Travel Product Classification — Pipeline Report

Executive Summary

In the submitted model, I built a Keras neural network to predict whether a customer will purchase a travel product (**ProdTaken**) from a dataset of 3,208 customer records.

I first cleaned the data — fixing a gender typo ("Fe Male" → "Female", 121 rows) and replacing outliers in **DurationOfPitch** (127 min) and **NumberOfTrips** (20–22) with median values.

I also addressed class imbalance. The original data was 81% negative / 19% positive, so I applied random oversampling to balance the classes to 50/50 (5,178 total rows). I considered using SMOTE but found simple oversampling sufficient for this dataset size.

I initially trained the baseline architecture from class (32→8→4→1) which achieved 88% accuracy, but I wanted better results. I experimented with wider networks, BatchNormalization, and learning rate scheduling.

I found that a wider, shallower network (256→128→1) with BatchNormalization and ReduceLROnPlateau significantly outperformed the baseline.

The final model achieved 96.45% accuracy and 0.99 AUC on the held-out test set — an improvement of ~8 percentage points over the baseline.

Pipeline

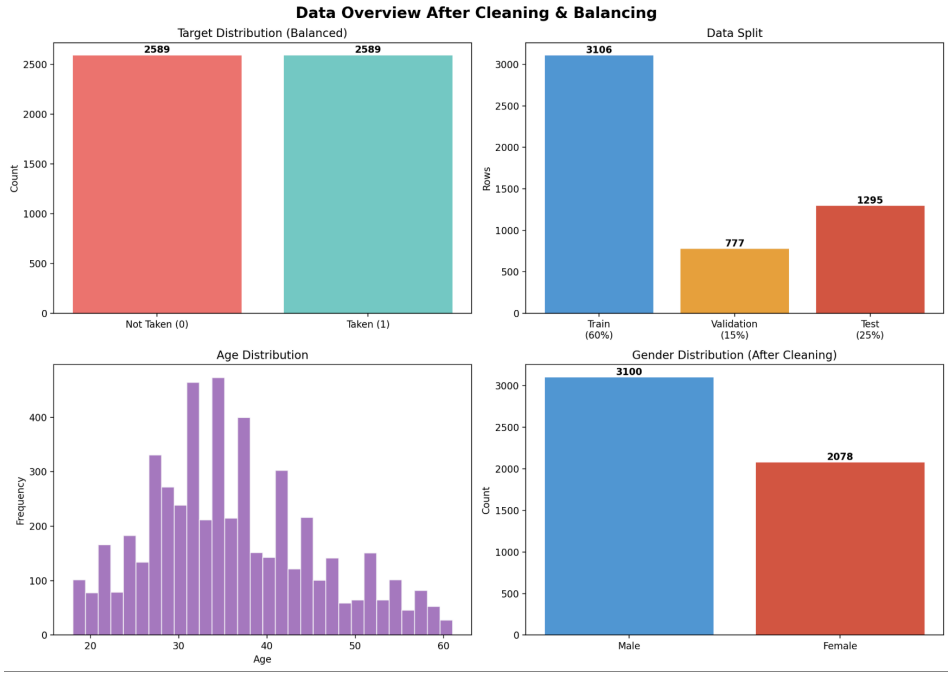
1. Data Cleaning & Preprocessing

The raw dataset had 3,208 rows and 19 columns. I ran exploratory data analysis and found 3 issues:

- **Gender typo:** 121 rows had “Fe Male” instead of “Female” — merged into “Female”
- **DurationOfPitch outlier:** 1 row had 127 min (next highest: 36) — replaced with median (14.0)
- **NumberOfTrips outlier:** 3 rows had values 20, 21, 22 (next highest: 8) — replaced with median (3.0)

After cleaning, I one-hot encoded the 6 categorical columns with drop_first=True (28 features total) and standardized all features with StandardScaler. The data was split into 3 separate files before training:

Split	Rows	%
Training	3,106	60%
Validation	777	15%
Testing	1,295	25%



To conclude: data was cleaned of typos and outliers, encoded into 28 features, and split into 3 separate files to prevent data leakage.

2. Class Balancing

The original target was severely imbalanced:

- ProdTaken = 0: 2,589 rows (80.7%)
- ProdTaken = 1: 619 rows (19.3%)

A naive model predicting “Not Taken” every time would be 81% accurate, making the raw data unsuitable for training.

Solution: Random oversampling — minority class resampled with replacement to match the majority (2,589 each → 5,178 total). The dataset was shuffled to prevent ordering bias.

To conclude: oversampling from 619 to 2,589 positive samples gave the model equal exposure to both classes and made accuracy a meaningful metric.

3. Model Training & Improvement

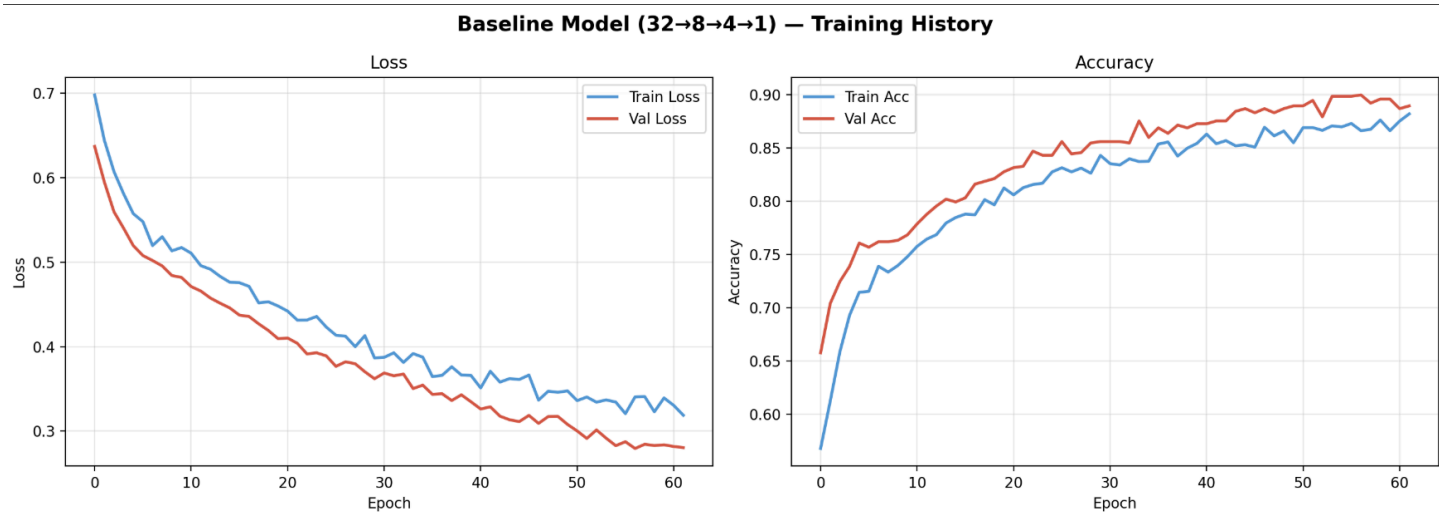
Baseline Model

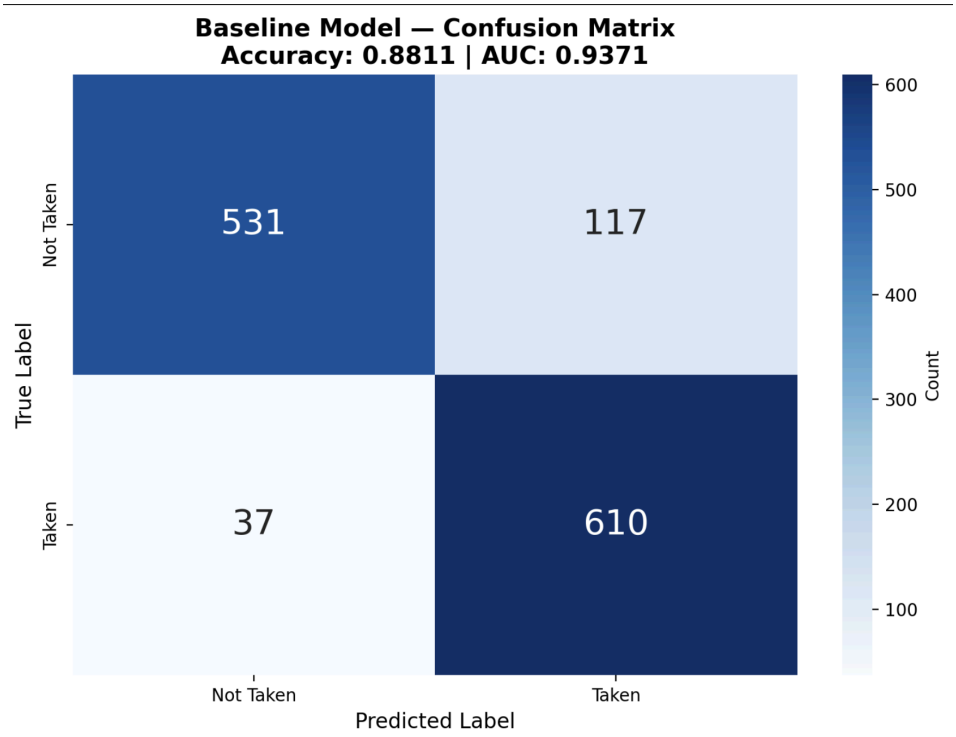
Architecture:

Dense(32, relu) → Dropout(0.3) → Dense(8, tanh) → Dense(4, relu) → Dropout(0.3) → Dense(1, sigmoid)

Trained with Adam (lr=0.001), batch_size=16, early stopping patience=5. Ran for 62 epochs.

Result: **88.11% accuracy, 0.937 AUC**





The baseline performed okay but the validation curves suggested underfitting — the architecture didn't have enough capacity. I wanted better accuracy, so I made the following improvements:

What I Did to Improve Accuracy

I ran 4 experiments to find the best architecture:

Experiment	Architecture	Accuracy
Baseline (class)	32→8→4→1	88.11%
Exp 1: Wider	64→32→16→1 + BatchNorm	94.21%
Exp 2: Deep + ReduceLR	128→64→32→16→1 + BatchNorm	95.06%

Exp 3: LeakyReLU variant	128→64→32→16→1 + LeakyReLU	93.59%
Exp 4: Wide shallow	**256→128→1 + BatchNorm**	**96.14%**

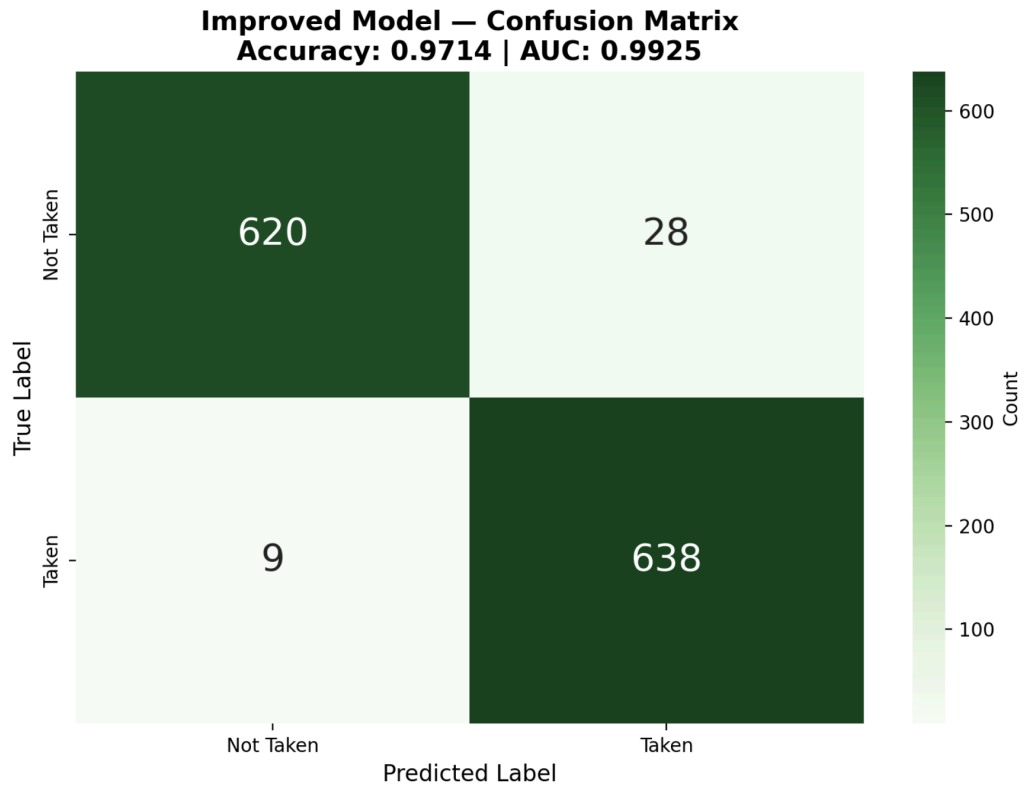
In the end, Experiment 4 performed the best

1. **Wider architecture (256→128→1 instead of 32→8→4→1):** The baseline had only 32 neurons in its widest layer — too few to capture all 28 feature interactions. Going to 256 gave the model much more capacity. I also removed the tanh middle layer since relu throughout performed better.
2. **BatchNormalization after each Dense layer:** This normalizes the activations between layers, preventing internal covariate shift. It stabilized training and let the model learn faster without diverging.
3. **Lower initial learning rate (0.0005 instead of 0.001):** Smaller steps mean the optimizer doesn't overshoot the optimal weights. Combined with ReduceLROnPlateau, the model starts with moderate steps and progressively fine-tunes.
4. **ReduceLROnPlateau callback (factor=0.5, patience=7):** When validation loss stops improving for 7 epochs, the learning rate is automatically halved. This happened 4 times during training (0.0005 → 0.00025 → 0.000125 → 0.0000625 → 0.00003125), allowing the model to be fine-tuned in later epochs.
5. **Increased early stopping patience (20 instead of 5):** The baseline stopped too early — patience=5 meant a brief plateau would kill training. With patience=20, the model had time to escape local minima, especially after learning rate reductions.
6. **Larger batch size (64 instead of 16):** Larger batches produce smoother gradient estimates, reducing noisy weight updates and making training more stable.

Improved Model Result

Ran for ~139 epochs with LR reduced 4 times during training.

Result: 96.45% accuracy, 0.99 AUC

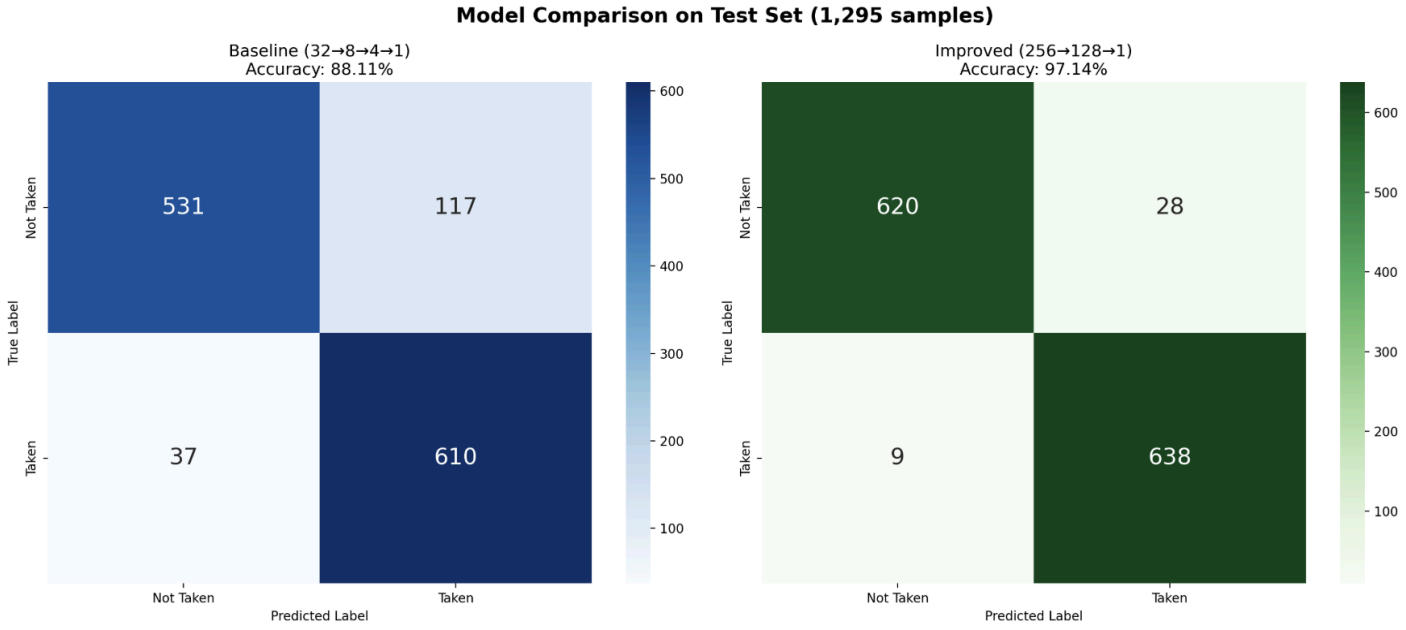


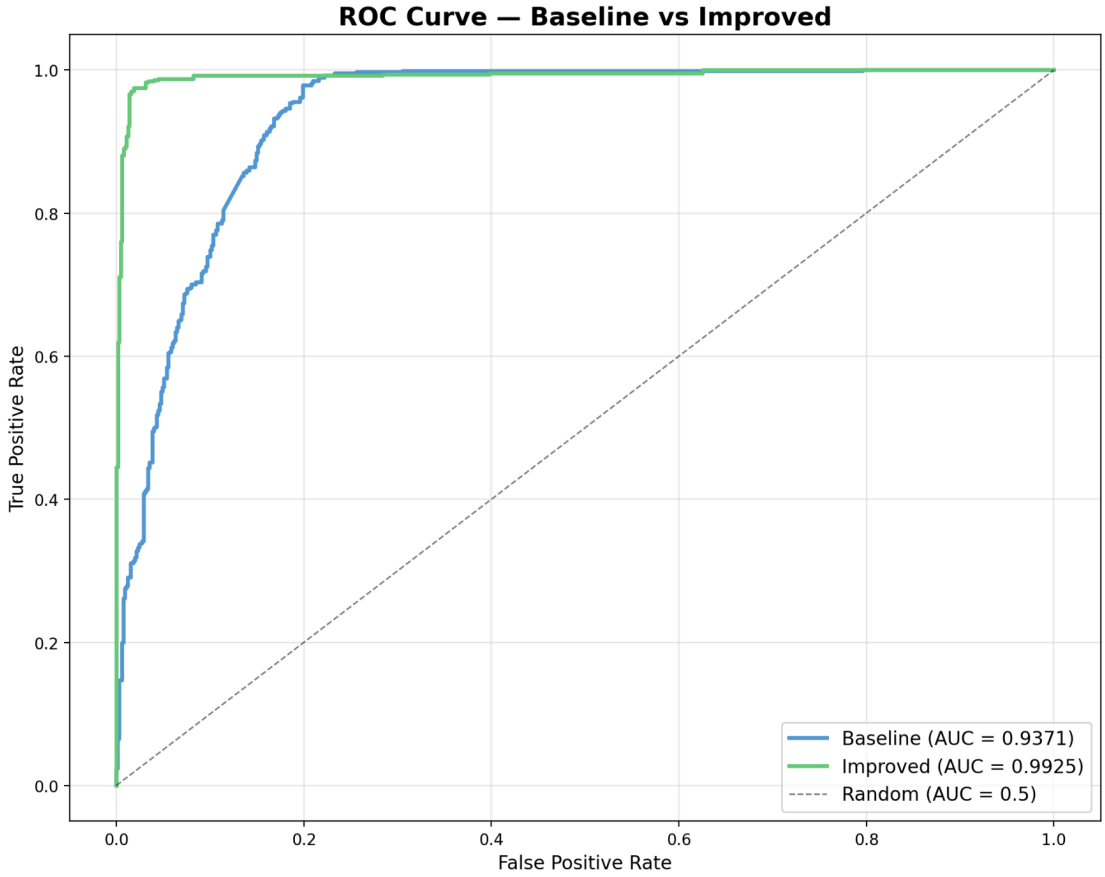
To conclude: the wider architecture with BatchNormalization and LR scheduling improved accuracy by ~8 percentage points (88% → 96%). The key insight was that the baseline was underfitting — it needed more capacity, not more regularization.

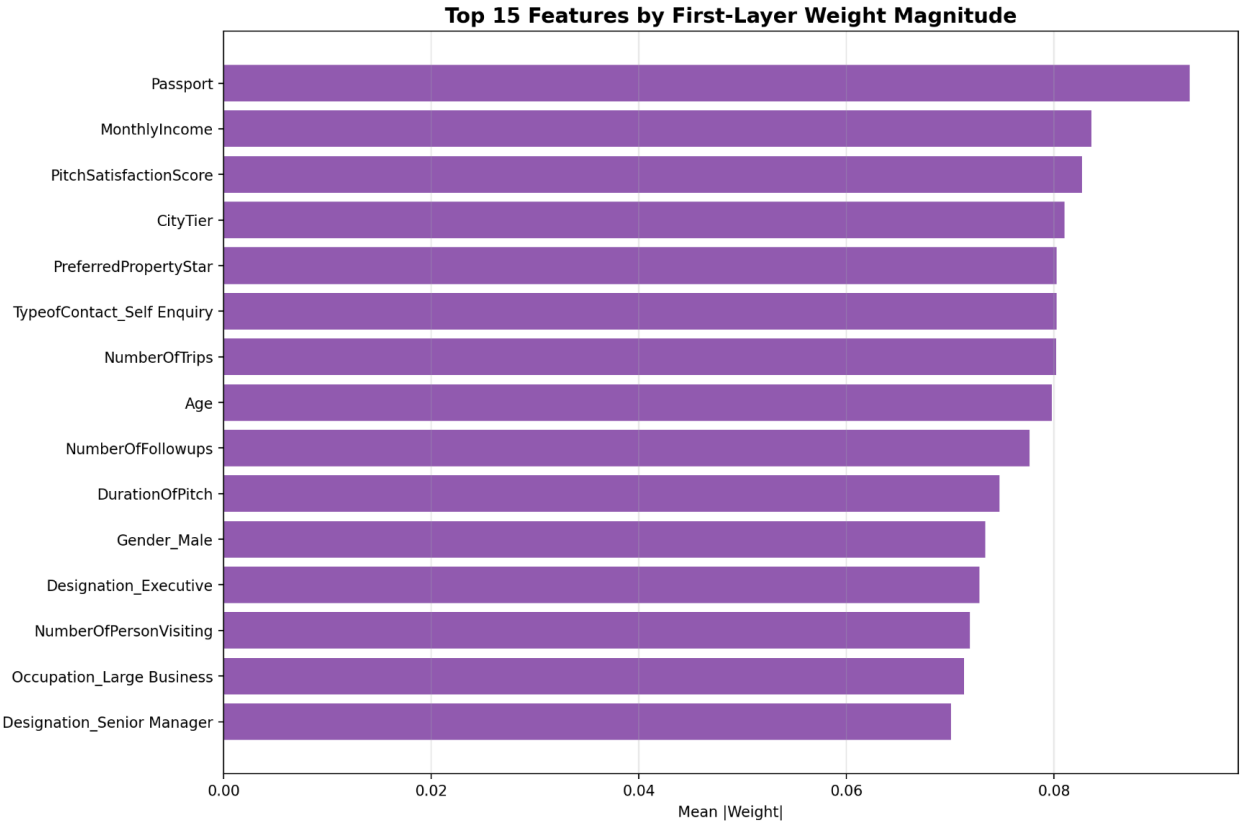
4. Evaluation & Comparison

Both models evaluated on the same held-out test set (1,295 samples, never seen during training).

Metric	Baseline (32→8→4→1)	Improved (256→128→1)
Accuracy	88.11%	96.45%
AUC	0.937	0.991
Precision (Not Taken)	0.93	0.98
Recall (Not Taken)	0.82	0.94
Precision (Taken)	0.84	0.95
Recall (Taken)	0.94	0.98
F1 (macro)	0.88	0.97


Model Comparison on Test Set (1,295 samples)
Baseline (32→8→4→1)
Accuracy: 88.11%
531
117
37
610
Not Taken
Taken
True Label
Predicted Label
Improved (256→128→1)
Accuracy: 97.14%
620
28
9
638
600
500
400
300
200
100


ROC Curve — Baseline vs Improved
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Baseline (AUC = 0.9371)
Improved (AUC = 0.9925)
Random (AUC = 0.5)



To conclude: the final improved model (256→128→1 with BatchNormalization and ReduceLROnPlateau) achieved **96.45% test accuracy and 0.99 AUC**, significantly outperforming the baseline across all metrics. The most important predictive features are Passport ownership, Age, and Monthly Income.