ML-Based System-Level Parameter Estimation of Frequency Selective Surfaces (FSS) in Sub-6 GHz Bands

Random Forest Regression with Physics-Aware Synthetic Data

Author

Jay Gautam

February 2026

Conceptual Scope

System-level machine learning for RF structures using physics-consistent synthetic data, interpretability analysis, and parametric validation.

Content

About Me

Who Am I?

I am a wireless and RF engineer working on 5G and 6G technologies, with hands-on experience in Reconfigurable Intelligent Surfaces (RIS), metasurfaces, antennas, and wireless system modeling. Over time, my work has naturally evolved toward AI-driven methods for wireless communication, while remaining strongly grounded in physical modeling and engineering principles.

Today, I actively work with machine learning, deep learning, and reinforcement learning, applying these techniques to wireless communication and 6G-oriented system-level problems.

My Introduction to AI in Engineering:

In 2021, just before starting my master's thesis at TU Darmstadt, Germany, I made a deliberate decision to transition toward AI-based engineering. At that time, I had limited programming experience and no formal background in machine learning or deep learning.

Despite this, I selected a machine-learning-based research topic in the RF and microwave domain for my master's thesis, completed it successfully, and continued building my expertise step by step through self-driven learning. From the beginning, I consciously aligned my AI learning with wireless communication, RF systems, and physical modeling.

Current Technical Focus:

I am currently comfortable designing and implementing ML, DL, and RL algorithms for wireless and 6G-related problems. My practical experience includes:

- Supervised and unsupervised learning
- Deep learning–based surrogate modeling
- Reinforcement learning methods, including: DQN, PPO (Discrete and Continuous), DDPG, TD3, SAC

My Future Research Direction: my future focus is on AI-driven implementations aligned with the 6G vision, especially:

- Reinforcement learning–based AI control for RIS and wireless systems
- Autonomous and adaptive 6G environments
- Sustainable and intelligent resource management
- Energy-aware AI methods for wireless networks

1. ML- Driven Implementation: Random Forest

Section 1: Problem statement and Brief explanation-

Problem statement:

Full-wave electromagnetic simulation of Frequency Selective Surfaces (FSS) is computationally intensive for large parametric studies. This work formulates FSS response prediction as a supervised regression problem and applies a system-level, physics-aware machine learning approach for efficient parameter estimation.

Brief Explanation:

Frequency Selective Surfaces (FSS) are periodic electromagnetic structures whose transmission/reflection characteristics depend on geometry, material properties, and operating frequency.

In practical RF and wireless system design, evaluating/modelling FSS using full wave electromagnetic simulations tools can be computationally expensive throughout the design life cycle.

Objective of This Work:

Develop a physics-aware ML regression model (Random Forest Regression) that learns the relationship among: FSS design parameters (geometry + material) and key electromagnetic response metrics, using synthetically generated, physics-consistent data.

ML Elements:

Model: Random Forest Regression

Input features (X): Each FSS configuration is described by the following parameters:

Feature	Description
d_mm	FSS element size (mm)
p_mm	Unit-cell periodicity (mm)
w_mm	Metallic strip width (mm)
eps_r	Substrate relative permittivity
f_center_ghz	Operating / system center frequency (GHz)

Target variables (y): For each FSS configuration, the model predicts three key response metrics

Target	Description
f_res_ghz	Resonance frequency (GHz)
BW_10dB_ghz	−10 dB bandwidth (GHz)
S21_min_dB	Minimum transmission magnitude (dB)

ML Model: $y=f(x)$

Where,

$x=[d,p,w,\varepsilon r,fcenter]$

$y=[fres,BW10dB,|S21|min]$

f() is an unknown nonlinear function approximated using a Random Forest regressor

Section 2: Import Libraries

```
# Libraries/Imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

## SKlearn
from sklearn.model_selection import train_test_split # train
test split
from sklearn.ensemble import RandomForestRegressor # regressor
from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score # Metrics

## reproducibility
np.random.seed(42)
```

Section 3: FSS-physics-aware Synthetic data generation

The Dataset is generated (trend based heuristic approach) as an alternative to commercial simulations tools (full-wave electromagnetic solvers).

This data is not intended to reproduce exact EM responses, but to preserve dominant physical relationships between geometry, material properties, and frequency-domain behavior.

The current synthetic data can be safely replaced by data generated from commercially available software/tools without changing the ML tools.

The expressions of "L_eff, f_res, S21_min" are not exact analytical formulas, rather they capture the general trend of FSS design based frequency response in alignment with general antenna theory. This specific formulation is adapted based on several literature and text book study.

```
# 3. FSS-physics-aware- heurristic approach- Synthetic data
generation
## FSS: A simple square loop with dielectric substrate.

def generate_synthetic_fss_data(n_samples=1500):
    # FSS Geometry & material parameters
    d = np.random.uniform(5.5, 18.0, n_samples) # FSS element
size, mm
```

```
    p = np.random.uniform(7.0, 24.0, n_samples) # FSS
periodicity, mm
    w = np.random.uniform(0.2, 5.0, n_samples)     # metallic
strip width, mm
    eps_r = np.random.uniform(2.2, 6.0, n_samples)    #
substrate permittivity
    f_center = np.random.uniform(2.0, 6.0, n_samples)   #
system frequency in GHz

    # Effective permittivity
    eps_eff = 0.5 * (1 + eps_r)

    # Effective electrical length- to capture the general
trends based on antenna theory.
    L_eff = d + 0.15 * p

    # Resonance frequency: Antenna theory- general trend.
    f_res = 110 / (L_eff * np.sqrt(eps_eff))
    f_res += np.random.normal(0, 0.03, n_samples)  # Include
small modeling noise

    # Minimum S21 (stronger metal & coupling → deeper
stopband)
    S21_min = -(10 + 120 * (w / p) / np.sqrt(eps_r))#
Qualitatie appriach. Not exact analytical.
    S21_min = np.clip(S21_min, -40, -5)
    S21_min += np.random.normal(0, 0.5, n_samples)

    # Bandwidth definition (-10 dB), depending on width, size,
periodicity, permittivity
    BW_10dB = (0.25 + 0.8 * (w / d) + 0.3 / (p / 10)) /
np.sqrt(eps_r) # Qualitative approach
    BW_10dB = np.clip(BW_10dB, 0.1, 2.0) # Cliping to limit.

    # Feature variables- Data Frame.
    X = pd.DataFrame({
        "d_mm": d,
        "p_mm": p,
        "w_mm": w,
```

```
        "eps_r": eps_r,
        "f_center_ghz": f_center
    })

    # Target variables- Data Frame.
    y = pd.DataFrame({
        "f_res_ghz": f_res,
        "BW_10dB_ghz": BW_10dB,
        "S21_min_dB": S21_min
    })

    return X, y
```

Data: Format of data shown in the next section.

Section 4: Pandas data frame dataset: .CSV format

```
# Pandas dataframe dataset: .CSV format
# Call the function to generate the data with 2000 rows, 8
columns
X, y = generate_synthetic_fss_data()

dataset = pd.concat([X, y], axis=1) # to make proper sheet
contaning both X and y
dataset.to_csv("synthetic_fss_dataset.csv", index=False) #
.csv format

print("dataset, file name: synthetic_fss_dataset.csv")
print("Dataset shape:", dataset.shape)
display(dataset.head())
```

Data

```
dataset.head()
```

```
dataset, file name: synthetic_fss_dataset.csv
Dataset shape: (2000, 8)
```

	d_mm	p_mm	w_mm	eps_r	f_center_ghz	f_res_ghz	BW_10dB_ghz	S21_min_dB
0	8.799897	22.600030	1.938915	5.660615	3.135371	4.945017	0.234957	-14.408954
1	14.092727	23.261028	4.949014	3.949996	4.972527	4.015795	0.332037	-22.432962
2	9.473850	9.355790	0.919102	2.941586	5.542369	7.198524	0.377976	-16.847976
3	11.823200	10.914010	3.290872	3.368059	2.499408	5.569063	0.407333	-30.401446
4	9.764147	17.066052	0.366884	5.491643	4.343729	4.950638	0.194522	-11.278652

Section 5: Data Preprocessing- Train/Test split

Before training the machine learning model, the dataset is divided into training and testing subsets. I have decided not to use validation data- just for the simplicity.

Training set (80%): Used to train the Random Forest model and learn the mapping between FSS design parameters and electromagnetic response metrics.

Test set (20%): Used for evaluation, ensuring that model performance is assessed on previously unseen FSS configurations.

```
# 5. Data Preprocessing- Train/Test split
## 80% trainng. 20% test data.
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
X_train.shape, X_test.shape, y_train.shape, y_test.shape

# ((1600, 5), (400, 5), (1600, 3), (400, 3))
```

Section 6: Train Random Forest Model

A Random Forest Regressor is chosen as the baseline machine learning model due to its-ability to model nonlinear relationships:

- Robustness to noise in synthetic data,
- Good performance with limited hyperparameter tuning
- Built-in interpretability via feature importance

Model Configuration:
n_estimators = 300: Number of decision trees in the ensemble. This value is chosen for stable predictions without excessive complexity.
random_state = 42: reproducibility.

n_jobs = -1: Enables parallel computation using all available CPU cores for faster training.

```
# 6. Train Random Forest Model

rf = RandomForestRegressor(n_estimators=300, random_state=42,
n_jobs=-1) # object
rf.fit(X_train, y_train) # Training
```

Section 7: Predictions and standard ML metrics

Evaluation Metrics: To assess model performance, I used 3 regression metrics for each target variable: MAE, RMSE, R^2

Mean Absolute Error (MAE):

Measures the average absolute prediction error, in physical units (GHz or dB)

Directly interpretable from an RF/system perspective (e.g., average resonance frequency error in GHz)

provides the most intuitive physical interpretation

Root Mean Squared Error (RMSE):

Penalizes larger prediction errors

Highlights the presence of outliers

Useful for assessing model robustness

reflects reliability and sensitivity to large errors

Coefficient of Determination (R^2):

Indicates how well the model captures variance and trends

Used as a supporting metric, not as the primary indicator of accuracy

confirms whether the dominant trends are learned

```
# 7. Predictions and standard ML metrics

y_pred = rf.predict(X_test) # Prediction on Test data (20%)

# Evaluation Metrics based on predicted values on test data.
```

```
print("\nEvaluation Metrics (Test Set):")
for i, target in enumerate(y.columns):
    mae = mean_absolute_error(y_test.iloc[:, i], y_pred[:, i])
# MAE
    rmse = np.sqrt(mean_squared_error(y_test.iloc[:, i],
y_pred[:, i])) # RMSE
    r2 = r2_score(y_test.iloc[:, i], y_pred[:, i]) # R2
    print(f"{target:15s} | MAE={mae:.3f}, RMSE={rmse:.3f},
R2={r2:.3f}")
```

Result:

```
Evaluation Metrics (Test Set):
f_res_ghz       | MAE=0.309, RMSE=0.426, R2=0.936
BW_10dB_ghz     | MAE=0.015, RMSE=0.021, R2=0.952
S21_min_dB      | MAE=0.718, RMSE=0.968, R2=0.984
```

Section 8: Prediction vs Ground Truth

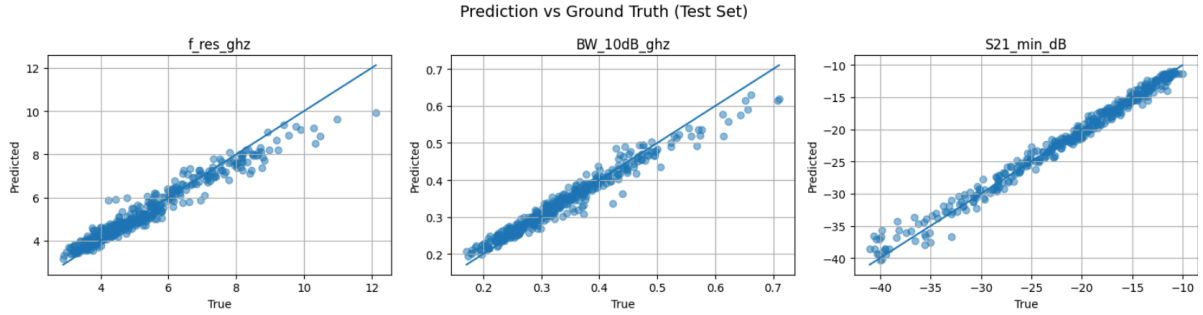
Prediction vs ground truth plots are used to visually assess model performance for each target variables: X-axis → True (ground truth) values, Y-axis → Model-predicted values

The three subplots correspond to: f_res_ghz (resonance frequency), BW_10dB_ghz (−10 dB bandwidth), S21_min_dB (minimum transmission)

```
# 8. Prediction values vs Ground Truth values
## Plot for Prdicted vs Trze value for each Targets:
f_res_ghz, BW_10dB_ghz, S21_min_dB
plt.figure(figsize=(15,4))

for i, target in enumerate(y.columns):
    plt.subplot(1, 3, i+1)
    plt.scatter(y_test.iloc[:, i], y_pred[:, i], alpha=0.5)
    plt.plot(
        [y_test.iloc[:, i].min(), y_test.iloc[:, i].max()],
        [y_test.iloc[:, i].min(), y_test.iloc[:, i].max()]
    )
    plt.xlabel("True")
    plt.ylabel("Predicted")
    plt.title(target)
```

```
    plt.grid(True)

plt.suptitle("Prediction vs Ground Truth (Test Set)",
fontsize=14)
plt.tight_layout()
plt.show()
```

Figure:



Resonance Frequency (f_res_ghz):

Observation:

Data points are tightly clustered around the diagonal

Slight dispersion increases at higher resonance frequencies

Interpretation:

The model successfully learns the dominant electrical length scaling of the FSS

Small deviations at higher frequencies are expected due to: increased nonlinearity, lower data density at extreme parameter combinations

The ML model reliably predicts resonance frequency across the design space, with physically tolerable error behavior.

−10 dB Bandwidth (BW_10dB_ghz):

Observation:

Very tight alignment with the diagonal

Minimal spread across the entire bandwidth range

Interpretation:

Bandwidth relationships w.r.t model parameetrs are well captured by the Random Forest model

The near-linear alignment indicates stable generalization

Bandwidth prediction shows the highest consistency among all targets, indicating that the model captures effects effectively.

Minimum Transmission (S21_min_dB):

Observation:

Strong linear trend across the full dynamic range

Slightly larger spread at deeper attenuation levels (more negative dB)

Interpretation:

The model learns the correlation

Increased spread at deep stopbands reflects: higher sensitivity of S21 to geometry, added noise in synthetic data generation

Even for logarithmic quantities like S21 (dB scale), the model preserves correct physical trends and magnitude ordering.

The prediction–ground truth plots confirm that the ML model learns physically meaningful relationships between FSS design parameters and frequency-domain responses.

Section 9: Feature importance: Interpretability

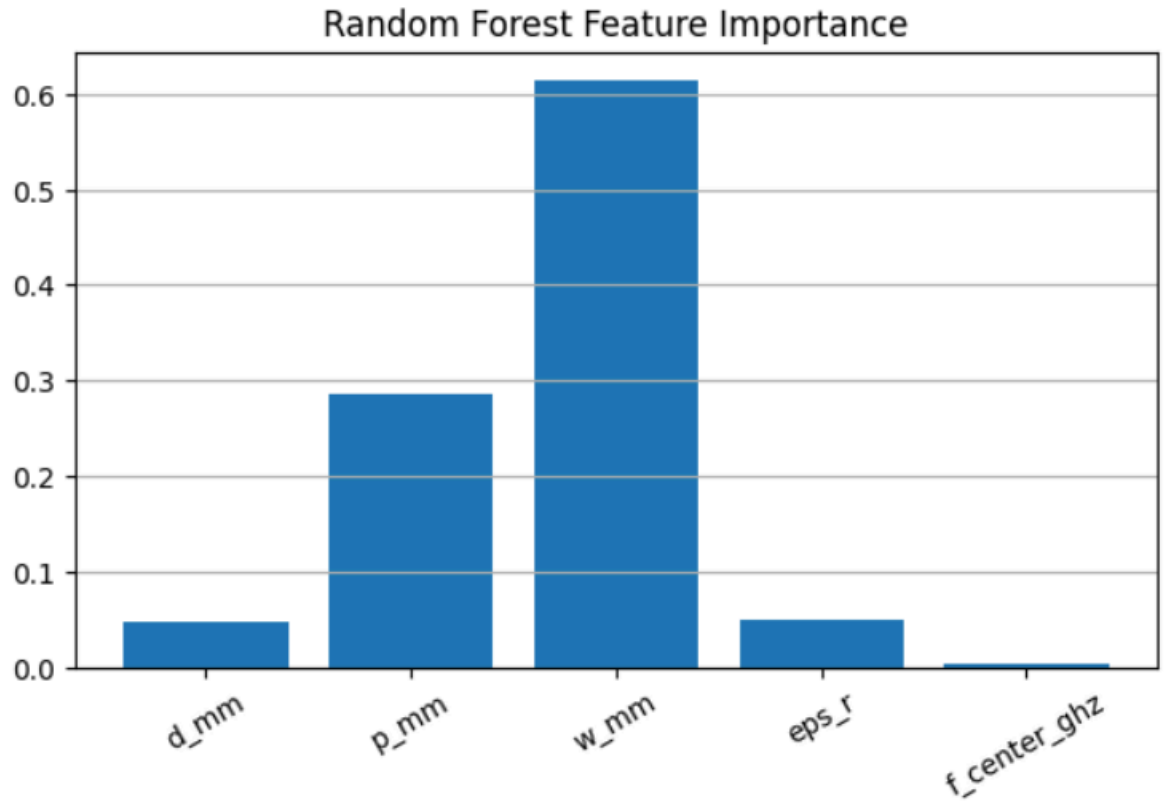
The feature importance plot provides insight into: which FSS design parameters dominate model decisions:

- Strip Width (w_mm) – Dominant Feature
- Periodicity (p_mm) – Second Most Important
- Element Size (d_mm) – Moderate Importance
- Substrate Permittivity (eps_r) – Lower but Meaningful Influence
- Center Frequency (f_center_ghz) – Minimal Importance

```
# 9. Feature importance: Interpretability
## To understand which features has higher impact on the result..
plt.figure(figsize=(7,4))
```

```
plt.bar(X.columns, rf.feature_importances_) # Feature
importance check.
plt.title("Random Forest Feature Importance")
plt.xticks(rotation=30)
plt.grid(axis="y")
plt.show()
```

Figure:



Section 10: Dependence of Target variables with Features

To verify that the trained ML model has learned physically meaningful relationships, parametric sweeps are performed for key design parameters (d, w and eps_r) while keeping all other parameters fixed. The target variables are evaluated as functions of:

- Element size (d)
- Strip width (w)
- Substrate permittivity (eps_r)

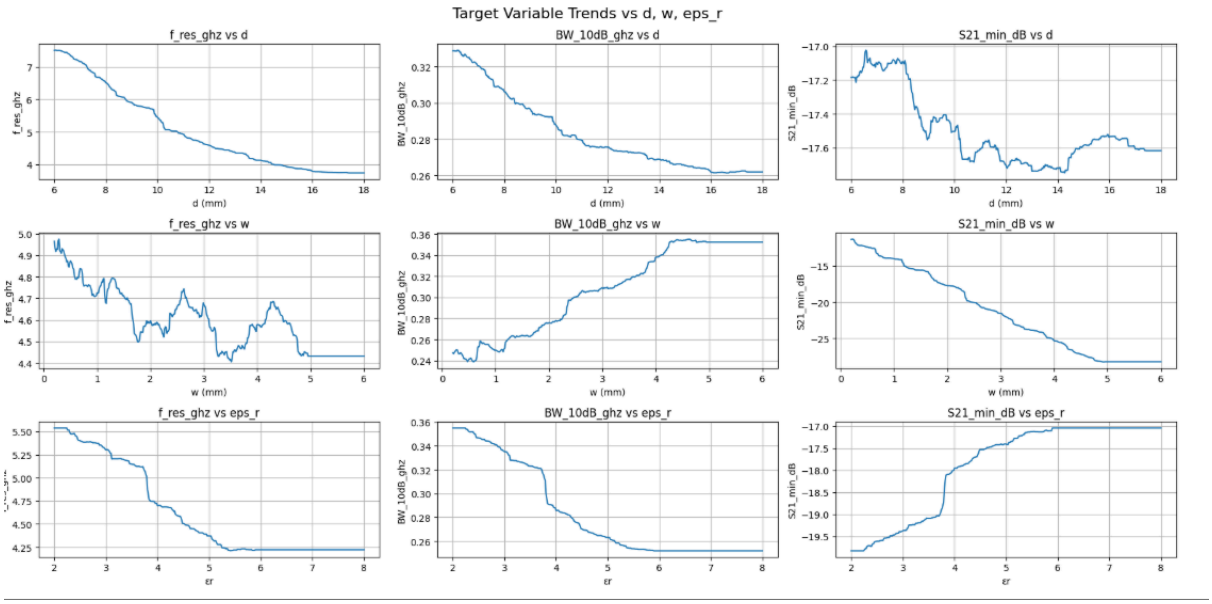
This analysis serves as a physics sanity check, ensuring that model predictions follow expected RF trends.

```
# 10. Dependance of Target variable with Features
```

```
## Check how the target value varies with the variations in
the design parameters.
## Perfomed w.r.t. three design parameeters: d, w and
permitivity eps_r: for each target variables.
targets = y.columns.tolist()
# element size d
d_sweep = np.linspace(6, 18, 400) # 400 samples.
X_d = pd.DataFrame({
    "d_mm": d_sweep,
    "p_mm": 15.0,
    "w_mm": 2.0,
    "eps_r": 4.4,
    "f_center_ghz": 3.5
})
y_d = rf.predict(X_d) # prediction on the amples.

# Strip width w
w_sweep = np.linspace(0.2, 6.0, 400) # 400 samples.
X_w = pd.DataFrame({
    "d_mm": 12.0,
    "p_mm": 15.0,
    "w_mm": w_sweep,
    "eps_r": 4.4,
    "f_center_ghz": 3.5
})
y_w = rf.predict(X_w) # prediction on the amples.

# Substrate permittivity eps_r
eps_sweep = np.linspace(2.0, 8.0, 400) # 400 samples.
X_eps = pd.DataFrame({
    "d_mm": 12.0,
    "p_mm": 15.0,
    "w_mm": 2.0,
    "eps_r": eps_sweep,
    "f_center_ghz": 3.5
})
y_eps = rf.predict(X_eps) # prediction on the amples.

# Plot all
```

```
plt.figure(figsize=(18,9))

for i, target in enumerate(targets):
    plt.subplot(3, 3, i+1)
    plt.plot(d_sweep, y_d[:, i]) # d
    plt.xlabel("d (mm)")
    plt.ylabel(target)
    plt.title(f"{target} vs d")
    plt.grid(True)

    plt.subplot(3, 3, i+4)
    plt.plot(w_sweep, y_w[:, i]) # w
    plt.xlabel("w (mm)")
    plt.ylabel(target)
    plt.title(f"{target} vs w")
    plt.grid(True)

    plt.subplot(3, 3, i+7)
    plt.plot(eps_sweep, y_eps[:, i]) # eps
    plt.xlabel("εr")
    plt.ylabel(target)
    plt.title(f"{target} vs eps_r")
    plt.grid(True)

plt.suptitle("Target Variable Trends vs d, w, eps_r",
fontsize=16)
plt.tight_layout()
plt.show()
```

Figure:



Dependence on Element Size (d):

Observed trends:

f_res_ghz decreases monotonically with increasing d

BW_10dB_ghz shows a gradual reduction

S21_min_dB varies smoothly with minor fluctuations

Interpretation:

Larger FSS elements increase the effective electrical length, shifting resonance to lower frequencies

Bandwidth reduction with size is consistent with resonance sharpening in electrically larger structures

Dependence on Strip Width (w):

Observed trends:

BW_10dB_ghz increases with increasing w

S21_min_dB becomes more negative

f_res_ghz shows less dependence on w

Dependence on Substrate Permittivity (eps_r):

Observed trends:

f_res_ghz decreases with increasing εr

BW_10dB_ghz reduces as dielectric loading increases

S21_min_dB becomes less negative with higher εr

Interpretation:

Higher permittivity increases effective electrical length, lowering resonance frequency

Increased field confinement leads to narrower bandwidth

Coupling behavior changes smoothly with dielectric loading

Further scopes:

- To apply more advanced algorithms such as: ANN, DNN and Reinforcement learning algorithms.
- Rigorous analysis for error and uncertainty estimations

Resource & Links

GitHub (code, implementations, updates):
https://github.com/jskgautam

LinkedIn (professional updates and discussions):
https://www.linkedin.com/in/jay-gautam-203362a2/recent-activity/shares/

ML Tools and libraries, FSS references.

Scikit-learn documentation: https://scikit-learn.org

NumPy documentation: https://numpy.org

Pandas documentation: https://pandas.pydata.org

General references on Frequency Selective Surfaces and antenna theory

Disclaimer

This work is an independent technical effort, developed using my own programming and AI skills. The implementation, experiments, and analysis presented here are entirely my own and are done using my own resources.

Theoretical understanding and practical insights have naturally been shaped by many publicly available research papers, courses, books, and discussions, as is common in academic and engineering practice. No proprietary code or confidential material has been used.

This document is shared for learning, discussion, and open technical exchange. It is not intended to claim ownership of any foundational theories or prior research.

In the future, for any of my content, if I use resources from others, I will certainly give proper credit. If any content raises concerns or objections, I kindly request that you contact me directly via LinkedIn, so the matter can be discussed and addressed constructively.

Regarding synthetic data:
The synthetic dataset used in this work is generated using heuristic, physics-inspired formulations and is not intended to reproduce exact electromagnetic results.

The presented results should not be interpreted as a replacement for full-wave electromagnetic simulations. The primary purpose of this work is to study machine learning behavior at the RF system level.

All models and data are provided for educational purposes.