Score out of 10	**9**

Strengths & Highlights:
Project Structure: Excellent organization with clear sections (Know Your Data, Data Wrangling, Visualization, Modeling, etc.) following a logical workflow.

Code Quality: Clean, well-commented, and production-ready code. Proper use of functions and standard libraries (pandas, sklearn, matplotlib, seaborn).

Data Understanding: Thorough initial analysis including missing values, duplicates, data types, and unique value checks for all datasets.

Data Cleaning & Preprocessing: Robust handling of missing values, outliers, categorical encoding, and textual data preprocessing. Justification of techniques used is clear and appropriate.

Visualization: A wide variety of charts (15+) with clear explanations for chart choice, insights found, and business impact. Both univariate and multivariate analysis are covered.

Hypothesis Testing: Well-defined hypotheses and appropriate use of statistical tests (Pearson correlation, ANOVA) with clear interpretations.

Feature Engineering: Meaningful new features created (e.g., DOSE_RATIO, DOSE_GAP) and effective feature selection methods applied.

Model Implementation: Multiple models tested (Linear Regression, Random Forest, Gradient Boosting). Good use of hyperparameter tuning (GridSearchCV, RandomizedSearchCV).

Model Evaluation: Comprehensive evaluation using MSE, RMSE, and R^2. Clear comparison across models.

Explainability: Use of SHAP and feature importance plots to interpret model predictions—a very good practice.

Deployment Readiness: Model saving (joblib), prediction on unseen data, and directory management show preparation for production.

Documentation: Detailed comments and Markdown explanations make the notebook easy to follow and reproduce.

Areas for Improvement & Suggestions:
Target Variable Leakage: Ensure that features like DOSES are not used to predict COVERAGE if COVERAGE is derived from DOSES/TARGET_NUMBER. This can inflate model performance artificially.

Class Imbalance Handling: Although you mentioned it, if COVERAGE was binned into classes for classification, SMOTE/undersampling should be implemented. For regression, target transformation (e.g., log) could help if skewed.

Cross-Validation: Always use cross-validation (e.g., cross_val_score) for more reliable performance estimation, not just a single train-test split.

Hyperparameter Tuning Scope: For Gradient Boosting, consider tuning subsample, max_features, and early_stopping to prevent overfitting and improve performance.
Error Handling: Add more try-except blocks for robustness, especially in file I/O and model training steps.

Model Comparison Table: Include a summary table comparing all models' metrics for quick reference.

Dependency Management: Include a requirements.txt file for easier replication in other environments.

Unit Tests: For production-grade code, add unit tests for critical functions (e.g., data validation, prediction consistency).

Business Impact & Conclusion:
Your work provides a strong foundation for predicting vaccination coverage, which can optimize resource allocation and improve public health strategies.
The use of explainable AI (SHAP) enhances trust and usability for policymakers.
The project is deployment-ready and could be integrated into a dashboard for real-time predictions.