Abnormal behaviour in cellular networks detection

Gil Jiménez Canellas
Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona (ETSETB)
Universitat Politècnica de Catalunya (UPC)
gil.jimenez@estudiantat.upc.edu

***Abstract*—Detecting abnormal network behaviors for optimal configuration of next-generation cell communication base stations presents a contemporary challenge. Addressing this as a binary classification problem, the author employ a Machine Learning approach. This report delves into a range of established data processing and machine learning techniques, covering data preprocessing, classification solutions, and the comprehensive training process with parameter tuning. Utilizing several ML models result in an f1-score exceeding 95%.**

I. Introduction

This report aims to address a binary classification challenge posed as a competition on the Kaggle InClass platform [1]. We've applied concepts and techniques from our coursework, along with additional methodologies from exploratory data analysis, feature selection, and classification research. All the code for this project can be found at my github [8], or the notebooks [9], and [10].

A critical aspect of cellular network design involves optimizing energy and resources to ensure smooth operation, even during peak traffic periods. Over time, cellular networks have been moving towards dynamic management and configuration to efficiently adapt to varying traffic demands. This approach aims to avoid resource over-provisioning and promote energy savings. Thus, the task at hand is clear: develop a network operator capable of anticipating these fluctuations in user traffic demands for optimized resource management.

To achieve this, we've employed a Machine Learning approach to explore methods for detecting unusual network utilization behaviors that may necessitate base station configuration changes.

II. Feature Analysis

The initial step in data pre-processing involves ensuring that all features are prepared for use, particularly the non-numerical ones: Time and CellName. Time exhibits a significant correlation with peak traffic hours, even though the day is not explicitly stated in the date; it remains a relevant feature. To prepare Time for analysis, various approaches can be considered. However, our chosen method involves converting the time (in minutes) into radians and splitting it into two features—one applying cosine and the other sine transformations.

Regarding CellName, we have opted not to use this variable as it does not show a significant correlation with the results. Therefore, it has not been included in our analysis.

With the entire dataset prepared for analysis, our next step was to explore the data, examining distributions and searching for potential correlations between variables. We employed the combination of a decision tree classifier, a random forest classifier, and adaboost classifier to gain insights into the dataset. This three methods allowed us to identify the important features that significantly contribute to the classification task. Upon analysis, we found that out of the 14 features initially considered, the classifiers were only priorizing very few features. The decision tree selected as most influential the meanUE_UL, PRBUsageUL and PRBUsageUL with 0.75%, 0.12% and 0.022%, respectively.

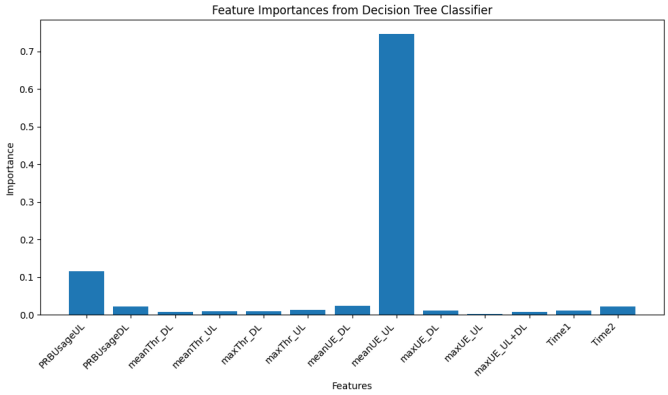

Fig. 1. Decision Tree Feature Importances

While the results from the Random Forest and ADABoost models were not as pronounced, they both highlighted the same three variables as the most important.

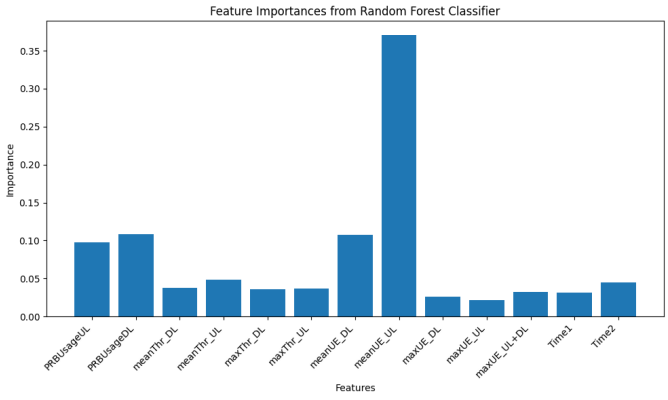

Fig. 2. Random Forest Feature Importances

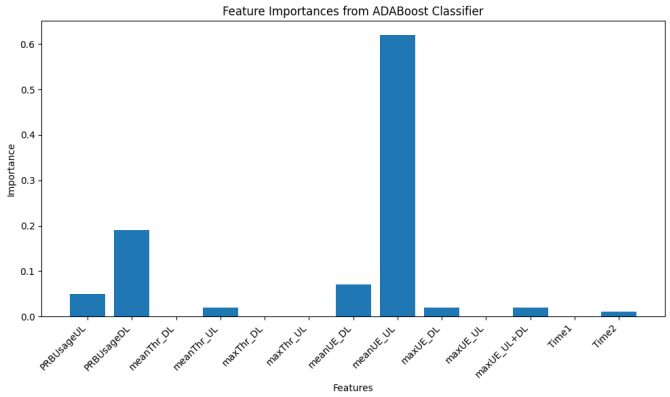

Fig. 3. ADABoost Feature Importances

Given this result, we made the decision to focus our modeling efforts on these three key features. This approach helps streamline the model training process, focusing on the most impactful variables and potentially improving the model's efficiency and interpretability.

III. Classification Methods

In this section, the classification methods proposed to address the classification problem are described. After trying various methods seen in class and in the laboratory, it has been decided to delve deeper into techniques that have yielded better initial results. Throughout the study, the classes provided by the Scikit-learn library [7] in Python have been used.

A. Decision Tree

Decision Trees [2] are a popular type of classifier that recursively partitions the input space into regions based on the feature values. Each internal node represents a feature, each branch represents a decision based on that feature, and each leaf node represents a class label.

The decision at each node is typically made by minimizing impurity or maximizing information gain. Two common impurity measures are Gini impurity and entropy:

- **Gini Impurity**:
  The Gini impurity for a node is calculated as:

$$\text{Gini}(t) = 1 - \sum_{i=1}^{c} (p(i|t))^2$$

  where c is the number of classes, and $p(i|t)$ is the proportion of samples of class i at node t.
- **Entropy**:
  Entropy is another measure of impurity and is defined as:

$$\text{Entropy}(t) = -\sum_{i=1}^{c} p(i|t) \log_2 p(i|t)$$

  where $p(i|t)$ is the same as in Gini impurity.

The decision tree algorithm recursively splits the data based on these measures until a stopping criterion (like a maximum depth or minimum number of samples per leaf) is reached.

B. Random Forest

Random Forest [3] is an ensemble learning method that constructs a multitude of decision trees during training. The final prediction is made by averaging the predictions of each tree (for regression tasks) or taking a majority vote (for classification tasks).

Each tree is trained on a random subset of the data and a random subset of the features. This randomness helps to reduce overfitting and improves the robustness of the model.

The prediction of the Random Forest can be written as:

$$\hat{y} = \text{mode}(f_1(x), f_2(x), ..., f_n(x))$$

where $\hat{y}$ is the predicted class, $f_i(x)$ is the prediction of the i-th tree, and mode is the function that returns the most frequent value.

C. AdaBoost

AdaBoost (Adaptive Boosting) [4] is an ensemble learning method that combines multiple weak learners (often simple decision trees) to create a strong classifier. It sequentially trains a series of weak learners, where each subsequent learner focuses on the mistakes made by the previous ones.

The final prediction of AdaBoost is a weighted sum of the predictions of the weak learners:

$$\hat{y} = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

where $\hat{y}$ is the predicted class, α_t is the weight assigned to the t-th weak learner, and $h_t(x)$ is the prediction of the t-th weak learner.

D. Gaussian NB

Gaussian Naive Bayes [5] is a simple probabilistic classifier based on Bayes' theorem with the assumption of independence between features. It is particularly suited for continuous-valued features.

The prediction of Gaussian Naive Bayes is based on the Gaussian probability density function:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}}$$

where $P(x_i|y)$ is the likelihood of feature x_i given class y, μ_y is the mean of feature x_i for class y, and σ_y^2 is the variance of feature x_i for class y.

The predicted class is the one with the highest posterior probability:

$$\hat{y} = \text{argmax}_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

E. KNeighborsClassifier

K-Nearest Neighbors [6] is a simple yet effective classifier based on instance-based learning. It makes predictions by selecting the majority class among the k nearest neighbors of a new data point.

The prediction is typically done by a majority vote, where the class with the highest frequency among the k neighbors is chosen:

$$\hat{y} = \text{argmax}_y \sum_{i=1}^{k} I(y_i = y)$$

where $\hat{y}$ is the predicted class, y_i are the classes of the k nearest neighbors, and $I(\cdot)$ is the indicator function.

F. Voting

1) Hard Voting: In hard voting, each model in the ensemble makes a class prediction, and the majority class among the models is chosen as the final prediction. This is commonly used for classification tasks.

The final prediction is determined by a simple majority vote among the individual model predictions. In other words, the class with the most votes from the models is selected as the final predicted class.

2) Soft Voting: In soft voting, each model in the ensemble produces a probability score for each class. These probabilities are averaged across all models, and the class with the highest average probability is chosen as the final prediction. This is often used for both classification and regression tasks.

Instead of selecting the class with the most votes, soft voting considers the average probability of each class across all models. The class with the highest average probability is then predicted as the final output.

IV. Evaluation Metrics

When evaluating the performance of our models, we chose to observe three key metrics: Accuracy, Recall, and F1 Score. These metrics provide different perspectives on the model's performance and are particularly relevant for classification tasks.

A. Accuracy

Accuracy is a common metric that measures the proportion of correctly classified samples out of the total number of samples. It provides an overall assessment of the model's correctness.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Samples}}$$

B. Recall (Sensitivity)

Recall, also known as Sensitivity or True Positive Rate (TPR), measures the ability of the model to correctly identify positive samples (correctly classified true positives) out of all actual positive samples.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

C. F1 Score

The F1 Score is the harmonic mean of precision and recall. It provides a balance between precision and recall, where higher values indicate better performance.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

D. Confusion Matrix

The confusion matrix is a table that visualizes the performance of a classification model. It presents a summary of the actual and predicted class labels, showing the counts of true positives, true negatives, false positives, and false negatives.

	Predicted Negative	Predicted Positive
Actual Negative	TN	FP
Actual Positive	FN	TP

- True Positives (TP): Samples correctly predicted as positive.

- True Negatives (TN): Samples correctly predicted as negative.

- False Positives (FP): Negative samples incorrectly predicted as positive.

- False Negatives (FN): Positive samples incorrectly predicted as negative.

V. Experiments and Results

A. Split the train dataset

In our experiments, we first split the training data into training and validation sets. This allowed us to train our models on the training set and evaluate their performance on the validation set to avoid overfitting.

B. Class imbalance

We observed that the train dataset was unbalanced, i.e., with a significant disparity in the number of rows between 'usual' and 'unusual' categories. The balancing helps prevent the model from being biased towards the majority class. After implementing this approach on the different models, the validation results showed improvement.

C. Hyperparameter Tuning

For hyperparameter tuning, we employed a grid search approach for every model in our study. Grid search is a technique used to find the optimal hyperparameters for a model by searching through a specified grid of parameter values. This systematic search allows us to evaluate the model's performance with different combinations of hyperparameters and select the ones that yield the best results.

D. Results

After analyzing the results, we found that the Decision Tree, Random Forest, and AdaBoost models performed favorably compared to the others. These models showed promising accuracy, recall, and F1 Score.

Given the favorable results obtained from the Decision Tree, Random Forest, and AdaBoost models, our final approach

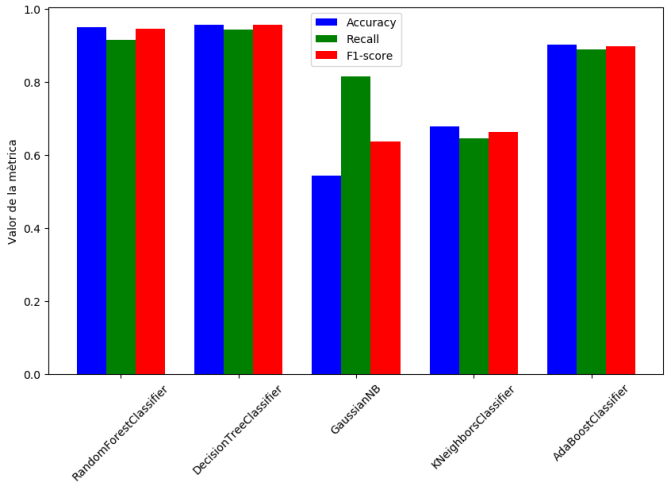

Fig. 4. Comparison of models

will combine these models using a soft voting. This approach leverages the strengths of each individual model to improve overall prediction accuracy.

This are our validation results:

Accuracy: 0.979317196531792
F1-score: 0.961596511822908
Recall: 0.9372343903236352
Confusion Matrix:

$$\begin{bmatrix} 7976 & 37 \\ 192 & 2867 \end{bmatrix}$$

Whereas in the Kaggle platform we perform a 0.95202% in the private score and 0.96181% in the public score, which result in the second place overall.

VI. Conclusion

In conclusion, our study aimed to address the challenge of detecting abnormal behaviors in network utilization for optimizing the configuration of base stations in next-generation cell communications. We explored various machine learning models including Decision Tree, Random Forest, AdaBoost, Gaussian Naive Bayes, and K-Nearest Neighbors.

Our experiments revealed promising results, with the Decision Tree, Random Forest, and AdaBoost models showing favorable performance. Through hyperparameter tuning using grid search, we optimized these models for improved accuracy, F1-score, and recall.

However, despite these positive outcomes, it is evident that there is room for improvement. The results, while not bad, could be enhanced further to achieve even higher accuracy and robustness. Future work could focus on refining the feature selection process, exploring more sophisticated ensemble techniques, and gathering additional data for training.

Overall, this study provides valuable insights into the application of machine learning for network optimization. By continuing to refine and innovate in this field, we can contribute to the development of more efficient and adaptive cellular networks for the future.

References

[1] Kaggle Platform
[2] Decision Trees Wikipedia
[3] Random Forest Wikipedia
[4] AdaBoost Classifier Wikipedia
[5] Gausian Naive Bayes Wikipedia
[6] KNeighbors Classifier Sklearn
[7] Sklearn
[8] GitHub Gil Jiménez Canellas
[9] Predictor Notebook
[10] Data Analysis Notebook