

In the submitted model for Class 5, I built a binary travel product classification pipeline using a dataset with **3,208 records** and **19 columns** ('18` input features and `1` target label, `ProdTaken`). The data was split into **2,566 training samples** and **642 test samples**, with class distribution kept visible during evaluation ('train: 0=2071, 1=495'; `test: 0=518, 1=124').

For data preparation, I separated features from the target (`ProdTaken`), applied one-hot encoding for categorical columns, and standardized the encoded feature matrix using `StandardScaler` so inference used the same feature scale as training. I also saved preprocessing artifacts (`scaler` and feature-column list) to enforce consistent test-time transformation. The test data was transformed with the exact training feature schema by adding missing one-hot columns and reordering columns to match training before prediction. This step prevented feature mismatch and made evaluation reproducible.

From analysis of the dataset and class distribution, I observed a clear class imbalance where class `0` (no product taken) is much more frequent than class `1` (product taken). This imbalance is visible in both train and test splits and directly affects recall for the positive class. Because of this, I treated **recall** as a key metric in addition to accuracy, since a high-accuracy model could still miss many actual positive cases. The confusion matrix and classification report were used as the main analysis evidence to verify behavior per class, not only overall score.

For feature extraction, the final pipeline used encoded categorical features plus scaled numeric representations. The decision to keep one-hot encoded categorical information came from the need to preserve non-ordinal category meaning, while scaling was applied to keep feature magnitudes consistent for model inference workflow. I also ensured the exact same engineered feature space between train and test by storing `feature_columns.pkl` and applying it during inference; this made the extracted feature vector stable and prevented leakage or accidental schema drift.

For model building, the inference pipeline loads a trained neural network (`travel_classification_model.h5`) and performs binary classification with probability thresholding at `0.5`. The implementation predicts probabilities, converts them to class labels, and computes evaluation metrics including confusion matrix and classification report. Supporting artifacts and outputs were generated in the Class 5 workspace (`models/` and `output5/`) so the model behavior can be traced from preprocessing through final predictions. This setup gives a complete deployable flow: preprocess -> align features -> scale -> predict -> evaluate.

For evaluation results on the held-out test set (`n=642`), the model achieved **accuracy = 0.9065 (90.65%)** and **recall = 0.7177 (71.77%)** for class `1`. The confusion matrix was `[[493, 25], [35, 89]]`, meaning the model correctly identified `89` true positives and missed `35` positives (false negatives), while correctly classifying `493` negatives. These numbers indicate strong overall performance with good positive-class detection, though recall leaves room for improvement if the goal is to reduce missed positive cases further. Overall, the

submitted pipeline is reproducible, quantitatively supported, and aligned with the required ML reporting structure.

