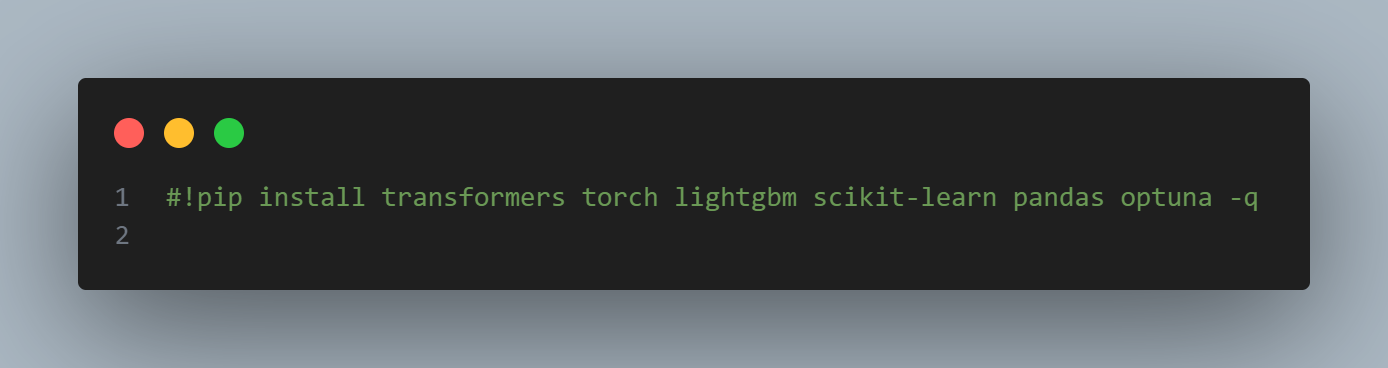
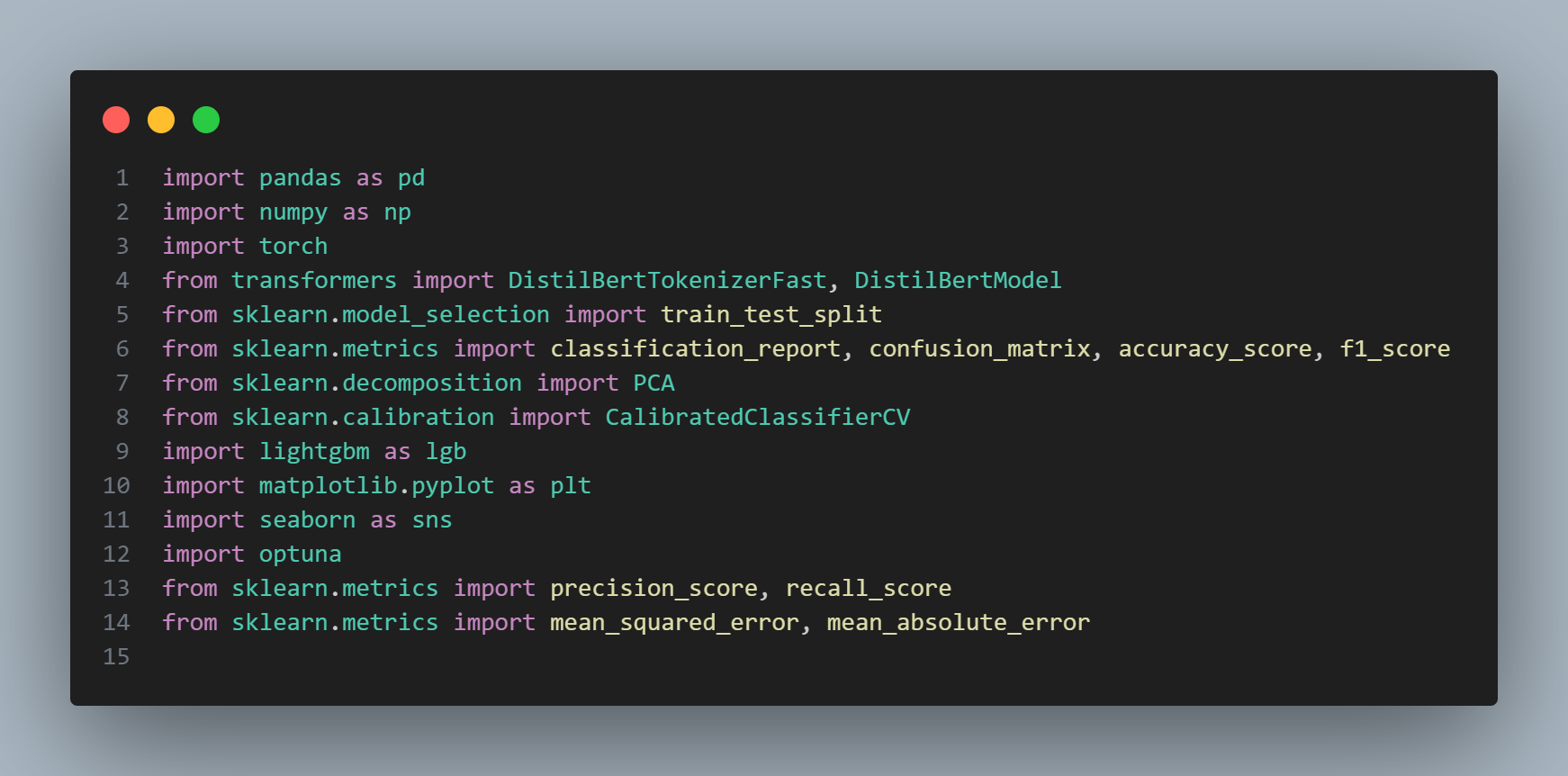
**Documentation for Hybrid BERT + LightGBM**

**With Feature Engineering and Hypertuning using Optuna**



This is a **Jupyter Notebook** or **Google Colab** style command used to install Python packages.

* #!pip install ...:  
  The #! at the beginning is a **comment** in Python, which means this line **won’t execute** unless the # is removed.
* pip install:  
  This is the **Python package installer**, used to install libraries from the Python Package Index (PyPI).
* **Packages listed**:
  + transformers: Hugging Face library for state-of-the-art NLP models like BERT, GPT, etc.
  + torch: PyTorch — a deep learning framework.
  + lightgbm: A fast, efficient gradient boosting framework by Microsoft.
  + scikit-learn: For traditional machine learning models and utilities.
  + pandas: For data manipulation and analysis (especially with DataFrames).
  + optuna: An automatic hyperparameter optimization framework.
* -q: This stands for **quiet mode**, which suppresses the output during installation to make logs cleaner.



This block imports all necessary libraries for the hybrid machine learning pipeline. It includes data handling (pandas, numpy), BERT models from Hugging Face, machine learning tools from scikit-learn, the LightGBM classifier, plotting tools (matplotlib, seaborn), and Optuna for hyperparameter tuning. These imports enable preprocessing, embedding generation, dimensionality reduction, model training, calibration, evaluation, and visualization.



This code loads a cleaned synthetic dataset and prepares it for analysis. It extracts the week number from the "Week" column using regex, sorts the data by product\_code and Week\_num, and adds a new feature was\_on\_special\_last\_week, indicating if a product was on special in the previous week. This lag feature is useful for modeling future discount predictions.



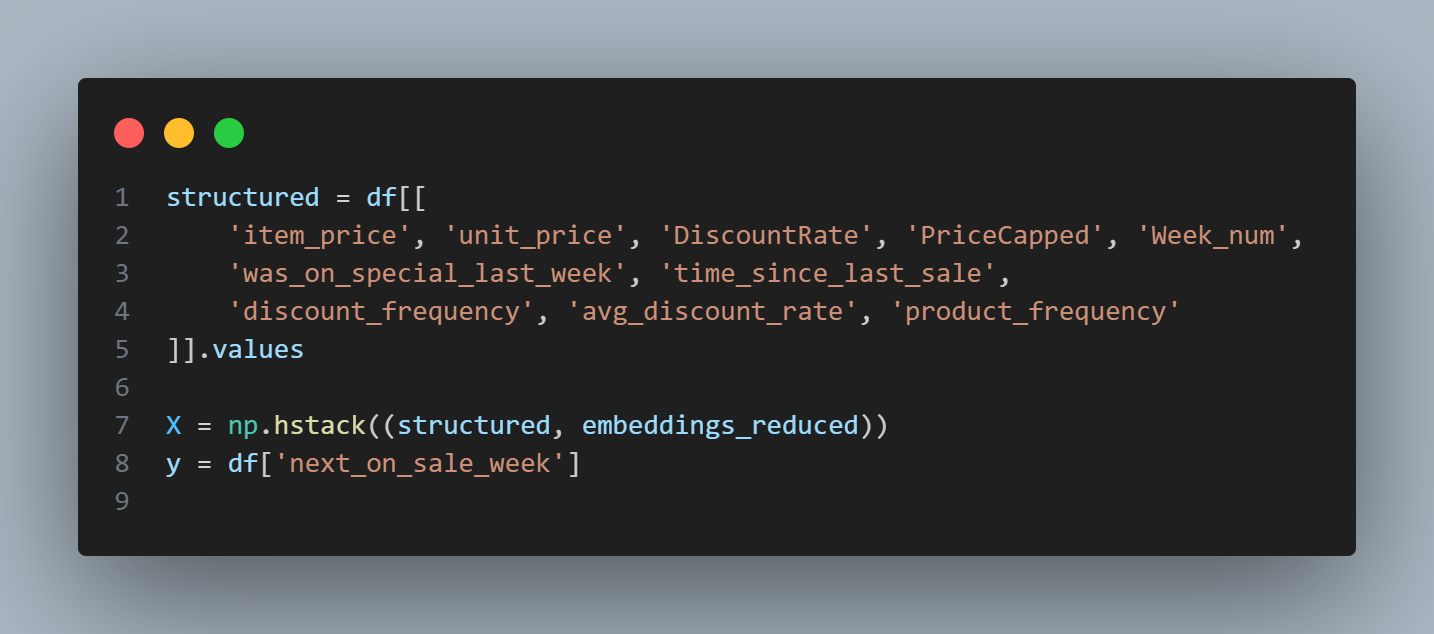
This code defines and applies a function to find the next week a product will go on special. For each product, it loops through its weekly records and checks if a future IsOnSpecial flag is set to 1. If so, it records that future week as the next\_on\_sale\_week; otherwise, it appends None. The function is applied across all product groups, and rows with missing next sale weeks are dropped. Finally, the resulting column is converted to integers, preparing it as a target variable for supervised learning.



This code adds several new features to enhance the dataset for modeling. The function add\_time\_since\_last\_sale calculates how many weeks have passed since each product was last on special. Afterward, additional group-level features are computed: discount\_count (total weeks on special), discount\_frequency (ratio of discount weeks to total weeks), avg\_discount\_rate (mean discount), and product\_frequency (number of weekly records per product). These features help the model learn product-specific discount patterns over time.



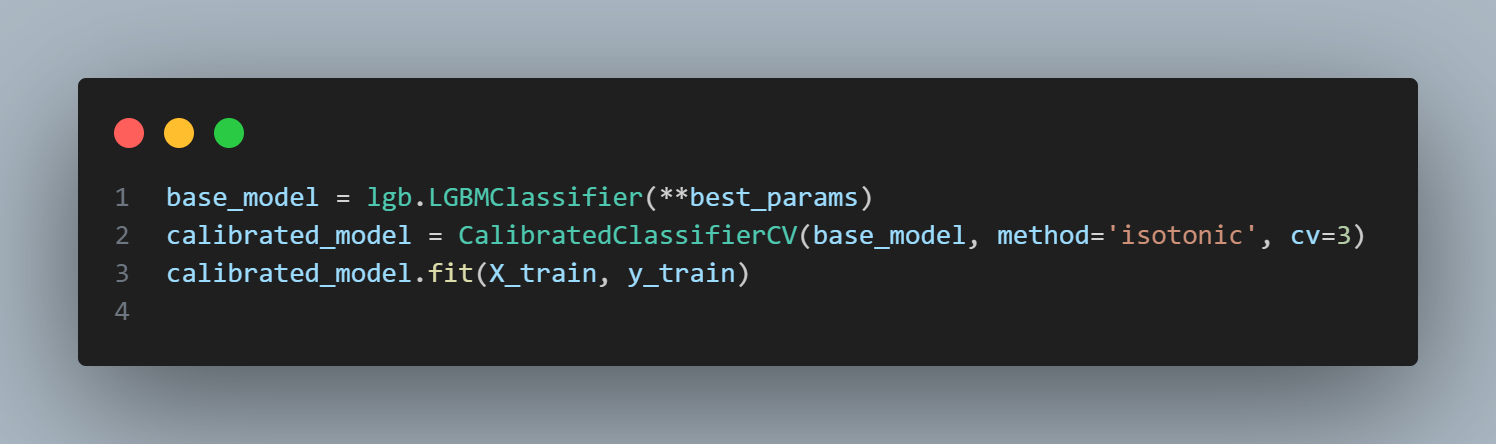
This code defines a function get\_bert\_embeddings that generates sentence embeddings using a pretrained DistilBERT model in batches. It automatically uses a GPU if available and disables gradient tracking for efficient inference. The CLS token embeddings are collected and stacked into a single array. The embeddings are then reduced to 100 dimensions using PCA for downstream use in machine learning models. Finally, the code applies this process to the item\_name column of the dataset.



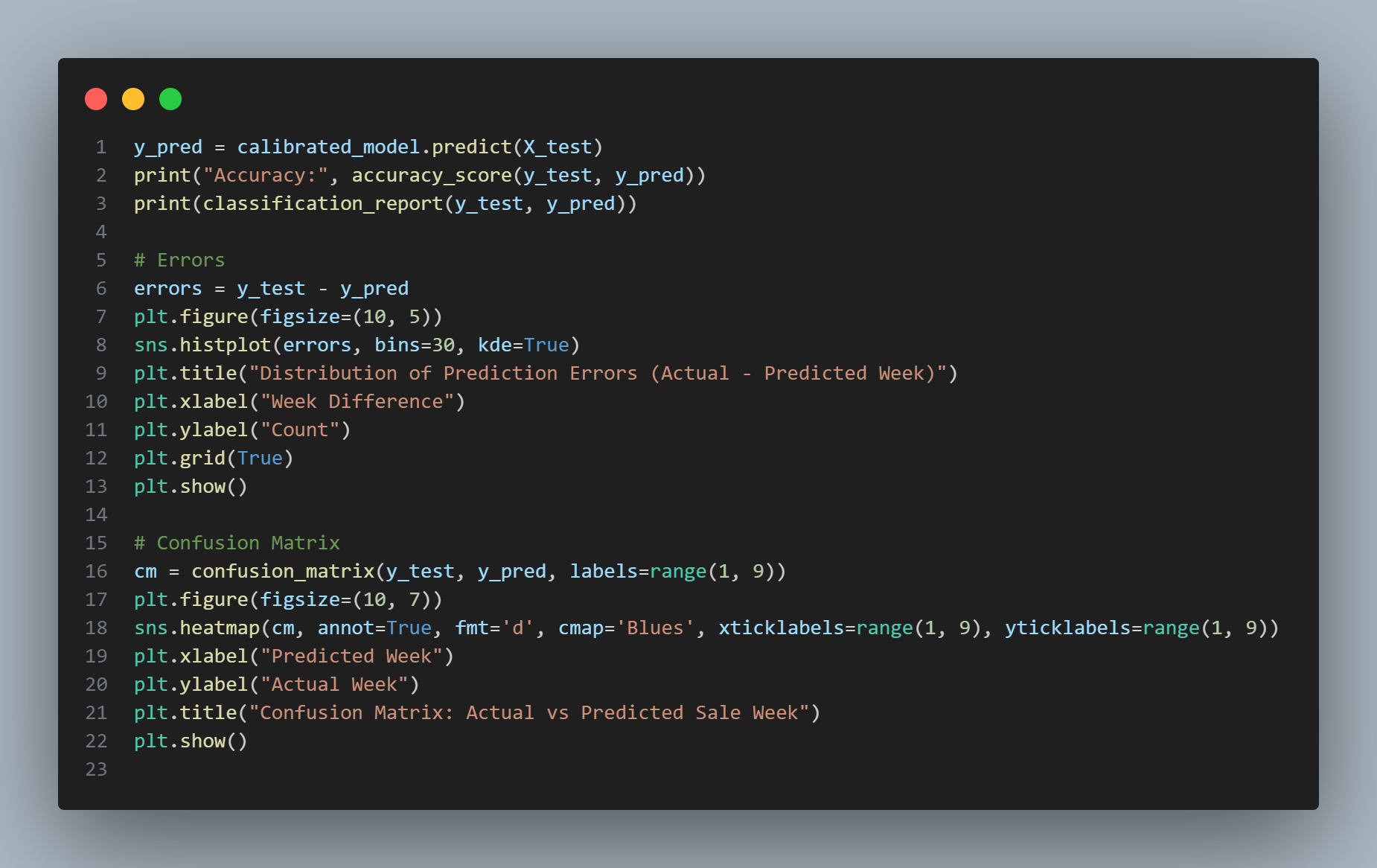
This code constructs the final feature matrix X and target vector y for training. It selects structured numerical features such as pricing, discount history, and product activity from the dataframe, then horizontally stacks them with the reduced BERT embeddings. The resulting X combines both semantic and numerical information, while y holds the target variable—next\_on\_sale\_week—to be predicted.



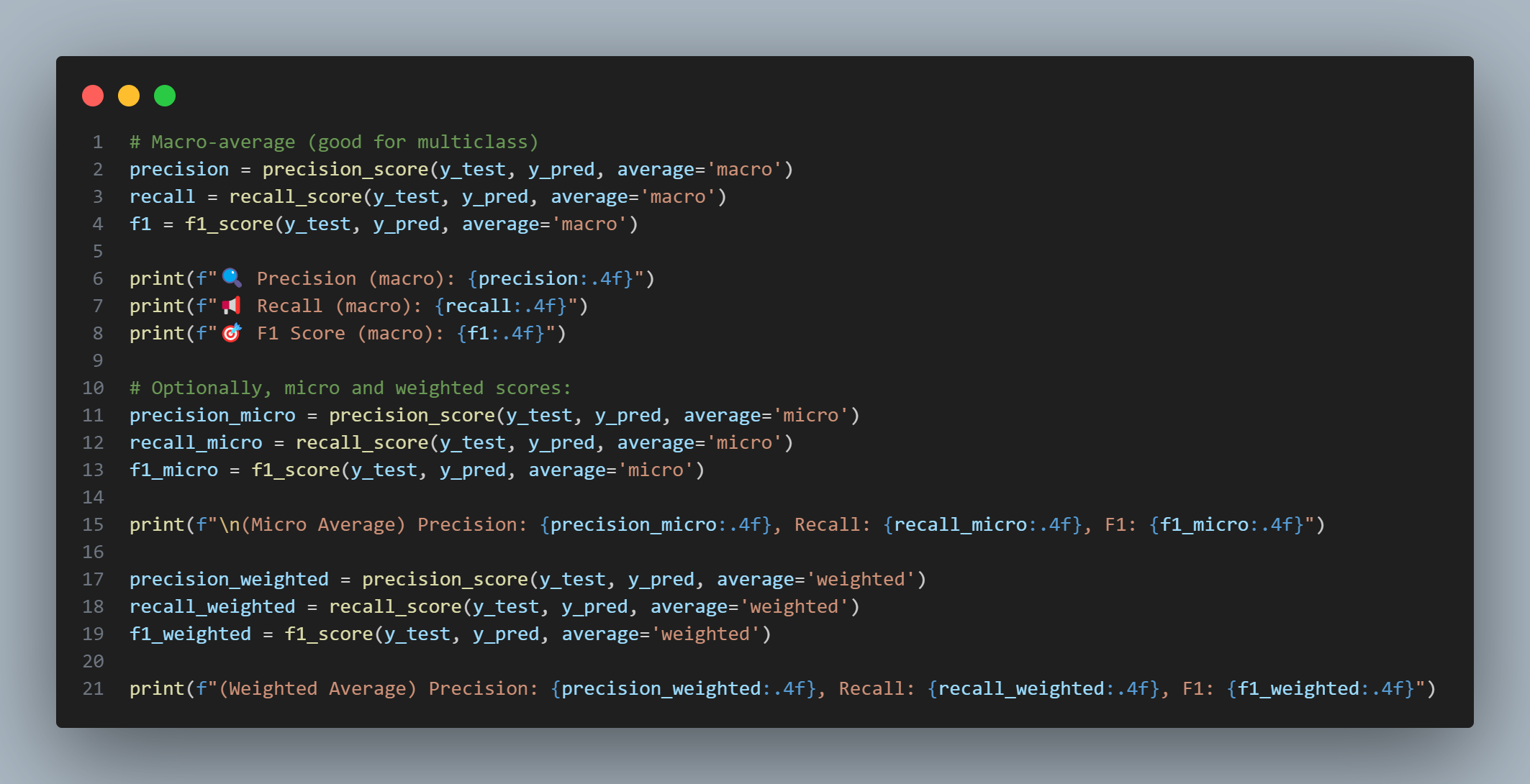
This code performs hyperparameter tuning for a LightGBM multiclass classifier using Optuna. It defines an objective function that suggests values for parameters like learning rate, number of leaves, max depth, and min data in leaf. The model is trained and evaluated using macro F1-score. Optuna then runs 30 trials to find the best parameter combination. Finally, the best parameters are updated with fixed values for objective and num\_class, preparing them for use in final model training.



This code uses the best hyperparameters from Optuna to create a LightGBM model and wraps it with CalibratedClassifierCV to improve the probability estimates. It uses isotonic regression with 3-fold cross-validation for calibration, then fits the calibrated model on the training data. This step ensures both strong predictive performance and reliable confidence scores.



This code evaluates the calibrated model's performance on the test set. It prints the accuracy and full classification report, plots a histogram of prediction errors (actual vs. predicted sale weeks), and displays a confusion matrix heatmap. These visualizations help identify how far off predictions are from the true weeks and which classes (weeks) the model confuses most frequently.



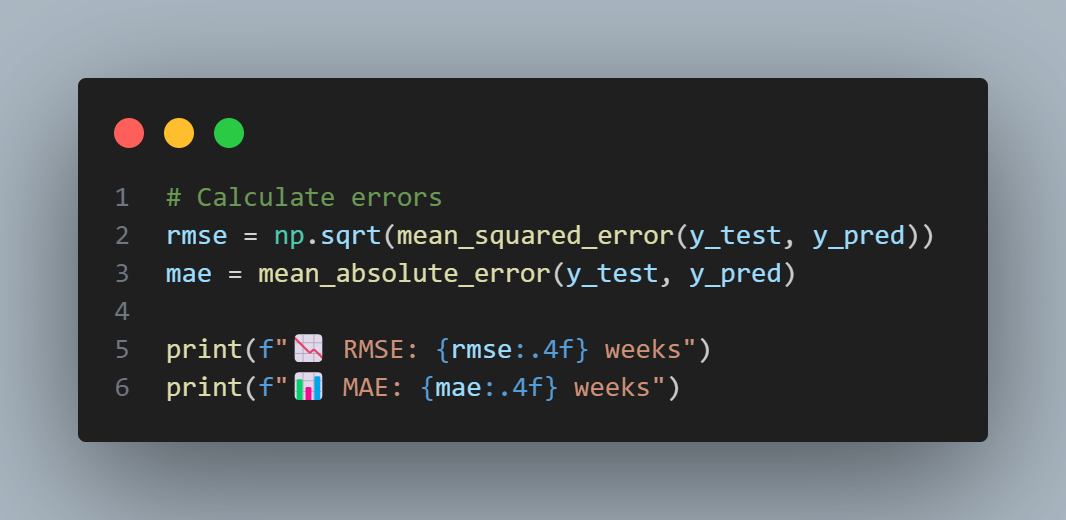
This code calculates and prints precision, recall, and F1 scores using three different averaging methods: **macro**, **micro**, and **weighted**. Macro averages treat all classes equally, micro averages aggregate over all instances, and weighted averages account for class frequency. These metrics provide a well-rounded evaluation of model performance, especially in multiclass settings with class imbalance.



This code generates a detailed classification report, extracts precision, recall, and F1-score for each sale week class (1–8), and visualizes them using a grouped bar chart. It helps analyze model performance per class and quickly identify underperforming or overfitted weeks, offering insights into how well the model predicts sales timing across the 8-week window.



This code performs an end-to-end prediction of the next sale week for a specific product using the trained hybrid model. It defines the item name and its associated structured features (like price, discount history, and frequency). These inputs are passed to the predict\_next\_sale\_week function, which combines BERT embeddings and structured data, runs the model, and returns both the predicted sale week and the full probability distribution across all weeks. The results are then printed for interpretation.



This code calculates and prints two key regression metrics: **Root Mean Squared Error (RMSE)** and **Mean Absolute Error (MAE)**. RMSE gives higher weight to larger errors, while MAE provides an average of absolute differences between actual and predicted sale weeks. These metrics are expressed in weeks and help evaluate the prediction accuracy of the model in temporal terms.