Sakaya Ammaruekhachok (6758100956)

Class 5 Assignment

Data Preparation

To start the project, I converted the provided dataset into a spreadsheet format and organized it into a clean, machine-learning-ready structure. After that, I performed data cleaning to fix inconsistent categorical values. One important correction was in the Gender column, where the value “Fe Male" was standardized to "Female" for consistency.

28	Self Enquiry	1	7	Large Business	Male	
32	Self Enquiry	1	8	Large Business	Male	
46	Self Enquiry	3	6	Small Business	Female	
36	Company Invited	1	32	Salaried	Fe male	
43	Self Enquiry	1	9	Small Business	Female	
42	Self Enquiry	3	20	Small Business	Female	
43	Self Enquiry	1	18	Salaried	Male	

I applied this cleaning to all split files (train, validation, and test), which ensured that category labels were uniform before training. In total, 121 values were corrected across the three files. This step is important because inconsistent category labels can create duplicate categories and reduce model performance.

How we split data

I split the dataset into three separate files for a proper machine learning workflow:

- Training set (70%)
- Validation set (15%)
- Test set (15%)

The split was done using a stratified split on ProdTaken, which means the class ratio (buyers vs non-buyers) was preserved across all three sets. This is important because the dataset is imbalanced, and stratification helps make training and evaluation fair.

The final split sizes were:

- Train: 2245 rows
- Validation: 481 rows
- Test: 482 rows

This setup allowed me to:

- Train the model on the training set,
- Tune hyperparameters and threshold using the validation set, and evaluate final performance on a completely unseen test set.

Analysis (Updated)

The dataset contains a mix of numerical and categorical features describing customer information and travel package-related attributes. The target variable is ProdTaken, which indicates whether a customer purchased the travel package (1 = Yes, 0 = No).

From the test-set confusion matrix results, the model performance shows that the data remains imbalanced, but the model is still able to detect the positive class reasonably well. The final test confusion matrix (CatBoost model) was:

True Negatives (TN): 372
False Positives (FP): 17
False Negatives (FN): 25
True Positives (TP): 68
This gives a test accuracy of:
0.9129

So the final test accuracy is 91.29%, which is a strong result.

I also generated a normalized confusion matrix, which shows:

For class 0 (non-buyers), the model correctly predicts about 93%

For class 1 (buyers), the model correctly predicts about 67%

This indicates the model is stronger at identifying non-buyers, but still performs reasonably well on the minority buyer class.

Feature Extraction

Since the dataset contains both text-based and numeric features, preprocessing is required before training.

I used two approaches during development:

(A) Neural Network pipeline (first approach)

For the ANN model, I used a preprocessing pipeline that included:

One-Hot Encoding for categorical columns

Standard Scaling for numerical columns

feature alignment between train/validation/test sets

class weighting for imbalance handling

This converted all features into a numeric format suitable for Keras.

(B) CatBoost pipeline (final approach)

For the final optimized model, I used CatBoostClassifier, which can:

handle categorical columns directly

handle missing values natively

work very well on tabular business/customer data

This was a major improvement because CatBoost is generally stronger than ANN for structured tabular datasets like this one.

Building Model

I developed and tested multiple models during this assignment:

Model 1: Neural Network (Keras ANN)

I first built a deep learning model using Keras (Sequential API) with multiple dense layers, dropout, batch normalization, and sigmoid output for binary classification. The model was trained using:

Binary Crossentropy loss
Adam optimizer
Early stopping
Class weights to address imbalance

This version produced good results (around the high-0.8x range) and helped build the full training/inference pipeline.

Model 2: Improved ANN (v2)

I then improved the neural network pipeline by:

adding a more robust preprocessing pipeline (ColumnTransformer)

handling missing values with imputers

tuning the classification threshold on the validation set

saving the best threshold for inference

This improved stability and performance, but the best accuracy was still below my target.

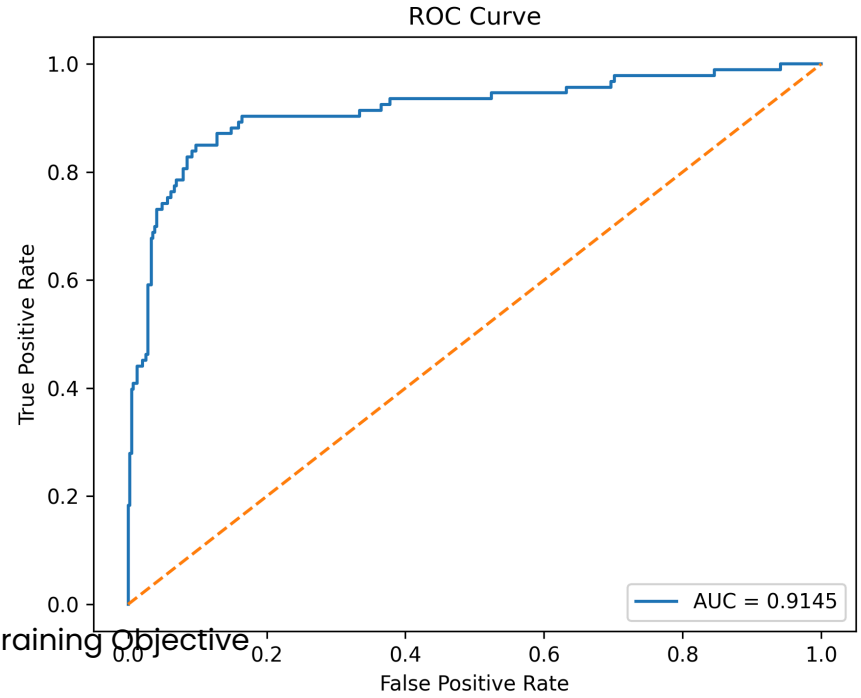
Model 3: CatBoost (Final model)

To push performance higher, I switched to CatBoostClassifier, which is better suited to tabular customer data.
The CatBoost model was trained with:

loss_function = "Logloss"
eval_metric = "AUC"
scale_pos_weight for class imbalance
early stopping using the validation set

threshold tuning on validation probabilities to maximize accuracy

This model gave the best results and achieved a test accuracy of 91.29% and ROC AUC = 0.9145.



Loss Function / Training Objective

This is a binary classification problem, so the core training objective was to predict whether a customer would purchase a package (ProdTaken = 1) or not (ProdTaken = 0).

ANN models

For the ANN models, I used:
Binary Crossentropy as the loss function
Accuracy and AUC as monitoring metrics, The goal was to minimize binary crossentropy while improving classification performance on the validation set.

CatBoost final model
For the final model, CatBoost used:
Logloss as the optimization objective
AUC as the evaluation metric during training

In addition, I tuned the final classification threshold on the validation set instead of always using 0.50. This improved the final test accuracy and made the model predictions more balanced for this dataset.

7. Results

The final model used for evaluation was CatBoost, and it produced the following test-set results:

Final Test Performance

Accuracy: 91.29%

ROC AUC: 0.9145

Confusion Matrix (Test Set)

TN = 372

FP = 17
FN = 25
TP = 68

Normalized Confusion Matrix
Class 0 recall: 0.93
Class 1 recall: 0.67

This shows the model performs very well on the majority class and still captures a useful portion of the minority class.

ROC Curve

The ROC curve reached an AUC of 0.9145, which indicates strong ranking performance (the model is good at separating positive and negative cases based on probability scores).

Training Curve (ANN experiment)

I also generated training history plots (accuracy and loss) for the ANN experiment. The plots showed:

training and validation accuracy both reached high values (~0.96–0.99),

validation loss remained relatively low,

but the final CatBoost model still gave better tabular-data performance overall.