SWINBURNE UNIVERSITY OF TECHNOLOGY

School of Science, Computing, and Engineering Technologies

Artificial Intelligence for Engineering COS40007

Project Outcome and Report

Studio 1-6 - Group 1 - Theme 5

STUDENT NAME & ID

BAO Nguyen – 103842645

Dang Khoa Le - 103844421

Nguyen Hung Nguyen - 104058197

Trung Kien Nguyen - 104053642

Saw Ko Ko Oo - 104150310

31/10/2024

Table of Contents

1. Introduction

1.1. Background and Motivation

The motivation behind this project is driven by the rapid evolution of 5G technology and its potential impact across multiple sectors, including telecommunications, manufacturing, aviation, and public infrastructure. Leveraging Ericsson Vietnam's dataset on 5G network performance, this project focuses on analyzing, categorizing, and predicting 5G network quality metrics across different geographical zones within southeast Vietnam. With the increasing demand for reliable and high-speed networks, this project aims to empower network engineers, telecom professionals, and infrastructure planners to gain actionable insights for improving network performance and user experience.

The model is primarily designed for:

- **Network Engineers and Analysts** who monitor, troubleshoot, and optimize network quality in real time.
- **Telecommunication Service Providers** who need to balance load, plan infrastructure strategically, and enhance service delivery.
- **Corporate and Government Decision-Makers** interested in proactive infrastructure maintenance, resource allocation, and strategic expansion of 5G networks.

Through this model, users will gain capabilities to monitor KPIs effectively, balance network load, perform predictive maintenance, and make data-driven decisions for 5G infrastructure planning and enhancement.

1.2. Project Objectives

The project’s objectives are to:

- **Assess Key Performance Indicators (KPIs)**: Focus on critical 5G performance metrics to provide actionable insights.
- **Develop Predictive Capabilities**: Predict values for KPIs such as traffic volume, latency, and error rates, enabling network planning and proactive maintenance.

- **Empower Decision-Makers**: Facilitate data-driven insights that lead to better resource allocation, improved user experience, and enhanced network resilience.

In engineering, especially telecommunications, this AI model will provide the technical workforce with rapid, precise analysis of network data, helping them maintain and optimize high-quality 5G services.

1.3. Summary of Outcomes

This project resulted in the development of two predictive models and a web UI [3][4][5][6]:

- **Zone Prediction Model** (Random Forest)
  - **Sector-Level Prediction Accuracy**: 80%
  - **Site-Level Prediction Accuracy**: 78%
  - This model accurately predicts the geographical zone (sector or site) based on input features.
- **Performance Prediction Model** (Artificial Neural Network)
  - **Model Loss**: 0.002
  - This model provides precise performance predictions with minimal error, indicating strong model accuracy and reliability.
- **Web-based UI** with models integrated that user can input feature columns and get predicted Performance and Zone

2. Dataset

2.1. Data Source

The **LTE_KPI.csv** dataset provides Ericsson's 5G network performance data across 10 test sites located in key regions: Ho Chi Minh City, its outskirts, Binh Duong, and Bien Hoa. These sites are divided into four main groups, targeting performance in diverse environments, including urban, industrial, and aviation sectors in Vietnam.

Dataset Overview:

- **Geographical Scope**: Covers multiple zones of southeast Vietnam.
- **Features**: Includes 74 performance-related features, capturing metrics like user throughput, traffic volume, setup success rate, and cell availability.
- **Key Feature Categories**:
  - **Traffic Metrics**: Measures uplink/downlink traffic and throughput.
  - **User Metrics**: Tracks the average and maximum number of users.
  - **Availability Metrics**: Monitors cell availability and downtime.

- **Error and Quality Metrics**: Includes drop call rates and modulation quality.

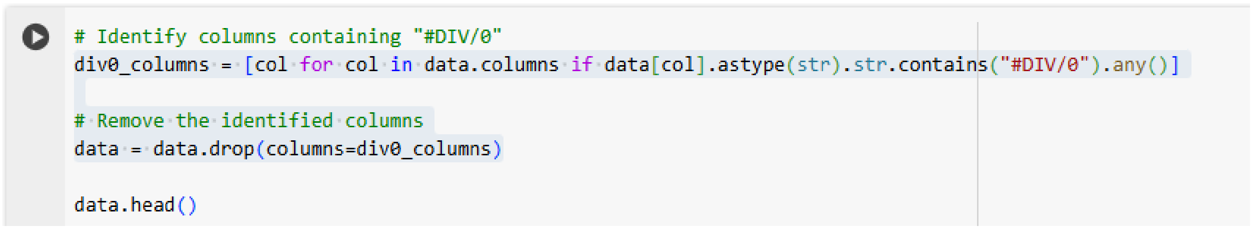
This dataset enables the creation of composite metrics for enhanced analysis, like **Traffic_Volume_and_Payload_Metrics**, which provides a comprehensive view of traffic and payload performance.

Acknowledgment: This dataset is granted from Ericsson Vietnam to be used explicitly and individually for this group and this project and for education purposes in general. This is an industrial dataset and has not been published publicly.

[1] provides the context of all 74 features in the dataset.

2.2. Data Processing

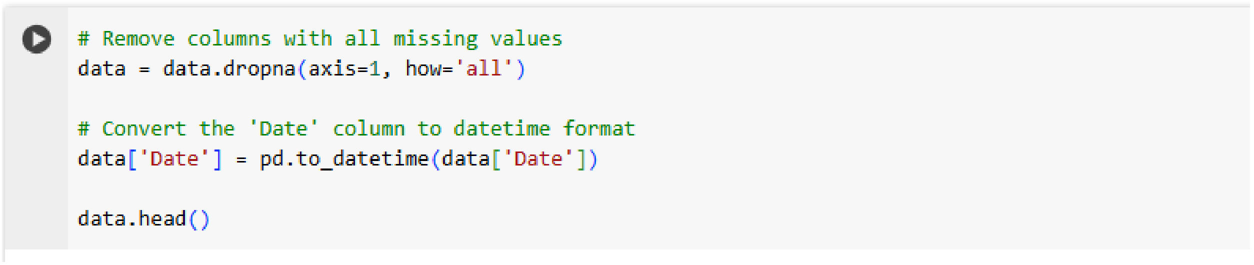
To begin with, we needed to identify missing values, errors in the code, and any duplicate rows for effective data preprocessing. We've discovered numerous errors across all columns of the dataset, specifically '#DIV/0' values, which we needed to remove as a first step.



```python
# Identify columns containing "#DIV/0"
div0_columns = [col for col in data.columns if data[col].astype(str).str.contains("#DIV/0").any()]

# Remove the identified columns
data = data.drop(columns=div0_columns)

data.head()
```

Figure 1. Removing #DIV/0' values

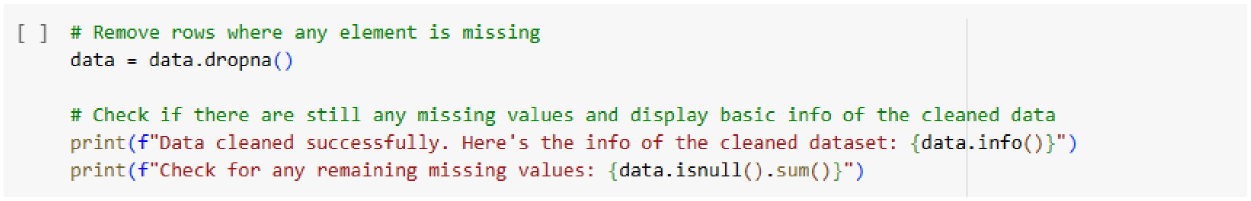
Additionally, we found a few columns that contain no values at all and some with missing entries. Therefore, we decided to remove these columns and converted the 'Date' column to a datetime format.



```python
# Remove columns with all missing values
data = data.dropna(axis=1, how='all')

# Convert the 'Date' column to datetime format
data['Date'] = pd.to_datetime(data['Date'])

data.head()
```

Figure 2. Removing missing values and date conversion

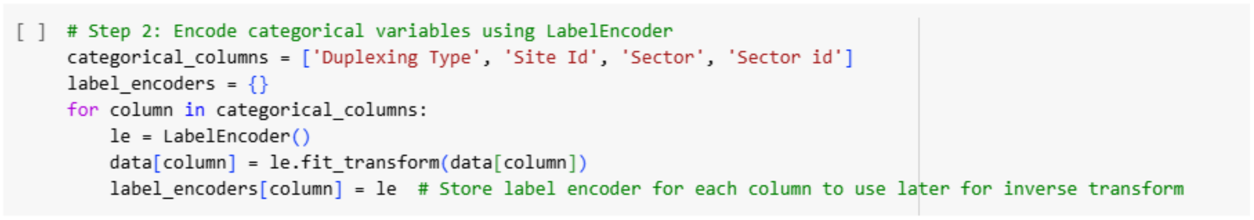
We also identified a few rows that contain no values, along with others that have missing entries. As a result, we decided to remove these rows.



```
[ ]  # Remove rows where any element is missing
     data = data.dropna()

     # Check if there are still any missing values and display basic info of the cleaned data
     print(f"Data cleaned successfully. Here's the info of the cleaned dataset: {data.info()}")
     print(f"Check for any remaining missing values: {data.isnull().sum()}")
```

Figure 3. Removing rows that have missing values

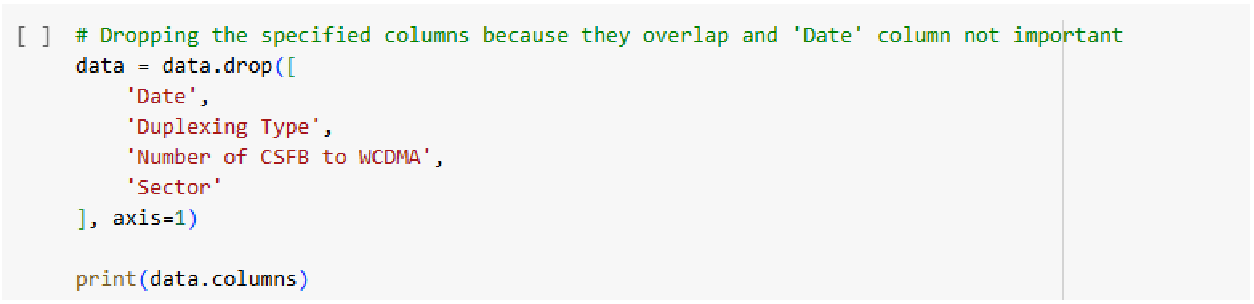
LabelEncoder was used to convert categorical values into unique integers, allowing models to process categorical data as numbers.



```
[ ]  # Step 2: Encode categorical variables using LabelEncoder
     categorical_columns = ['Duplexing Type', 'Site Id', 'Sector', 'Sector id']
     label_encoders = {}
     for column in categorical_columns:
         le = LabelEncoder()
         data[column] = le.fit_transform(data[column])
         label_encoders[column] = le  # Store label encoder for each column to use later for inverse transform
```

Figure 4. Labelencoder for categorical values

Unnecessary columns were dropped, including those containing only zero values, such as 'duplexing type' and 'number of CSFB to WCDMN.' The 'Sector' column was also removed, as 'Sector ID' and 'Site ID' already provided sufficient location identifiers for each site. Additionally, the 'date' column was removed.



```
[ ]  # Dropping the specified columns because they overlap and 'Date' column not important
     data = data.drop([
         'Date',
         'Duplexing Type',
         'Number of CSFB to WCDMA',
         'Sector'
     ], axis=1)

     print(data.columns)
```

Figure 5. Drop columns that have the same values for all of the elements and dropped date column

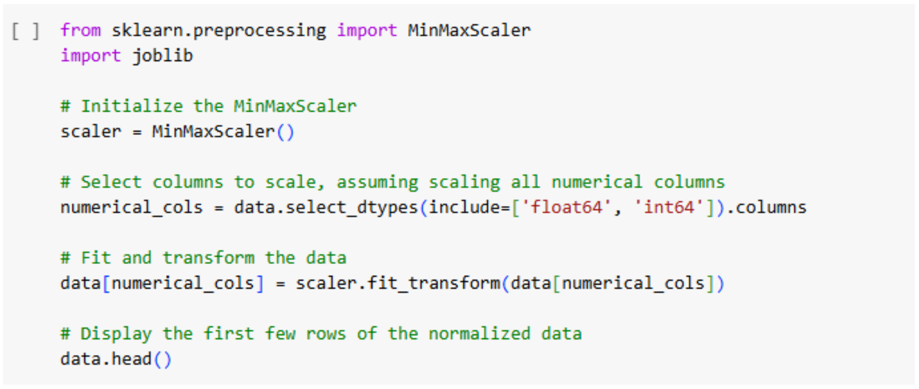
The data processing process took inspiration from [2].

3. AI model development

3.1. Feature engineering/Feature extraction

3.1.1. Normalization

We then normalized the numerical columns using MinMaxScaler to accelerate model training, enhance accuracy by preventing large-scale features from dominating, and facilitate faster convergence for gradient-based methods. This approach also promotes fairer regularization, resulting in a more balanced model performance.



```
[ ] from sklearn.preprocessing import MinMaxScaler
    import joblib

    # Initialize the MinMaxScaler
    scaler = MinMaxScaler()

    # Select columns to scale, assuming scaling all numerical columns
    numerical_cols = data.select_dtypes(include=['float64', 'int64']).columns

    # Fit and transform the data
    data[numerical_cols] = scaler.fit_transform(data[numerical_cols])

    # Display the first few rows of the normalized data
    data.head()
```

Figure 6. Normalised numerical values

3.1.2. Composite Features

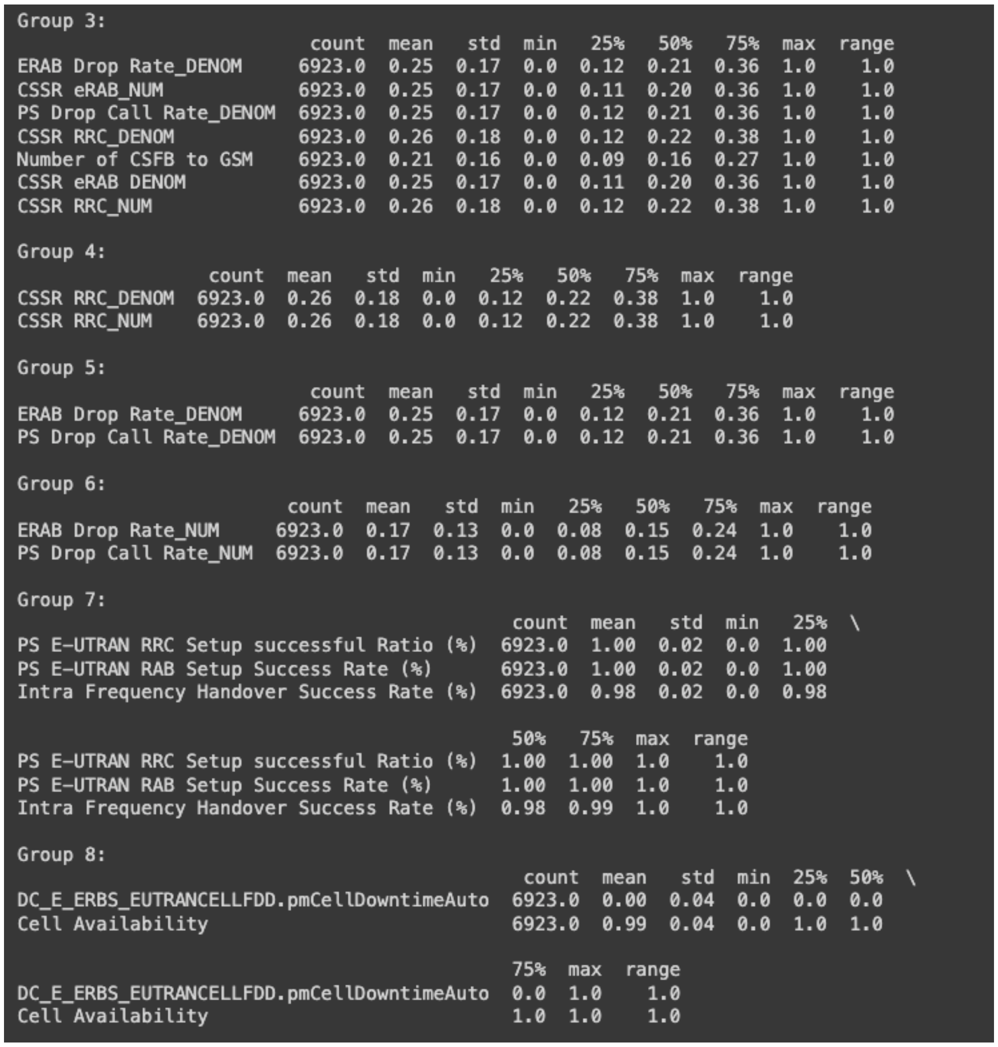
In this project, we performed feature engineering by creating composite features from highly correlated raw metrics. The raw data contained numerous individual metrics with significant redundancy due to high intercorrelation, so we consolidated these features to streamline analysis and enhance interpretability.

Composite Features: We identified groups of similar features using a correlation threshold of 0.7 and grouped them based on their relationships. For groups 1, 2, 7 and 8, we computed composite features by taking the average of the correlated metrics, which allowed us to capture the essential information of each group while reducing the number of features. The resulting composite features include:

- **Secondary_Cell_Performance**: representing metrics related to secondary cell data volume and traffic.
- **Traffic_Volume_and_Payload_Metrics**: encompassing various metrics related to traffic volume and payload.
- **Setup_Success_Rate**: covering metrics associated with connection setup success rates.

- **Cell_Availability_and_Downtime**: representing availability and downtime metrics.

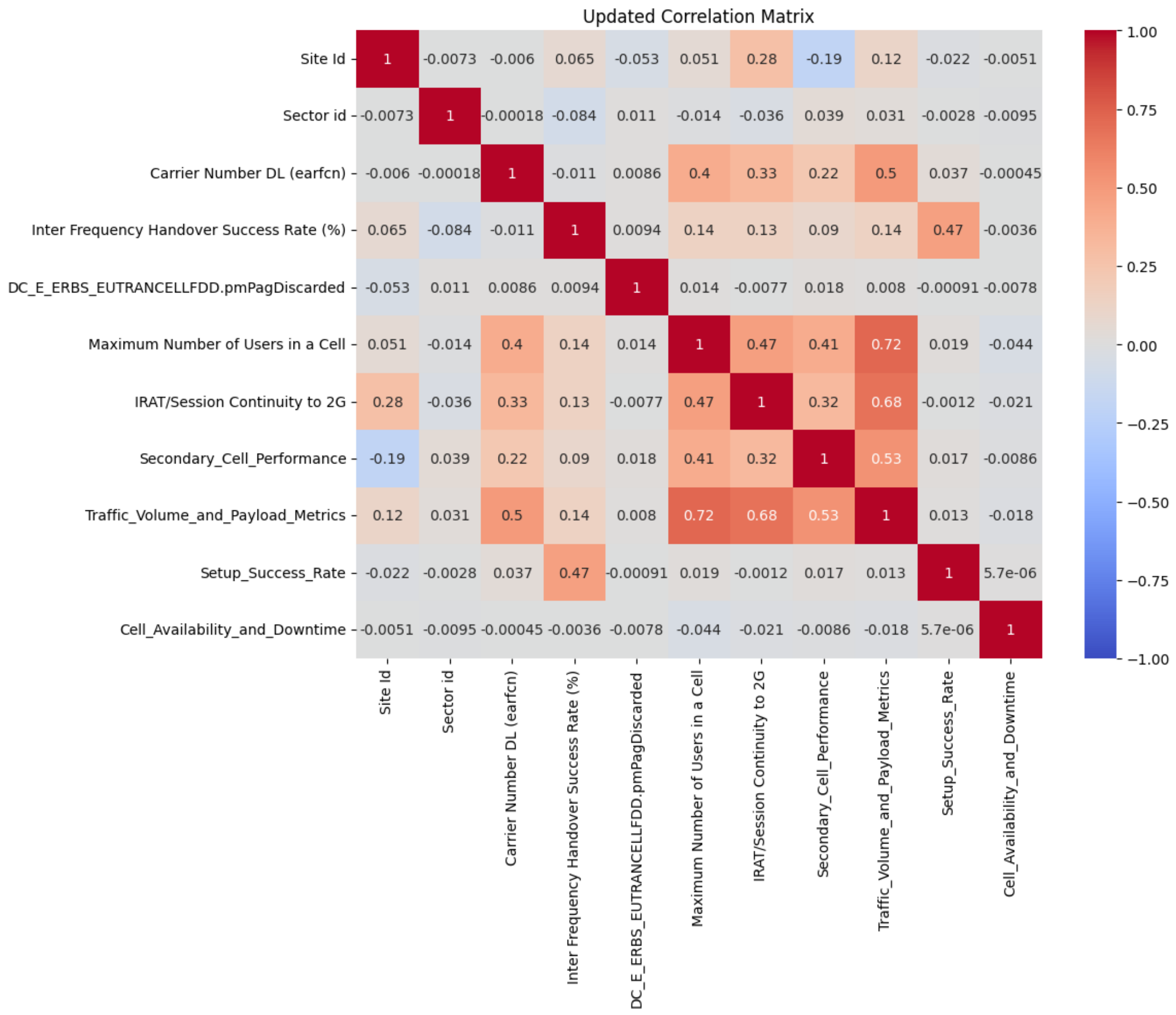
Group 3 to 6 are dropped because they are similar to each other and all represent Success Rate.



```
Group 3:
                          count  mean   std  min   25%   50%   75%  max  range
ERAB Drop Rate_DENOM     6923.0  0.25  0.17  0.0  0.12  0.21  0.36  1.0    1.0
CSSR eRAB_NUM            6923.0  0.25  0.17  0.0  0.11  0.20  0.36  1.0    1.0
PS Drop Call Rate_DENOM  6923.0  0.25  0.17  0.0  0.12  0.21  0.36  1.0    1.0
CSSR RRC_DENOM           6923.0  0.26  0.18  0.0  0.12  0.22  0.38  1.0    1.0
Number of CSFB to GSM    6923.0  0.21  0.16  0.0  0.09  0.16  0.27  1.0    1.0
CSSR eRAB DENOM          6923.0  0.25  0.17  0.0  0.11  0.20  0.36  1.0    1.0
CSSR RRC_NUM             6923.0  0.26  0.18  0.0  0.12  0.22  0.38  1.0    1.0

Group 4:
                 count  mean   std  min   25%   50%   75%  max  range
CSSR RRC_DENOM  6923.0  0.26  0.18  0.0  0.12  0.22  0.38  1.0    1.0
CSSR RRC_NUM    6923.0  0.26  0.18  0.0  0.12  0.22  0.38  1.0    1.0

Group 5:
                          count  mean   std  min   25%   50%   75%  max  range
ERAB Drop Rate_DENOM     6923.0  0.25  0.17  0.0  0.12  0.21  0.36  1.0    1.0
PS Drop Call Rate_DENOM  6923.0  0.25  0.17  0.0  0.12  0.21  0.36  1.0    1.0

Group 6:
                        count  mean   std  min   25%   50%   75%  max  range
ERAB Drop Rate_NUM     6923.0  0.17  0.13  0.0  0.08  0.15  0.24  1.0    1.0
PS Drop Call Rate_NUM  6923.0  0.17  0.13  0.0  0.08  0.15  0.24  1.0    1.0

Group 7:
                                            count  mean   std  min   25%  \
PS E-UTRAN RRC Setup successful Ratio (%)  6923.0  1.00  0.02  0.0  1.00
PS E-UTRAN RAB Setup Success Rate (%)      6923.0  1.00  0.02  0.0  1.00
Intra Frequency Handover Success Rate (%)  6923.0  0.98  0.02  0.0  0.98

                                            50%   75%  max  range
PS E-UTRAN RRC Setup successful Ratio (%)  1.00  1.00  1.0    1.0
PS E-UTRAN RAB Setup Success Rate (%)      1.00  1.00  1.0    1.0
Intra Frequency Handover Success Rate (%)  0.98  0.99  1.0    1.0

Group 8:
                                             count  mean   std  min  25%  50%  \
DC_E_ERBS_EUTRANCELLFDD.pmCellDowntimeAuto  6923.0  0.00  0.04  0.0  0.0  0.0
Cell Availability                           6923.0  0.99  0.04  0.0  1.0  1.0

                                            75%  max  range
DC_E_ERBS_EUTRANCELLFDD.pmCellDowntimeAuto  0.0  1.0    1.0
Cell Availability                           1.0  1.0    1.0
```

Figure 7. Composite feature groups study (Group 3-8)

After composite features are created, we plot a correlation matrix to show the relationship between the updated features.



	Site Id	Sector id	Carrier Number DL (earfcn)	Inter Frequency Handover Success Rate (%)	DC_E_ERBS_EUTRANCELLFDD.pmPagDiscarded	Maximum Number of Users in a Cell	IRAT/Session Continuity to 2G	Secondary_Cell_Performance	Traffic_Volume_and_Payload_Metrics	Setup_Success_Rate	Cell_Availability_and_Downtime
Site Id	1	-0.0073	-0.006	0.065	-0.053	0.051	0.28	-0.19	0.12	-0.022	-0.0051
Sector id	-0.0073	1	-0.00018	-0.084	0.011	-0.014	-0.036	0.039	0.031	-0.0028	-0.0095
Carrier Number DL (earfcn)	-0.006	-0.00018	1	-0.011	0.0086	0.4	0.33	0.22	0.5	0.037	-0.00045
Inter Frequency Handover Success Rate (%)	0.065	-0.084	-0.011	1	0.0094	0.14	0.13	0.09	0.14	0.47	-0.0036
DC_E_ERBS_EUTRANCELLFDD.pmPagDiscarded	-0.053	0.011	0.0086	0.0094	1	0.014	-0.0077	0.018	0.008	-0.00091	-0.0078
Maximum Number of Users in a Cell	0.051	-0.014	0.4	0.14	0.014	1	0.47	0.41	0.72	0.019	-0.044
IRAT/Session Continuity to 2G	0.28	-0.036	0.33	0.13	-0.0077	0.47	1	0.32	0.68	-0.0012	-0.021
Secondary_Cell_Performance	-0.19	0.039	0.22	0.09	0.018	0.41	0.32	1	0.53	0.017	-0.0086
Traffic_Volume_and_Payload_Metrics	0.12	0.031	0.5	0.14	0.008	0.72	0.68	0.53	1	0.013	-0.018
Setup_Success_Rate	-0.022	-0.0028	0.037	0.47	-0.00091	0.019	-0.0012	0.017	0.013	1	5.7e-06
Cell_Availability_and_Downtime	-0.0051	-0.0095	-0.00045	-0.0036	-0.0078	-0.044	-0.021	-0.0086	-0.018	5.7e-06	1

Figure 8. Full correlation matrix after creating composite features

3.1.3. Removing predictors with low correlation scores to targets

To improve model training effectiveness for both the Performance and Zone prediction models, features with low correlation scores to the target variables (Traffic Volume and Payload Metrics, Sector Id, and Site Id) were removed.

For the performance prediction model, features with an absolute correlation score below 0.1 with Traffic Volume and Payload Metrics were excluded, retaining only those that demonstrate stronger relationships with the target. This reduction of features streamlined the dataset, allowing the ANN and XGBoost models focus on the most relevant information.

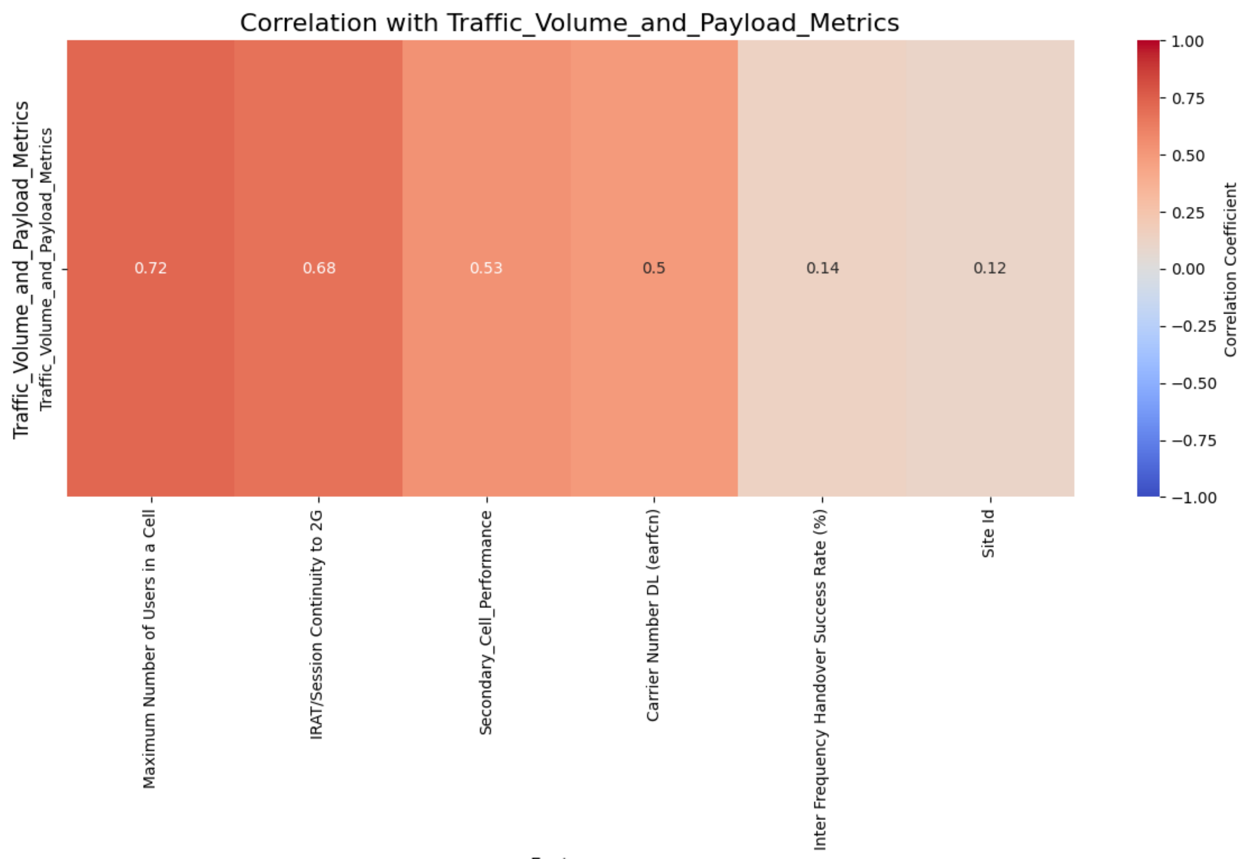


Figure 9. Correlation scores of selected features and Performance

For ANN zone prediction models, features with correlation scores above 0.01 with either Sector Id or Site Id were retained to capture even marginally relevant predictors. This step ensured that the models maintained a balance between simplifying the feature set and preserving potential predictive value.

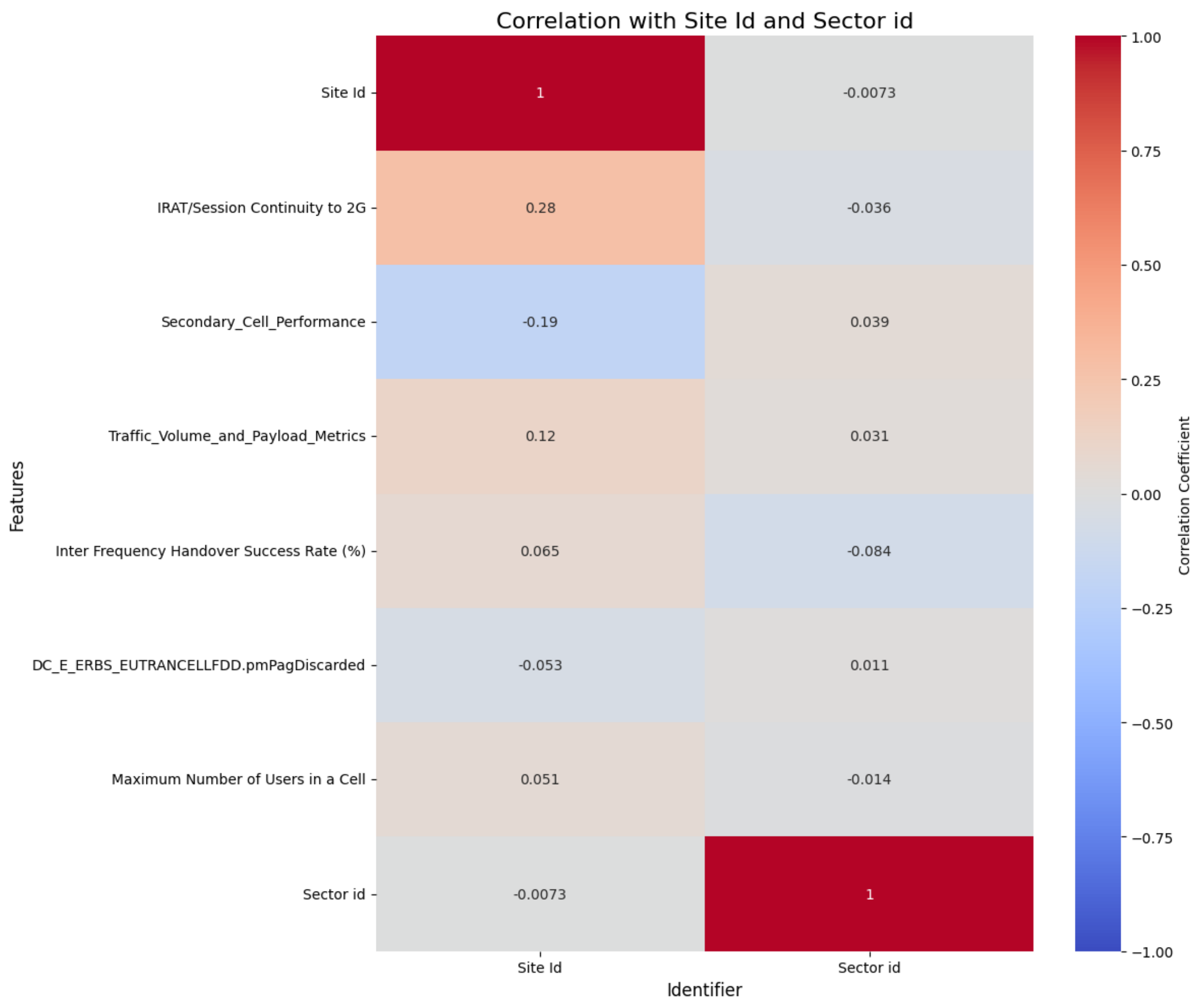


Figure 10. Correlation scores of selected features and Zone

In contrast, the Random Forest zone prediction models used a different approach, leveraging the feature_importances_ function from the sklearn library to retain only significant features. (Section 3.3.1.2)

3.2. Train/Test split

In this project, we employed an 80/20 train-test split for all models, balancing data availability and evaluation rigor. This decision was guided by the following considerations:

- **Sufficient Test Data**: Given the 6923 rows in the dataset, a 20% test set is sufficient. Therefore, the rest can be used for training set.
- **Simplicity of the Task**: Due to the relatively straightforward nature of this task and no overfitting occured, a separate validation set was unnecessary.

3.3. Training models

For model training stage, we utilized several Python libraries, including: **Scikit-learn, Tensorflow,** and **XGBoost.**

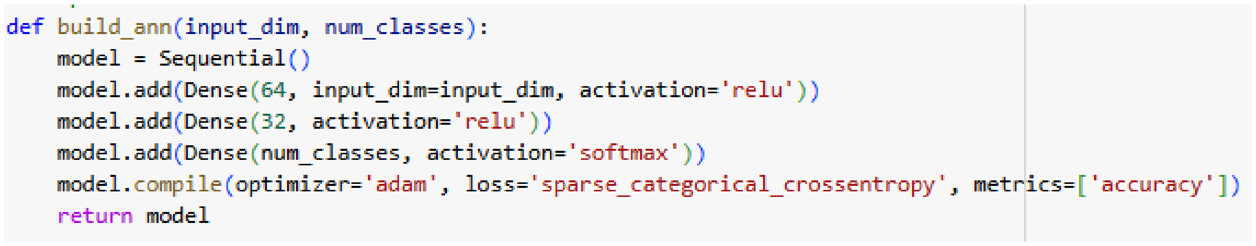
3.3.1. Zone prediction

3.3.1.1. Zone ANN

We developed two **artificial neural network (ANN)** models to classify *sector_id* and *site_id*. Our approach began with data preparation, including encoding categorical variables and normalizing the feature set for consistent scaling. After setting up the dataset, we split it into training and testing sets, ensuring an 80-20 split to evaluate the model's performance.

The model architecture was optimized iteratively, using two hidden layers with 64 and 32 neurons, and *ReLU* activation functions. The output layer utilized *softmax* for multi-class classification. We employed *adam* optimizer and *sparse_categorical_crossentropy* as the loss function. Throughout training, we tested hyperparameters such as batch size and epoch count, finally settling on 100 epochs and a batch size of 32 based on performance stability.

We trained multiple models on both *sector_id* and *site_id* labels. Each model's performance was tracked, and we compared accuracy to identify the most robust configuration. Below is a summary table of model performance:



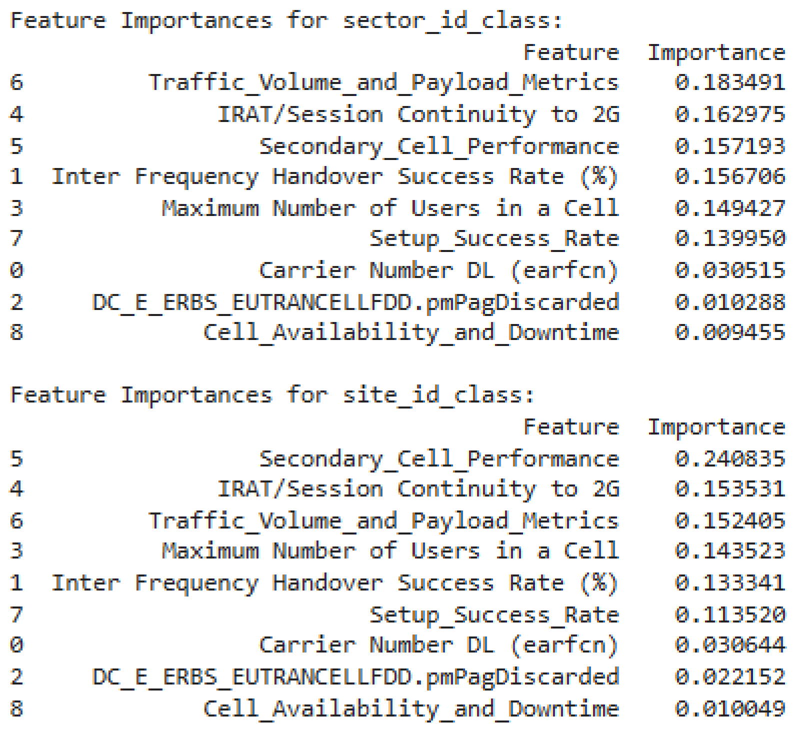
```
def build_ann(input_dim, num_classes):
    model = Sequential()
    model.add(Dense(64, input_dim=input_dim, activation='relu'))
    model.add(Dense(32, activation='relu'))
    model.add(Dense(num_classes, activation='softmax'))
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model
```

Figure 11. Zone ANN model Structure

3.3.1.2. Random Forest

We chose a **Random Forest Classifier** due to its robust performance in classification tasks and its ability to handle feature importance analysis effectively. We used a step-by-step approach, training separate models for each target class: *sector_id_class* and *site_id_class.* To optimize the model performance, we experimented with different numbers of top features for each target, based on feature importance scores calculated after initial model training. This iterative approach allowed us to identify the optimal set of features, enhancing accuracy progressively as more relevant features were included.

We performed hyperparameter tuning for the Random Forest model, setting parameters such as *n_estimators* to 100 and fixing a random seed to ensure reproducibility. After comparing multiple feature sets, we observed that using around 8 top features provided a good balance between accuracy and computational efficiency. Additionally, a tabular summary of accuracy scores for different feature subsets allowed us to compare performance and select the final feature set.



```
Feature Importances for sector_id_class:
                                     Feature  Importance
6         Traffic_Volume_and_Payload_Metrics    0.183491
4              IRAT/Session Continuity to 2G    0.162975
5                 Secondary_Cell_Performance    0.157193
1  Inter Frequency Handover Success Rate (%)    0.156706
3           Maximum Number of Users in a Cell    0.149427
7                         Setup_Success_Rate    0.139950
0                  Carrier Number DL (earfcn)    0.030515
2     DC_E_ERBS_EUTRANCELLFDD.pmPagDiscarded    0.010288
8              Cell_Availability_and_Downtime    0.009455

Feature Importances for site_id_class:
                                     Feature  Importance
5                 Secondary_Cell_Performance    0.240835
4              IRAT/Session Continuity to 2G    0.153531
6         Traffic_Volume_and_Payload_Metrics    0.152405
3           Maximum Number of Users in a Cell    0.143523
1  Inter Frequency Handover Success Rate (%)    0.133341
7                         Setup_Success_Rate    0.113520
0                  Carrier Number DL (earfcn)    0.030644
2     DC_E_ERBS_EUTRANCELLFDD.pmPagDiscarded    0.022152
8              Cell_Availability_and_Downtime    0.010049
```

Figure 12 Evaluation of Zone Random Forest model

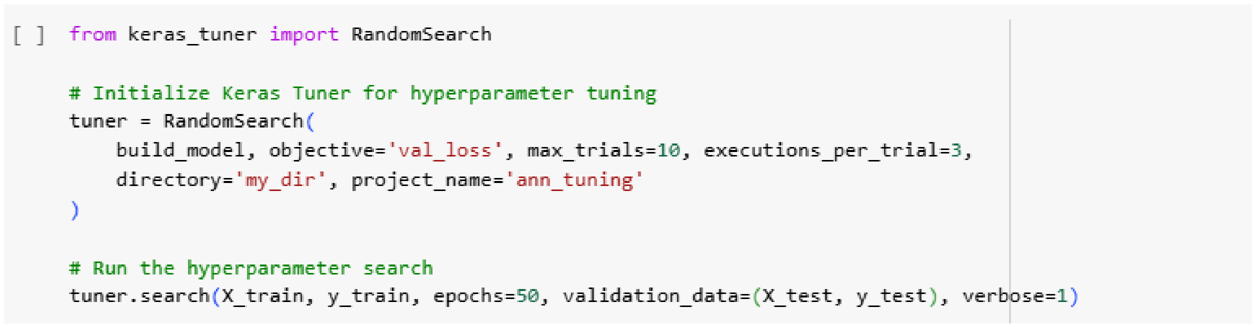
3.3.2. Performance prediction

3.3.2.1. Performance ANN

Following feature engineering, we used a Performance dataset, which includes several input features necessary for forecasting network traffic measurements. A set of normalised characteristics is included in this processed dataset to improve the predictive power of the model. *Traffic_Volume_and_Payload_Metrics*, our target variable, is the result we hope to predict. After preparing the dataset, we constructed, adjusted, and trained our Artificial Neural Network (ANN) model to determine the optimal topology for performance.

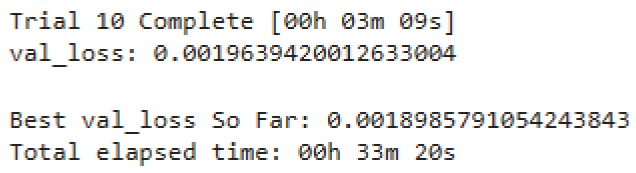
We began by defining the model structure and then used Keras Tuner to find the best configurations by varying hyperparameters like the number of units and layers. To conduct this tuning, we used RandomSearch, configuring it to minimise validation loss between the trials and optimise the model's accuracy. The key parameters are:

- **objective='val_loss'**: Specifies that the tuner should optimize for the lowest validation loss, aiming for the best generalization.
- **max_trials=10**: Limits the tuner to exploring 10 different hyperparameter combinations, balancing thoroughness with computational efficiency.
- **executions_per_trial=3**: Each combination is tried three times to get a reliable performance estimate, reducing the impact of random variations.
- **epochs=50**: Sets each trial to train for 50 epochs, allowing sufficient learning for each configuration.



```
[ ]  from keras_tuner import RandomSearch

     # Initialize Keras Tuner for hyperparameter tuning
     tuner = RandomSearch(
         build_model, objective='val_loss', max_trials=10, executions_per_trial=3,
         directory='my_dir', project_name='ann_tuning'
     )

     # Run the hyperparameter search
     tuner.search(X_train, y_train, epochs=50, validation_data=(X_test, y_test), verbose=1)
```

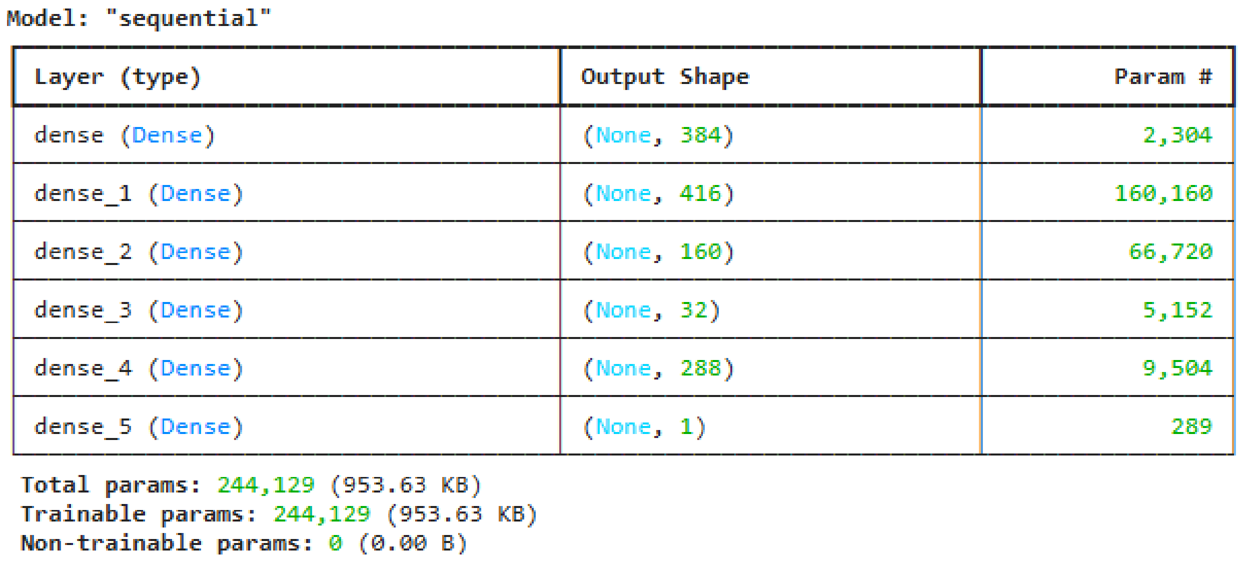
Figure 13. Performance ANN model Hyperparameter Tuning Code



```
Trial 10 Complete [00h 03m 09s]
val_loss: 0.0019639420012633004

Best val_loss So Far: 0.0018985791054243843
Total elapsed time: 00h 33m 20s
```

Figure 14. Tuning output: After over 30 mins of tuning, the best validation loss (MSE) found is about 0.00189

This hyperparameter tuning allowed us to determine the best model setup for our dataset.



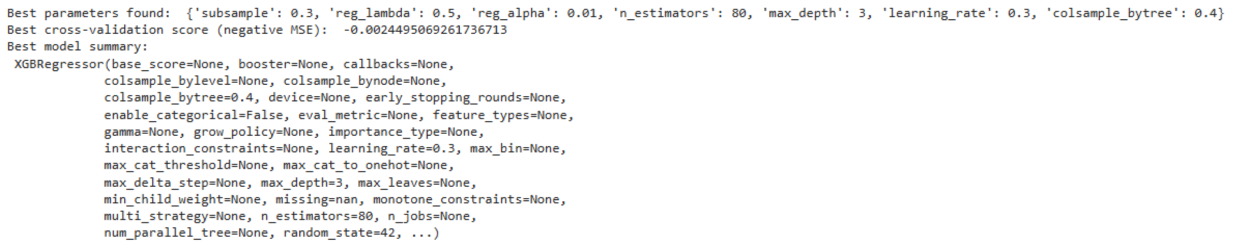
```
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━┓
┃ Layer (type)                     ┃ Output Shape              ┃        Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━┩
│ dense (Dense)                    │ (None, 384)               │          2,304 │
├──────────────────────────────────┼───────────────────────────┼────────────────┤
│ dense_1 (Dense)                  │ (None, 416)               │        160,160 │
├──────────────────────────────────┼───────────────────────────┼────────────────┤
│ dense_2 (Dense)                  │ (None, 160)               │         66,720 │
├──────────────────────────────────┼───────────────────────────┼────────────────┤
│ dense_3 (Dense)                  │ (None, 32)                │          5,152 │
├──────────────────────────────────┼───────────────────────────┼────────────────┤
│ dense_4 (Dense)                  │ (None, 288)               │          9,504 │
├──────────────────────────────────┼───────────────────────────┼────────────────┤
│ dense_5 (Dense)                  │ (None, 1)                 │            289 │
└──────────────────────────────────┴───────────────────────────┴────────────────┘
 Total params: 244,129 (953.63 KB)
 Trainable params: 244,129 (953.63 KB)
 Non-trainable params: 0 (0.00 B)
```

Figure 15. Best Performance ANN model: includes 1 input layer, 4 hidden layers and 1 output layer

After determining the optimal setup, we used the training data to train the ANN model and the test set to assess its performance. With this strategy and the feature-engineered dataset, we were able to produce a reliable model that was prepared for use.

3.3.2.2. XGBoost

In order to compare the performance of the ANN model and an XGBoost model, we additionally trained and adjusted the latter using the feature-engineered dataset. The number of estimators, maximum depth, learning rate, subsample rate, column sample rate, regularisation parameters (reg_alpha and reg_lambda), and other configurations were among the parameters we searched for using RandomizedSearchCV. Finding the ideal parameter combination to reduce the Mean Squared Error (MSE) on the validation set was the goal of the tuning procedure. After tuning, we selected the best-performing XGBoost model and evaluated it on the test set of performance data.



```
Best parameters found:  {'subsample': 0.3, 'reg_lambda': 0.5, 'reg_alpha': 0.01, 'n_estimators': 80, 'max_depth': 3, 'learning_rate': 0.3, 'colsample_bytree': 0.4}
Best cross-validation score (negative MSE):  -0.0024495069261736713
Best model summary:
 XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=0.4, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.3, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=3, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=80, n_jobs=None,
             num_parallel_tree=None, random_state=42, ...)
```

Figure 16. Best XGBoost model summary

3.4. Evalution of AI models

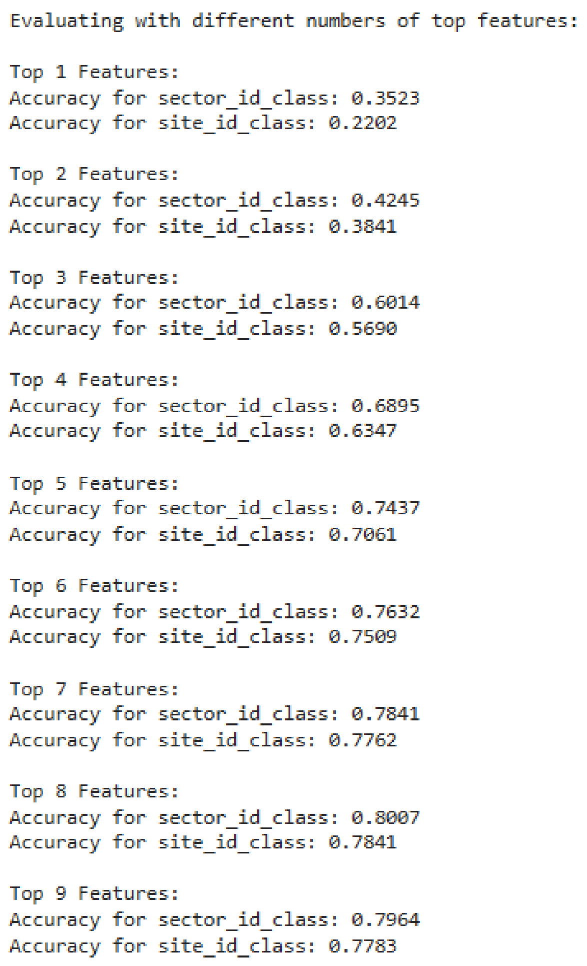
3.4.1. Zone prediction

3.4.1.1. Random Forest

The evaluation of our model was based on **accuracy scores** for both *sector_id_class* and *site_id_class* classifications. By iteratively selecting different numbers of top features, we evaluated each model on a test set to measure its predictive accuracy. This approach highlighted how the inclusion of additional features improved model performance until reaching a peak around the top 8 features.

Our evaluation metrics included accuracy for each target class, with accuracy scores progressively increasing as more features were considered. For instance, with 8 features, the model achieved accuracies of 0.8007 for *sector_id_class* and 0.7841 for *site_id_class*. These results were pivotal in deciding the final model configuration, ensuring that both targets were classified with high accuracy while maintaining model efficiency.

This evaluation and selection process ensured a well-optimized model suited for real-world application.



```
Evaluating with different numbers of top features:

Top 1 Features:
Accuracy for sector_id_class: 0.3523
Accuracy for site_id_class: 0.2202

Top 2 Features:
Accuracy for sector_id_class: 0.4245
Accuracy for site_id_class: 0.3841

Top 3 Features:
Accuracy for sector_id_class: 0.6014
Accuracy for site_id_class: 0.5690

Top 4 Features:
Accuracy for sector_id_class: 0.6895
Accuracy for site_id_class: 0.6347

Top 5 Features:
Accuracy for sector_id_class: 0.7437
Accuracy for site_id_class: 0.7061

Top 6 Features:
Accuracy for sector_id_class: 0.7632
Accuracy for site_id_class: 0.7509

Top 7 Features:
Accuracy for sector_id_class: 0.7841
Accuracy for site_id_class: 0.7762

Top 8 Features:
Accuracy for sector_id_class: 0.8007
Accuracy for site_id_class: 0.7841

Top 9 Features:
Accuracy for sector_id_class: 0.7964
Accuracy for site_id_class: 0.7783
```

Figure 17 Evaluation of Random Forest with Feature Extraction

3.4.1.2. Zone ANN

In evaluating our AI model, we utilized two key metrics: **F1-score** and **Accuracy**. These measures allowed us to assess the model's precision and ability to classify correctly.

Before Feature Engineering:

- For **Sector ID**, the model achieved an F1-score of 0.73, with an accuracy of 72.6%.
- For **Site ID**, the F1-score was similarly strong at 0.72, and accuracy was 72.0%.

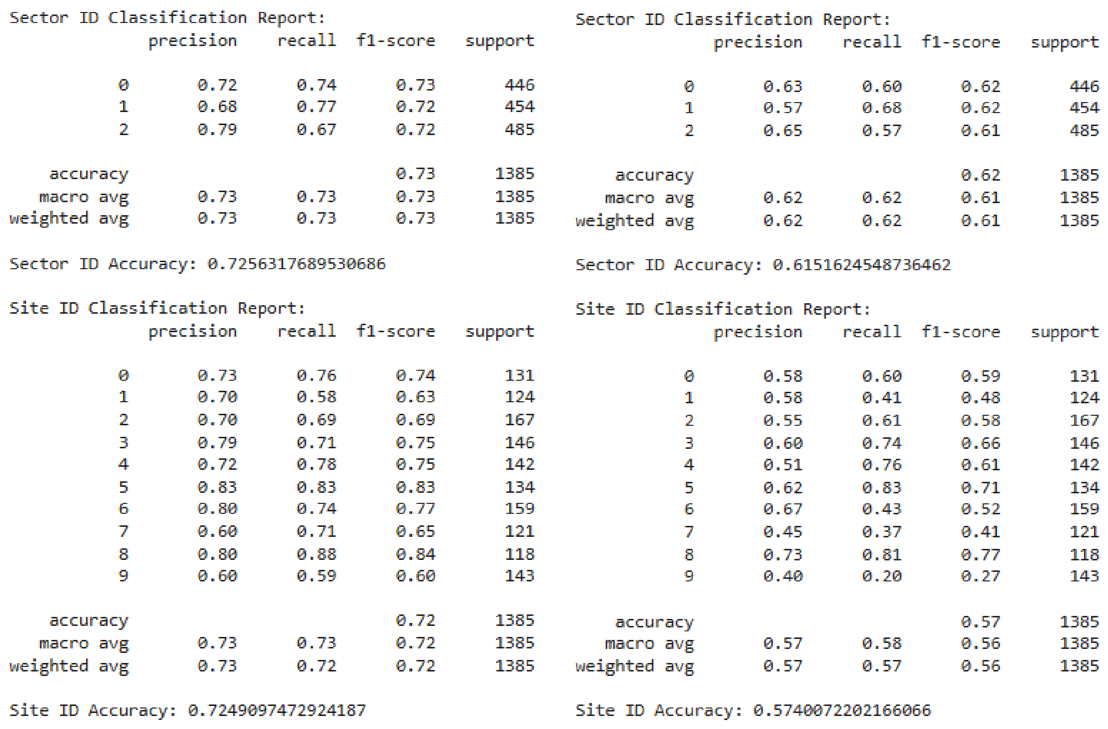
These scores reflect a balanced performance across sectors and sites, showing the model’s effectiveness in distinguishing between categories.

After Feature Engineering:

- Post feature engineering, there was a decline in performance for both Sector ID and Site ID classifications. For Sector ID, accuracy dropped to 61.5% and the F1-score to 0.61. Similarly, Site ID accuracy fell to 57.4%, with an F1-score of 0.56.

The decrease in performance post-feature engineering could be attributed to the introduction of features that, rather than enhancing, added noise or complexity, leading to less effective decision boundaries. This highlights that while feature engineering often enhances model performance, it can sometimes introduce redundant or misleading information, reducing the model's ability to generalize.

The results indicate that using an Artificial Neural Network (ANN) for this classification task may not be optimal due to its limited performance and interpretability. With accuracy scores of **61.5%** for *sector_id_class* and **57.4%** for *site_id_class*, the ANN struggles to generalize well across classes. The model's precision, recall, and F1-scores vary significantly across different classes, showing inconsistencies and revealing that it fails to handle certain classes effectively. For example, the F1-score for some *site_id_class* categories is as low as **0.27**, which suggests potential overfitting to specific patterns in the training set while underperforming on others. Additionally, ANNs with dense layers provide limited interpretability, making it difficult to understand the decision-making process or gain insights into feature importance, which are often valuable in real-world applications. Given these limitations, alternative models like Random Forest or Gradient Boosting may offer better classification accuracy and consistency across classes, along with improved interpretability and robustness in handling diverse feature sets and class imbalances.



```
Sector ID Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.74      0.73       446
           1       0.68      0.77      0.72       454
           2       0.79      0.67      0.72       485

    accuracy                           0.73      1385
   macro avg       0.73      0.73      0.73      1385
weighted avg       0.73      0.73      0.73      1385

Sector ID Accuracy: 0.7256317689530686

Site ID Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.76      0.74       131
           1       0.70      0.58      0.63       124
           2       0.70      0.69      0.69       167
           3       0.79      0.71      0.75       146
           4       0.72      0.78      0.75       142
           5       0.83      0.83      0.83       134
           6       0.80      0.74      0.77       159
           7       0.60      0.71      0.65       121
           8       0.80      0.88      0.84       118
           9       0.60      0.59      0.60       143

    accuracy                           0.72      1385
   macro avg       0.73      0.73      0.72      1385
weighted avg       0.73      0.72      0.72      1385

Site ID Accuracy: 0.7249097472924187
```

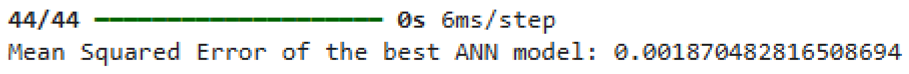
```
Sector ID Classification Report:
              precision    recall  f1-score   support

           0       0.63      0.60      0.62       446
           1       0.57      0.68      0.62       454
           2       0.65      0.57      0.61       485

    accuracy                           0.62      1385
   macro avg       0.62      0.62      0.61      1385
weighted avg       0.62      0.62      0.61      1385

Sector ID Accuracy: 0.6151624548736462

Site ID Classification Report:
              precision    recall  f1-score   support

           0       0.58      0.60      0.59       131
           1       0.58      0.41      0.48       124
           2       0.55      0.61      0.58       167
           3       0.60      0.74      0.66       146
           4       0.51      0.76      0.61       142
           5       0.62      0.83      0.71       134
           6       0.67      0.43      0.52       159
           7       0.45      0.37      0.41       121
           8       0.73      0.81      0.77       118
           9       0.40      0.20      0.27       143

    accuracy                           0.57      1385
   macro avg       0.57      0.58      0.56      1385
weighted avg       0.57      0.57      0.56      1385

Site ID Accuracy: 0.5740072202166066
```

Figure 18. Evalution of Zone ANN mode before and after Feature Engineeringl

3.4.2. Performance predition

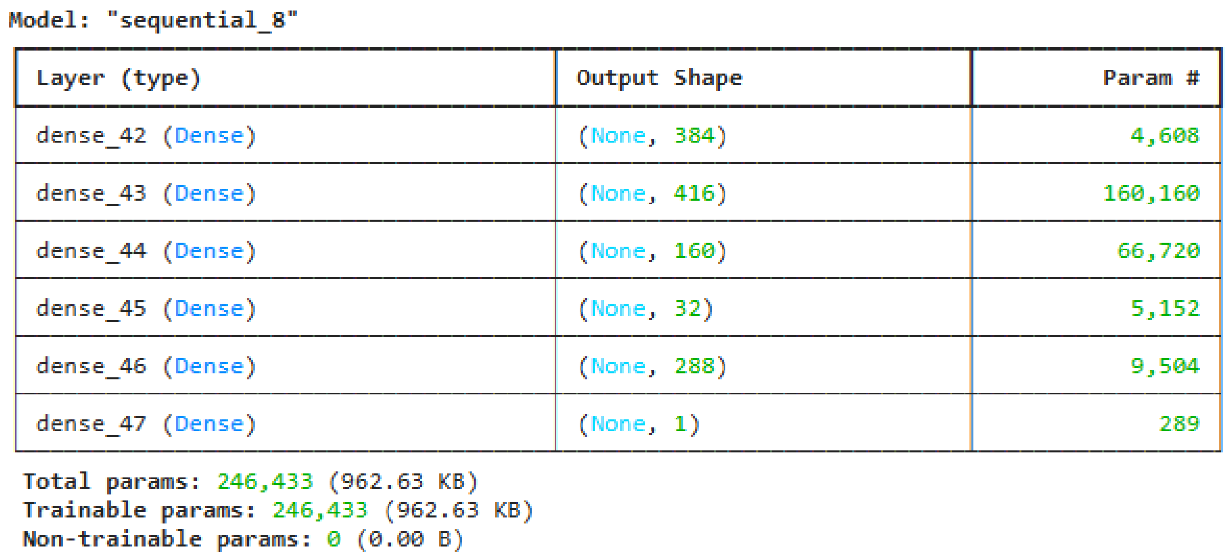
3.4.2.1. Performance ANN

We used the test set of performance dataset to assess the ANN model's predictive accuracy after tuning it on the feature-engineered train dataset. With this feature-engineered dataset, the Mean Squared Error (MSE) was remarkably low, suggesting that the model could accurately predict Traffic_Volume_and_Payload_Metrics using the fine-tuned inputs. Because the input features were optimised during the feature-engineering process, the model was able to capture more pertinent patterns and relationships, as evidenced by the reduced MSE.



```
44/44 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Mean Squared Error of the best ANN model: 0.001870482816508694
```

On top of that, we reconstructed a comparable ANN model using the same architecture as our tuned model, but we changed the shape of the input layer to match the structure of the original dataset prior to feature engineering (11 input features instead of 5) in order to evaluate the effect of feature engineering. When evaluated on the raw dataset, this rebuilt model produced a significantly higher MSE than its feature-engineered equivalent.

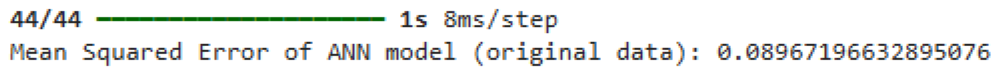


Model: "sequential_8"

Layer (type)	Output Shape	Param #
dense_42 (Dense)	(None, 384)	4,608
dense_43 (Dense)	(None, 416)	160,160
dense_44 (Dense)	(None, 160)	66,720
dense_45 (Dense)	(None, 32)	5,152
dense_46 (Dense)	(None, 288)	9,504
dense_47 (Dense)	(None, 1)	289

```
Total params: 246,433 (962.63 KB)
Trainable params: 246,433 (962.63 KB)
Non-trainable params: 0 (0.00 B)
```

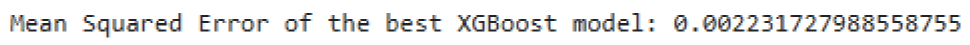
The benefits of feature engineering are evident from its results (≈0.08967 vs ≈0.00187). The ANN model may learn more efficiently if we reduced noise and redundancy in the input features by normalizing and improving them. On the other hand, the model's accuracy was constrained by the original dataset's raw and possibly less informative features. The significance of feature engineering in enhancing model performance is demonstrated by this contrast, which allows the model to concentrate on the most pertinent elements of the data, thus producing predictions that are more accurate and trustworthy.



```
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step
Mean Squared Error of ANN model (original data): 0.08967196632895076
```

3.4.2.2. XGBoost

On the identical feature-engineered dataset, the XGBoost model's MSE was significantly greater than the ANN model's, despite its respectable accuracy (≈0.00223 vs ≈0.00187 of ANN). Even while XGBoost has a solid reputation for handling tabular data efficiently, the ANN model performed better in this instance, indicating that the architecture of the neural network was more appropriate for identifying intricate correlations in this specific dataset.



```
Mean Squared Error of the best XGBoost model: 0.002231727988558755
```

Given its lower MSE, this comparison demonstrates how well our ANN model predicts Traffic_Volume_and_Payload_Metrics when compared to XGBoost. The ANN's greater performance indicates that it can learn and generalise from the feature-engineered data more effectively, proving that deep learning techniques can sometimes perform better than more conventional ensemble techniques like XGBoost.

There is also a model trained with dataset before feature engineering which results in 0.00223 in loss.

```
Mean Squared Error of the best XGBoost model: 0.0024503418068854447
```

3.5. Final models selection

		Zone ANN	Random Forest
Before feature engineering	Site	0.724	0.77
	Sector	0.725	0.79
After feature engineering	Site	0.574	0.78
	Sector	0.615	0.8

Table 1. Zone models comparison

	Performance ANN	XGBoost
Before feature engineering	0.0897	0.00245
After feature engineering	0.00187	0.00223

Table 2. Performance models comparison

After evaluation, we selected the best models for each task, with substantial improvements observed from feature engineering:

- **Zone Prediction (Random Forest over ANN)**:
    - **Improved Accuracy**: Feature engineering (<u>using 8 instead of 10 features</u>) reduced redundancy and selected only the most relevant predictors, enhancing model accuracy and stability.
    - **Efficiency**: Fewer, composite features streamlined the model, enabling it to capture essential patterns more effectively.
    - **Interpretability**: Consolidated features provided clearer insights into key metrics for zone classification.
- **Performance Prediction (ANN over XGBoost)**:
    - **Noise Reduction**: Removing low-correlation features (<u>using 5 instead of 10 features</u>) helped the ANN model focus on the most impactful data, resulting in a lower Mean Squared Error (MSE).
    - **Higher Predictive Accuracy**: Composite features enabled the model to learn complex patterns, improving generalization and accuracy.
    - **Efficient Learning**: The optimized feature set reduced training time and improved model robustness on new data.

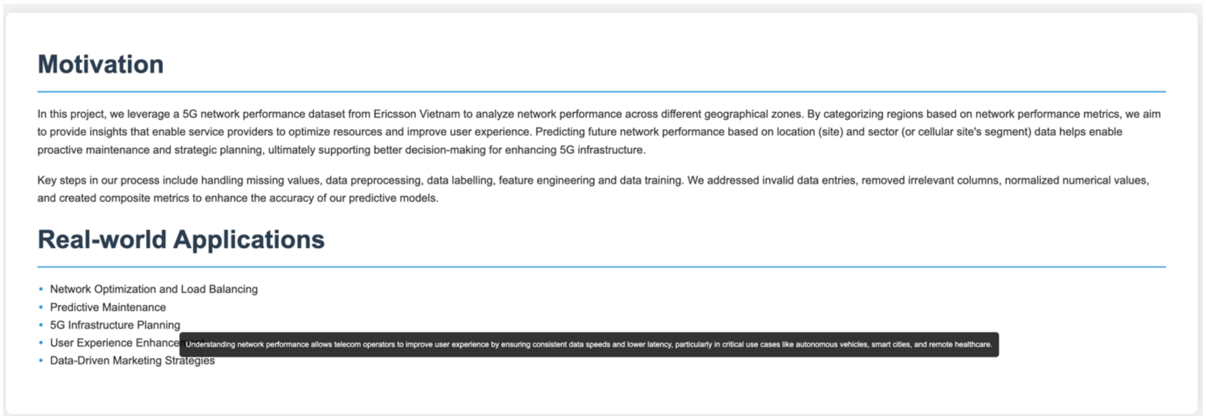
4. AI demonstrator

4.1. Demonstrator Analysis:

The frontend consists of several structured sections in index.html, JavaScript scripts for interactivity, and CSS styling. Below is a breakdown of the HTML structure and associated JavaScript files:

4.1.1. Key HTML Sections

- **Introduction Section:** Contains the project title.
- **Project Information Section**
  - Motivation and Applications:
    - This section outlines the primary motivations behind the project.
    - Key applications are listed with tooltips to elaborate on each point when users hover over them.



Motivation

In this project, we leverage a 5G network performance dataset from Ericsson Vietnam to analyze network performance across different geographical zones. By categorizing regions based on network performance metrics, we aim to provide insights that enable service providers to optimize resources and improve user experience. Predicting future network performance based on location (site) and sector (or cellular site's segment) data helps enable proactive maintenance and strategic planning, ultimately supporting better decision-making for enhancing 5G infrastructure.

Key steps in our process include handling missing values, data preprocessing, data labelling, feature engineering and data training. We addressed invalid data entries, removed irrelevant columns, normalized numerical values, and created composite metrics to enhance the accuracy of our predictive models.

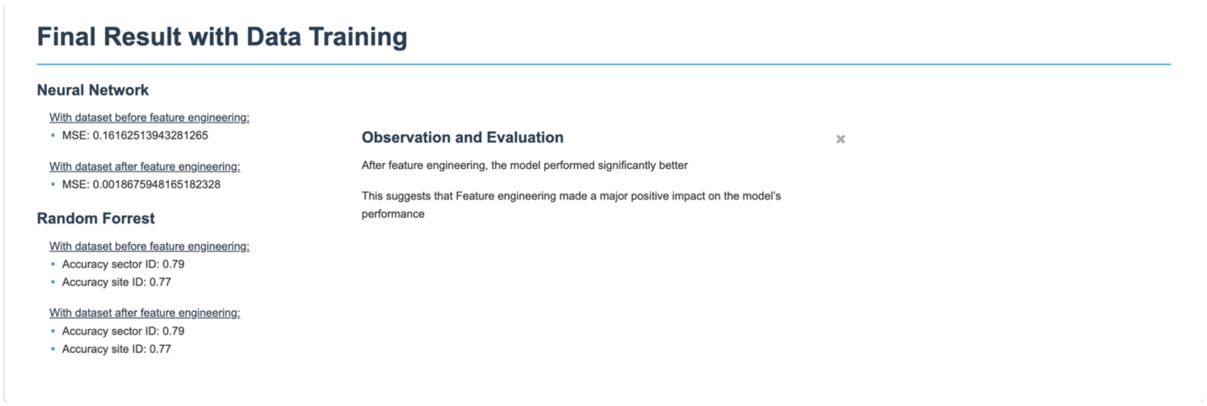
Real-world Applications

- Network Optimization and Load Balancing
- Predictive Maintenance
- 5G Infrastructure Planning
- User Experience Enhance
- Data-Driven Marketing Strategies

Understanding network performance allows telecom operators to improve user experience by ensuring consistent data speeds and lower latency, particularly in critical use cases like autonomous vehicles, smart cities, and remote healthcare.

Figure 19. Project Information section.

- **Final Results Section**
  - Data Training Results:
    - Displays performance metrics (MSE and accuracy) for neural networks and random forest models, both before and after feature engineering.
    - Enable modal on click to explore the context.



Final Result with Data Training

Neural Network

With dataset before feature engineering:
- MSE: 0.16162513943281265

With dataset after feature engineering:
- MSE: 0.0018675948165182328

Random Forrest

With dataset before feature engineering:
- Accuracy sector ID: 0.79
- Accuracy site ID: 0.77

With dataset after feature engineering:
- Accuracy sector ID: 0.79
- Accuracy site ID: 0.77

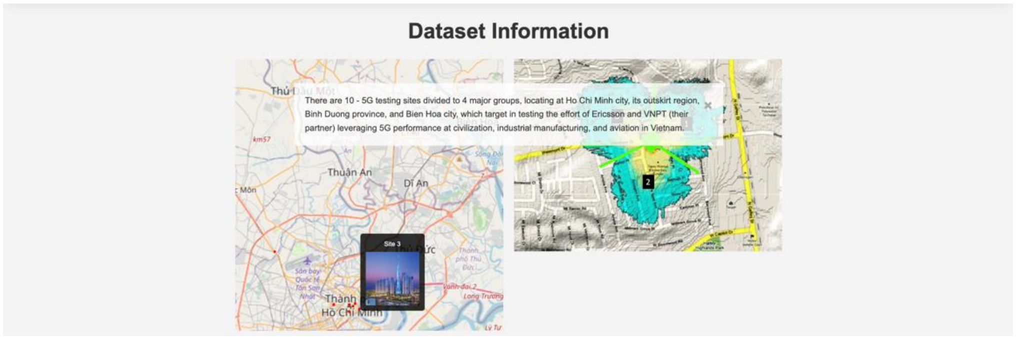
Observation and Evaluation

After feature engineering, the model performed significantly better

This suggests that Feature engineering made a major positive impact on the model's performance

Figure 20. Final Result section with modal window.

- **Dataset Information Section**
  - **Map and Markers**: An interactive image map displays 5G testing sites across geographic locations. Each site marker provides a tooltip with information about the site and an image when hovered over.
  - **Sector Information**: A separate section explains how cellular towers cover areas by dividing them into sectors (A, B, and C) for efficient signal distribution.

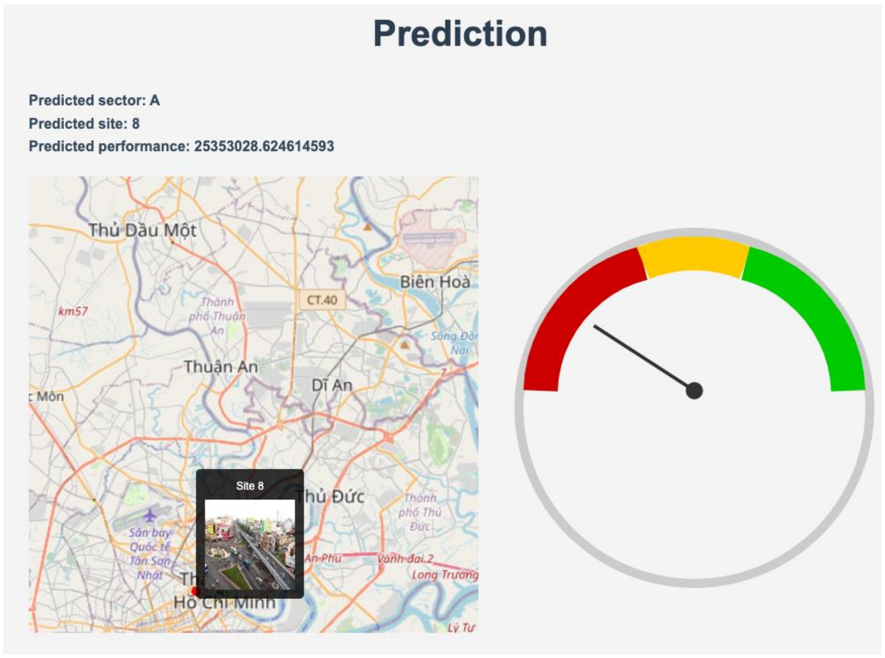


Dataset Information

There are 10 - 5G testing sites divided to 4 major groups, locating at Ho Chi Minh city, its outskirt region, Binh Duong province, and Bien Hoa city, which target in testing the effort of Ericsson and VNPT (their partner) leveraging 5G performance at civilization, industrial manufacturing, and aviation in Vietnam.

Figure 21. Dataset Information section with modal window and tooltip describing the site location.

- **Prediction Section**
  - **Feature Inputs**:
    - Users enter data into form fields for features required.
    - Composite features like Secondary Cell Performance and Setup Success Rate include dropdowns for additional details.
  - **Prediction Outputs**:
    - Displays the predicted sector, site, and performance metrics after running predictions using the Flask API.


Prediction

Predicted sector: A
Predicted site: 8
Predicted performance: 25353028.624614593

Figure 22. Prediction outcomes show the result of performance, site and sector id; the site id with a marker on the map with tooltip shows on mouseover; a speedometer illustrates the performance speed.

4.1.2.JavaScript Scripts

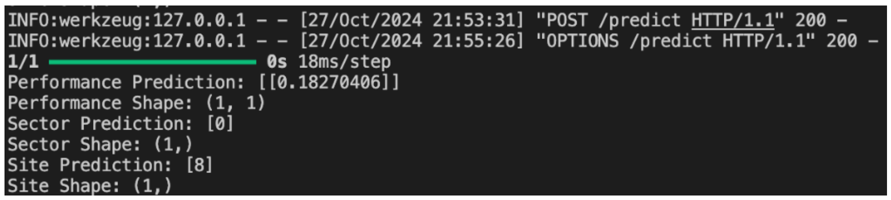
- **add_pins.js**
  - **Purpose**: Manages the addition of pins on the location map and adds tooltips for site information on mouseover.
  - **Functionality**: It finds the appropriate pin based on the pin_id, creates the pin element, and dynamically adds it to the map. The pins display tooltips with site information when hovered.
- **needle.js**
  - **Purpose**: Controls the movement of the speedometer needle in the Prediction section, visualizing the predicted performance on a scale.
- **prediction_helpers.js**
  - **Purpose**: Contains helper functions to manage the prediction process by handling form inputs, gathering data, and formatting output for the prediction section.
- **tooltip.js**
  - **Purpose**: Adds tooltips to various UI elements such as form labels, list items, and site markers.
  - **Functionality**: When the user hovers over a specified element, the tooltip provides additional information, such as feature explanations or site images.

- **modal.js**
  - **Purpose**: Handles the opening and closing of modal dialogs.
  - **Functionality**: Listens for clicks on various elements and manages multiple modal displays to show detailed project or dataset information as needed.

4.1.3. Backend (Flask API flask_app.py)

The backend, implemented in Flask (flask_app.py), acts as an API that processes user input, runs predictions using pre-trained models, and returns the results. Key components and functionality include:

- **Endpoints:**
  - The main endpoint is /predict, which accepts JSON data from the frontend and returns predictions for performance, sector ID, and site ID.
- **Model Integration:**
  - **Loaded Models**: Three models are used: best_ann_model_speed (predicts performance metrics), model_top_sec.pkl (predicts sector ID), and model_top_site.pkl (predicts site ID).
  - **Scalers**: Separate scalers (scaler_performance.pkl, scaler_sector.pkl, scaler_site.pkl) normalize the input data, ensuring compatibility with model training.
- **Prediction Workflow:**
  - **Input Data Processing**: The JSON input is transformed into DataFrames and scaled using the corresponding scalers.
  - **Predictions**: The scaled inputs are fed into the models, which output predicted values for sector, site, and performance.
  - **Output**: Results are formatted as JSON and sent back to the frontend to display in the Prediction section.
- **Error Handling:**
  - The API includes try-except blocks to catch exceptions, returning error messages as JSON if an issue arises during prediction.



```
INFO:werkzeug:127.0.0.1 - - [27/Oct/2024 21:53:31] "POST /predict HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [27/Oct/2024 21:55:26] "OPTIONS /predict HTTP/1.1" 200 -
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
Performance Prediction: [[0.18270406]]
Performance Shape: (1, 1)
Sector Prediction: [0]
Sector Shape: (1,)
Site Prediction: [8]
Site Shape: (1,)
```

Figure 23. Flask API backend predicting outcome on terminal console

This analysis provides an overview of each component's role in the 5G network prediction system, from interactive data entry and informative UI elements in the frontend to the model-based prediction logic managed by the Flask API.

4.2. Feature Prediction

- **Raw Input Features**:
    - **Carrier Number DL (earfcn)**: Downlink carrier frequency; affects signal coverage and capacity.
    - **Inter Frequency Handover Success Rate (%)**: Measures successful handovers between frequencies, critical for uninterrupted service.
    - **DC_E_ERBS_EUTRANCELLFDD.pmPagDiscarded**: Tracks discarded paging attempts, indicating network congestion or instability.
    - **Maximum Number of Users in a Cell**: Maximum capacity of users that a cell can handle, affecting load balancing.
    - **IRAT/Session Continuity to 2G**: Percentage of sessions transitioning smoothly to 2G networks, crucial for fallback support.
- **Composite Input Features**:
    - **Secondary Cell Performance**: Reflects secondary cell’s effectiveness, incorporating:
        - ScellVolume_DL_MAC: Measures downlink traffic on the secondary cell.
        - Scell Traffic Volume: Overall traffic on the secondary cell.
    - **Setup Success Rate**: Measures initial connection success rate, supporting network reliability. Including:
        - PS E-UTRAN RRC Setup Success Ratio (%): Evaluates Radio Resource Control setup success
        - Intra Frequency Handover Success Rate (%): Measures success rate within the same frequency.
        - PS E-UTRAN RAB Setup Success Rate (%): Success rate for establishing Radio Access Bearers.
    - **Cell Availability and Downtime**: Captures cell’s uptime and reliability, derived from:
        - PS E-UTRAN RRC Setup Success Ratio: Establishment success of initial connection requests.
        - Cell Availability: Percentage of time the cell is operational.
- **Output Features (Model Predictions)**:
    - **Predicted Sector**: Identifies the optimal sector ID (A, B, or C) for enhanced performance within a cellular tower.

- **Predicted Site**: Identifies the site ID (1-10) associated with the prediction, helping network engineers focus on specific geographical zones.
- **Predicted Performance (Traffic_Volume_and_Payload_Metrics)**: This is a composite metric combining multiple traffic-related features to predict overall network performance, focusing on throughput and payload capacity, including metrics such as PS Drop Call Rate_DENOM, CSSR eRAB DENOM, Average DL PDCP Throughput (Mbit/s), Average UL PDCP Throughput (Mbit/s), DL User Throughput(Mbps)_DENOM, PcellVolume_DL_MAC, UL Traffic (GB), ERAB Drop Rate_DENOM, Average User Throughput in UpLink (Mbps)_NUM, DL Payload(GB), Total Traffic (GB), and others. This composite feature reflects the ability of the network to manage high traffic and maintain quality, providing a holistic measure of 5G network capacity.

This composite feature allows engineers to gain insight into the network's payload efficiency and potential bottlenecks, supporting informed decisions for optimizing 5G infrastructure performance.

5. Conclusion

This project successfully addressed the challenge of analyzing, predicting, and optimizing 5G network performance using a dataset provided by Ericsson Vietnam. By leveraging advanced AI techniques, we developed a robust framework that empowers network engineers and decision-makers to enhance 5G network quality across geographical zones. The project encompassed comprehensive data processing, feature engineering, model training, and the deployment of a predictive AI demonstrator.

The key achievements include the development of two predictive models: the Zone Prediction Model, using Random Forest, which achieved high accuracy in identifying geographical zones, and the Performance Prediction Model, based on an Artificial Neural Network, which provided accurate predictions for network performance metrics with minimal error. The results demonstrate the effectiveness of feature engineering and model selection in improving prediction accuracy and reliability.

Furthermore, the project delivered a user-friendly AI demonstrator that visualizes predictions, making the insights accessible to non-technical stakeholders. This tool facilitates data-driven decision-making, enabling network engineers to monitor, optimize, and plan 5G infrastructure more efficiently.

In conclusion, the project demonstrates the potential of AI in telecommunications, particularly in the realm of 5G network optimization. By providing actionable insights and predictive capabilities, the developed models and demonstrator pave the way for more resilient and efficient network infrastructures.

6. References

[1] Google. "Feature Analysis" Google Drive. https://drive.google.com/file/d/1PCl_FpMKpiaUGn-DWpWW9qVxfcNLSiwe/view?usp=sharing

[2] LakeFS “data-preprocessing-in-machine-learning". https://lakefs.io/blog/data-preprocessing-in-machine-learning/

7. Appendix

[3] Google Colaboratory. "5G Network Performance Prediction." https://colab.research.google.com/drive/1kvDXMyeGtO75S1gFHM3Sq-FOYrxzaQDV#scrollTo=nBq_CPqybhNt.

[4] Google. "Intermediate data files, final model files, UI and backend (Flask API) Source Codes." Google Drive. https://drive.google.com/drive/folders/13VtUnM-ACXs5D_AVDp2KBFmleyBZB3B5?usp=sharing

[5] Google. "Test Cases (8) for LTE_KPI.csv Dataset Sample Inputs." Google Drive. https://drive.google.com/file/d/1uAlI0o-WYHSh_LTzAfF6eTOoelPAxSb0/view?usp=sharing

[6] Google. "Demonstrator (UI) Instruction Video." Google Drive. https://drive.google.com/file/d/11dI1Rnilu-w0u1QKXERyJHk8alZQVxxB/view?usp=sharing