Name:
Bank Term Deposit Subscription Prediction

Project Title:
Predicting Customer Subscription to Bank Term Deposits Using Machine Learning

Business/Problem Context:
Banks often run marketing campaigns to encourage customers to subscribe to term deposits. However, reaching out to all customers is costly and inefficient. Many customers are unlikely to subscribe, while a small subset is highly likely. By predicting which customers are most likely to subscribe, banks can:

- Reduce marketing costs
- Increase campaign effectiveness
- Improve customer targeting and satisfaction

Objective Statement:
Build and deploy a machine learning model to predict whether a customer will subscribe to a term deposit. The goal is to achieve a high predictive performance while handling class imbalance, enabling the bank to prioritize customers with the highest probability of conversion.

Problem Type:
This is a **supervised machine learning classification problem**.
The goal is to predict whether a customer will subscribe to a term deposit after a marketing campaign.

Target Variable:
y → Binary categorical outcome:

- "yes" → client subscribed
- "no" → client did not subscribe

Key Input Features (Predictors):

- **Demographic / Client Information**
  - age (numeric)
  - job (categorical: management, technician, etc.)
  - marital (categorical: married, single, divorced)
  - education (categorical: primary, secondary, tertiary, unknown)
  - default (binary: yes, no)
  - housing (binary: yes, no)
  - loan (binary: yes, no)
- **Contact & Campaign-Related Information**
  - contact (categorical: unknown, cellular, telephone)
  - day (numeric: last contact day of the month)
  - month (categorical: jan, feb, ..., dec)
  - duration (numeric: last contact duration in seconds)
  - campaign (numeric: number of contacts during this campaign)
  - pdays (numeric: days since client was last contacted; -1 = never contacted)
  - previous (numeric: number of contacts before this campaign)
  - poutcome (categorical: outcome of previous campaign)

Assumptions:

1. All records in the dataset are independent.
2. Target variable y is correctly labeled.
3. Past contact information (pdays, previous, poutcome) is reliable and consistent.

Dataset Source:

Link - bank-full-dataset.csv (As given by administrator)

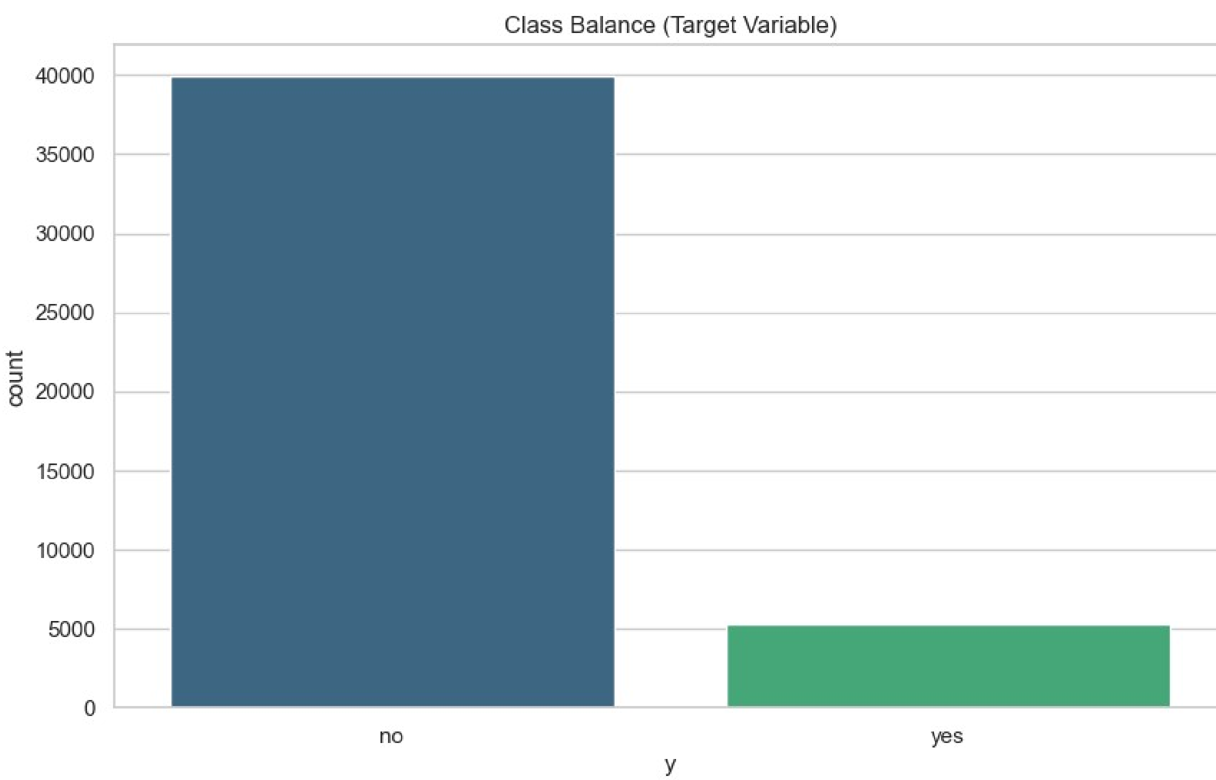
- Dataset was quite clean
- The target balance was not quite favorable (like 1:10 ratio, kind of), which might have impacted my recall rate.

Data Description:

- Shape - (45211, 17)
- Features - ['age', 'job', 'marital', 'education', 'default', 'balance', 'housing', 'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays', 'previous', 'poutcome', 'y']
- Missing Value Check –

```
Missing Values per Column:
 age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

- Class Balance (Target Variable – ‘y’):

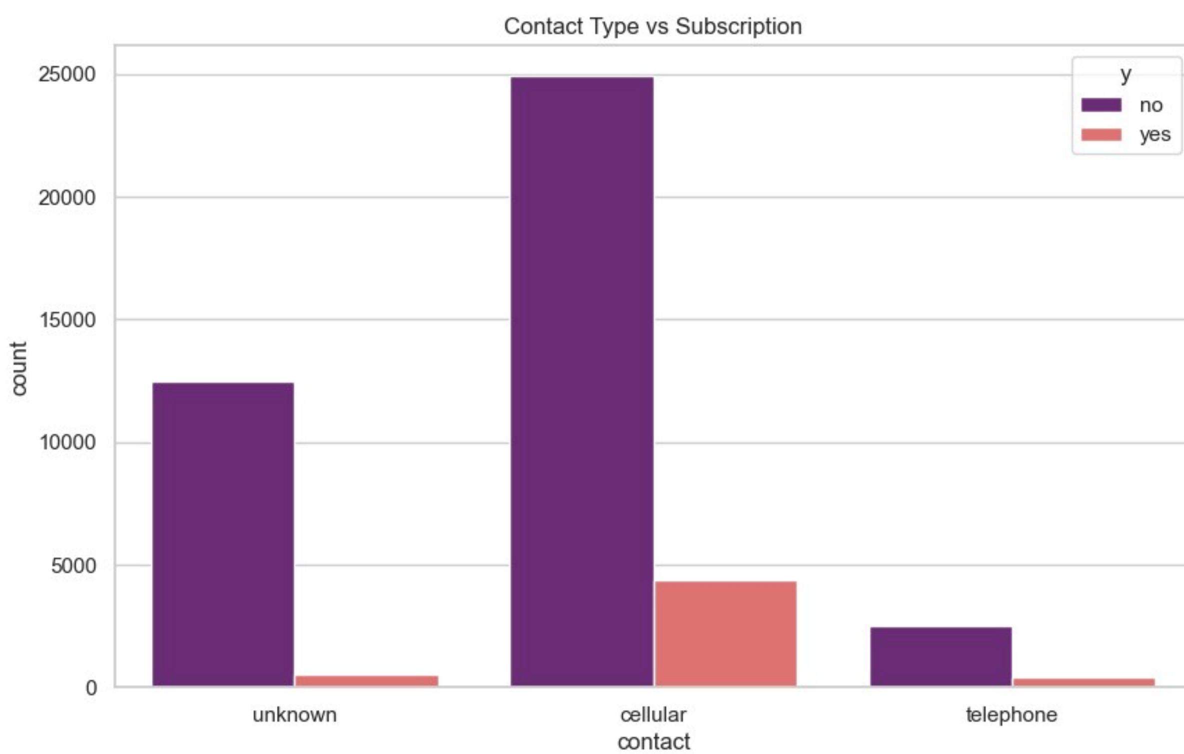


Preprocessing:

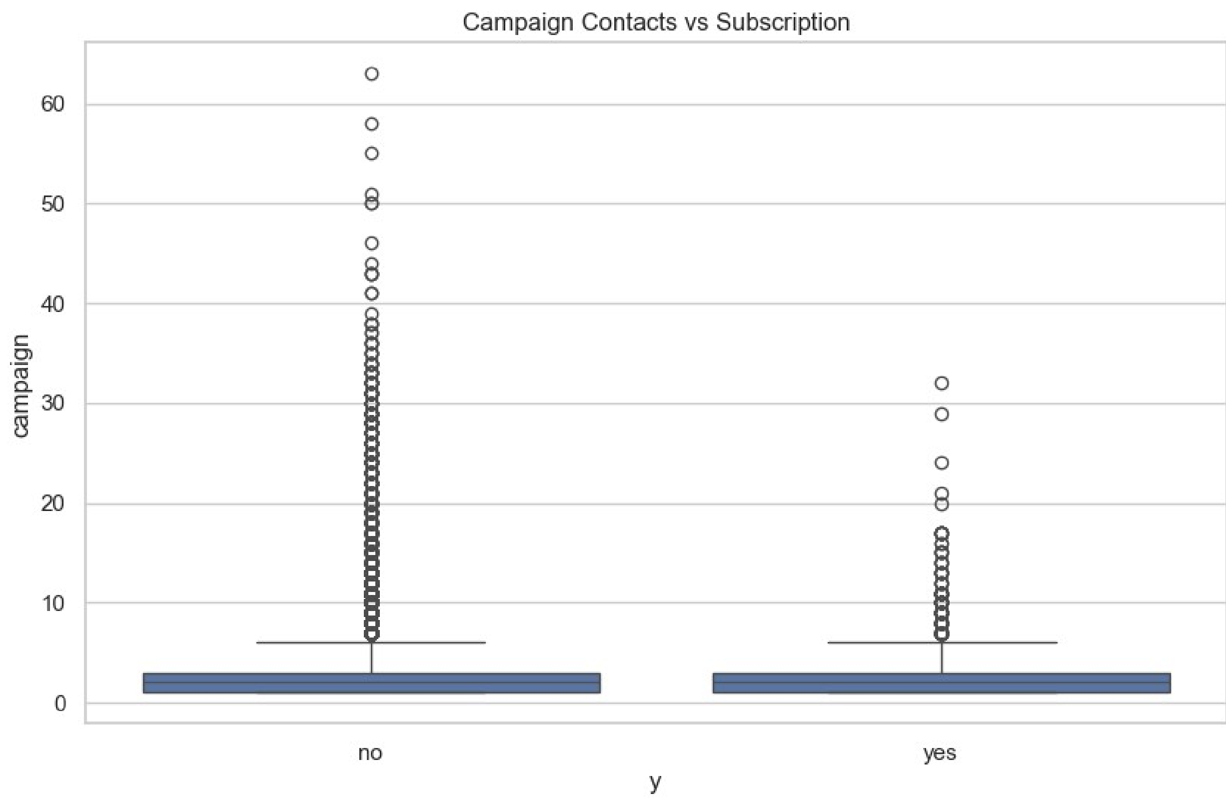
- Outlier Detection –
  - These are **real customer cases**, not data errors. Removing them could throw away valuable information about high-net-worth customers, who might behave very differently in term deposit subscriptions.
  - The dataset is already **imbalanced** (~11–12% "Yes", ~88–89% "No"). Outlier removal tends to disproportionately drop rare positive cases (e.g., people who subscribed but had unusual balances/durations).
  - Unlike linear models, **tree-based models are robust to outliers**. So outliers don't distort the model — they just become part of separate branches.
- Encoding –
  - For features like age, balance, duration, campaign, etc., you kept them **as is** (the model handles numeric features natively). No scaling/normalization was needed because tree-based models (like CatBoost) split based on thresholds, not distances.
  - For features like job, marital, education, default, housing, loan, contact, month, poutcome → you applied **OneHotEncoder** inside a ColumnTransformer. This means each category (e.g., job=admin. or job=technician) was converted into a **binary column** (0/1).

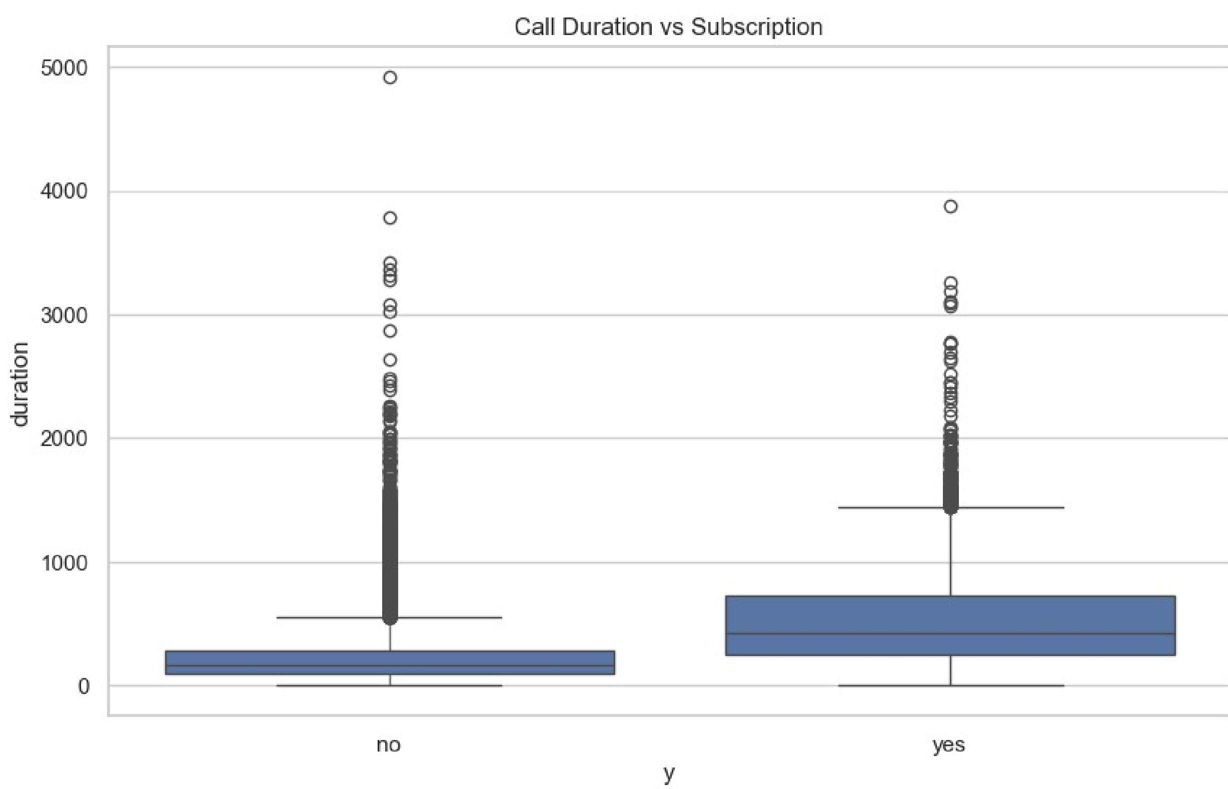
Explatory Data Analysis:

- **Contact Type vs Subscription (Target)**



- **Campaign Variables vs Subscription (Target)**





Models:

1. Logistic Regression (Baseline)

- It’s a simple, interpretable linear model.
- Provides probability outputs, which are useful for business decision-making (e.g., probability of term deposit subscription).
- Serves as a baseline model to compare more advanced methods against.

2. Decision Tree

- Intuitive and easy to visualize/interpret — good for explaining “decision rules” to business users.
- Naturally handles both numeric and categorical data (after encoding).
- Captures nonlinear relationships and feature interactions.

3. LightGBM (Gradient Boosting Trees)

- One of the fastest and most efficient boosting algorithms.
- Handles large datasets and imbalanced classes effectively.
- Known for high accuracy in tabular data problems.

4. CatBoost (Categorical Boosting)

- Specifically designed to handle categorical features directly, without heavy preprocessing (like one-hot encoding).
- Excellent performance on structured/tabular data, especially marketing datasets with many categorical variables.
- Provides state-of-the-art performance with less hyperparameter tuning compared to other boosting methods.

Feature Engineering:

- **Domain Knowledge Features:**
    - Age Bucket - Instead of using age as a raw continuous variable, you're grouping customers into **age categories**. Bucketing helps capture **nonlinear relationships** — for example, "being middle-aged" may matter more than the exact age.
    - Contact History Indicators - **contacted_before:** Converts the "pdays" column (days since last contact) into a binary flag → whether the customer was contacted before or not. **repeat_contact:** Checks if the client had multiple contacts before the current campaign. **Past contact history is a strong predictor of whether someone will subscribe.**
    - Call Duration Flag - Creates a binary feature that identifies whether the call duration was **short (< 1 min)** or not. Duration is one of the **most important predictors** in this dataset but converting it into a flag makes it easier for some models to interpret.
- Transformations:
    - Creates a combined feature that multiplies account balance with age. Captures the **interaction** between wealth and age → e.g., a high balance for a young customer may mean something different than the same balance for an older customer.
    - Creates a binary variable showing if a customer has **both a housing loan and a personal loan**. Customers with multiple loans may be **financially constrained** and less likely to invest in a term deposit.

Hyperparameter Tuning:

- **Decision Tree (GridSearchCV):**
    - For each combination, it:
        1. Trains the decision tree.
        2. Evaluates performance using cross-validation (e.g., ROC-AUC).
        3. Picks the best hyperparameter set.
    - The hyperparameters (e.g., max_depth, min_samples_split, criterion) are discrete and manageable in number.
    - This ensures we **don't miss** the best hyperparameters within the search grid.

- **LightGBM (Optuna):**
  - It proceeds as follow:
    1. Starts with random trials.
    2. Uses results to build a probability model of the search space.
    3. Chooses the next hyperparameters based on past performance (tries promising regions more often).
  - LightGBM has many hyperparameters (e.g., num_leaves, learning_rate, feature_fraction, bagging_fraction). Searching them exhaustively with GridSearch would be too slow. Optuna efficiently finds a **good combination in fewer trials**.
- **Catboost (Optuna):**
  - We tuned hyperparameters like:
    1. iterations (number of boosting rounds),
    2. depth (tree depth),
    3. learning_rate,
    4. l2_leaf_reg (regularization),
    5. bagging_temperature (sampling strategy),
    6. random_strength (feature randomness).
  - CatBoost has **continuous + discrete hyperparameters**.
  - Optuna can explore promising regions more intelligently, improving **ROC-AUC** with fewer trials.

Pipeline:

1. **Built a preprocessing block**
   - Scaled all numeric features so they are on the same range.
   - Converted categorical features into one-hot encoded vectors (dummy variables).
   - Ensured new/unseen categories don't break the model (ignore option).
2. **Wrapped preprocessing + model into a pipeline**
   - Each model (Logistic Regression, Decision Tree, LightGBM, CatBoost) was combined with the same preprocessing block.
   - way, you didn't need to preprocess separately for each model.
3. **Integrated with model training and tuning**
   - Logistic Regression was used as a baseline.
   - Decision Tree was tuned with GridSearch.
   - LightGBM and CatBoost were tuned with Optuna.
   - Each optimized model was placed inside its pipeline.

4. **Evaluated all models consistently**
    - Trained each pipeline on the training set.
    - Predictions were made using the same pipeline (so preprocessing and prediction were done in one go).
    - Metrics like Accuracy, Precision, Recall, F1, ROC-AUC, RMSE were calculated and compared.
5. **Saved the final pipeline**
    - Instead of saving only the trained model, you saved the entire pipeline.
    - This makes deployment easy (e.g., in Streamlit/Flask) because raw input data goes straight into the pipeline → preprocessing → model → prediction.

Evaluation Metrics:

	Model	Accuracy	Precision	Recall	F1	ROC-AUC	RMSE
3	CatBoost	0.910870	0.663212	0.483932	0.559563	0.935487	0.298546
2	LightGBM	0.908769	0.653088	0.469754	0.546454	0.932280	0.302044
0	Logistic Regression (Baseline)	0.845737	0.418244	0.814745	0.552741	0.907922	0.392763
1	Decision Tree	0.796970	0.350269	0.860113	0.497812	0.892971	0.450588

Error Analysis:

1. **Class Imbalance Effect**
    - The dataset has far more “No Term Deposit” cases than “Yes”.
    - Even though class weights and threshold adjustments were used, the model sometimes still predicts “No” too often.
    - This leads to **false negatives** (customers who actually subscribed but were predicted as non-subscribers).
2. **Short Call Durations**
    - Records with very short call durations (e.g., < 1 minute) often confuse the model.
    - Some customers might still subscribe even after a short call, but the model tends to treat short calls as a strong “No” signal.

3. **Ambiguous Socio-Economic Profiles**
    - Middle-aged customers with average balance and education fall into a "gray zone".
    - The model sometimes misclassifies these groups because their patterns overlap between "Yes" and "No".
4. **Previous Contact History**
    - Customers not previously contacted are harder to predict.
    - The model struggles because "no contact history" can either mean a missed opportunity or genuine disinterest.

Business Interpretation:

1. **High ROC-AUC (≈0.93+)**
    - The model is very effective at ranking customers by likelihood to subscribe.
    - This means marketing teams can **prioritize who to call first**, increasing efficiency.
2. **Precision vs. Recall Trade-off**
    - If threshold is set at 0.5, the model favors precision (avoiding calling uninterested customers) but misses some real subscribers.
    - By lowering the threshold (e.g., 0.3), recall improves → fewer missed opportunities, but at the cost of more calls to uninterested clients.
3. **Business Actionable Insight**
    - The model identifies **key groups more likely to subscribe** (e.g., longer calls, previous positive contact, certain age/education groups).
    - Marketing campaigns can be tailored to focus on these profiles.
4. **Cost Savings & ROI**
    - By targeting high-probability customers, the bank can reduce wasted calls.
    - This saves agent time, reduces campaign costs, and improves conversion rates.