CS5011 - Practical 1 Report

1.1 Design Choices and Rationale

1. **Date Preprocessing**
   - Implementation: For each row, I converted "date_recorded" to a datetime object, then extracted "year_recorded" and "month_recorded".
   - Rationale: This transformation reduces dimensionality while preserving temporal patterns, such as seasonal fluctuations, which can impact water pump functionality.
2. **Missing Value Handling**
   - **Numeric Features: Used median imputation to replace missing values.**
     - Rationale: The median is robust against outliers, making it suitable for variables with skewed distributions such as population.
   - **Categorical Features: Used mode imputation to replace missing values.**
     - Rationale: Preserves the most frequent category, ensuring consistency without introducing artificial categories.
3. **Categorical Encoding**
   - **OneHotEncoder**: Expanded categorical variables into binary features; used TruncatedSVD to mitigate high dimensionality.
   - **OrdinalEncoder**: Assigned integer values to categorical labels, assuming an implicit order.
   - **TargetEncoder**: Encoded categories using their mean target values, balancing informativeness with the risk of target leakage.
4. **Numeric Preprocessing**
   - "None" (no scaling) vs. StandardScaler.
   - Models like LogisticRegression and MLPClassifier frequently see significant benefits from scaled numeric features, whereas tree-based methods (e.g., RandomForestClassifier) typically do not.
5. **Model Selection**
   - Five classifiers are supported:
     - LogisticRegression: Linear, sensitive to data scaling, used as a solid baseline. (It gives "Increase the number of iterations" warning in the output of the code which is not an error)
     - RandomForestClassifier: A bagging ensemble robust to unscaled features and missing data, excellent baseline performance.
     - GradientBoostingClassifier and HistGradientBoostingClassifier: Boosting techniques that incrementally improve on errors, often top performers on tabular data.

- MLPClassifier: A neural-network approach with hidden layers is sensitive to scaling and can overfit if not carefully tuned.

1.2 Analysis and Experimental Findings

1. **Cross-Validation Strategy**

   - **5-fold cross-validation** was used to ensure robust evaluation of model performance across different data partitions, preventing overfitting and improving generalization.

2. **Impact of Encoding**

   - **OneHotEncoder:** Performed well for tree-based models but required TruncatedSVD to handle high-cardinality categorical features efficiently. **TargetEncoder:** Outperformed other encoding techniques in boosting models, allowing numerical representation without excessive dimensionality. **OrdinalEncoder:** Worked but introduced potential ordinal biases, affecting models relying on numerical interpretations.

3. **Model Observations**

   - **LogisticRegression:** Performed best with scaled features and TargetEncoder, achieving 0.7849 training accuracy and 0.7634 cross-validation accuracy. **RandomForestClassifier:** Showed strong baseline performance (CV: ~0.8088) across all encodings, benefiting from TargetEncoder. **GradientBoostingClassifier:** Provided a consistent performance boost (~0.7802 CV accuracy) when paired with TargetEncoder. **HistGradientBoostingClassifier:** Achieved ~0.7995 CV accuracy, indicating strong generalization. **MLPClassifier:** MLPClassifier: Required StandardScaler to prevent overfitting, achieving the best results with TargetEncoder (~0.7851 CV accuracy).

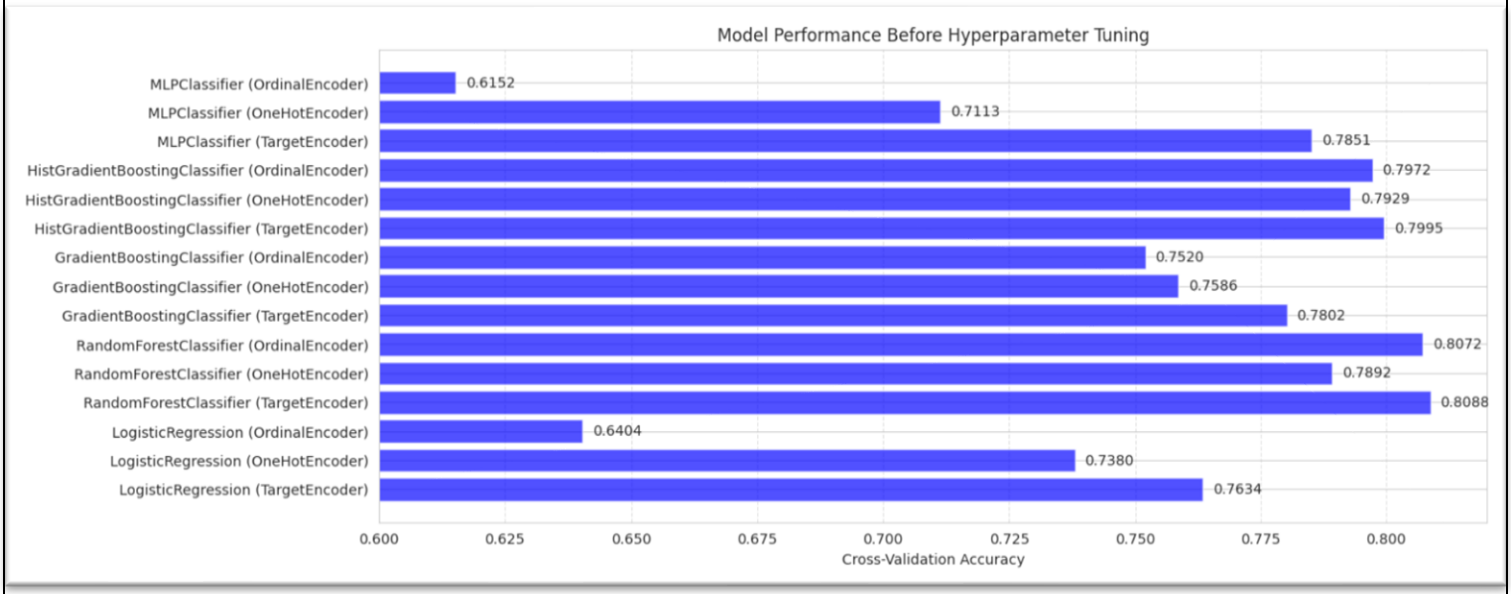
4. **Summary of Key Findings (Part 1)**

   - Tree-Based Models (RandomForest, GradientBoosting) demonstrated stable performance across encoding types, with TargetEncoder yielding the best results.
     - Reason: These models naturally handle categorical splits, but TargetEncoder provided an added numerical structure for improved decision boundaries.
   - LogisticRegression and MLPClassifier required feature scaling to optimize accuracy and convergence.
     - Reason: These models rely on gradient-based optimization, which struggles with raw categorical variables or unscaled numerical inputs.
   - TargetEncoder proved to be the most effective categorical encoding method, especially for boosting models.
     - Reason: It efficiently represents categorical features while retaining label distribution information, improving models that rely on gradient-based learning.
   - OneHotEncoder + TruncatedSVD was crucial for handling high-cardinality categorical data efficiently.
     - Reason: Reducing dimensions prevented unnecessary complexity while maintaining the most important variance in categorical data.

CS5011 - Practical 1 Report

Model	Numeric	Categorical	Training Accuracy	Cross-Validation Accuracy	Std Â±
LogisticRegression	None	OneHotEncoder	0.7387	0.7374	Â± 0.0018
LogisticRegression	None	OrdinalEncoder	0.6401	0.6395	Â± 0.0084
LogisticRegression	None	TargetEncoder	0.7809	0.7589	Â± 0.0055
LogisticRegression	StandardScaler	OneHotEncoder	0.739	0.738	Â± 0.0016
LogisticRegression	StandardScaler	OrdinalEncoder	0.6412	0.6404	Â± 0.0074
LogisticRegression	StandardScaler	TargetEncoder	0.7849	0.7634	Â± 0.0048
RandomForestClassifier	None	OneHotEncoder	0.9982	0.7892	Â± 0.0032
RandomForestClassifier	None	OrdinalEncoder	0.9983	0.8072	Â± 0.0022
RandomForestClassifier	None	TargetEncoder	0.8752	0.8082	Â± 0.0024
RandomForestClassifier	StandardScaler	OneHotEncoder	0.9982	0.7888	Â± 0.0030
RandomForestClassifier	StandardScaler	OrdinalEncoder	0.9983	0.8081	Â± 0.0029
RandomForestClassifier	StandardScaler	TargetEncoder	0.8746	0.8088	Â± 0.0038
GradientBoostingClassifier	None	OneHotEncoder	0.7686	0.7581	Â± 0.0029
GradientBoostingClassifier	None	OrdinalEncoder	0.7562	0.752	Â± 0.0038
GradientBoostingClassifier	None	TargetEncoder	0.8016	0.7802	Â± 0.0060
GradientBoostingClassifier	StandardScaler	OneHotEncoder	0.7695	0.7586	Â± 0.0044
GradientBoostingClassifier	StandardScaler	OrdinalEncoder	0.7562	0.752	Â± 0.0038
GradientBoostingClassifier	StandardScaler	TargetEncoder	0.8024	0.7792	Â± 0.0050
HistGradientBoostingClassifier	None	OneHotEncoder	0.8401	0.7929	Â± 0.0020
HistGradientBoostingClassifier	None	OrdinalEncoder	0.8213	0.7972	Â± 0.0051
HistGradientBoostingClassifier	None	TargetEncoder	0.8322	0.7989	Â± 0.0051
HistGradientBoostingClassifier	StandardScaler	OneHotEncoder	0.8375	0.7922	Â± 0.0023
HistGradientBoostingClassifier	StandardScaler	OrdinalEncoder	0.8213	0.7972	Â± 0.0051
HistGradientBoostingClassifier	StandardScaler	TargetEncoder	0.831	0.7995	Â± 0.0074
MLPClassifier	None	OneHotEncoder	0.7465	0.7113	Â± 0.0341
MLPClassifier	None	OrdinalEncoder	0.6253	0.6152	Â± 0.0870
MLPClassifier	None	TargetEncoder	0.6512	0.6463	Â± 0.1115
MLPClassifier	StandardScaler	OneHotEncoder	0.844	0.7792	Â± 0.0068
MLPClassifier	StandardScaler	OrdinalEncoder	0.626	0.6131	Â± 0.0886
MLPClassifier	StandardScaler	TargetEncoder	0.8168	0.7851	Â± 0.0075

CS5011 - Practical 1 Report



Plot 1: Model Performance Before Hyperparameter Optimization (HPO)

Description: This bar chart compares the cross-validation accuracy of different models before hyperparameter tuning, using various preprocessing and encoding strategies.

Section 2: Design, Analysis, and Key Findings (Part 2)

2.1 Hyperparameter Optimization (HPO) with Optuna

1. **Search Space**
   - **Preprocessing:**
     - **Numeric Features:** None, StandardScaler}
     - **Categorical Features:** OneHotEncoder, OrdinalEncoder, TargetEncoder}
   - **Model Choice**: {LogisticRegression, RandomForestClassifier, GradientBoostingClassifier, HistGradientBoostingClassifier, MLPClassifier}
   - **Model-Specific Hyperparameters**:
     - RandomForestClassifier: n_estimators, max_depth, min_samples_split}
     - Logistic Regression: C, solver, max_iter}
     - GradientBoostingClassifier: n_estimators, learning_rate, max_depth}
     - HistGradientBoostingClassifier: max_iter, learning_rate, max_depth}
     - MLPClassifier: hidden_layer_sizes}
2. **Objective Function & Evaluation**
   - **Cross-Validation Strategy**: Used StratifiedKFold to ensure class balance across splits.

- **Optimization Approach**: Optuna adaptively explored hyperparameters through Bayesian and random search over 100 trials, maximizing cross-validation accuracy.

3. **Insights from the Optimization**

- **GradientBoostingClassifier** achieved the best performance with:
  - n_estimators: 256, learning_rate: 0.0755, max_depth: 10, Cross-validation accuracy: 0.8108 (highest among models)
- **RandomForestClassifier** tended to overfit, achieving nearly perfect training accuracy (~0.9983) but lower validation accuracy (~0.8082).
- **MLPClassifier** was highly sensitive to preprocessing and exhibited inconsistent performance, with accuracy ranging from 0.6152 to 0.7851.
- **LogisticRegression** performed best with TargetEncoder, but tree-based models consistently outperformed it.

2.2 Key Findings and Performance Gains

1. Preprocessing vs. Model Hyperparameters

- **TargetEncoder** emerged as the best encoding method for boosting models due to its ability to provide a numerical representation of categorical variables while preserving relationships with the target variable.
- **OneHotEncoder + TruncatedSVD** helped mitigate high-cardinality issues, leading to more stable and compact tree-based models.

2. Model-Specific Performance Gains

- Hyperparameter tuning led to a 2–5% performance boost, especially for RandomForest and GradientBoosting.
- LogisticRegression showed significant accuracy gains when optimal C and solver values were selected.
- MLPClassifier required careful tuning of hidden layers (e.g., (100,50)) and feature scaling for stable convergence.

3. Pipeline Robustness & Generalization

- Automated HPO eliminated manual guesswork, systematically refining hyperparameters and model selection.
- Tree-based models consistently generalized well, while boosting methods outperformed traditional approaches.
- GradientBoostingClassifier emerged as the most reliable model, balancing accuracy and robustness.

Section 3: Conclusion

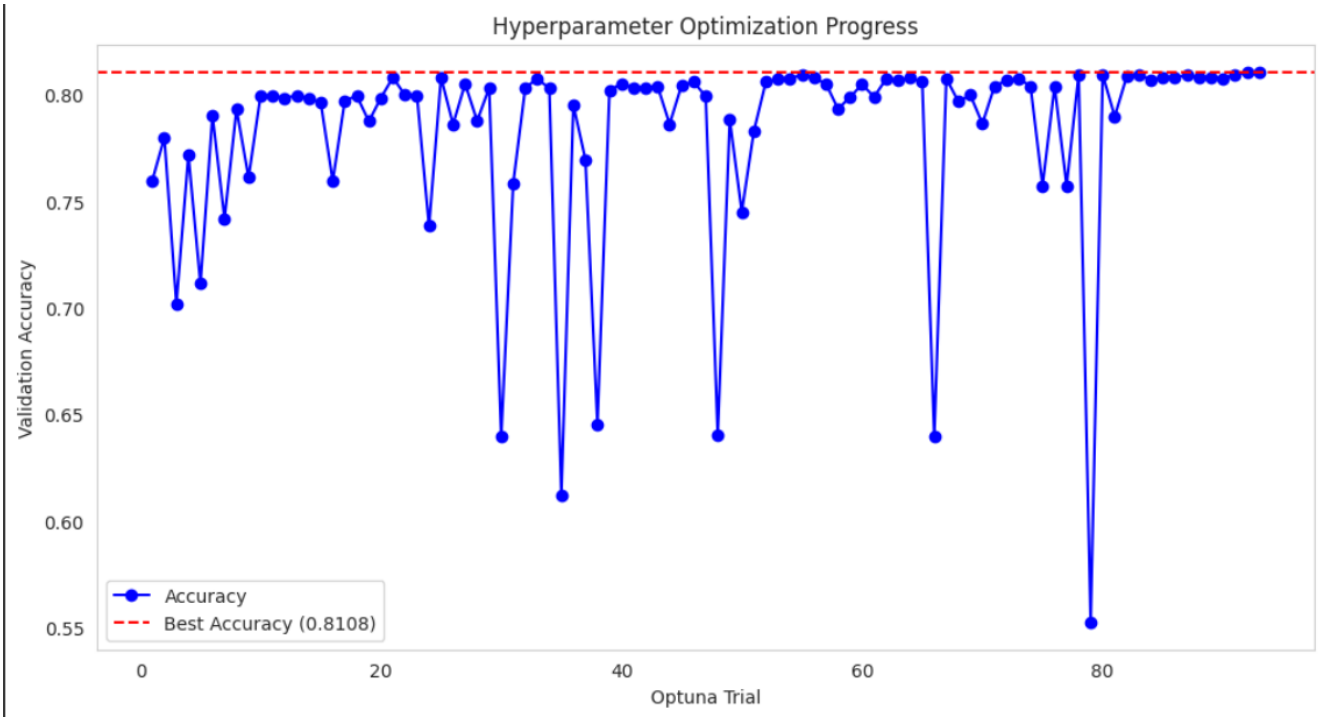
The comprehensive data preprocessing strategy played a critical role in optimizing model performance by effectively handling missing values, categorical encoding, and feature scaling. Among encoding techniques, TargetEncoder emerged as the best choice for GradientBoostingClassifier, preserving category-target relationships without inflating feature space. OrdinalEncoder performed well for RandomForestClassifier,

aligning with its tree-based structure. These encoding decisions significantly impacted model generalization, demonstrating that categorical data transformation is as crucial as model selection. Additionally, feature scaling was essential for LogisticRegression and MLPClassifier, as they rely on numerical gradients, while tree-based models remained robust without scaling.

The GradientBoostingClassifier with TargetEncoder achieved the highest accuracy (0.8108), outperforming all other models due to its ability to iteratively correct prediction errors and refine decision boundaries. Optimal hyperparameters, including n_estimators = 256, learning_rate = 0.0755, and max_depth = 10, ensured a balance between model complexity and generalization. Although RandomForestClassifier exhibited competitive validation accuracy (0.8082), its training accuracy of ~0.9983 indicated severe overfitting, suggesting a need for additional regularization. Meanwhile, MLPClassifier displayed unstable performance (0.6123–0.7451), heavily reliant on preprocessing, reaffirming the importance of feature scaling and structured hyperparameter tuning.

The impact of Hyperparameter Optimization (HPO) was substantial, leading to accuracy improvements of up to 5% across models, with the most significant gains in GradientBoostingClassifier. HPO allowed fine-tuning of key parameters, ensuring an optimal trade-off between underfitting and overfitting. The automated, data-driven hyperparameter selection through Optuna eliminated the inefficiencies of manual tuning, highlighting the power of systematic optimization. These findings reinforce the importance of preprocessing, encoding selection, and hyperparameter tuning in machine learning pipelines. Ultimately, GradientBoostingClassifier with TargetEncoder proved to be the best approach, offering the highest predictive accuracy while maintaining robustness, setting a benchmark for similar classification tasks.

Plot 2: Hyperparameter Optimization Progress with Optuna



Description: This line plot visualizes the accuracy improvement of over 100 Optuna trials during hyperparameter optimization (HPO).