Springboard Capstone two:

“Classifying Permit Types from San Francisco Building Permit Data Using Machine Learning.”

Benoit Loze

18/07/2025

Table of Contents

Introduction:

This report presents the development of a supervised machine learning model to classify building permit types in San Francisco using structured features from the 2013–2018 open permit dataset. Given the high volume and complexity of permit applications, automating permit type classification aims to improve processing efficiency and decision support within municipal workflows.

The project compares multinomial logistic regression with tree-based models (Random Forest), evaluates multiple preprocessing strategies to address class imbalance, and applies both randomized and Bayesian hyperparameter search.

The final model, Random Forest with Bayesian Optimization, achieved a strong F1 score (0.81) and demonstrated balanced precision and recall across all eight permit categories. The full modeling workflow, including data wrangling, EDA, preprocessing, and performance evaluation, is documented in this report.

1 Problem statement:

1.1 Problem:

To develop a supervised machine learning model to classify San Francisco building permit types (e.g., New Construction, Electrical) using structured features, targeting ≥80% accuracy on a held-out test set, within a 4-week timeline.

1.2 Context:

Municipal offices handle large volumes of permits. Automating permit classification using historical data can streamline workflows, detect anomalies, and optimize processing. The San Francisco Building Permits dataset (2013–2018, Kaggle) will be used.

1.3 Success Criteria:

- ≥80% classification “accuracy”.
- Complete Jupyter Notebook (data wrangling, EDA, preprocessing, modeling).
- Clear report and slide deck for non-technical stakeholders.

1.4 Solution Scope:

- Supervised classification only.
- Comparison of model types.

1.5 Constraints:

- Missing/inconsistent data.
- Potential class imbalance.
- Need for feature engineering.
- Assumes correct labeling in the source dataset.

2 Data wrangling:

2.1 Data inspection and definition:

- Loaded the dataset and reviewed general information.
- Inspected null values for each feature.
- Identified features with less than 5% missing values.
- Marked features expected to be in date format.
- Marked features expected to be categorical.
- Identified numerical features.
- Detected redundant features.
- Confirmed absence of fully duplicated rows.
- Identified presence of partial duplicates.

2.2 Data cleaning:

- Dropped features with more than 50% missing values.
- Removed rows with missing values in features having <5% nulls.
- Imputed missing values in numerical features:
    - Used mean values by default.
    - Used median for skewed distributions (e.g., Estimated Cost).
- Converted date-type columns and applied forward fill by Permit Number.
- For categorical features:
    - Detected and corrected fuzzy duplicate entries.
    - Corrected data types.
    - Imputed missing values using the mode (most frequent category).

2.3 Outliers detection:

- Flagged numerical features which are likely to contain outliers.
- Visualized distributions using boxplots and histograms.
- Counted outliers per numerical feature.

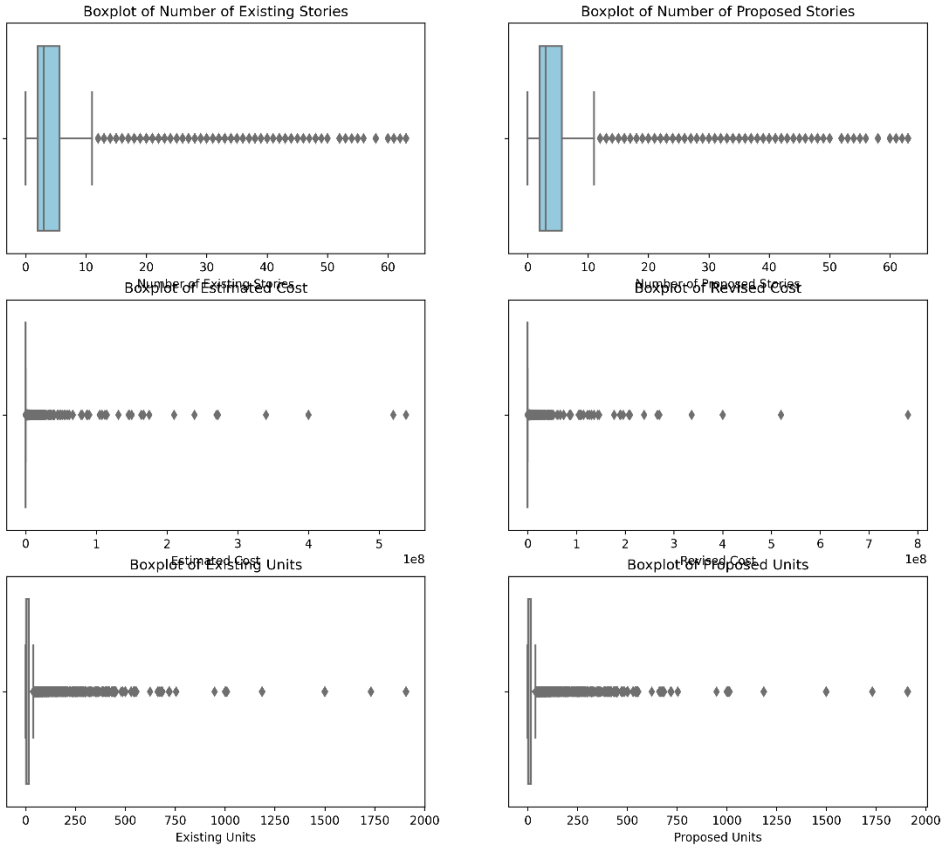

Figure 1: Outliers detection, boxplots.

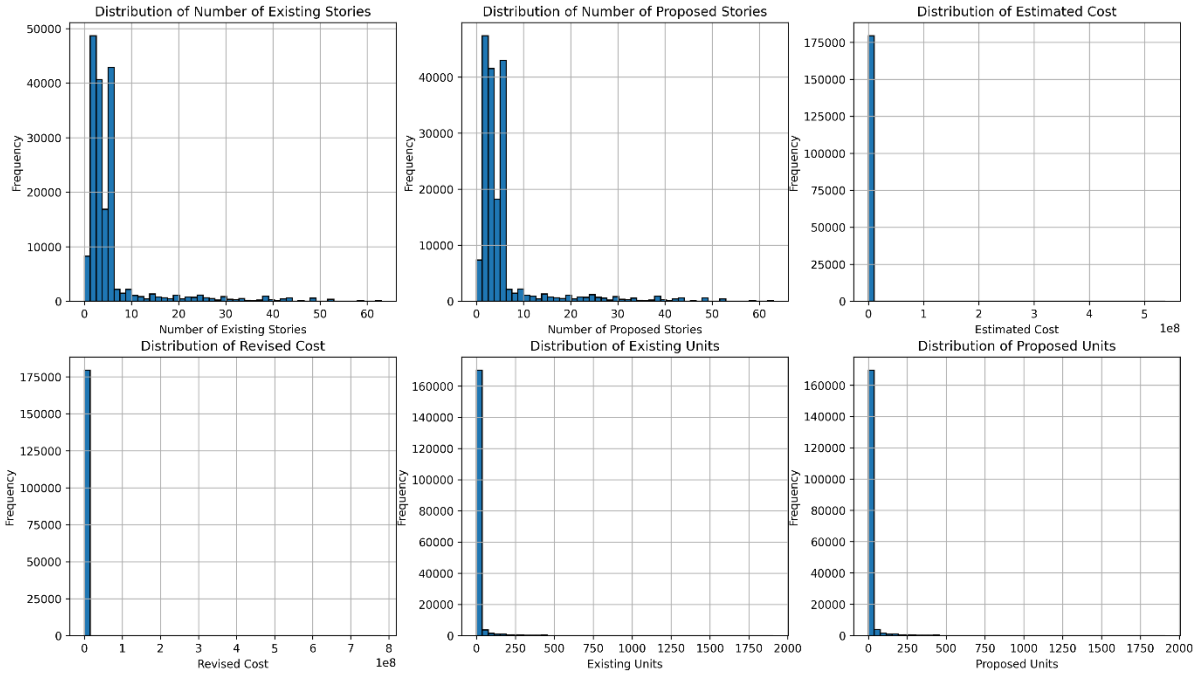

Figure 2: Outliers detection, histogram distribution.

2.4 Redundant features:

- Removed previously identified redundant features.

2.5 Partial duplicates:

- Logged presence of partial duplicates for further stages (EDA and preprocessing).

2.6 Data save:

- The cleaned data is saved to a .csv file for further stages.

3 Exploratory data analysis - EDA:

3.1 Build data profiles and tables:

- Revision of the categorical data from the data wrangling phase.
- Revision of the outliers identified in the data wrangling phase:
  - First, a distinction is made (split into two separate datasets) between “high-rise” and “low-rise” buildings. This makes plausible sense in construction and urban planning analysis as both building types behave differently across costs, timeline, permitting and project types.
  - Second, np.log1p() will be applied to normalize the skewness of the following features: Estimated Cost, Revised Cost, Existing Units, and Proposed Units. These variables are likely to be right skewed due to their scale and distribution characteristics. Since they contain valuable information relevant to the predictive task, applying a log transformation helps reduce skewness, stabilize variance, and improve the performance and reliability of machine learning models.

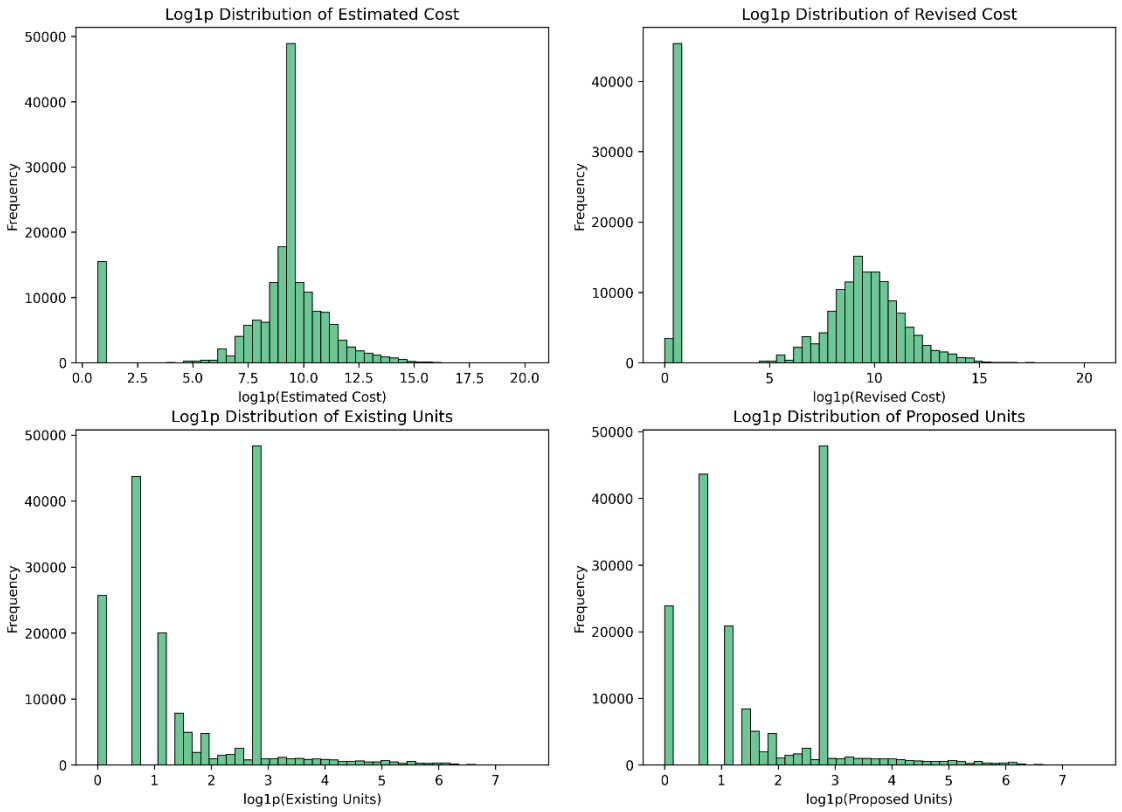

Figure 3: Outliers log transformation.

- Partial duplicates with the same Permit Number are retained, as they reflect meaningful variations (e.g., extensions or modifications) and contribute valuable information for classifying permit types.
- Retaining the original data ensures more precise input for learning. To avoid multicollinearity and preserve model reliability, range-based features are dropped, allowing the model to focus on the most informative variables.
- Inferential statistics: features preselection and feature engineering.
- Inferential statistics: the significance of each feature variable—whether categorical or numerical—has been evaluated with respect to the "Permit Type" target variable using appropriate statistical tests. A p-value threshold of 0.05 has been applied to determine statistical significance.

3.2 Explore data relationships:

- The final goal is to use Logistic Regression and Random Forest to classify the target variable, Permit Type.
  - **Logistic Regression** assumes independence among predictors and is sensitive to multicollinearity, which can distort coefficient estimates and inflate standard errors.
  - **Random Forest** is more robust to collinearity, but high correlation between features can still affect feature importance interpretation.

- To mitigate multicollinearity, it is good practice to drop one of the correlated features when the correlation coefficient exceeds 0.80.
  The following feature pairs are affected:
  - Revised Cost vs Estimated Cost.
  - Proposed Construction Type vs Existing Construction Type.
  - Proposed Units vs Existing Units.

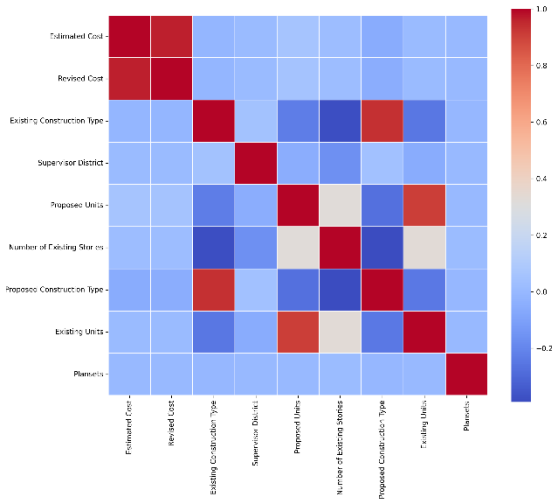

Figure 4: Collinearity heatmap.

- Verification of the data distribution of the categorical features with count plot.

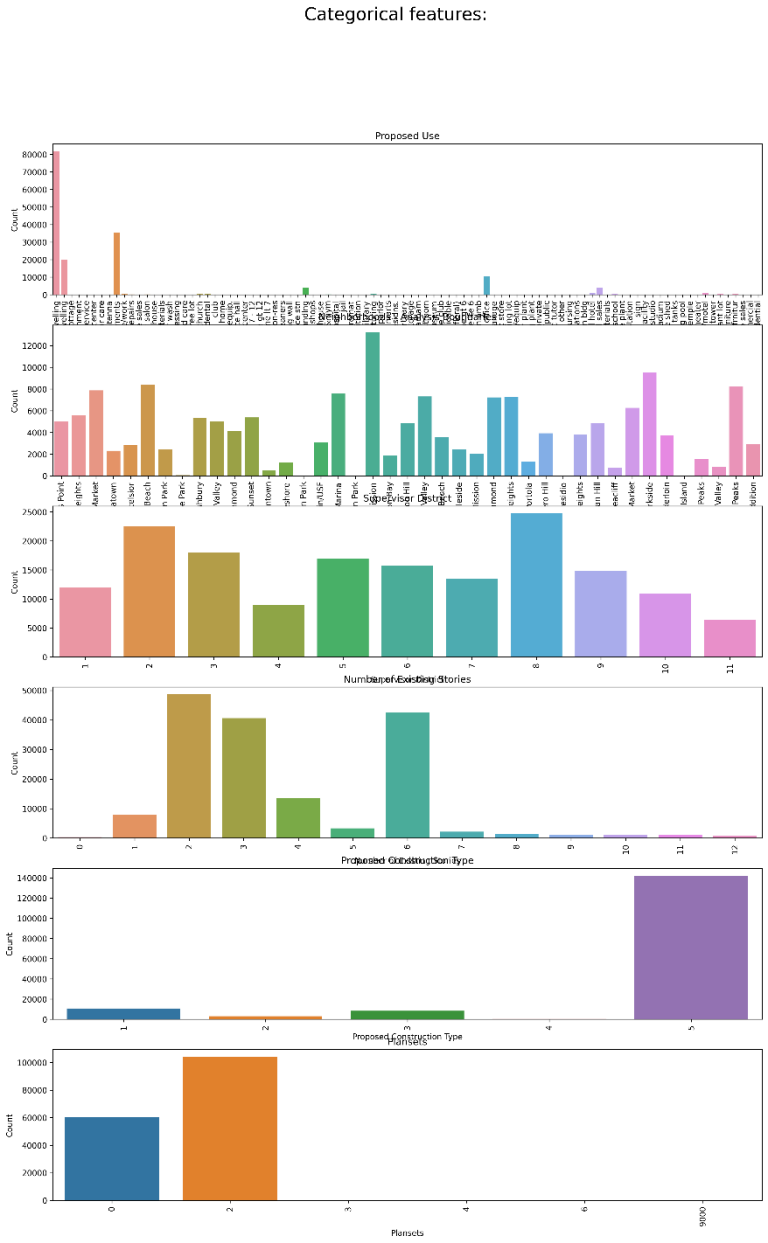

Figure 5: Categorical count plots.

- One hot encoding applied to the data. However, this step will be revised and refined in the further stage of data preprocessing.

3.3 Data save:

- The data is saved to a .csv file for further stages.

4 Data preprocessing:

4.1 Predictors check:

- The loaded dataset has been re-checked to ensure consistency and correctness.
- Numerical features with confirmed skewed distributions undergo log transformation to reduce skewness and enhance model performance.

4.2 Target data check:

- The distribution of the dependent variable reveals a clear class imbalance across the eight categories of Permit Type.

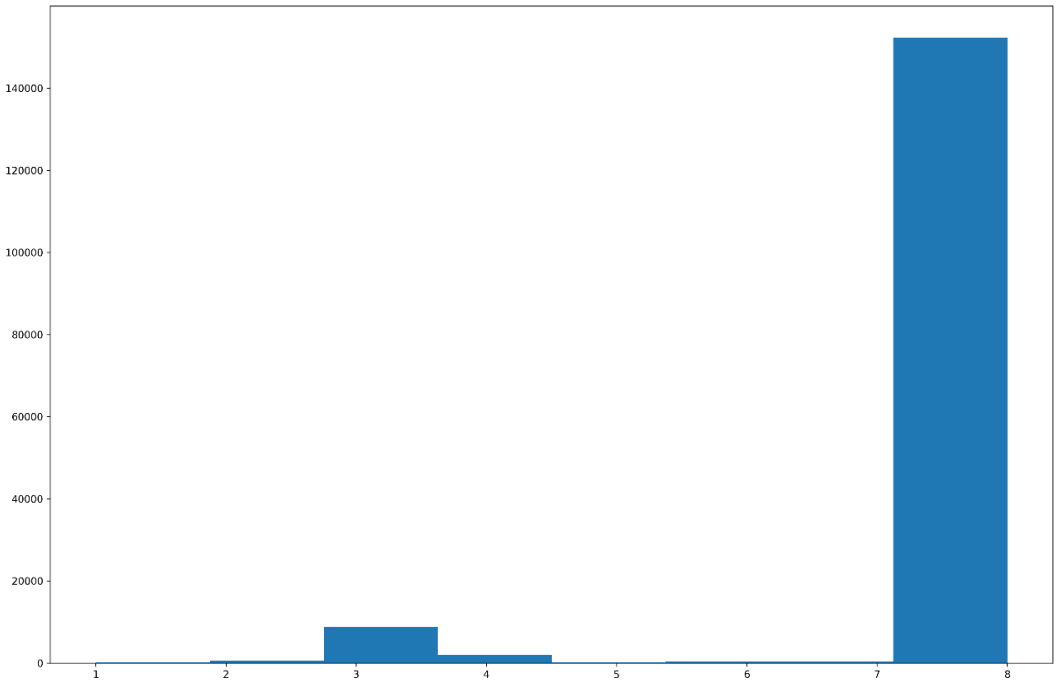

Figure 6: Target features, class imbalance.

4.3 Categorical data handling:

- Regression models (e.g., linear, or logistic regression) are favored by one-hot encoding.
- Tree-based models can natively handle categorical features.

4.4 Workflow explanation:

- A key priority in this workflow is to avoid data leakage, which can lead to artificially inflated model performance during training and failure when deployed in real-world scenarios. To maintain model integrity and generalizability, the following steps are applied:

  - **Data Splitting**:
    - The dataset is first split into training and test sets.
    - Using "stratify=y" ensures that all 8 permit classes, including rare ones, are proportionally represented in both training and test sets. Without it, minority classes may be missing from one split, leading to biased training and unreliable evaluation.
  - **Handling Class Imbalance** (SMOTE Alternative):
    - Initially, SMOTE (Synthetic Minority Oversampling Technique) was considered, but compatibility issues with the imblearn package prompted a shift.
    - Instead, we now leverage the "class_weight" parameter in machine learning models to adjust for class imbalance during training.
  - **Categorical Feature Encoding:**
    - <u>Tree-Based Models (e.g., Random Forest):</u>
      - Apply Target Encoding to categorical variables.
      - The encoder is fit only on the training data to avoid leakage and applied consistently to test data.
    - <u>Regression-Based Models (e.g., Logistic Regression):</u>
      - Use One-Hot Encoding on categorical features.
      - Align training and test datasets to ensure consistent feature columns across both.
  - **Feature Scaling:**
    - Scaling is applied only to regression-based models, as tree-based models are scale-invariant.
    - Standard scaling techniques (e.g., StandardScaler) are fit on the training data only and then applied to the test data.

- This pipeline avoids data leakage, ensures compatibility with the current environment, and maintains a clear distinction between training and test data

throughout preprocessing. It also adapts feature engineering techniques to suit the model type while handling class imbalance.

4.5 Data save:

- The data is saved to a .csv file for further stages.

5 Modelling:

- Due to severe class imbalance in the target variable, where Class 7 dominates the dataset (~90%), the accuracy score is misleading.
- Instead of accuracy, the evaluation focused on more informative metrics:
  - F1-score.
  - Precision.
  - Recall.
  - Confusion Matrix.
- Models' metrics files are all available in the subfolder: "model_metrics_files".

5.1 Regression-based model:

- Use of the multinomial logistic regression model.

Hyperparameter tuning:

- Both "Randomized Search" and "Bayesian Search" are used to tune the hyperparameters:
  - "Randomized Search" best hyperparameters:
    {'max_iter': 1000, 'C': 10}
  - "Bayesian Search" best hyper parameters:
    {'C': 100, 'max_iter': 700}

Performance metrics:

- In both case of "Randomized Search" and "Bayesian Search", the performance metrics and the confusion matrices are similar.
- The confusion matrices show:
  - A large number of samples from the class 7 were misclassified either as class 2, class 4, class 6.
  - A little bit more than half the class 7 was correctly predicted.
  - This may show persistent confusion due to the class imbalance despite the balancing efforts.
  - Some samples of class 2 were misclassified as the class 7.
- High recall compared to low precision can be explained is a sign that the model is overpredicting the dominant class.
- F1 score is the harmonic mean of precision and recall and precision drags the effectiveness down.
- The F1 score makes the model prediction **moderate** but in its **lower range**.

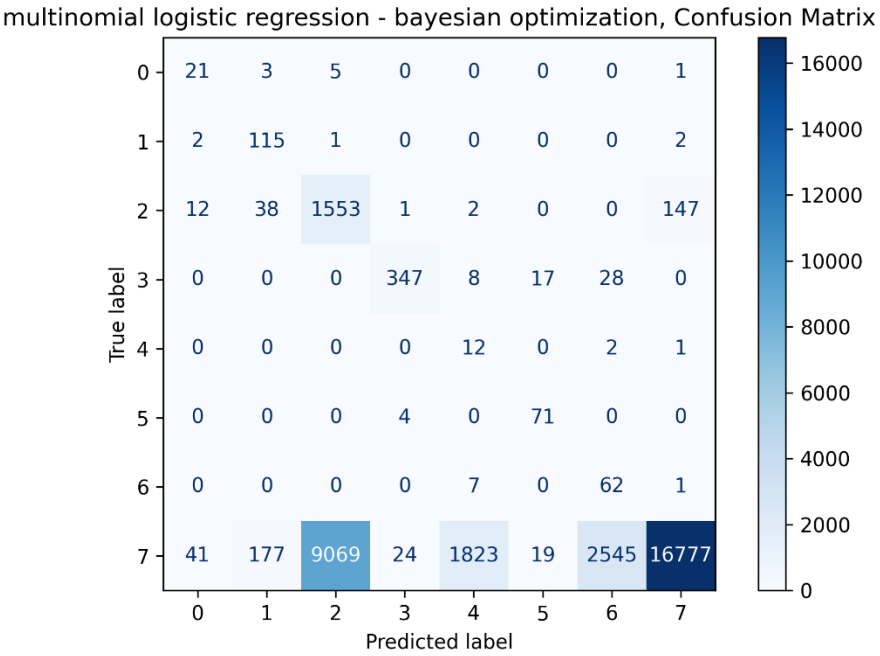

Figure 7: Multinomial logistic regression, Bayesian optimization, confusion matrix.

5.2 Tree-based model:

- Use of the random forest model.

Hyperparameter tuning:

- Both "Randomized Search" and "Bayesian Search" are used to tune the hyperparameters:
  - "Randomized Search" best hyperparameters: {'n_estimators': 300, 'min_samples_split': 10, 'min_samples_leaf': 1, 'max_features': 'sqrt', 'max_depth': None, 'criterion': 'entropy', 'class_weight': 'balanced_subsample', 'bootstrap': True}
  - "Bayesian Search" best hyper parameters: {'n_estimators': 400, 'max_depth': 40, 'min_samples_split': 2, 'min_samples_leaf': 1, 'bootstrap_val': 0.11734270640368927, 'criterion_val': 0.16922819233831066, 'max_features_val': 0.9182114435034318, 'class_weight_val': 0.644863330672783}

Performance metrics:

- Tree-based models, especially Random Forest, perform significantly better than logistic regression on imbalanced data.
- Bayesian search yields more balanced precision and recall compared to randomized search.
- Class 7 predictions improved: fewer misclassifications as class 2, raising precision.

- Class 2 sees some misclassification into class 7, slightly reducing recall.
- Despite tradeoffs, the F1 score remains strong, indicating overall robustness.
- Conclusion: Tree-based models, particularly with Bayesian optimization, handle imbalance more effectively and provide balanced performance across classes.

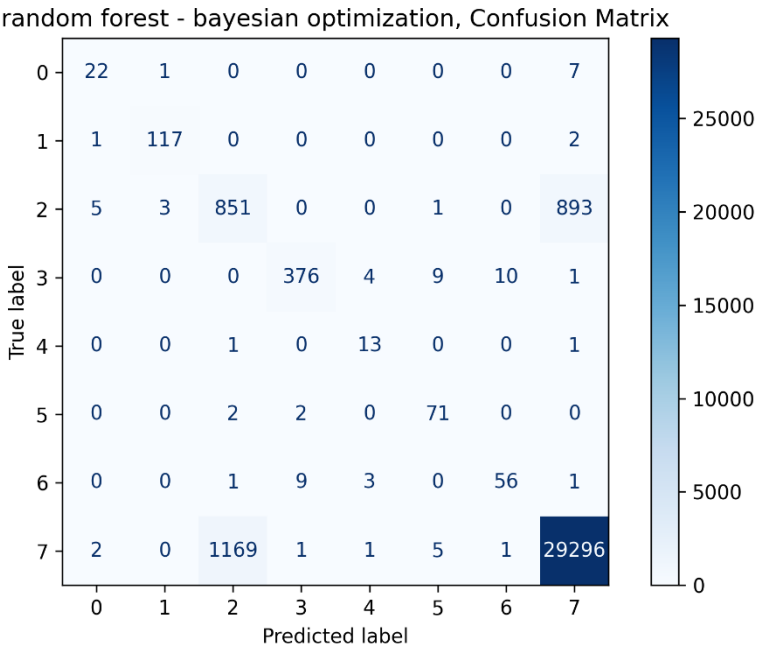

Figure 8: Random Forest, Bayesian optimization, confusion matrix.

5.3 Model selection & key takeaways:

- Final Model Selection: **Random Forest model (Bayesian Optimization)**
  - Best performance across all tested models, including logistic regression and grid search.
  - F1 Score: 0.81 — strong overall balance.
  - Precision: 0.79, Recall: 0.84 — effective and reliable across 8 classes.
  - Handles class imbalance well without overfitting.
  - Bayesian optimization improved tuning efficiency and consistency.

5.4 Data save:

- The model metrics file (.txt) are saved under a “model_metrics_file” directory.

****