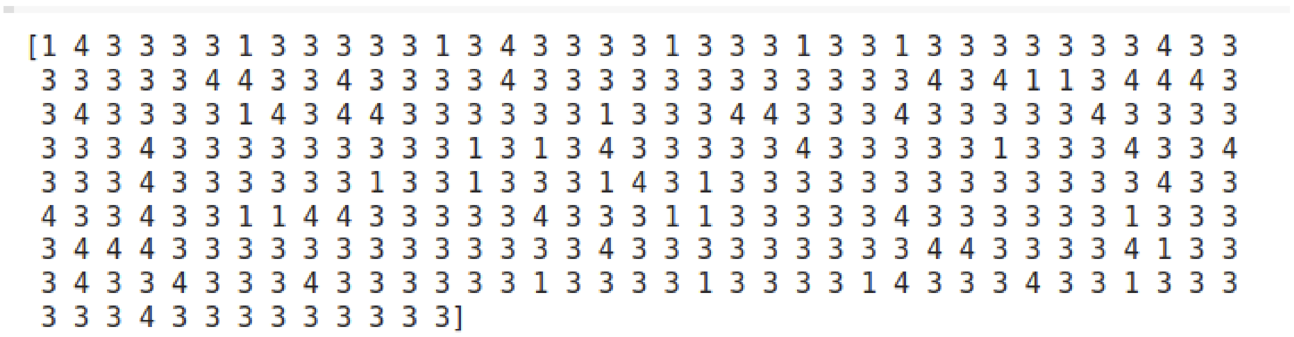
```
[2 5 4 4 4 4 2 4 4 2 4 4 2 4 5 4 4 4 3 4 4 1 4 2 4 5 2 4 4 4 3 4 4 4 5 2 4
 4 4 3 3 5 4 5 4 4 4 3 4 4 4 5 5 4 5 4 4 4 4 4 4 4 4 4 4 4 5 2 2 4 4 5 5 4
 4 4 2 4 4 4 2 4 5 4 5 4 4 4 4 4 4 2 4 4 2 4 4 5 4 4 4 2 4 4 4 4 4 4 5 5 4
 4 4 3 5 4 2 4 4 4 4 4 5 4 2 4 2 4 4 4 4 4 3 4 4 4 4 4 4 5 2 4 4 4 4 4 4 5
 4 3 3 4 4 4 4 4 3 4 2 4 4 3 4 4 4 2 4 4 2 4 4 4 4 3 3 4 5 4 4 4 3 4 4 4 4
 5 4 4 5 4 4 2 2 4 5 4 4 4 4 4 4 5 4 4 2 2 3 4 3 4 4 5 4 4 4 4 4 4 2 4 3 4
 4 4 4 5 4 3 4 4 4 4 4 4 4 4 4 4 4 5 4 3 4 4 4 4 4 4 4 4 4 4 4 4 4 5 2 4 4
 4 4 4 4 4 4 4 4 4 3 4 4 4 4 4 2 4 2 4 3 2 3 4 4 4 2 4 4 4 4 4 4 4 2 3 4 4
 4 4 4 5 4 4 4 3 4 4 4 4 4]
0.5242718446601942
```

The Mean square error, Root mean square error and mean absolute error are as follows:



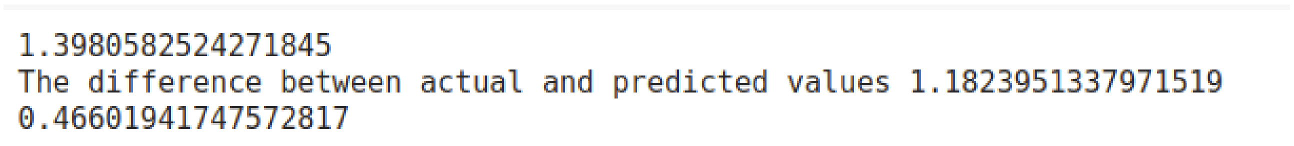
```
0.627831715210356
The difference between actual and predicted values 0.7923583250085506
0.46601941747572817
```

6. Stochastic Gradient Descent (SGD) classifier:



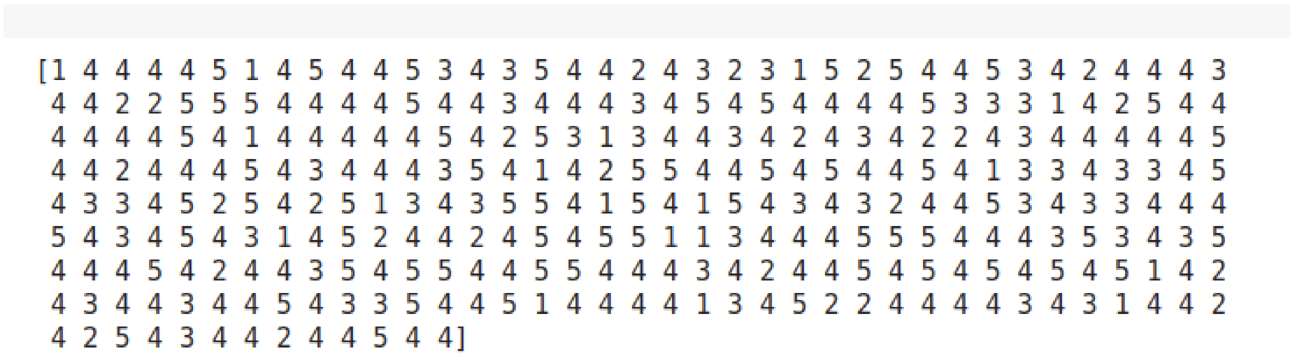
```
[1 4 3 3 3 3 1 3 3 3 3 3 1 3 4 3 3 3 3 1 3 3 3 1 3 3 1 3 3 3 3 3 3 3 4 3 3
 3 3 3 3 3 4 4 3 3 4 3 3 3 3 4 3 3 3 3 3 3 3 3 3 3 3 3 4 3 4 1 1 3 4 4 4 3
 3 4 3 3 3 3 1 4 3 4 4 3 3 3 3 3 3 1 3 3 3 4 4 3 3 3 4 3 3 3 3 3 4 3 3 3 3
 3 3 3 4 3 3 3 3 3 3 3 3 3 1 3 1 3 4 3 3 3 3 3 4 3 3 3 3 3 1 3 3 3 4 3 3 4
 3 3 3 4 3 3 3 3 3 3 1 3 3 1 3 3 3 1 4 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 4 3 3
 4 3 3 4 3 3 1 1 4 4 3 3 3 3 3 4 3 3 3 1 1 3 3 3 3 3 4 3 3 3 3 3 3 1 3 3 3
 3 4 4 4 3 3 3 3 3 3 3 3 3 3 3 3 3 4 3 3 3 3 3 3 3 3 3 4 4 3 3 3 3 4 1 3 3
 3 4 3 3 4 3 3 3 4 3 3 3 3 3 3 1 3 3 3 3 1 3 3 3 3 1 4 3 3 3 4 3 3 1 3 3 3
 3 3 3 4 3 3 3 3 3 3 3 3 3]
```

The Mean square error, Root mean square error and mean absolute error are as follows:



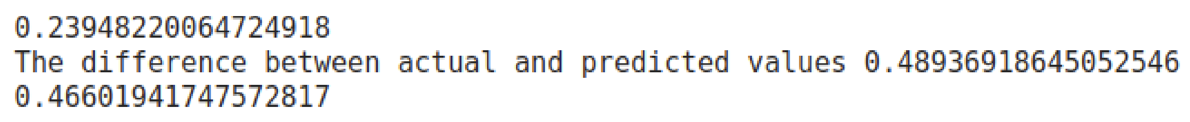
```
1.3980582524271845
The difference between actual and predicted values 1.1823951337971519
0.46601941747572817
```

7. Decision Tree (DT) classifier:



```
[1 4 4 4 4 5 1 4 5 4 4 5 3 4 3 5 4 4 2 4 3 2 3 1 5 2 5 4 4 5 3 4 2 4 4 4 3
 4 4 2 2 5 5 5 4 4 4 4 5 4 4 3 4 4 4 3 4 5 4 5 4 4 4 4 5 3 3 3 1 4 2 5 4 4
 4 4 4 4 5 4 1 4 4 4 4 4 5 4 2 5 3 1 3 4 4 3 4 2 4 3 4 2 2 4 3 4 4 4 4 4 5
 4 4 2 4 4 4 5 4 3 4 4 4 3 5 4 1 4 2 5 5 4 4 5 4 5 4 4 5 4 1 3 3 4 3 3 4 5
 4 3 3 4 5 2 5 4 2 5 1 3 4 3 5 5 4 1 5 4 1 5 4 3 4 3 2 4 4 5 3 4 3 3 4 4 4
 5 4 3 4 5 4 3 1 4 5 2 4 4 2 4 5 4 5 5 1 1 3 4 4 4 5 5 5 4 4 4 3 5 3 4 3 5
 4 4 4 5 4 2 4 4 3 5 4 5 5 4 4 5 5 4 4 4 3 4 2 4 4 5 4 5 4 5 4 5 4 5 1 4 2
 4 3 4 4 3 4 4 5 4 3 3 5 4 4 5 1 4 4 4 4 1 3 4 5 2 2 4 4 4 4 3 4 3 1 4 4 2
 4 2 5 4 3 4 4 2 4 4 5 4 4]
```

The Mean square error, Root mean square error and mean absolute error are as follows:



```
0.23948220064724918
The difference between actual and predicted values 0.48936918645052546
0.46601941747572817
```

Colab Notebook Link:

https://colab.research.google.com/drive/1sud5XTFB_0j5ZRe3ruCZKNsw9u10_g1o?usp=sharing

Use case 3:QoS optimization

Motivation:

QoS (Quality of Service) optimization in ORAN (Open Radio Access Network) is a crucial aspect of ensuring that the network provides the best possible service to its users. ORAN is designed to be more open and flexible, enabling multi-vendor interoperability and innovation.

Optimizing QoS in ORAN requires a combination of advanced network management techniques, intelligent algorithms, and cooperation among stakeholders. By focusing on QoS, ORAN operators can deliver better service quality and enhance the overall user experience.

Problem statement:

Ensuring a high level of Quality of Service (QoS) is crucial to meet the diverse needs of users and applications while maximizing network efficiency.

The challenge lies in optimizing QoS in an ORAN ecosystem, which involves managing several key aspects, such as resource allocation and latency reduction.

The successful optimization of QoS in ORAN is critical to delivering enhanced user experiences, meeting service level agreements (SLAs), and supporting emerging applications and services.

Reference Paper Used:

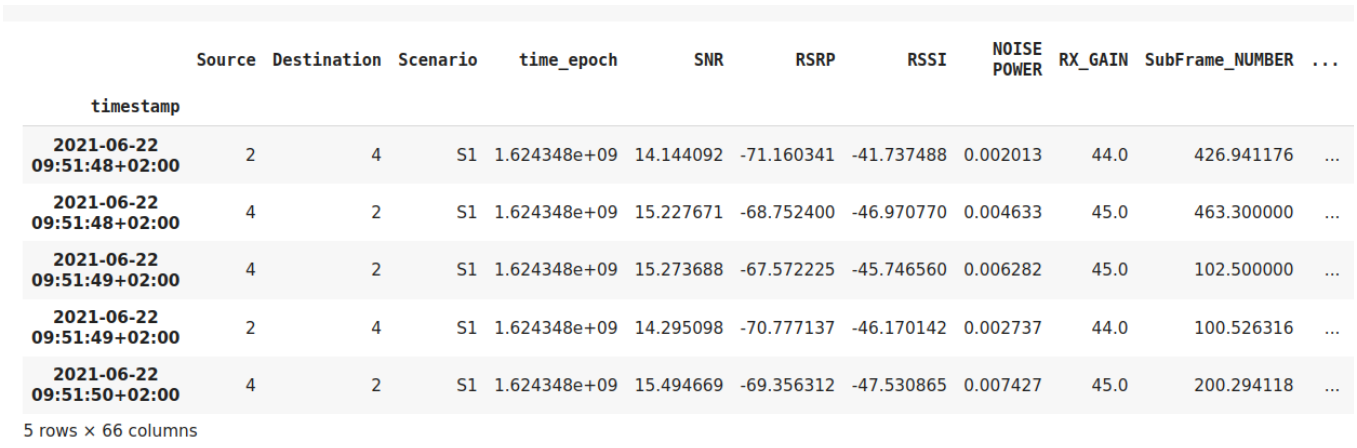
https://www.mdpi.com/1424-8220/23/5/2660

Rizzi, M.; Ferrari, P.; Flammini, A.; Sisinni, E. Evaluation of the IoT LoRaWAN solution for distributed measurement applications. *IEEE Trans. Instrum. Meas.* **2017**, *66*, 3340–3349.

Dataset Source and Particularities:

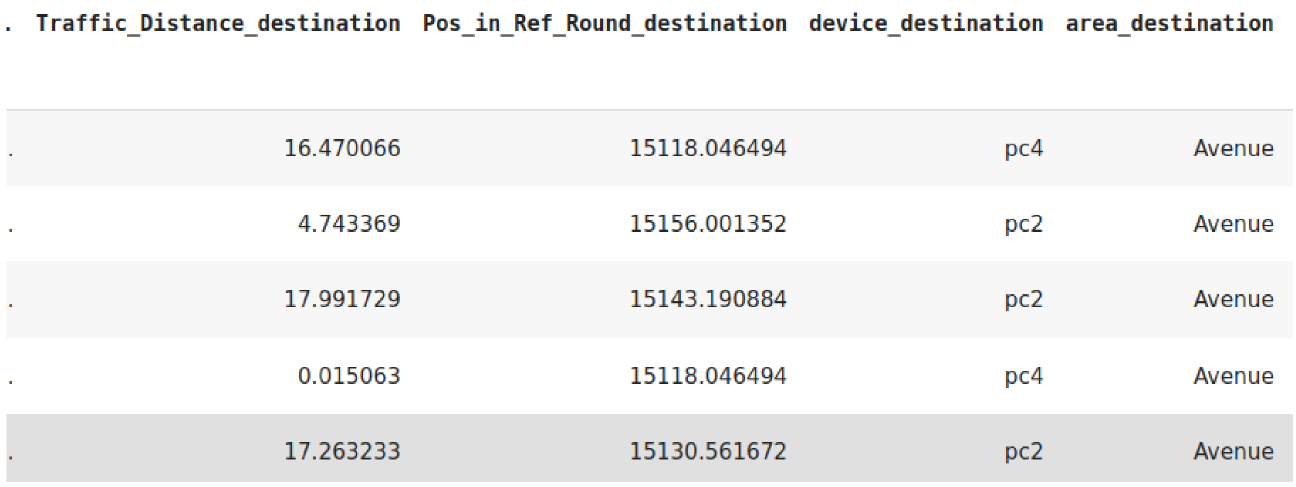
https://ieee-dataport.org/open-access/berlin-v2x

dataframe.head():

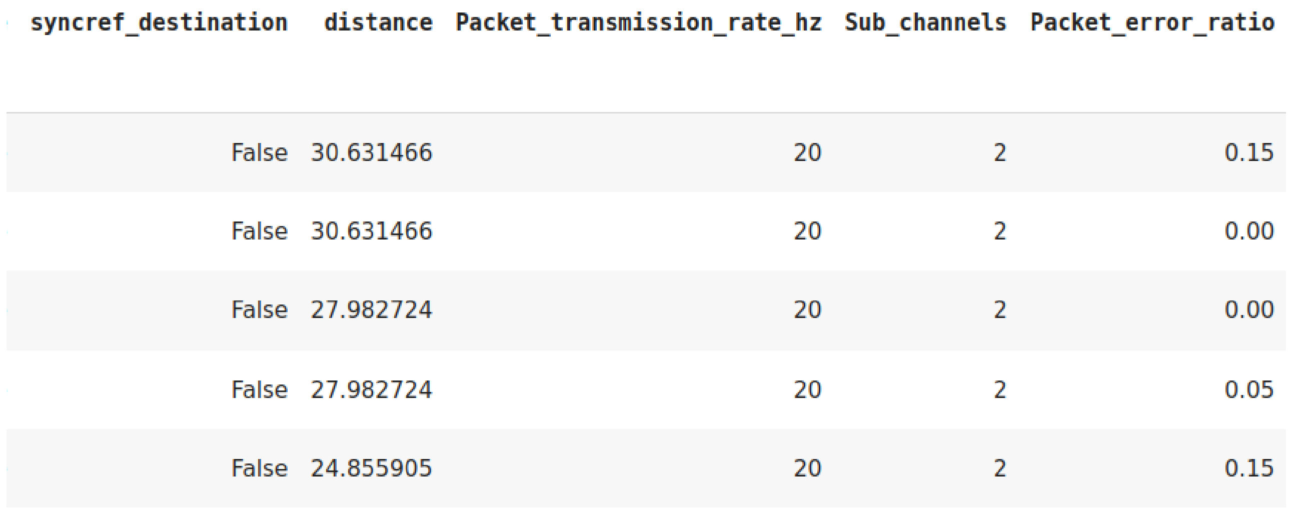


	Source	Destination	Scenario	time_epoch	SNR	RSRP	RSSI	NOISE POWER	RX_GAIN	SubFrame_NUMBER	...
timestamp											
2021-06-22 09:51:48+02:00	2	4	S1	1.624348e+09	14.144092	-71.160341	-41.737488	0.002013	44.0	426.941176	...
2021-06-22 09:51:48+02:00	4	2	S1	1.624348e+09	15.227671	-68.752400	-46.970770	0.004633	45.0	463.300000	...
2021-06-22 09:51:49+02:00	4	2	S1	1.624348e+09	15.273688	-67.572225	-45.746560	0.006282	45.0	102.500000	...
2021-06-22 09:51:49+02:00	2	4	S1	1.624348e+09	14.295098	-70.777137	-46.170142	0.002737	44.0	100.526316	...
2021-06-22 09:51:50+02:00	4	2	S1	1.624348e+09	15.494669	-69.356312	-47.530865	0.007427	45.0	200.294118	...

5 rows × 66 columns



.	Traffic_Distance_destination	Pos_in_Ref_Round_destination	device_destination	area_destination
.	16.470066	15118.046494	pc4	Avenue
.	4.743369	15156.001352	pc2	Avenue
.	17.991729	15143.190884	pc2	Avenue
.	0.015063	15118.046494	pc4	Avenue
.	17.263233	15130.561672	pc2	Avenue



syncref_destination	distance	Packet_transmission_rate_hz	Sub_channels	Packet_error_ratio
False	30.631466	20	2	0.15
False	30.631466	20	2	0.00
False	27.982724	20	2	0.00
False	27.982724	20	2	0.05
False	24.855905	20	2	0.15

dataframe.columns:

```
print(df.columns.values)
```

```
['Source' 'Destination' 'Scenario' 'time_epoch' 'SNR' 'RSRP' 'RSSI'
 'NOISE_POWER' 'RX_GAIN' 'SubFrame_NUMBER' 'SubFrame_LENGHT' 'Rx_power'
 'MCS' 'Received_Packets' 'ts_gps_source' 'Latitude_source'
 'Longitude_source' 'Altitude_source' 'speed_kmh_source' 'COG_source'
 'precipIntensity_source' 'precipProbability_source' 'temperature_source'
 'apparentTemperature_source' 'dewPoint_source' 'humidity_source'
 'pressure_source' 'windSpeed_source' 'cloudCover_source' 'uvIndex_source'
 'visibility_source' 'Traffic_Jam_Factor_source'
 'Traffic_Street_Name_source' 'Traffic_Distance_source'
 'Pos_in_Ref_Round_source' 'device_source' 'area_source'
 'ts_gps_destination' 'Latitude_destination' 'Longitude_destination'
 'Altitude_destination' 'speed_kmh_destination' 'COG_destination'
 'precipIntensity_destination' 'precipProbability_destination'
 'temperature_destination' 'apparentTemperature_destination'
 'dewPoint_destination' 'humidity_destination' 'pressure_destination'
 'windSpeed_destination' 'cloudCover_destination' 'uvIndex_destination'
 'visibility_destination' 'Traffic_Jam_Factor_destination'
 'Traffic_Street_Name_destination' 'Traffic_Distance_destination'
 'Pos_in_Ref_Round_destination' 'device_destination' 'area_destination'
 'syncref_source' 'syncref_destination' 'distance'
 'Packet_transmission_rate_hz' 'Sub_channels' 'Packet_error_ratio']
```

Dataframe is then filtered to extract only the required columns:

qos.head():

Source	Destination	SNR	RSRP	RSSI	NOISEPOWER	RX_GAIN	Rx_power	MCS	ReceivedPackets
2	4	14.144092	-71.160341	-41.737488	0.002013	44.0	0.053156	8	17
4	2	15.227671	-68.752400	-46.970770	0.004633	45.0	0.098266	8	20
4	2	15.273688	-67.572225	-45.746560	0.006282	45.0	0.123603	8	20
2	4	14.295098	-70.777137	-46.170142	0.002737	44.0	0.066103	8	19
4	2	15.494669	-69.356312	-47.530865	0.007427	45.0	0.095582	8	17

speed_kmh_destination	distance	Packet_transmission_rate_hz	Sub_channels	Packet_error_ratio
22.9648	30.631466	20	2	0.15
31.8544	30.631466	20	2	0.00
33.1508	27.982724	20	2	0.00
22.4092	27.982724	20	2	0.05
32.5952	24.855905	20	2	0.15

qos.columns:

```
qos.columns
```

```
Index(['Source', 'Destination', 'SNR', 'RSRP', 'RSSI', 'NOISE POWER',
       'RX_GAIN', 'Rx_power', 'MCS', 'Received Packets',
       'speed_kmh_destination', 'distance', 'Packet_transmission_rate_hz',
       'Sub_channels', 'Packet_error_ratio'],
      dtype='object')
```

ML Algorithms Used:

With the goal of improving LoRaWAN QoS performance, many researchers developed resource allocation schemes using strong AI algorithms:

1. Support Vector regressor

2. Decision Tree

3. Linear regressor

4. KNN algorithm

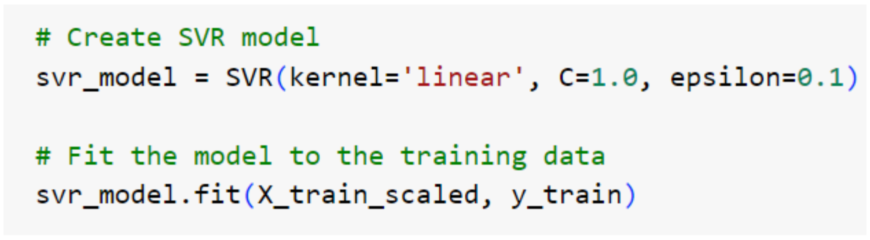
Through simulation with different traffic loads, it presented a significant enhancement in terms of data extraction rate.

Their simulation results showed an improvement in the packet delivery ratio. This algorithm was evaluated to achieve an enhancement in throughput and packet delivery rate.

Output:

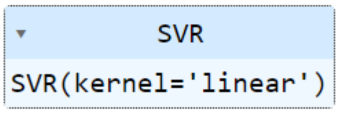
Error ratios - Mean Square Error [Optimal Algorithm → Decision Tree]

1. Support Vector regressor



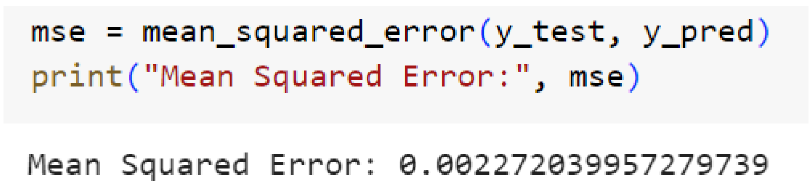
```
# Create SVR model
svr_model = SVR(kernel='linear', C=1.0, epsilon=0.1)

# Fit the model to the training data
svr_model.fit(X_train_scaled, y_train)
```



```
▾          SVR
SVR(kernel='linear')
```

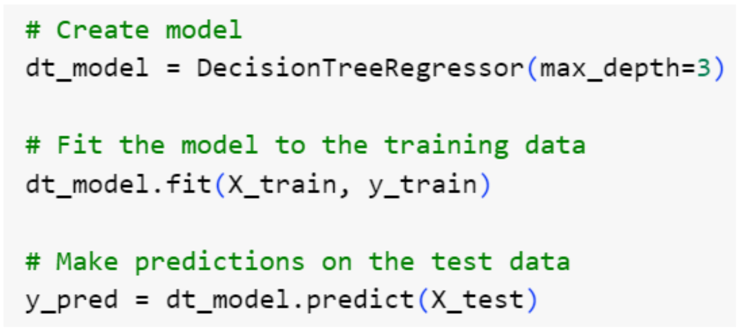
The Mean square error is as follows:



```
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 0.002272039957279739
```

2. Decision Tree

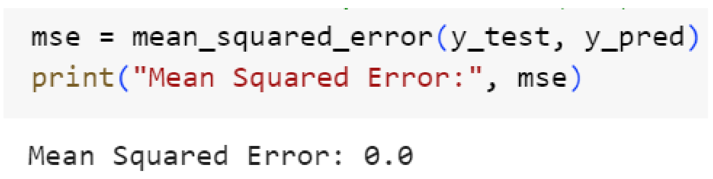


```
# Create model
dt_model = DecisionTreeRegressor(max_depth=3)

# Fit the model to the training data
dt_model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = dt_model.predict(X_test)
```

The Mean square error is as follows:



```
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 0.0
```

3. Linear regressor

```
from sklearn.linear_model import LinearRegression
# Create a Linear Regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

The Mean square error is as follows:

```
# Calculate Mean Squared Error
mse = mean_squared_error(y_test, predictions)

print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
print("Mean Squared Error:", mse)
```

```
Coefficients: [ 2.97985349e-15  1.22124533e-15  6.31439345e-16 -3.74179854e-15
  4.23554421e-15  1.06757898e-14 -2.66106581e-15  4.25007252e-16
  1.50000000e+00 -5.44703171e-16 -2.70616862e-16 -2.21177243e-16
  3.00000000e+00 -1.07550145e-15]
Intercept: 2.000000000000057
Mean Squared Error: 3.3424172360200654e-28
```

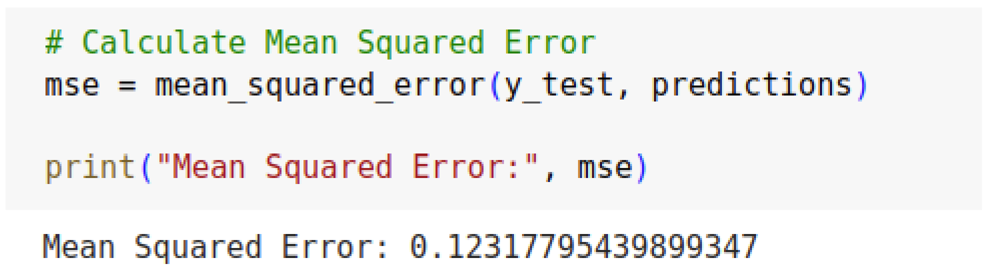
4. KNN algorithm

```
k = 5  # Number of neighbors
knnmodel = KNeighborsRegressor(n_neighbors=k)

# Fit the model to the training data
knnmodel.fit(X_train, y_train)
```

```
▾ KNeighborsRegressor
KNeighborsRegressor()
```

The Mean square error is as follows:



```
# Calculate Mean Squared Error
mse = mean_squared_error(y_test, predictions)

print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 0.12317795439899347
```

Colab Notebook Link:

https://colab.research.google.com/drive/1yXr1SRE6u8yjGpvNjZV3vh9gwSQubJgW?usp=sharing

Use case 4: Local Indoor Positioning in RAN

Motivation:

For local indoor scenarios, the positioning function inside RAN is envisioned to be a promising solution. It not only reduces the latency of positioning but also can reuse the edge cloud infrastructure in RAN. In the context of O-RAN architecture, the positioning function can be deployed as a positioning xApp in the Near-RT RIC. The positioning xApp computes the UE location and optional velocity based on the positioning measurement obtained via the E2 interface.

By leveraging multi-point positioning, field strength positioning and other positioning algorithms, positioning xApp inside Near-RT RIC can conduct a real-time position of UE.

Problem Statement:

Local Indoor Positioning has emerged as a crucial requirement in modern Radio Access Networks (RANs), enabling a wide range of location-based services and applications within indoor environments. The challenge lies in developing a robust and accurate Local Indoor Positioning system that can seamlessly integrate with existing RAN infrastructure while delivering high precision and reliability.

Reference Papers Used:

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9531633

https://www.researchgate.net/publication/350579120_Machine_Learning_Based_Indoor_Localization_using_Wi-Fi_and_Smartphone

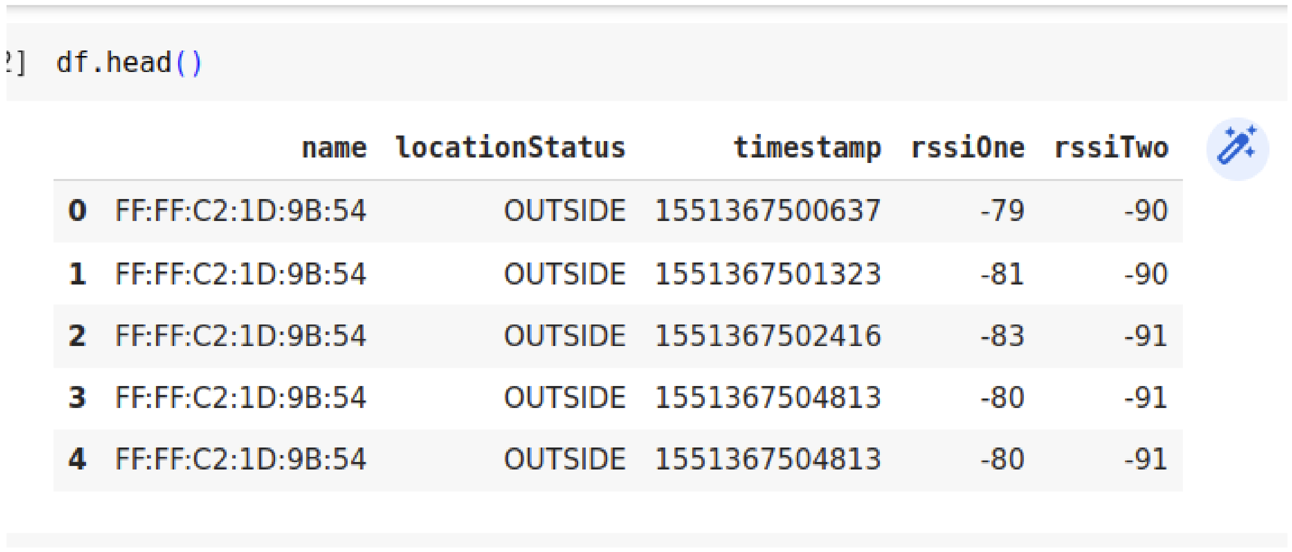
Dataset Source and Particularities:

https://archive.ics.uci.edu/dataset/586/ble+rssi+dataset+for+indoor+localization

Attribute Information:

- name - MAC address of iTAG
- locationStatus - one of three possible iTAG location: INSIDE, IN_VESTIBULE, OUTSIDE
- timestamp - timestamp in milliseconds
- rssiOne - RSSI on first smartphone
- rssiTwo - RSSI on second smartphone

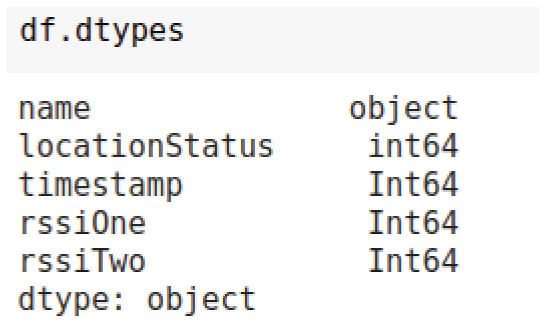
dataset.head():



```
2] df.head()
```

	name	locationStatus	timestamp	rssiOne	rssiTwo
0	FF:FF:C2:1D:9B:54	OUTSIDE	1551367500637	-79	-90
1	FF:FF:C2:1D:9B:54	OUTSIDE	1551367501323	-81	-90
2	FF:FF:C2:1D:9B:54	OUTSIDE	1551367502416	-83	-91
3	FF:FF:C2:1D:9B:54	OUTSIDE	1551367504813	-80	-91
4	FF:FF:C2:1D:9B:54	OUTSIDE	1551367504813	-80	-91

dataset.columns:



```
df.dtypes
```

```
name              object
locationStatus     int64
timestamp          Int64
rssiOne            Int64
rssiTwo            Int64
dtype: object
```

ML Algorithms Used:

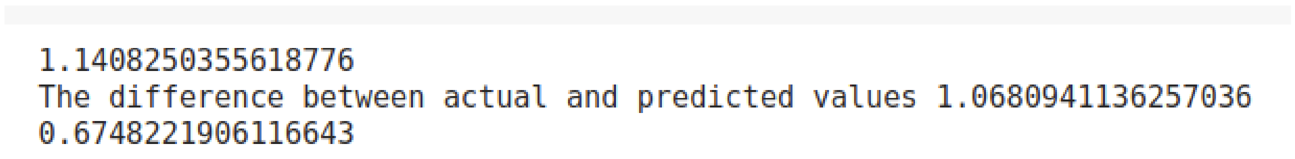
1. Random forest classifier:
2. KNN classifier
3. Gradient Boost Classifier
4. Logistic Regression Classifier
5. Support Vector Machine Classifier

Output:

Error ratios - Mean Square Error [Optimal Algorithm → Gradient Boost classifier]

1. Random forest classifier:

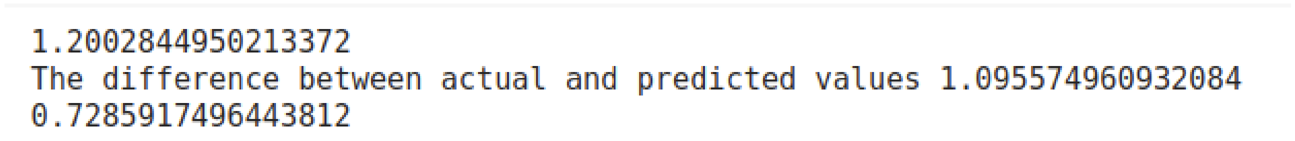
   The Mean Square Error, Root Mean Square Error and Mean Absolute error are as follows:



```
1.1408250355618776
The difference between actual and predicted values 1.0680941136257036
0.6748221906116643
```

2. KNN classifier

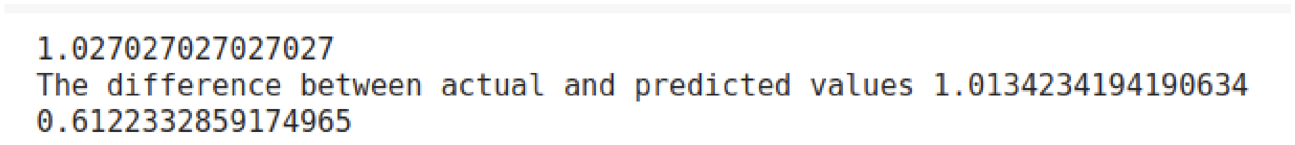
   The Mean Square Error, Root Mean Square Error and the Mean Absolute error are as follows:



```
1.2002844950213372
The difference between actual and predicted values 1.095574960932084
0.7285917496443812
```

3. Gradient Boost classifier

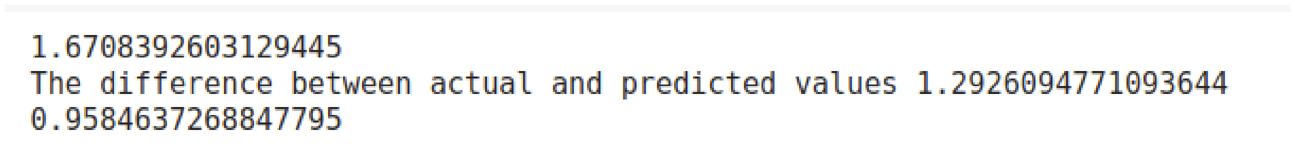
   The Mean Square Error, Root Mean Square Error and the Mean Absolute error are as follows:



```
1.027027027027027
The difference between actual and predicted values 1.0134234194190634
0.6122332859174965
```

4. Logistic Regression Classifier

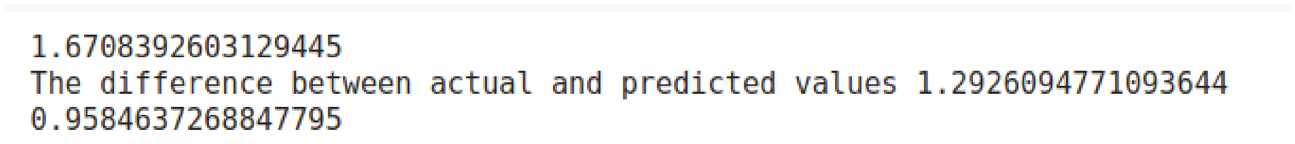
   The Mean Square Error, Root Mean Square Error and the Mean Absolute error are as follows:



```
1.6708392603129445
The difference between actual and predicted values 1.292609477109364
0.9584637268847795
```

5. Support Vector Machine Classifier

   The Mean Square Error, Root Mean Square Error and the Mean Absolute error are as follows:



```
1.6708392603129445
The difference between actual and predicted values 1.292609477109364
0.9584637268847795
```

Colab Notebook Link:

https://colab.research.google.com/drive/1oLj9P4SwfepSuzOrjSIsajrwm-wuRDZo?usp=sharing

Use case 5: Energy Saving

Motivation:

ES for legacy and 5G networks can be carried out using manual configuration or SON functions. 3GPP defines both centralized and distributed ES features [6], which are mainly targeting intra- or inter-RAT cell on/off switching. The motivation of the ES use case is to leverage on ORAN AI/ML services and open interfaces in order to introduce optimized ES and EE solutions involving switching off/on of different network components at different time scales. Will be considered off/on switching solutions supported by 3GPP configurations or supported by proprietary implementations that require additional standardization.

Problem Statement:

To predict the cell state as Idle or Downloading based on the mobility parameters.

It is stated in the paper that the base stations(LTE's eNodeBs) can be either in active state (handling user equipment's (UE) data or in an idle state (transmitting only downlink control signaling)

The classifier determines its next class for the next period of time. For this purpose, classification algorithms are used namely:

1. Random Forest Classifier

2. k-means clustering

Reference Paper Used:

https://upcommons.upc.edu/bitstream/handle/2117/179819/SestoGarcia_WCNC19_preprint.pdf

K. Hiltunen, "Improving the energy-efficiency of dense LTE networks by adaptive activation of cells," in Proc. of IEEE International Conference on Communications (ICC), pp. 1150-1154, 2013

Dataset Source and Particularities:

https://www.kaggle.com/datasets/aeryss/lte-dataset

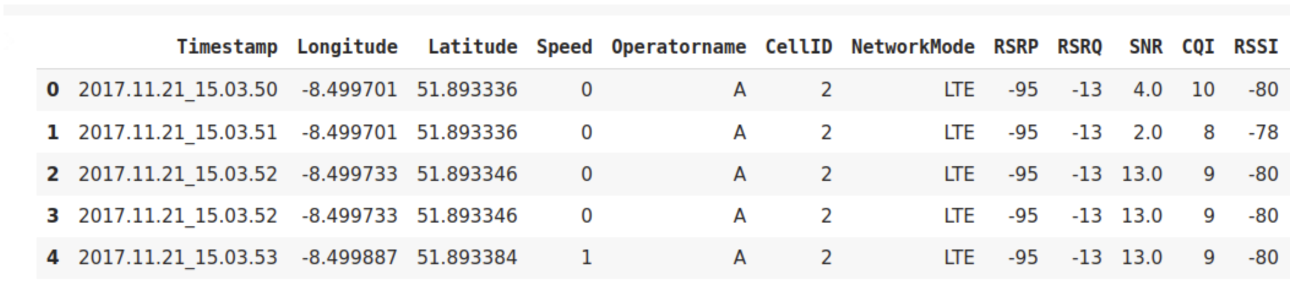
4G measurement trials (unless otherwise stated) across six different mobility patterns:

Column names & Description:

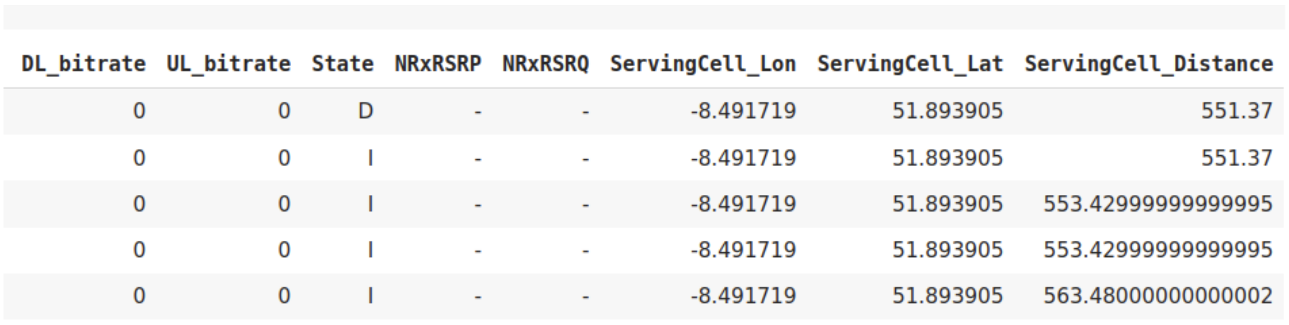
- Timestamp: timestamp of sample
- Longitude and Latitude: GPS coordinates of mobile device
- Velocity: velocity in kph of mobile device
- Operatorname: cellular operator name (anonymised)
- CellId: Serving cell for mobile device
- NetworkMode: mobile communication standard (2G/3G/4G)

- RSRQ: value for RSRQ. RSRQ Represents a ratio between RSRP and Received Signal Strength Indicator (RSSI). Signal strength (signal quality) is measured across all resource elements (RE), including interference from all sources (dB).
- RSRP: value for RSRP. RSRP Represents an average power over cell-specific reference symbols carried inside distinct RE. RSRP is used for measuring cell signal strength/coverage and therefore cell selection (dBm).
- RSSI: value for RSSI. RSSI represents a received power (wideband) including a serving cell and interference and noise from other sources. RSRQ, RSRP and RSSI are used for measuring cell strength/coverage and therefore cell selection (handover) (dBm).
- SNR: value for signal-to-noise ratio (dB).
- CQI: value for CQI of a mobile device. CQI is a feedback provided by UE to eNodeB. It indicates data rate that could be transmitted over a channel (highest MCS with a BLER probability less than 10%), as the function of SINR and UE's receiver characteristics. Based on UE's prediction of the channel, eNodeB selects an appropriate modulation scheme and coding rate.
- DL_bitrate: download rate measured at the device (application layer) (kbit/s)
- UL_bitrate: uplink rate measured at the device (application layer) (kbit/s)
- State: state of the download process. It has two values, either I (idle, not downloading) or D (downloading)
- NRxRSRQ & NRxRSRP: RSRQ and RSRP values for the neighbouring cell.
- Cell_Longitude & Cell_Latitude: GPS coordinates of serving eNodeB. We use OpenCellid4, the largest community open database providing GPS coordinates of cell towers.
- Distance: distance between the serving cell and mobile device in metres.

dataset.head():

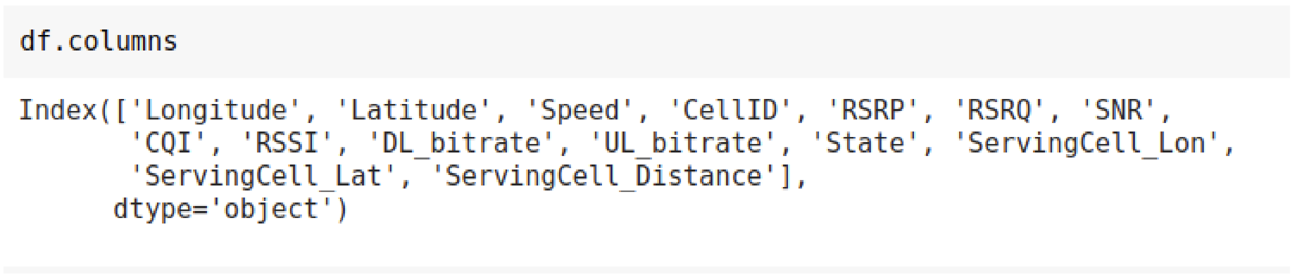


	Timestamp	Longitude	Latitude	Speed	Operatorname	CellID	NetworkMode	RSRP	RSRQ	SNR	CQI	RSSI
0	2017.11.21_15.03.50	-8.499701	51.893336	0	A	2	LTE	-95	-13	4.0	10	-80
1	2017.11.21_15.03.51	-8.499701	51.893336	0	A	2	LTE	-95	-13	2.0	8	-78
2	2017.11.21_15.03.52	-8.499733	51.893346	0	A	2	LTE	-95	-13	13.0	9	-80
3	2017.11.21_15.03.52	-8.499733	51.893346	0	A	2	LTE	-95	-13	13.0	9	-80
4	2017.11.21_15.03.53	-8.499887	51.893384	1	A	2	LTE	-95	-13	13.0	9	-80



DL_bitrate	UL_bitrate	State	NRxRSRP	NRxRSRQ	ServingCell_Lon	ServingCell_Lat	ServingCell_Distance
0	0	D	-	-	-8.491719	51.893905	551.37
0	0	I	-	-	-8.491719	51.893905	551.37
0	0	I	-	-	-8.491719	51.893905	553.42999999999995
0	0	I	-	-	-8.491719	51.893905	553.42999999999995
0	0	I	-	-	-8.491719	51.893905	563.48000000000002

dataset.columns:



```
df.columns
```

```
Index(['Longitude', 'Latitude', 'Speed', 'CellID', 'RSRP', 'RSRQ', 'SNR',
       'CQI', 'RSSI', 'DL_bitrate', 'UL_bitrate', 'State', 'ServingCell_Lon',
       'ServingCell_Lat', 'ServingCell_Distance'],
      dtype='object')
```

ML Algorithms Used:

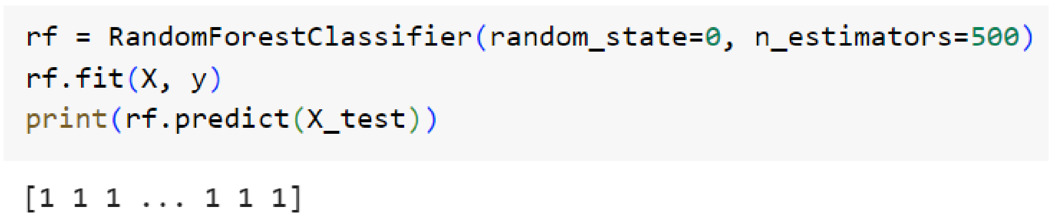
1. Random Forest Classifier

2. K-means Clustering

Output:

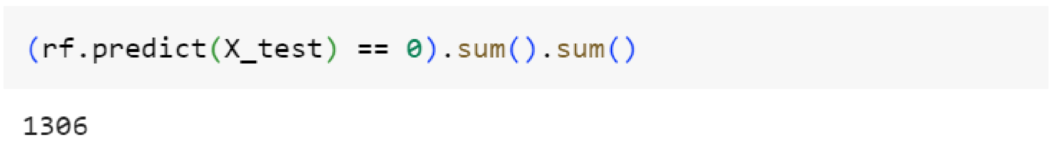
Error ratios - Mean Square Error [Optimal Algorithm → Random Forest Classifier]

1. Random Forest Classifier



```
rf = RandomForestClassifier(random_state=0, n_estimators=500)
rf.fit(X, y)
print(rf.predict(X_test))
```

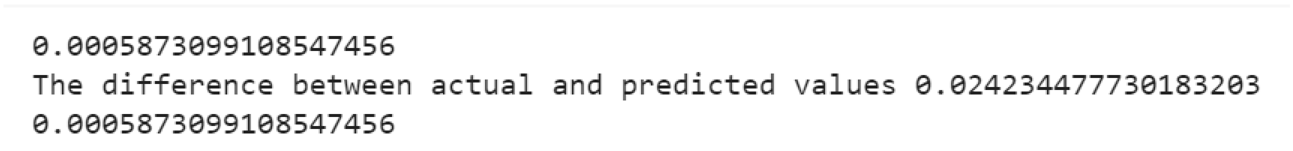
```
[1 1 1 ... 1 1 1]
```



```
(rf.predict(X_test) == 0).sum().sum()
```

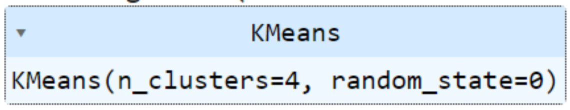
```
1306
```

The Mean Square Error, Root Mean Square Error and the Mean Absolute error are as follows:

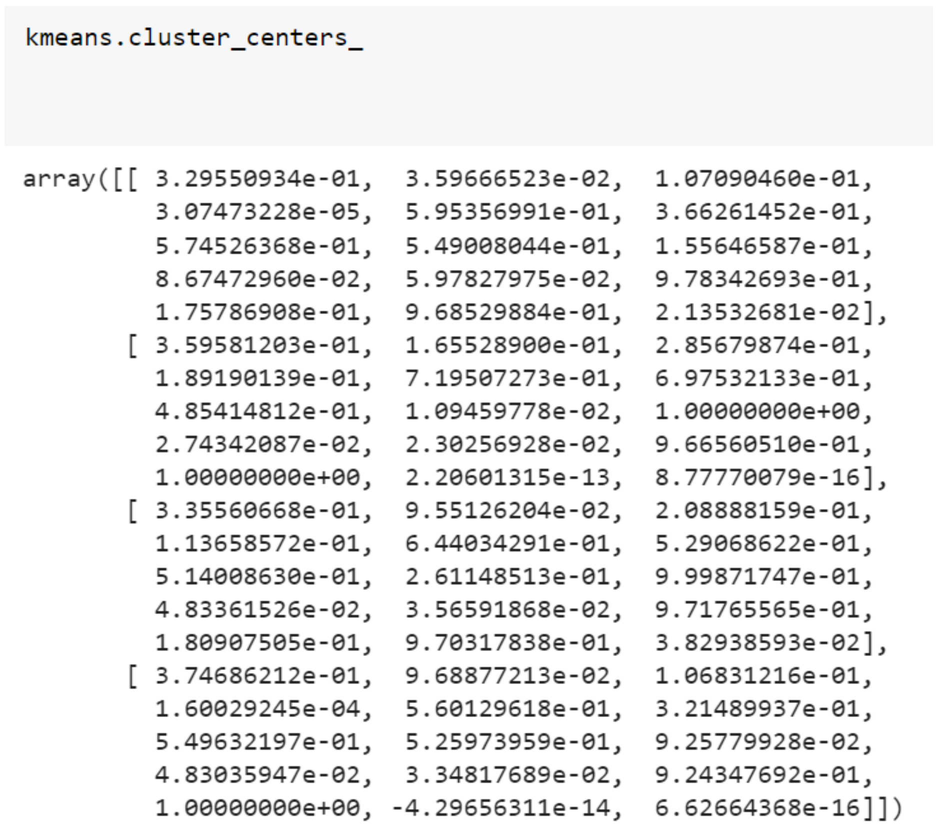


```
0.0005873099108547456
The difference between actual and predicted values 0.024234477730183203
0.0005873099108547456
```

2. K-means Clustering



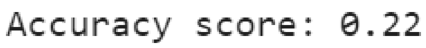
```
KMeans
KMeans(n_clusters=4, random_state=0)
```



```
kmeans.cluster_centers_
```

```
array([[ 3.29550934e-01,  3.59666523e-02,  1.07090460e-01,
         3.07473228e-05,  5.95356991e-01,  3.66261452e-01,
         5.74526368e-01,  5.49008044e-01,  1.55646587e-01,
         8.67472960e-02,  5.97827975e-02,  9.78342693e-01,
         1.75786908e-01,  9.68529884e-01,  2.13532681e-02],
       [ 3.59581203e-01,  1.65528900e-01,  2.85679874e-01,
         1.89190139e-01,  7.19507273e-01,  6.97532133e-01,
         4.85414812e-01,  1.09459778e-02,  1.00000000e+00,
         2.74342087e-02,  2.30256928e-02,  9.66560510e-01,
         1.00000000e+00,  2.20601315e-13,  8.77770079e-16],
       [ 3.35560668e-01,  9.55126204e-02,  2.08888159e-01,
         1.13658572e-01,  6.44034291e-01,  5.29068622e-01,
         5.14008630e-01,  2.61148513e-01,  9.99871747e-01,
         4.83361526e-02,  3.56591868e-02,  9.71765565e-01,
         1.80907505e-01,  9.70317838e-01,  3.82938593e-02],
       [ 3.74686212e-01,  9.68877213e-02,  1.06831216e-01,
         1.60029245e-04,  5.60129618e-01,  3.21489937e-01,
         5.49632197e-01,  5.25973959e-01,  9.25779928e-02,
         4.83035947e-02,  3.34817689e-02,  9.24347692e-01,
         1.00000000e+00, -4.29656311e-14,  6.62664368e-16]])
```

The Accuracy Score is as follows:



```
Accuracy score: 0.22
```

Colab Notebook Link:

https://colab.research.google.com/drive/1s01Rl4FQdIAHMBtp4hJGFDeFuerIglkQ?usp=sharing

Future Scope:

The future scope of handover prediction in Open Radio Access Network (ORAN) holds significant promise and potential for improving network performance and enhancing user experience. Handover prediction is a critical aspect of seamless mobility management, where predicting and executing handovers in advance can reduce latency, maintain connection continuity, and optimize resource utilization. As technology continues to advance, handover prediction will play a pivotal role in shaping the next-generation wireless networks and meeting the ever-increasing demands of mobile users and applications.

The future scope of Quality of Experience (QoE) optimization in Open Radio Access Network (ORAN) is promising, driven by advancements in technology and the evolution of wireless communication networks. As ORAN continues to gain traction and deployment, the focus on QoE optimization will become even more critical to cater to the diverse needs of users and applications.

The Open Radio Access Network (ORAN)'s (future) Quality of Service (QoS) improvement has the potential to completely alter the wireless communication environment. Optimizing QoS will be more and more important as ORAN grows and develops to meet the various demands of users, applications, and developing technologies.

The potential for indoor localization via the Open Radio Access Network (ORAN) to change a variety of industries and user experiences is exciting. Indoor localization becomes a crucial enabler for a variety of location-based services and applications as ORAN develops and allows multi-vendor deployments with open interfaces.

As the telecommunications sector strives to increase sustainability and lower its carbon footprint, the future potential for energy-saving in Open Radio Access Network (ORAN) is of the utmost importance. Innovative energy-saving techniques can be used across the network ecosystem thanks to ORAN's adaptable and open architecture.

Conclusion

By leveraging the capabilities of the Python programming language and various machine learning libraries, this project successfully developed solutions for O-RAN use cases in 5G NR networks. The chosen technologies played a critical role in enabling efficient data preprocessing, model development, evaluation, and visualization, ultimately leading to enhanced QoE, QoS prediction, throughput prediction, energy savings, and indoor localization in 5G NR networks.

Developing solutions for use cases of ORAN aligns with the industry's vision for more open, flexible, and innovative wireless networks. It enables the delivery of enhanced services, improved user experiences, and sustainable growth in the telecommunications sector.