Machine Learning Assignment

Camilo Mercado
Pablo Alfaro
Victor Ramirez

Meet The Team

Pablo
Alfaro

Data Scientist

Camilo
Mercado

Data Scientist

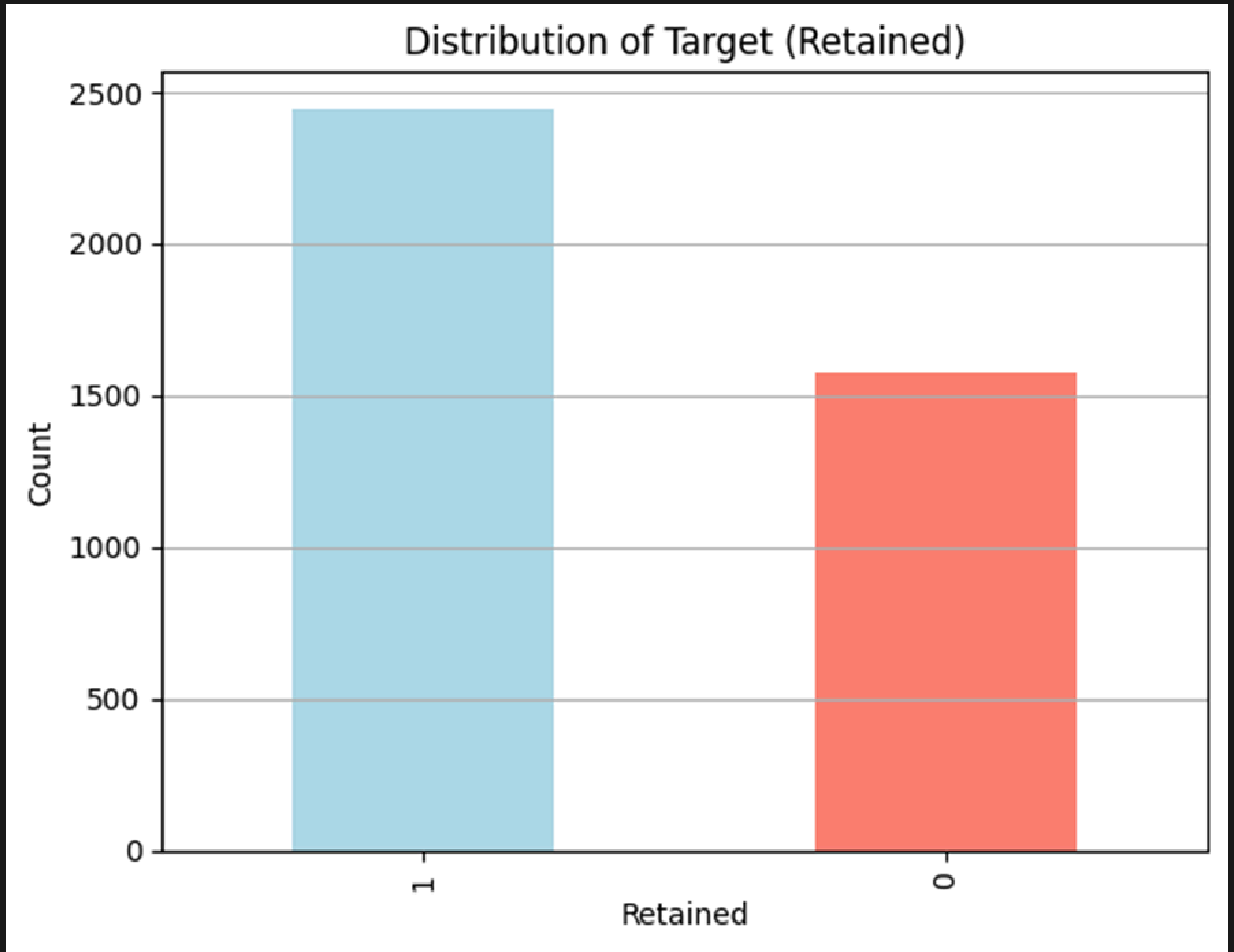
Victor
Ramirez

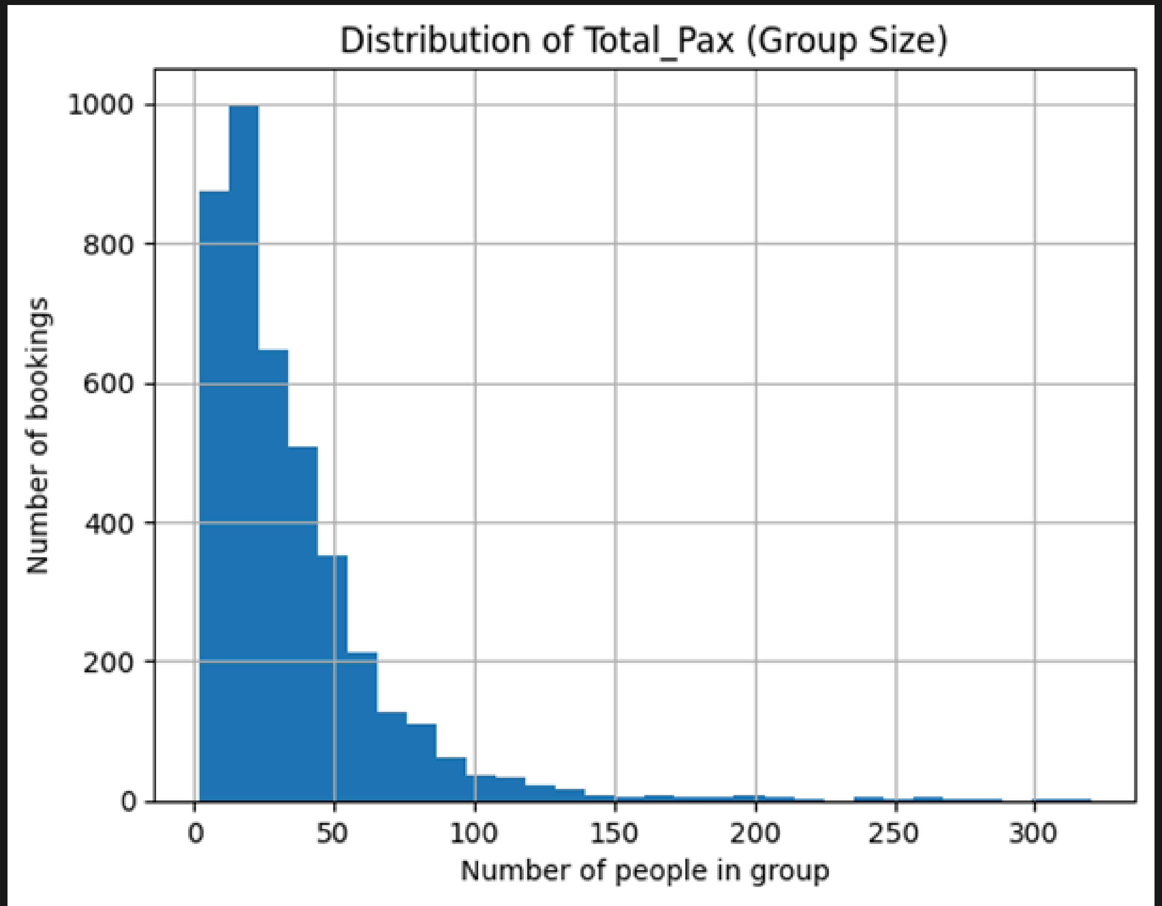
Data Scientist

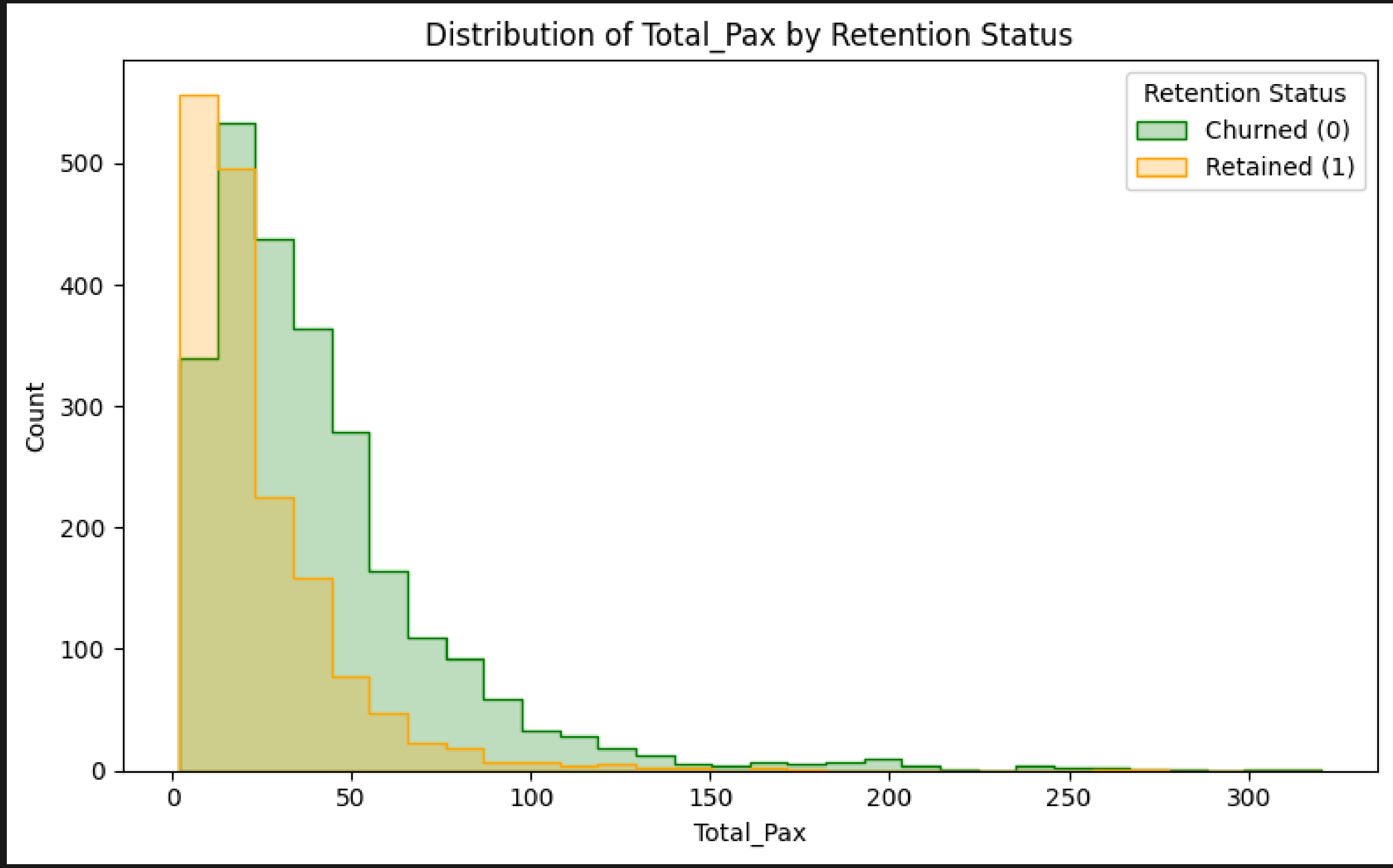
Agenda

EDA

Exploratory Data Analysis about the features and target



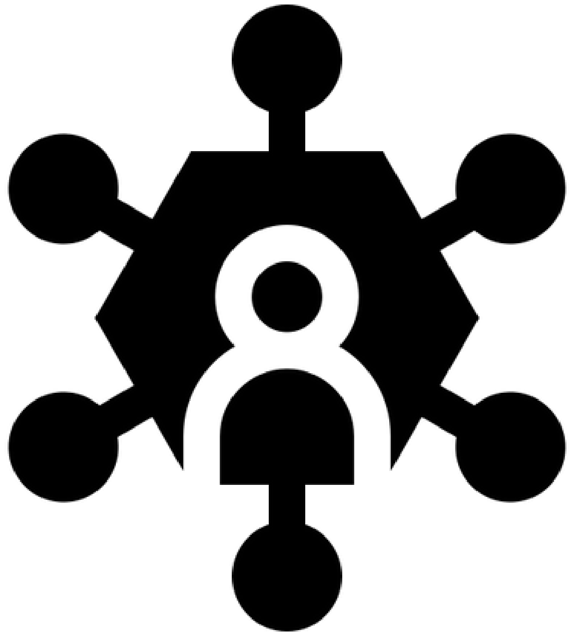




Data Preparation

Sales

Provides details about each trip, including location, dates, participants, and program type — helping us understand how each experience was planned.



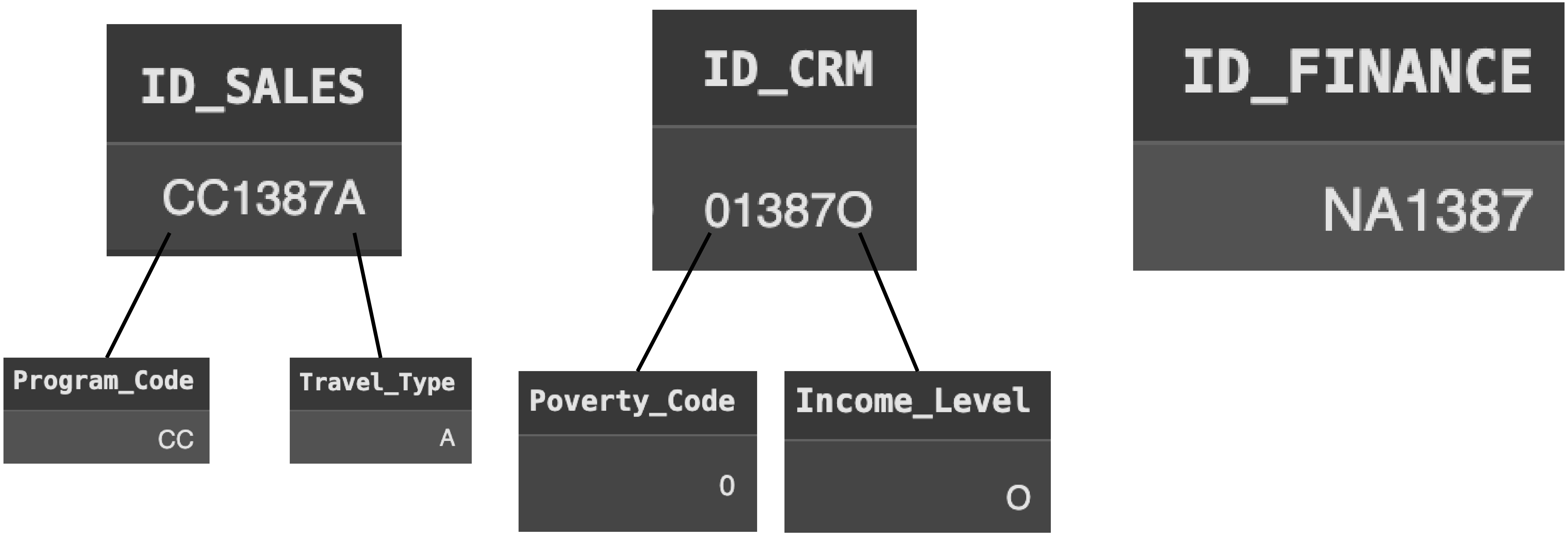
CRM

Covers school and family engagement, income levels, and parent meeting activity — giving insight into each school's background and involvement.

Finance

Includes pricing, payment behavior, sponsorship, and insurance use — showing how families pay and whether they commit to the trip.

Data Preparation



Back to Agenda Page

Data Preparation

```
df['Departure_Date'] = pd.to_datetime(df['Departure_Date'])
df['Early_RPL'] = pd.to_datetime(df['Early_RPL'])
df['Latest_RPL'] = pd.to_datetime(df['Latest_RPL'])
df['Deposit_Date'] = pd.to_datetime(df['Deposit_Date'])

df['Departure_Week_Number'] = df['Departure_Date'].dt.isocalendar().week
df['Early_RPL_Departure_Difference'] = (df['Departure_Date'] - df['Early_RPL']).dt.days
df['Late_RPL_Departure_Difference'] = (df['Departure_Date'] - df['Latest_RPL']).dt.days
df['Deposit_Departure_Difference'] = (df['Departure_Date'] - df['Deposit_Date']).dt.days
```

```
df['FPP_to_School_enrollment'] = df['FPP_to_School_enrollment'].apply(lambda x: x.replace(',', '.'))
df['FPP_to_School_enrollment'] = pd.to_numeric(df['FPP_to_School_enrollment'])
```

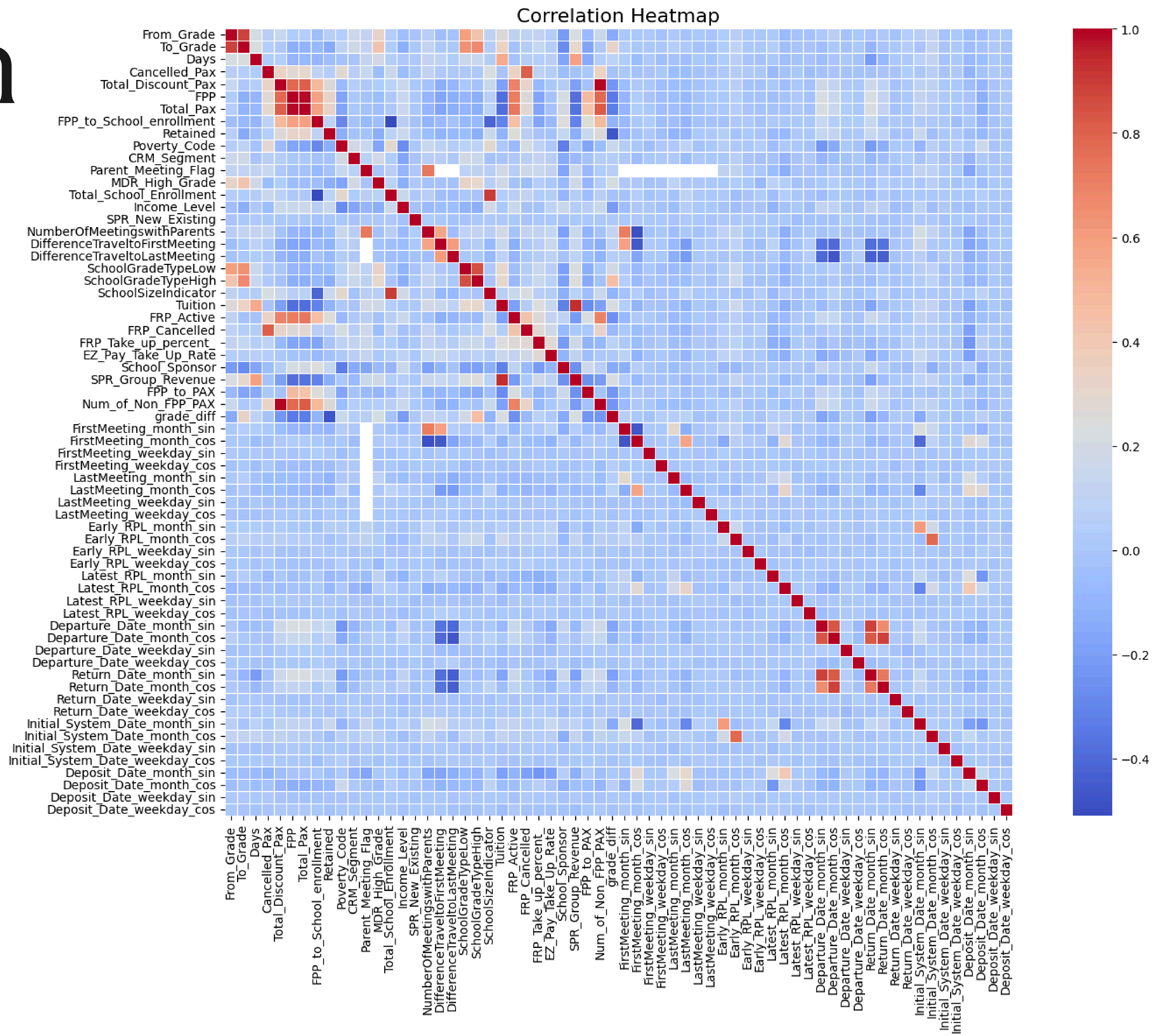
🧹 **Date Cleanup & Feature Creation**

- Converted date columns to proper datetime format
- Created new time-based features (e.g., gaps between key dates)

Data Preparation

Elimination of variables that are too correlated

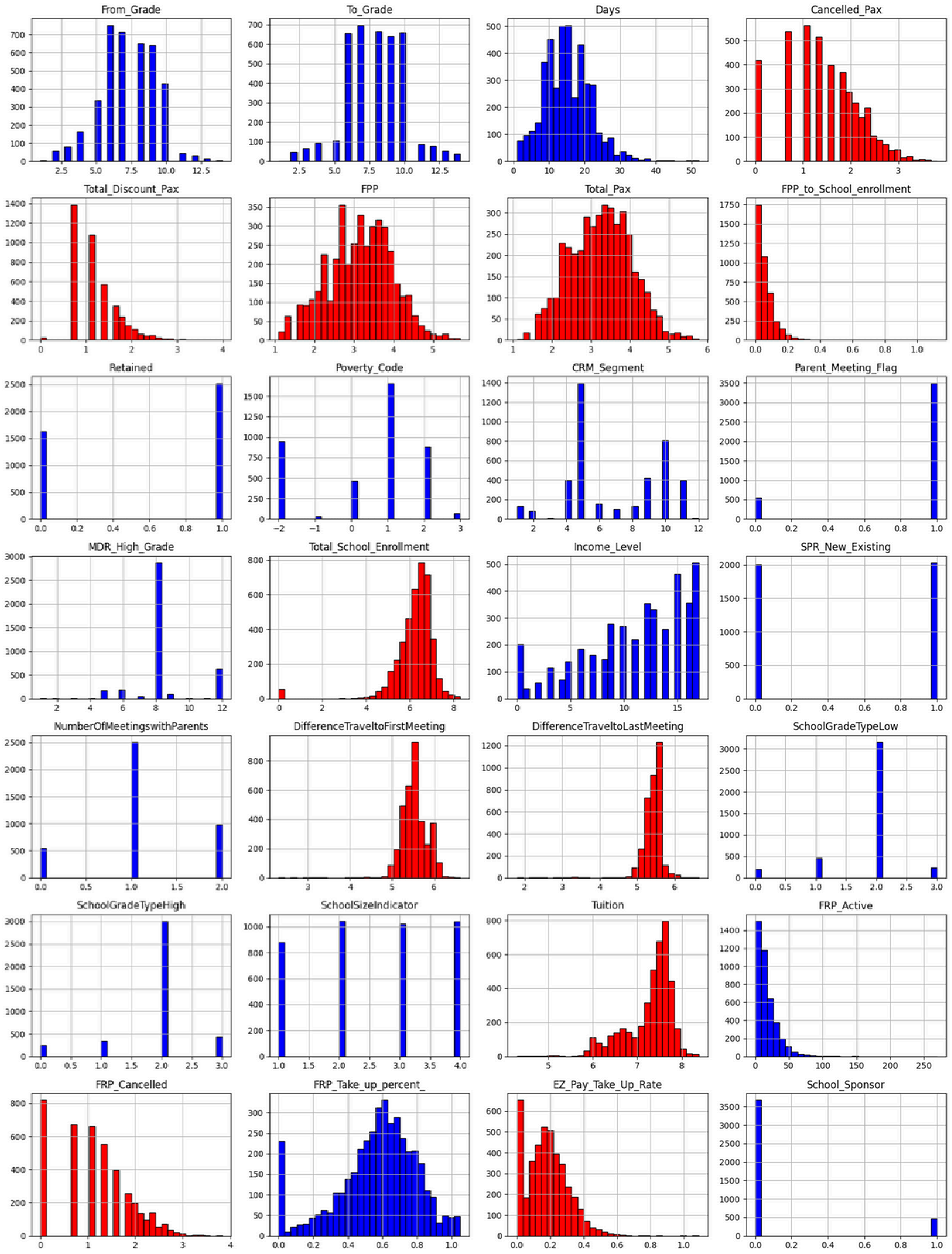
When finding a pair of variables that are correlated above 0.8
we take the one with the highest bivariate gini with the target
and discard the other one



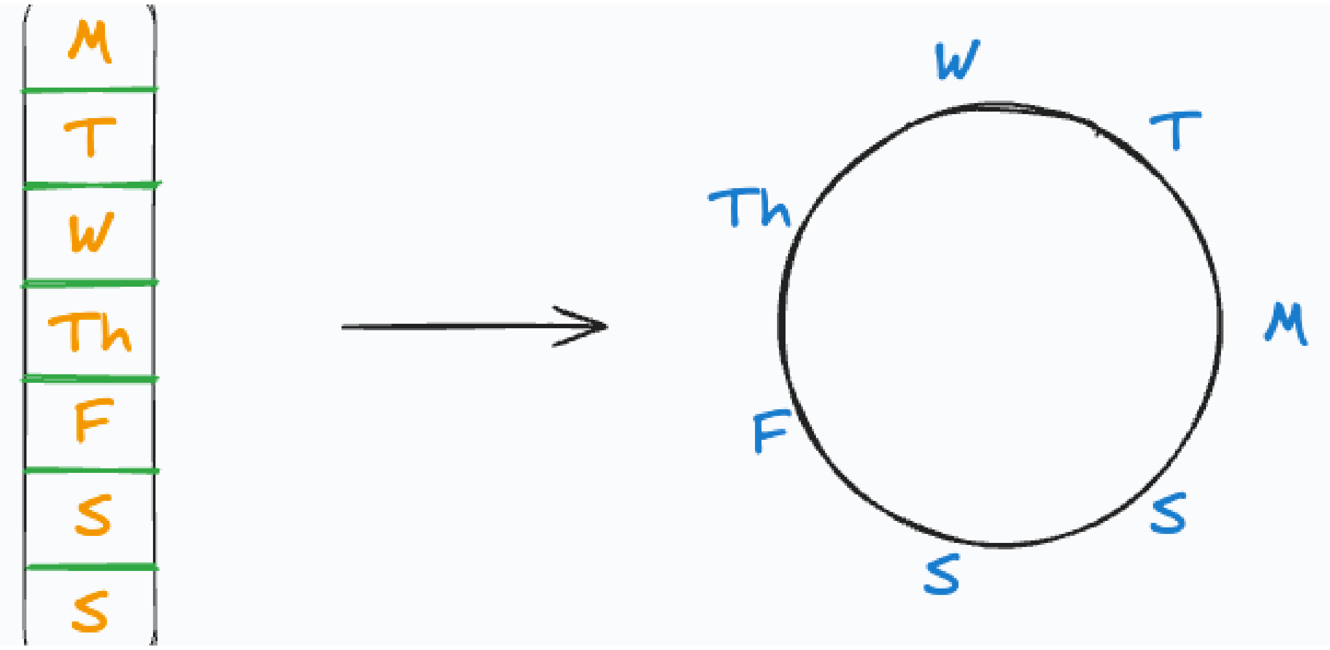
Data Preparation

Log transformation of skewed variables

We performed a log transformation to variables that presented high right skewness.



Feature encoding



Cyclical Feature Engineering for date columns

```python
# Ordinal mappings
grade_mapping = {'Undefined': 0, 'Elementary': 1, 'Middle': 2, 'High': 3}
size_mapping = {'nan': 0, 'S': 1, 'S-M': 2, 'M-L': 3, 'L': 4}
poverty_map = {'A': 0,'B': 1,'C': 2,'D': 3,'E': -1, '0': -2}

# Ordinal encoding for CRM
df_crm['Poverty_Code'] = df_crm['Poverty_Code'].map(poverty_map).fillna(-3)
df_crm['SchoolGradeTypeLow'] = df_crm['SchoolGradeTypeLow'].map(grade_mapping)
df_crm['SchoolGradeTypeHigh'] = df_crm['SchoolGradeTypeHigh'].map(grade_mapping)
df_crm['SchoolSizeIndicator'] = df_crm['SchoolSizeIndicator'].map(size_mapping)
```

Ordinal feature encoding

```python
# Threshold: top 15 states kept, others grouped
top_states = merged_df['Group_State'].value_counts().nlargest(15).index
merged_df['Group_State'] = merged_df['Group_State'].apply(lambda x: x if x in top_states else 'Other')
merged_df['Group_State'] = merged_df['Group_State'].astype('category')
```

Reduction of excessive number of classes

Experimenting with Random Forest and XGBoost — With and Without SMOTE

Retained Variable Balance

Retained: **2522** **Non-Retained:** **1631**

Data Split

80/20 train-test split

SMOTE Application

SMOTE applied only to the training set

Model Candidates

RandomForestClassifier

XGBoost

Hyperparameter Tuning

RF Parameters: n_estimators, max_depth, min_samples_split, min_samples_leaf, etc.

XGB Parameters: max_depth, learning_rate, n_estimators, subsample, colsample_bytree, gamma, etc.

Evaluation Metric: Focused on F1 score (via scoring='f1')

Model Evaluation

Checked **Train Accuracy**, **Test Accuracy**, **Confusion Matrix**, and **Classification Report**

Compared **SMOTE** vs. **non-SMOTE** performance for each model

Best Result

$$\begin{pmatrix} 197 & 76 \\ 59 & 354 \end{pmatrix}$$

XGBoost without SMOTE

Test AUC: 0.874 *Train AUC: 0.989*

	precision	recall	f1-score	support
0	0.77	0.72	0.74	273
1	0.82	0.86	0.84	413
accuracy			0.80	686
macro avg	0.80	0.79	0.79	686
weighted avg	0.80	0.80	0.80	686

Experimenting with LightGBM and Catboost

Retained Variable Balance

Retained: 2522 **Non-Retained: 1631**

Data Split

75/25 train-test split

Balancing

Didn't apply any balancing technique

Model Candidates

LigthGBM

Catboost

Hyperparameter Tuning

LightGBM Parameters: num_leaves, max_depth, learning_rate, n_estimators, subsample, colsample_bytree, reg_alpha, reg_lambda, min_child_samples, boosting_type, etc.

CatBoost Parameters: iterations, depth, learning_rate, l2_leaf_reg, bootstrap_type, bagging_temperature, border_count, rsm, etc.

Evaluation Metric: Focused on ROC-AUC

Model Evaluation

Checked **Train AUC**, **Test AUC**, **Confusion Matrix**, and **Classification Report**

Compared model AUC on both **train and test**

Best Result

Catboost

	precision	recall	f1-score	support
0	0.74	0.73	0.74	376
1	0.85	0.86	0.85	663
accuracy			0.81	1039
macro avg	0.80	0.79	0.79	1039
weighted avg	0.81	0.81	0.81	1039

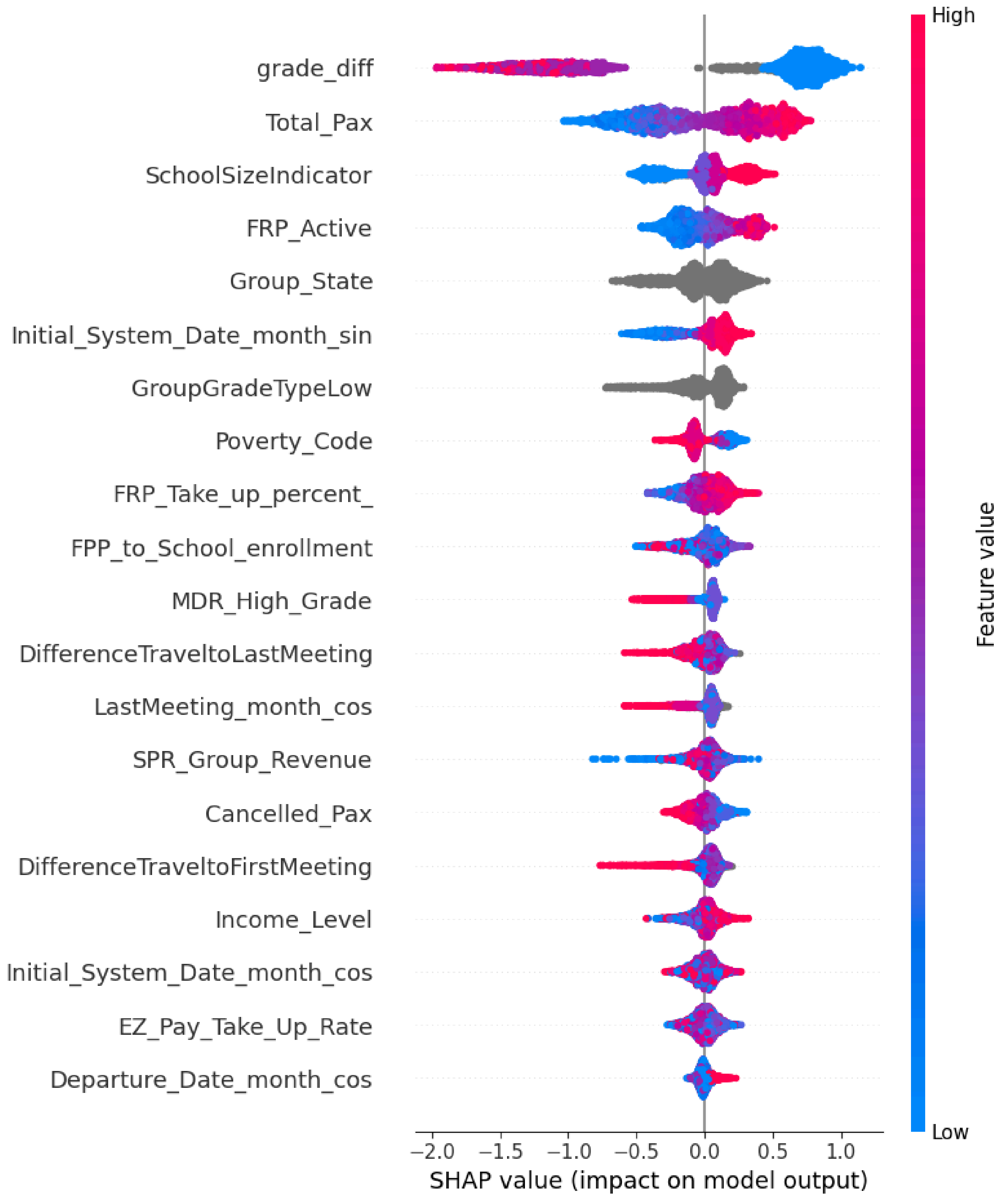
A note on overfitting

Different models gave very good metrics on the test set. Nonetheless, they had perfect metrics on the Train set

```
Fitting 10 folds for each of 20 candidates, totalling 200 fits
Best Parameters: {'learning_rate': 0.08, 'l2_leaf_reg': 2, 'iterations': 100, 'depth': 9, 'border_count': 32, 'bagging_temperature': 0.8}
Train AUC: 0.9982
Test AUC: 0.915
Test LogLoss: 0.3654
CPU times: user 2.92 s, sys: 572 ms, total: 3.49 s
Wall time: 2min 37s
```

```
Fitting 10 folds for each of 20 candidates, totalling 200 fits
Best Parameters: {'learning_rate': 0.1, 'l2_leaf_reg': 8, 'iterations': 100, 'depth': 8, 'border_count': 64, 'bagging_temperature': 0.2}
Train AUC: 0.9873
Test AUC: 0.891
Test LogLoss: 0.404
CPU times: user 2.62 s, sys: 337 ms, total: 2.96 s
Wall time: 2min 21s
```

```
Best Parameters: {'rsm': 0.8, 'learning_rate': 0.05, 'l2_leaf_reg': 10, 'iterations': 300, 'grow_policy': 'SymmetricTre
Train AUC: 0.9644
Test AUC: 0.8689
Test LogLoss: 0.4365
CPU times: user 5.33 s, sys: 759 ms, total: 6.09 s
Wall time: 4min 29s
```



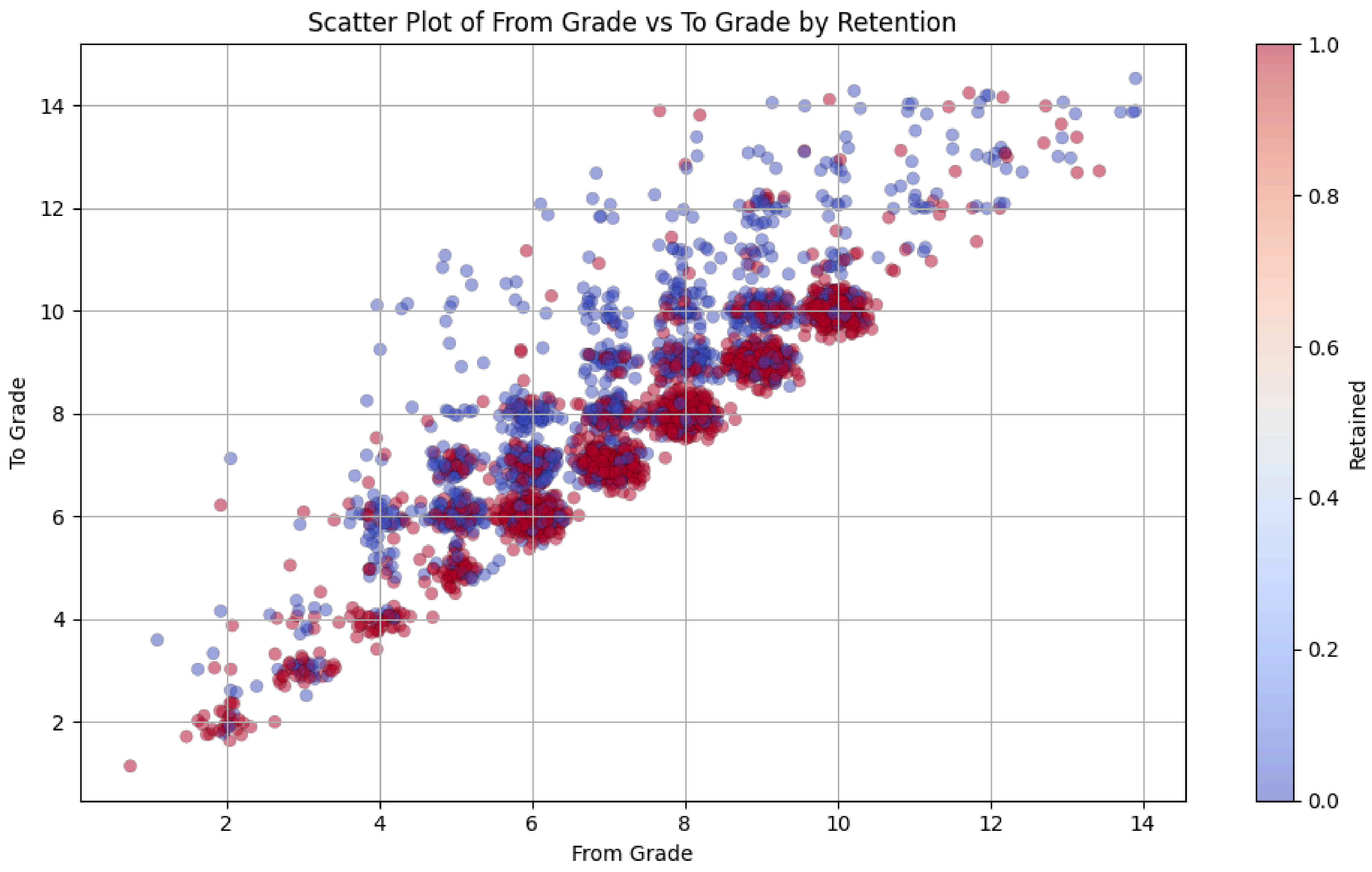
Feature Interpretation

- We can see that the most important variables are, in that order, the grade_diff (To_Grade-From_Grade) Total_Pax and SchoolSizeIndicator.

- In grade_diff, for example, the blue color in the right hand side of the 0.0 indicates that low values of the feature have a positive impact on the probability of a sale of being retained for the next year, while high values tend to have a negative impact.

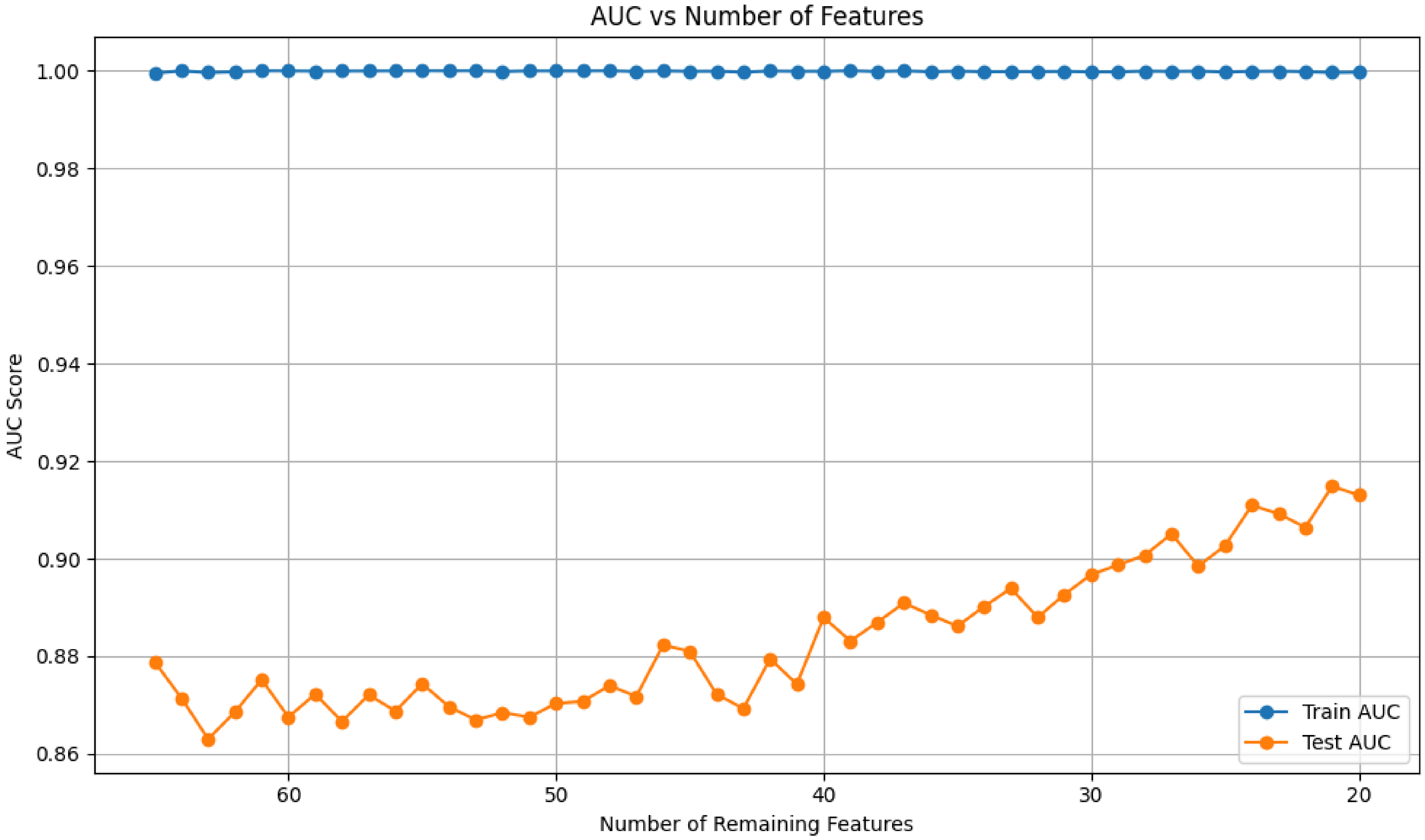
- On the other hand, in Total_Pax, the red color in the right hand side of the 0.0 indicates that high values of the variable have a positive impact on the probability of a sale of being retained for the next year, while low values have a negative impact.

Why is Grade_diff so important?



- On this scatter plot some jitter was applied to see the points that are overlapped
- We can see that most of the retained customers had trips that had the From_Grade and To_Grade as the same grade
- Thats why creating a variable called grade_diff was so good at summarizing that information

Dimensionality reduction



We used the feature importance according to SHAP to iteratively delete the least important variable on each iteration

Hyperparameter tunning

```
# Parameter grid
param_dist = {
    'iterations': [200, 300],                          # Slightly fewer iterations to reduce complexity
    'learning_rate': [0.01, 0.03, 0.05],                # Lower LR = smoother learning
    'depth': [4, 5, 6],                                 # Shallow trees generalize better
    'l2_leaf_reg': [10, 20, 30],                        # Stronger L2 regularization
    'bagging_temperature': [1.0, 1.5, 2.0],             # Increase randomness for generalization
    'rsm': [0.6, 0.8],                                  # Feature subsampling (randomness + regularization)
    'border_count': [32, 64],                           # Less granularity in splits = less overfitting
    'bootstrap_type': ['Bayesian'],                     # Keep Bayesian (good uncertainty handling)
    'grow_policy': ['SymmetricTree']                    # Avoid overly deep trees with Oblivious splits
}
# Cross-validation setup
cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Random search with early stopping passed in fit_params
random_search = RandomizedSearchCV(
    estimator=catboost_model,
    param_distributions=param_dist,
    n_iter=20,
    scoring='roc_auc',
    cv=cv,
    verbose=1,
    n_jobs=-1,
    random_state=42
)
```

We used Randomized Search CV to find the best hyper parameters that optimize AUC in the training dataset

Confusion Matrix - Retained Class (1)

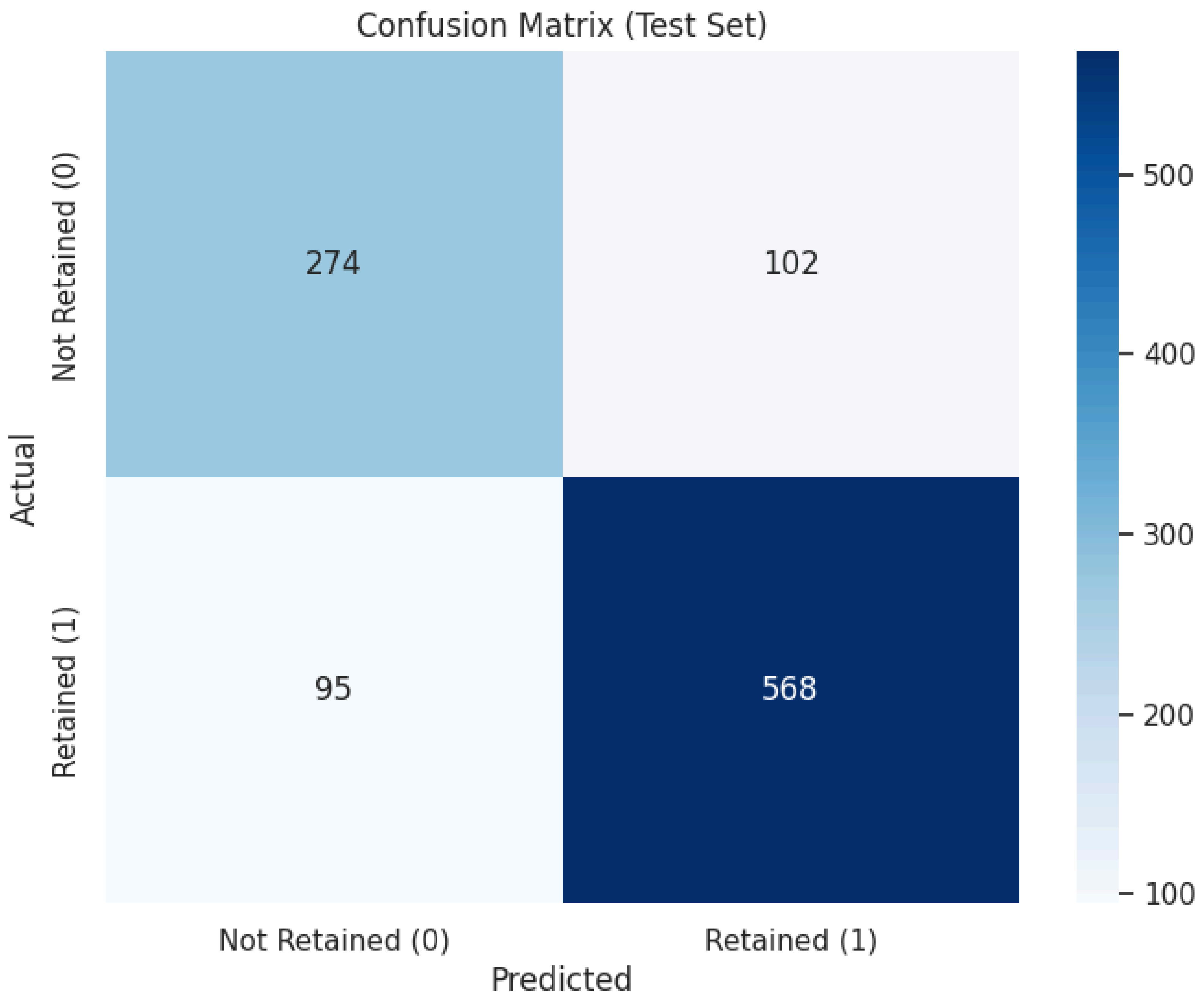
Precision: 84.8%

Of all customers predicted as retained, 84.8% were actually retained.

Recall (Sensitivity): 85.7%

Of all customers who actually were retained, the model correctly identified 85.7%.

- High Precision → The model is very accurate when it says a customer will be retained.
- High Recall → The model captures nearly all retained customers.



F1 score

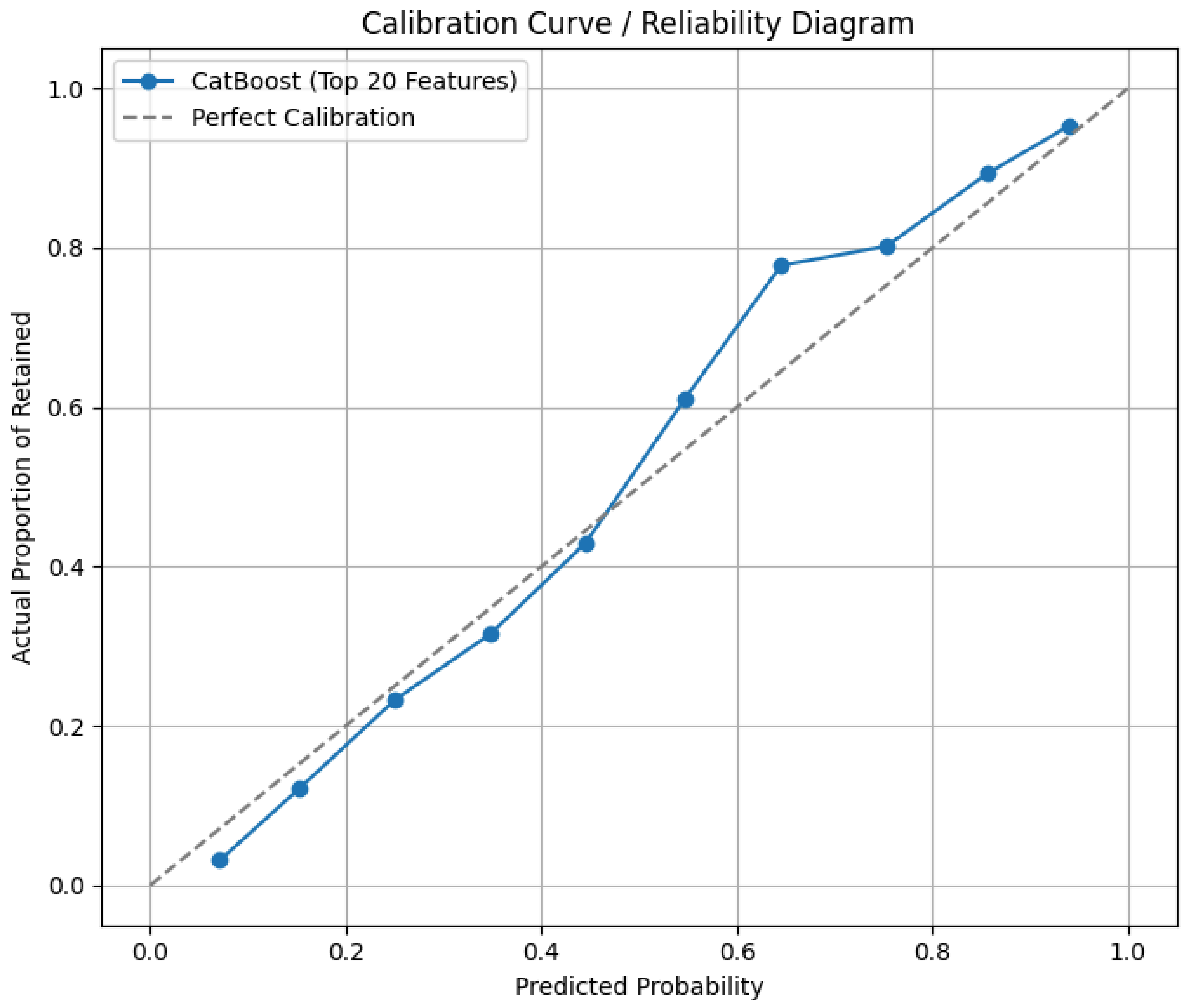
Since we don't have reliable estimates for the cost of losing a customer (false negative) or the cost of targeting a non-retained one(false positive), **we assume they are equally important**.

Using **F1-score (β = 1)** gives a balanced evaluation of the model, weighing both precision and recall equally. It helps ensure we're neither too aggressive nor too conservative in predicting customer retention.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = 2 \cdot \frac{0.848 \cdot 0.857}{0.848 + 0.857} = \mathbf{0.852}$$

Calibration curve check

We want to make sure the model gives positive predictions at the same rate they're observed in the actual data



How can we use the model?

To lower the number of clients that churn we will **focus our marketing efforts** to those customers that have the lowest probability of being retained according to the model

For example we could implement discounts or have our customer support specialists get in contact with the customers who got the lowest score in the model (low probability of being retained)

Timeline

EDA

Getting a feel of the data and the business objective

Model development

We try different models to try to get the best fitting model

A/B testing

We run an A/B Test to see if the model recommendations lower churn

Production deployment

Given a positive uplift in the A/B Test we can roll out the model to production

Thank you

Team:

Camilo Mercado

Pablo Alfaro

Victor Ramirez