**CatBoost Tutorial: Handling Categorical Features in Gradient Boosting**

**Introduction**

In the landscape of supervised machine learning, gradient boosting has emerged as a highly effective ensemble technique. While methods like XGBoost and LightGBM have become mainstream, they still require significant preprocessing of categorical variables, such as label encoding or one-hot encoding. This often introduces complexity, performance trade-offs, or even data leakage. To solve these problems, CatBoost was introduced by Yandex in 2018 as an open-source, high-performance gradient boosting library that naturally handles categorical features (Prokhorenkova et al., 2018).

This tutorial presents a deep dive into CatBoost, explaining its innovations and strengths, and demonstrating its application using the Bank Marketing dataset from the UCI repository. As your instructor for this session, I will guide you step by step through the theory, code, and evaluation, much like we would in a graduate-level machine learning seminar.

**Real-World Analogy**

Imagine a chef preparing dishes for guests with unique dietary preferences. Traditional boosting algorithms (like XGBoost) ask the chef to manually translate every guest's needs into a binary format. In contrast, CatBoost allows the chef to directly understand and work with each preference in its original form. This makes the chef (i.e., the model) quicker, more accurate, and less error-prone.

**Why Use CatBoost?**

CatBoost excels in tasks involving **mixed-type tabular data**, where categorical variables are abundant. Its main advantages include:

* **Native Categorical Feature Handling**: Avoids manual encoding
* **Ordered Boosting**: Reduces overfitting and prediction shift
* **Symmetric Trees**: Improve training speed and robustness
* **Strong Default Parameters**: Minimal hyperparameter tuning needed

These characteristics make CatBoost suitable for production-grade applications, especially in finance, telecom, healthcare, and marketing.

**Dataset: Bank Marketing (UCI Repository)**

The dataset used here predicts whether a customer will subscribe to a term deposit. It includes 41,188 records with 20 features: age, job, marital status, education, contact method, previous campaign outcomes, economic indicators, and more.

Target variable: y (binary: "yes" or "no")

Link: [UCI Bank Marketing Dataset](https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank-additional.zip)

**Key Innovations in CatBoost**

**1. Ordered Target Statistics**

Traditional label encoding uses the entire dataset to convert categorical variables to numeric format. This can lead to data leakage. CatBoost solves this by using **ordered statistics**, where each example is encoded using only previous data points, simulating a real online learning process (Prokhorenkova et al., 2018).

**2. Ordered Boosting**

Standard gradient boosting can suffer from prediction shift if the same data is used to build trees and compute gradients. CatBoost introduces **ordered boosting**, using permutations of the dataset to compute gradients in an out-of-fold manner, thus reducing overfitting (Dorogush et al., 2018).

**3. Symmetric Trees (Oblivious Trees)**

Instead of arbitrarily shaped decision trees, CatBoost uses **symmetric trees**, where each level splits on the same feature and condition across all branches. This leads to:

* Faster inference
* Easier regularization
* Compatibility with hardware optimization (Dorogush et al., 2018)

**4. Minimal Tuning and Fast Training**

With intelligent default settings and automatic preprocessing, CatBoost requires very little manual configuration. This makes it ideal for rapid prototyping and enterprise use.

**Implementation Steps:**

Let us walk through the implementation phase like we would in a hands-on tutorial workshop. Each step is paired with code and an explanation to solidify understanding.

**Step 1: Exploratory Data Analysis (EDA)**

We begin by loading the dataset and inspecting the shape, column types, and missing values. No nulls were found, and 11 features were identified as categorical. The target class was imbalanced (~88% "no", 12% "yes"). Understanding this imbalance is important for evaluating model performance beyond simple accuracy.

A computer code with black text

AI-generated content may be incorrect.

A close-up of a white background

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

*Caption: Bar plot showing the distribution of the target variable (class imbalance)*

**Step 2: Preprocessing**

The target variable y was converted from string labels ("yes", "no") to numeric binary values (1, 0). CatBoost does not require manual encoding of categorical variables like one-hot or label encoding. Instead, we identify categorical columns and pass them directly.

A screen shot of a computer code

AI-generated content may be incorrect.

**Step 3: Train-Test Split**

We split the dataset into training and testing sets using stratification to maintain the same class distribution.

X = df.drop('y', axis=1)

y = df['y']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, stratify=y, test\_size=0.2, random\_state=42)

*Explanation: This maintains class proportions in both training and testing datasets.*

**Step 4: Model Training**

Let us now build our model. We'll configure a CatBoost classifier to train for 500 boosting iterations and evaluate it on the test set.

A screenshot of a computer code

AI-generated content may be incorrect.

**Step 5: Evaluation**

After training, we evaluate the classifier's performance.

A computer screen shot of a program code

AI-generated content may be incorrect. A blue squares with numbers and a white background

AI-generated content may be incorrect.

Figure Confusion matrix showing prediction breakdown

A screen shot of a computer code

AI-generated content may be incorrect.

A graph of a positive rate

AI-generated content may be incorrect.

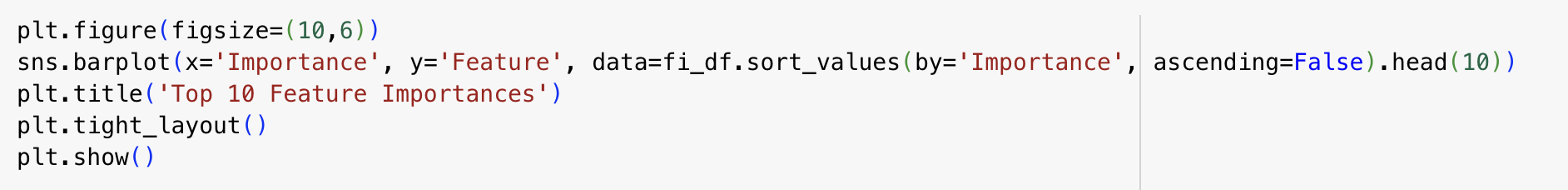
Figure ROC curve illustrating the model's discriminative ability

**Step 6: Feature Importance**

Let’s examine which features contributed most to the model.

A close-up of a computer code

AI-generated content may be incorrect.

**

A graph with blue bars

AI-generated content may be incorrect.

Figure Top 10 features influencing CatBoost decisions

**Final Summary**

This tutorial presented a comprehensive, instructor-led walkthrough of CatBoost, a state-of-the-art gradient boosting algorithm designed specifically to handle categorical features natively. Through a real-world use case using the Bank Marketing dataset, we explored CatBoost's innovative handling of data via ordered target statistics, ordered boosting, and symmetric tree structures. These innovations enable more robust, efficient, and leak-resistant model training compared to traditional boosting libraries.

The implementation section guided graduate-level students step-by-step through the entire modeling process: from data exploration and preprocessing to training, evaluation, and feature interpretation. We demonstrated how CatBoost eliminates the need for manual encoding, and showcased its superior performance through metrics like ROC-AUC, confusion matrices, and feature importance plots—all with accessibility in mind.

By combining theoretical insights with reproducible Python code and visual aids, this tutorial equips students and practitioners alike to confidently apply CatBoost to real-world tabular datasets where categorical features play a central role. A GitHub repository ensures full transparency, accessibility, and reusability of the tutorial content for independent learning and deployment.

**Accessibility Statement**

To ensure this tutorial is accessible to the widest possible audience, the following measures have been taken:

**Visual Accessibility**

* **Colorblind-Friendly Palettes**: All plots use color schemes like Blues and viridis to accommodate colorblind users.
* **Alt-Text Descriptions**: Captions provided for each visual describe the content and purpose.
* **Large Fonts and High Contrast**: Code outputs and figures are formatted for readability.

**Screen Reader Support**

* **Proper Headings**: The tutorial is organized using Markdown heading levels to support navigation.
* **Descriptive Titles**: Figures, code blocks, and sections use meaningful labels for clarity.
* **Semantic Structure**: Lists and subheadings allow screen readers to parse and relay the structure effectively.

**Captioning and Code Comments**

* **Markdown Commentary**: Each code block is paired with written explanations for those who rely on screen readers or prefer reading over visual output.
* **Step-by-Step Instructions**: Code is documented and explained thoroughly to help all learners follow the logic regardless of their preferred learning style.

These efforts ensure that the tutorial aligns with inclusive education standards and offers a productive experience for users with diverse learning needs.

**GitHub Repository Accessibility**

All code, notebook, tutorial PDF, and supporting files are made available on GitHub for reproducibility.

**GitHub Repository:** <https://github.com/Muhammadnawab/individual_assignment_machine_learning.git>

This repository includes:

* Jupyter Notebook (.ipynb)
* Tutorial Report (.pdf)
* README.md with installation instructions
* Open-source MIT License

**References**

Dorogush, A. V., Ershov, V., & Gulin, A. (2018). *CatBoost: gradient boosting with categorical features support*. arXiv preprint arXiv:1810.11363. https://arxiv.org/abs/1810.11363

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). *CatBoost: unbiased boosting with categorical features*. Advances in Neural Information Processing Systems, 31. https://papers.nips.cc/paper\_files/paper/2018/hash/14491b756b3a51daac41c24863285549-Abstract.html

UCI Machine Learning Repository. (n.d.). *Bank Marketing Dataset*. https://archive.ics.uci.edu/ml/datasets/bank+marketing