**Attempt 01:**

We implemented a machine learning pipeline that combines **Principal Component Analysis (PCA)** for dimensionality reduction and an **Artificial Neural Network (ANN)** for regression to predict Path Loss (dB) based on various geographic and environmental factors. PCA (Principal Component Analysis) is a dimensionality reduction technique that transforms correlated features into a smaller set of uncorrelated principal components, preserving maximum variance. It helps in noise reduction and improving computational efficiency. On the other hand, ANN (Artificial Neural Networks) are inspired by the human brain and are used for classification and regression tasks. They consist of layers of neurons that process inputs through weighted connections, activation functions, and backpropagation to minimize errors. While PCA simplifies data representation, ANN learns complex patterns to make predictions.

However, this endeavor did not yield much success. In the end, our mean squared error was 78.35.

**Attempt 02:**

Our updated code improves the ANN model for Path Loss (dB) prediction by incorporating regularization, dropout, and early stopping, which help prevent overfitting and improve generalization.

**Key Improvements:**

* **More Robust Against Overfitting:** Regularization (L2), dropout layers, and early stopping help improve generalization.
* **More Training Control:** Early stopping prevents unnecessary epochs while allowing up to 100 if needed.
* **Better Performance Analysis:** Added scatter plot to compare actual vs. predicted values.

Overall, the updated ANN model is more optimized, stable, and generalizable.

Due to these modifications, the mean squared error decreased to 64.12.

**Attempt 03:**

This time, we introduced major improvements by integrating **ANN, Random Forest** anda **Voting Regressor** alongside **PCA**, making it a hybrid machine learning approach.

**Key Enhancements & Benefits**

**More Robust Model**:

Instead of relying solely on ANN, we can now compare and combine it with Random Forest, leveraging both deep learning and traditional ML.

**Better Generalization**:

* Random Forest captures non-linear relationships without requiring much tuning.
* ANN is optimized using ReduceLROnPlateau, dynamically adjusting learning rates.
* Voting Regressor combines strengths of both models, reducing bias and variance.

**More Flexible ANN Architecture**:

* The create\_model() function allows quick tuning of neurons, dropout, and optimizer, making it more adaptable.

**More Insights from Visualizations**:

* Scatter plot for Actual vs. Predicted values of all models helps compare performance.
* Learning curve of ANN helps assess training behavior.

This new model is significantly more advanced and optimized for real-world predictive tasks.

While comparing the Random Forest model to the ANN model, we clearly observed that the Random Forest model performs better. The Random Forest Model produced an 8.63 mean squared error, whereas the Best ANN model produced a 93.45 mean squared error.

**Attempt 04:**

Our new implementation removes the **Artificial Neural Network (ANN)** and ensemble learning, leaving only the **Random Forest Regressor** (RF) as the primary model.

**Key Takeaways from the New Model**

**Simplified Approach**:

* No deep learning (ANN) means faster training and evaluation.
* Easier to tune hyperparameters (e.g., n\_estimators, max\_depth, etc.).

**Only Uses Random Forest**:

* Random Forest works well for structured tabular data.
* Doesn't require extensive hyperparameter tuning like ANN.
* No need for feature scaling.

This resulted in a slight less error than before.

**Attempt 05:**

Our new Random Forest model has significant changes and improvements over the previous version. We have significantly expanded our feature set, added categorical encoding, and adjusted the PCA components to match the increased input dimensions. We worked with a much larger synthetic dataset this time, which is a great step toward dealing with real data.

**Comparative Analysis:**

| **Aspect** | **Previous Code** | **Updated Code** |
| --- | --- | --- |
| **Dataset** | CSV file (sample1.csv) | Excel file (Processed\_Satellite\_Data.xlsx) |
| **Number of Features** | 6 | 26+ features (including environmental and satellite parameters) |
| **Categorical Encoding** | Not included | One-hot encoding for categorical features |
| **PCA Components** | 6 | 27 (to match the expanded feature space) |

**Key Takeaways:**

**More features →** Potentially higher accuracy  
**Categorical encoding →** Better handling of non-numeric features  
**More PCA components →** Retains more variance

This time the efficiency stood around 73%.

**Attempt 06:**

Our latest model introduces **XGBoost (XGBRegressor)** alongside **Random Forest (RF)**.

**Comparative Analysis: Random Forest vs. XGBoost Model**

| **Feature** | **Previous Model (RF only)** | **Current Model (RF + XGBoost)** | **Advancements** |
| --- | --- | --- | --- |
| **Model Types** | Random Forest (RF) | RF + XGBoost (XGBRegressor) | Added Gradient Boosting Model |
| **RF Hyperparameters** | n\_estimators=100 | Same | No change |
| **XGBoost Added** | Not present | n\_estimators=100, random\_state=42 | New model for boosting performance |
| **Performance Metrics** | Only RF results | RF vs. XGB comparison | Can compare performance |
| **Visualizations** | RF predictions only | Separate plots for RF & XGB | Clear model comparison |
| **Computational Cost** | Moderate | Higher (XGB takes longer than RF) | XGB requires tuning |

**Key Takeaways:**

**XGBoost Introduced**: More advanced boosting model to improve predictive accuracy  
**Direct Model Comparison**: Helps determine if XGBoost outperforms RF  
**Same Preprocessing Pipeline**: Ensures a fair comparison

Here we witnessed that XGBoost beats RF Model in terms of accuracy. XGBoost offers 82% accuracy , where RF offers 73%.

**Attempt 07:**

Our updated Current Model introduces Optimized XGB with Stacking.

**Comparative Analysis:**

| **Aspect** | **Previous Code** | **Updated Code** |
| --- | --- | --- |
| **Models Used** | Random Forest, XGBoost | Random Forest, Optimized XGBoost, Stacking Regressor |
| **Hyperparameter Tuning** | None | Added RandomizedSearchCV for XGBoost |
| **Feature Importance** | Not included | Visualized feature importance (with/without PCA) |
| **Dimensionality Reduction** | Fixed PCA component count | Dynamic PCA component count based on feature availability |
| **Ensemble Learning** | Independent models | Stacked ensemble with Linear Regression as meta-learner |
| **Result Visualization** | Separate plots for each model | Visual comparisons for optimized and stacked models |

**What This Model Does Better**

**Optimized XGBoost:** RandomizedSearchCV improves learning rate, depth, estimators  
**Feature Importance Analysis:** Helps understand which variables drive predictions  
**Stacking Regression:** Combines Random Forest + XGBoost, leading to higher accuracy

**Performance Comparison**

| **Model** | **Mean Squared Error (MSE)** | **R² Score** |
| --- | --- | --- |
| **Initial XGBoost** | {mse\_xgb} | {r2\_xgb} |
| **Optimized XGBoost** | {mse\_best\_xgb} | {r2\_best\_xgb} |
| **Stacked Model (RF + XGB + Linear Regression)** | {mse\_stacked} | {r2\_stacked} |

The initial XGB Model used to offer 89.62% accuracy , whereas this model exceeds it and stands at 90.62% accuracy.

**Attempt 08:**

This time we have gone back to Random Forest Model with some noticeable changes. The model has been significantly enhanced through several key optimizations. Initially, a standard Random Forest model using raw features and default settings was employed. The current optimized model now incorporates one-hot encoding for categorical data, tuned hyperparameters via RandomizedSearchCV, and feature importance visualization, resulting in improved accuracy and a better understanding of key drivers, while maintaining standard performance tracking with MSE and R².

**Comparative Analysis:**

| **Aspect** | **Previous Code (XGBoost + Stacking)** | **Current Code (Optimized Random Forest)** |
| --- | --- | --- |
| **Model Types** | XGBoost, Random Forest, Stacking | Random Forest only |
| **Optimization** | RandomizedSearchCV for XGBoost | RandomizedSearchCV for Random Forest |
| **Dimensionality Reduction** | PCA applied | No PCA (uses all features with feature importance) |
| **Feature Importance** | Based on PCA components | Based on original features (more interpretable) |
| **Model Complexity** | Higher (ensemble stacking) | Lower (single optimized model) |
| **Training Speed** | Slower (due to stacking and XGBoost) | Faster (Random Forest is quicker to train) |
| **Result Visualization** | Plots for both models (XGBoost + Stacking) | Single plot for Random Forest predictions |

**What This Model Does Well:**

**RandomizedSearchCV Optimization:** Tunes key hyperparameters.  
**Feature Importance Analysis:** Identifies the most significant predictors.  
**Visual Performance Evaluation:** Scatter plot for actual vs. predicted values.

**Performance Comparison**

| **Model** | **Mean Squared Error (MSE)** | **R² Score** |
| --- | --- | --- |
| **Initial RF Model** | {mse\_rf} | {r2\_rf} |
| **Optimized RF Model** | {mse\_best\_rf} | {r2\_best\_rf} |

The optimized RF model should outperform the initial RF model in terms of lower MSE and higher R².

This optimized RF model has shown an impressive 91.36% accuracy.