**ML Applications in Business and Economics**

**High-Level Objectives**

1. **Data Understanding and Exploration**  
   The Jupyter notebook should first provide a thorough understanding of the dataset by performing exploratory data analysis (EDA) and referencing the data dictionary to clarify feature meanings.
2. **Data Preprocessing and Feature Engineering**  
   Prepare the dataset for modeling by handling missing values, encoding categorical features, and engineering additional predictive features from the given variables.
3. **Model Development and Hyperparameter Tuning**  
   Train at least one predictive model (e.g., a gradient boosting machine) and tune its hyperparameters for optimal performance.
4. **Model Evaluation and Interpretation**  
   Evaluate the model’s expected revenue impact based on the given cost-benefit structure. Additionally, apply interpretability techniques (e.g., feature importance, SHAP values) to understand the model’s predictions.
5. **Final Predictions and Submission**  
   Use the best model to generate predictions on the test dataset, apply all preprocessing steps, and export the results as test-predictions.csv containing only the ‘customernumber’ and the model’s predicted probabilities (or binary predictions).

**Detailed Step-by-Step Plan**

**A. Jupyter Notebook Structure and Narrative**

* **Introduction and Problem Statement**:  
  Start with a markdown cell describing the business problem. Explain that the retailer wants to identify which customers should receive a €5 voucher to maximize revenue. Summarize the cost/revenue logic:
  + If predicted “no purchase” → send voucher:
    - If they actually wouldn’t have purchased: +€2.50 expected revenue.
    - If they actually would have purchased: -€5.00 loss.
  + If predicted “purchase” → no voucher:
    - If they purchase: no extra cost/revenue.
    - If they don’t purchase: no extra cost/revenue.
* **Data Sources and Dictionary**:  
  Include a section describing the provided features, using the data dictionary. Summarize key features:
  + **Customer Metadata**: customernumber, date, datecreated
  + **Demographics & Customer Profile**: salutation, title, domain (email provider), newsletter subscription, paymenttype, deliverytype
  + **Geographic Info**: invoicepostcode, delivpostcode (could indicate region or patterns in purchasing)
  + **Order/Item Attributes**: case (value segment of goods), numberitems, gift (whether it was a gift), entry (entry channel), points redeemed, shippingcosts indicator, weight, remi (number of remitted items), cancel (number cancelled), used (number used items)
  + **Product Categories**: w0 to w10 representing different book and media categories ordered
  + **Target Variable**: target90 (1 = reorder within 90 days, 0 = no)

**B. Exploratory Data Analysis (EDA)**

* **Data Loading**:  
  Load train.csv and test.csv into pandas DataFrames. Display the first few rows to understand the structure.
* **Basic Statistics**:  
  Show value counts and distributions:
  + target90 distribution to understand class imbalance.
  + Categorical features (e.g., salutation, newsletter) distribution.
  + Numeric features (e.g., numberitems, weight, w0...w10) summary statistics (mean, median, std).
* **Missing Values**:  
  Identify which columns have missing values and their percentages. Visualize with a heatmap or bar plot for quick insight.
* **Feature Relationships**:
  + Correlation matrix for numeric features to identify highly correlated variables.
  + Grouped statistics: For instance, average numberitems for target90=1 vs target90=0.
  + Possibly investigate geographic or categorical factors that might influence repeat purchasing.
* **Key Insights**:  
  Note which features seem predictive. For instance, customers who ordered more items (numberitems) or certain categories (w2, w3 etc.) may be more or less likely to reorder. Those who subscribed to the newsletter or used certain payment/delivery types might show different reorder behavior.

**C. Data Preprocessing**

* **Data Cleaning**:
  + Handle Missing Values:  
    For categorical variables (like domain), consider imputing missing values with a special category or using mode imputation. For numeric variables, consider median or mean imputation. If date fields are missing, consider whether that record should be excluded or if the missing date conveys a message.
* **Feature Encoding**:
  + Convert categorical variables (salutation, title, domain, newsletter, paymenttype, deliverytype, gift, entry, points, shippingcosts) into numeric form.  
    Consider one-hot encoding or, if high-cardinality (for something like domain), maybe grouping rare domains under “others” or using target encoding.
* **Feature Creation & Transformation**:
  + Date Features: From date (first order date) and datecreated (account opening), create features like “days since account creation at order time”, “month/season of first order” to capture seasonality.
  + Aggregate categories: Combine w0 to w10 into total ordered items, or ratios of certain categories to total items. Possibly encode diversity of purchase (number of categories >0).
  + Shipping/Returns: High remi (remitted items), cancel, or used items might indicate dissatisfaction or browsing behavior. Include these as is or create a ratio metric (remitted items / numberitems).
  + Value Segmentation: case indicates the value range. Consider treating case as ordinal and use it directly or bin into fewer meaningful segments.
  + Region-based encoding from postcodes: If feasible, extract region or cluster postcodes (if they correlate with behavior), but be careful to avoid overfitting.

**D. Modeling Approach**

* **Model Selection**:
  + Start with a **Logistic Regression** as a baseline (simple, interpretable).
  + Move to more sophisticated models: **Random Forest** or **Gradient Boosted Trees (e.g., XGBoost, LightGBM)**, as they tend to perform well on structured data.
* **Probability Calibration**:
  + The model’s predicted probability must be well-calibrated. After training, check calibration curves and consider Isotonic Regression or Platt scaling if probabilities are off.
* **Hyperparameter Tuning**:
  + Use a validation set or cross-validation approach.
  + For tree-based models, tune parameters like max\_depth, learning\_rate, n\_estimators, subsample etc.
  + Use automated methods (GridSearchCV, RandomizedSearchCV, or Bayesian optimization) to find the best parameters balancing performance and complexity.

**E. Custom Evaluation Metric: Expected Revenue**

* **Defining the Threshold**: Recall that the expected revenue for sending a voucher is positive if p < 0.333. Thus, once probabilities are obtained, predictions should be:
  + If p(customer buys anyway) < 0.333, predict “no purchase” → send voucher.
  + Otherwise, predict “purchase” → no voucher.
* **Measuring Model Performance**:
  + Compute expected revenue on the validation set:
    - For each customer i:
      * p\_i = predicted probability that the customer will buy anyway.
      * If we predict voucher (i.e., p\_i < 0.333):
        + True scenario: target90=0 → +€2.50
        + False scenario: target90=1 → -€5.00
      * If we predict no voucher (p\_i ≥ 0.333):
        + No incremental revenue or loss.
    - Calculate total expected revenue over all customers.
  + Compare this expected revenue to a baseline (e.g., sending vouchers to everyone yields +€2.50 \* proportion of non-buyers).
* **Other Metrics**:
  + AUC-ROC or precision/recall might still be informative, but the final decision and evaluation focus on revenue gain.

**F. Interpretation of the Model**

* **Global Interpretability**:
  + Feature importance from Random Forest / Gradient Boosted Trees:  
    Identify which features most strongly influence predictions (e.g., number of items, subscription to newsletter, certain product categories).
* **Local Interpretability**:
  + Use SHAP values (SHapley Additive exPlanations) to see how each feature contributes to the prediction for individual customers.
  + Show a few examples in the notebook to illustrate which characteristics lead the model to assign low probabilities (and thus send a voucher) vs. high probabilities.

**G. Finalizing the Model and Generating Test Predictions**

* **Retrain Best Model on Full Training Data**: After deciding on hyperparameters and final preprocessing steps, retrain the model on the entire training set for best performance.
* **Apply Preprocessing to Test Data**: Perform the exact same preprocessing steps (imputation, encoding, feature engineering) on the test.csv data.
* **Predict on Test Set**:
  + Generate predicted probabilities for each test customer.
  + Apply the probability threshold of 0.333 to decide on “voucher vs. no voucher”.
  + The instructions for submission say to produce a CSV with customernumber and prediction. Here, clarify if they expect a binary prediction (0/1) or a probability. The description “Only include the customernumber and prediction columns” suggests providing a predicted label. Since the solution’s main decision is binary (send voucher or not), output prediction = 1 if voucher is sent (i.e., p < 0.333) and prediction = 0 if not.
* **Export test-predictions.csv**: The CSV should contain:

customernumber,prediction

<id1>,<0 or 1>

<id2>,<0 or 1>

...

**H. Documentation and Visualization**

* **Notebook Documentation**:
  + Use markdown cells to explain each step.
  + Provide reasoning behind feature engineering, choice of model, and evaluation methods.
  + Include plots:
    - Histograms of numerical features
    - Count plots of categorical features
    - Calibration plot for model probabilities
    - SHAP summary plot for interpretability
* **Results Discussion**:
  + Summarize the final expected revenue gain vs. baseline.
  + Discuss any limitations and potential improvements.

**I. Potential Enhancements (If Time Allows)**

* **Advanced Feature Engineering**:
  + Segment customers by domain (e.g., free email providers vs. corporate domains).
  + Investigate postal codes to see if certain regions correlate with higher reorder rates.
* **Time-Based Validation**:
  + If the data spans different time periods, consider a time-based split for validation to ensure generalization to future customers.
* **Ensembling**:
  + Combine predictions from multiple models (e.g., Logistic Regression, Random Forest, Gradient Boosted Trees) using a weighted average or stacking to potentially improve calibration and performance.

**Summary**

By following these steps, the resulting Jupyter notebook will:

1. Clearly lay out the problem and data understanding.
2. Perform EDA and meaningful data preprocessing.
3. Develop a well-calibrated predictive model, tuned for hyperparameters.
4. Evaluate success using a custom expected revenue metric tied directly to the business objective.
5. Provide both global and local interpretations of the model results.
6. Produce final predictions on a test set and write them out to test-predictions.csv for submission.