Fleet Classification Model: Technical Report & Methodology

Date: September 25, 2025

Author(s): Olivia Chen

Executive Summary

This report details the methodology, development, and performance of a machine learning model designed to classify companies by estimated fleet size. The primary business objective is to enhance lead generation by identifying high-potential prospects for our fleet services. A core challenge in this endeavor is the frequent absence of reliable `fleet_size` data for non-customers.

To overcome this, we have constructed a multi-class classification model that serves as a foundational proxy. By training this model on our rich internal customer data—where fleet size is known or can be reliably inferred—we can predict a probable `fleet_size` category for any given company, including new prospects. This predicted fleet size is a critical engineered feature that will subsequently be used in downstream models to predict key business metrics such as `outstanding_card_count`, thereby providing a robust, data-driven system for prioritizing sales and marketing efforts.

This document covers the project's strategic goals, the complex data integration process, the challenges encountered during modeling (such as severe class imbalance), and the techniques employed to build a high-performing and interpretable model.

1. Project Goal and Strategy

1.1. The Business Objective: Finding Prospective Leads

The commercial fleet market is vast and competitive. To maximize the efficiency and effectiveness of our sales organization, a strategic, data-informed approach to lead generation is required. The primary goal of this project is to move beyond broad-based prospecting and develop a system that can accurately identify and prioritize companies most likely to become valuable, high-volume customers.

1.2. The Core Problem: Missing `fleet_size` Data

A direct indicator of a company's potential is its fleet size. However, this crucial piece of information is an internal metric that is rarely available for prospective clients. Relying solely on data where `fleet_size` is present would severely limit our ability to score and rank the broad universe of potential leads.

1.3. Our Solution: A Proxy Model for Fleet Size Classification

To solve the missing data problem, we adopted a two-stage strategy with this classification model as the cornerstone of the first stage.

1. **Stage 1: Predict Fleet Size (Classification Model).** We built the model detailed in this report to predict a `fleet_size` category (`<= 10`, `11-50`, `> 50`) for any company, using a wide array of firmographic and vehicular data. This model is trained on existing customers where we have a reliable ground truth for fleet size.
2. **Stage 2: Predict Business Value (Downstream Model).** The predicted `fleet_size` category becomes a powerful new feature. This feature will be fed into subsequent regression or scoring models to predict key performance indicators like `outstanding_card_count`.

This approach allows us to leverage the full depth of our available data to create a robust proxy for the missing `fleet_size` value, enabling us to score the entire market of potential leads, not just those for whom we have complete data.

2. Data Sourcing, Integration, and Strategy

A robust model is built upon a comprehensive and well-integrated dataset. Our approach involved merging data from three distinct sources to create a 360° view of each business. Furthermore, a key strategic decision was made regarding the scope of our training data to ensure the model could learn the nuanced patterns of smaller, yet highly valuable, customer segments.

2.1. Core Data Sources

Our model's predictive power is derived from the synthesis of three unique datasets:

1. **D&B (Dun & Bradstreet):** This serves as our foundational source for **firmographic data**. It provides essential business characteristics such as company size (employee count), annual sales volume, industry classification (NAICS codes), years in business, and corporate structure.
2. **Polk Data:** This dataset provides critical **automotive and fleet-specific details**. It contains vehicle registration data linked to businesses, offering insights into existing fleet size, vehicle composition (e.g., light-duty trucks vs. sedans), makes, models, and vehicle age.

3. **Havill Data:** This is our **proprietary internal data**, containing historical interactions and customer behavior. It includes information on past service records, contract details, and other engagement metrics that signal a company's operational profile.

2.2. Data Pipeline and Cohort Strategy

The construction of our final training dataset involved a multi-step pipeline that extracted and merged data from the sources above. A critical strategic choice during this process was determining the inclusion criteria for our "significant customers" cohort, which forms the basis of our training data. We analyzed multiple thresholds, primarily based on `outstanding_card_count`, to find the optimal balance between data volume and signal quality.

The pipeline logs for two such scenarios—a threshold of >=10 cards and >=20 cards—are summarized below.

Comparison of Data Pipeline Outputs

Data Pipeline Stage	Threshold: >= 10 Cards	Threshold: >= 20 Cards	Strategic Implication
Initial "Significant Customers"	160,635	78,764	A lower threshold nearly doubles the initial pool of companies.
D&B Records Fetched	89,513	44,226	More companies in the initial pool lead to more potential matches in the D&B data.
Polk Records Fetched	55,508	28,106	Captures a much wider range of fleet information.
Havill Records Fetched	22,547	13,411	Includes more internal customer history.

Final Qualified & Merged Rows	**82,565**	**39,746**	The final dataset for the model is over twice as large with the lower threshold.

2.3. Rationale for Adopting the 10-Card Threshold

After careful analysis, we chose the `Card Count Threshold: 10` strategy for building our final training dataset. While a higher threshold (like 20 cards) selects for more obviously large customers, it introduces significant sample bias and reduces the overall volume of data available for training.

The decision to use a lower threshold was driven by two key objectives:

1. **Increasing Overall Data Volume:** A larger dataset with 82,565 qualified companies provides the model with more examples to learn from, leading to better generalization and robustness.
2. **Improving Small-Class Representation:** The most critical challenge in this project is the natural rarity of very large fleets (the `> 50` class). A high threshold of 20 cards would filter out many companies in the medium (`11-50`) and even some in the large (`> 50`) categories. By setting the bar at 10 cards, we intentionally include a more diverse set of smaller and mid-sized businesses. This provides the model with more examples of companies on the cusp of different fleet sizes, helping it learn the subtle features that distinguish a medium fleet from a large one, thereby directly addressing the class imbalance problem.

3. Modeling Methodology: Improving Accuracy and Tackling Imbalance

The development of a successful classification model required a systematic approach to feature engineering, addressing the severe class imbalance inherent in our data, and optimizing model performance through rigorous tuning. This section details the technical strategies implemented, with direct references to the `fleet_classification.py` pipeline script.

3.1. Defining the Target: The Binning Strategy

The raw `fleet_size` is a continuous numerical value. To frame this as a classification problem, we converted this target variable into discrete, business-relevant categories. Our script supports multiple binning strategies, including logarithmic and quantile-based methods. However, for maximum business utility, we selected an explicit strategy.

- **Implementation:** The `explicit3` binning strategy was chosen, which segments the data into three clear, interpretable bins: `<= 10`, `11-50`, and `> 50`. This is defined in the `binning_strategies` dictionary within the script:

```
None
binning_strategies = {
    # ... other strategies
    "explicit3": {"mode": "explicit", "edges": [-np.inf, 10.0,
50.0, np.inf], "name": "explicit_3_bins"}
}
```

The binning itself is performed using the `pandas.cut` function, which applies these edges to the `y_train` and `y_test` series to create the categorical labels used for training.

3.2. A Multi-Faceted Approach to Class Imbalance

Our dataset is naturally imbalanced, with far fewer companies in the `> 50` category than in the smaller bins. A naive model would achieve high accuracy by simply ignoring this rare class. To combat this, we implemented and tested several advanced techniques.

3.2.1. SMOTE (Synthetic Minority Over-sampling Technique)

One approach we explored was to artificially balance the dataset by generating new data points for the minority classes.

- **Implementation:** Our pipeline includes the capability to use SMOTE via the `imblearn` library. When the `--imbalance_strategy SMOTE` flag is used, SMOTE is added as a step within our machine learning pipeline. This ensures that the over-sampling is applied only to the training data during each cross-validation fold, preventing data leakage.

```
None
if imbalance_strategy == 'SMOTE':
    # ...
    # Add SMOTE as a step *before* the classifier
    steps.append(('oversampler',
SMOTE(random_state=RANDOM_STATE)))
```

3.2.2. Class Weighting (The Chosen Strategy)

After experimentation, the most effective strategy proved to be class weighting. This method does not alter the data but instead modifies the model's learning process by applying a higher penalty for misclassifying examples from the minority classes.

- **Implementation:** We implemented a flexible `calculate_class_weights` function that supports several weighting schemes. The best-performing model utilized the `simple_manual` strategy, which assigns a weight of `1.0` to the small-fleet class, `1.1` to the medium, and a heavy weight of `3.0` to the high-value `> 50` class.

```
None
def calculate_class_weights(y_train, class_weight_strategy):
    # ...
    elif class_weight_strategy == 'simple_manual':
        return {'<= 10': 1.0, '11-50': 1.1, '> 50': 3.0}
```

These weights are passed to the model's `.fit()` method via the `sample_weight` parameter, forcing the algorithm to pay significantly more attention to correctly identifying large-fleet customers.

3.3. Model Tuning and Optimization

To achieve the highest possible accuracy, we used an automated and robust process for selecting the best model and its optimal settings.

- **Hyperparameter Tuning:** We utilized `RandomizedSearchCV` from Scikit-learn to efficiently search through a wide range of hyperparameters for our RandomForest model. This technique randomly samples a fixed number of parameter combinations (`n_iter`) from a defined distribution. The parameters tuned for RandomForest include:
    - `n_estimators`: The number of trees in the forest.
    - `max_depth`: The maximum depth of each tree.
    - `max_features`: The number of features to consider when looking for the best split.

```
None
# Parameter distribution for RandomForest
param_dist = { 'classifier__n_estimators': randint(100, 500),
```

```
                'classifier__max_depth': randint(5, 20),
                'classifier__max_features': ['sqrt', 'log2', 0.5,
0.75, None]
            }

# Randomized Search implementation
random_search = RandomizedSearchCV(pipe,
param_distributions=param_dist, n_iter=n_iter, cv=3, ...)
```

4. Feature Engineering: Creating Predictive Signals

Raw data alone is often insufficient for building a high-performance model. The process of **feature engineering**—creating new, informative variables from the existing data—is critical for uncovering deeper patterns. Our strategy focused on creating unified features, business-centric ratios, and powerful geospatial signals.

4.1. Coalescing and Creating Ratio Features

To create a clean and consistent set of inputs, we first unified columns from different data sources and then engineered new ratios to provide normalized, comparable metrics.

- **Coalescing Data:** Our raw dataset contained similar information from multiple sources, such as `state_code_dnb` and `state_code_polk`. To create a single, reliable feature, we coalesced these columns. The `fleet_classification.py` script prioritizes the D&B data and fills any missing values with data from Polk.

```
None
# From fleet_classification.py
print("Coalescing state_code_dnb and state_code_polk into
state_code...")
df['state_code'] =
df['state_code_dnb'].fillna(df['state_code_polk'])
```

- **Engineering Business Ratios:** Raw numbers like total sales or employee count can be misleading; a company with 1,000 employees and $10M in sales has a very different

profile from one with 100 employees and $10M in sales. We created ratio features to capture these nuances:

    - `company_age`: Calculated from the `year_started` column.
    - `sales_per_employee`: Normalizes sales volume by the total number of employees.
    - `employee_concentration_ratio`: Measures the proportion of employees at a specific location versus the total.

These are created within the `create_features` function in `fleet_classification.py`, providing the model with richer, more contextual information than the raw numbers alone.

4.2. Geospatial Feature Enrichment

A company's location is a powerful predictor of its operational needs. To capture this, we enriched our data with geospatial features derived from public, third-party sources.

- **Data Acquisition:** The `download_data.py` script automates the download of shapefiles from government sources, ensuring we have the necessary geographic data. The key sources include:
    - **U.S. Census Bureau:** Provides shapefiles for all US counties and defines urban vs. rural areas.
    - **Federal Highway Administration:** Provides the National Highway Freight Network, mapping the most critical freight corridors in the country.

4.3. Implementing Geospatial Logic

The `geo_features.py` script contains the functions that process this third-party data and generate new features for each company based on its latitude and longitude.

- **Urban/Rural Classification:** The `add_urban_rural_tag` function performs a spatial join between our company locations and the Census Bureau's urban area polygons. Each company is tagged as being in an 'urban', 'cluster' (smaller urban area), or 'rural' location. This helps the model learn patterns related to population density and infrastructure.
- **Proximity to Logistics Hubs:** The `add_distance_to_freight_corridor` function calculates the straight-line distance (in kilometers) from a company's location to the nearest segment of the National Highway Freight Network.

Why Geospatial Features Improve Predictions: These features provide crucial context that firmographic data alone cannot.

- A business in a **rural** area, far from highways, likely has different logistical needs than an urban one.

- Proximity to a **freight corridor** is a strong signal that a business is involved in or reliant upon logistics and transportation. A company located just 2km from a major freight highway is far more likely to operate a large fleet than a company 50km away, even if their employee counts and sales figures are identical. By quantifying these geographic realities, we provide the model with powerful predictors of fleet size.

5. Summary and Final Model Performance

After extensive experimentation with data sourcing strategies, class imbalance techniques, and hyperparameter tuning, the optimal results were achieved using a **RandomForest Classifier**. The model was trained on the expanded dataset (10-card threshold) and employed a `simple_manual` class weighting strategy to focus learning on high-value fleet segments.

5.1. Final Performance Metrics

The final model delivers a strong balance of overall accuracy and targeted precision, making it a valuable tool for lead qualification.

Metric	Value	Description
Overall Accuracy	62%	The model correctly predicts the fleet size for 62% of all businesses.
Weighted Avg F1-Score	61%	A balanced measure of precision and recall across all classes.
Large Fleet (>50) Precision	45%	When the model predicts a large fleet, it is correct 45% of the time.
Large Fleet (>50) Recall	39%	The model successfully finds 39% of all actual large fleets in the data.
Mid-Sized (11-50) Precision	66%	This is the model's most precise category, offering a highly reliable list.

The **45% precision** for the `> 50` fleet segment represents a significant achievement. It provides the sales team with a high-quality, focused list where nearly half the prospects are confirmed to be in our most valuable target segment, drastically improving efficiency over untargeted outreach.

5.2. Model Interpretation and Insights

Understanding *why* the model makes its decisions is as important as the predictions themselves. The following visualizations provide insight into the model's behavior.

Overall Feature Importance The model identified industry codes (NAICS) and company scale (employee count) as the most influential predictors across all classes. This confirms that a company's industry and size are fundamental indicators of its likely fleet needs.

Drivers for the Large Fleet (`> 50`) Class The SHAP plot reveals what specifically drives a prediction for our highest-value segment. High employee counts and sales volume are the most powerful positive indicators. This confirms our business intuition: larger companies with more resources are the most likely to operate large fleets.

Model Discriminative Power (ROC Curve) The ROC curve demonstrates the model's ability to distinguish between classes. The curve for the `> 50` class (green line) has an Area Under Curve (AUC) of **0.82**. This is a strong score, indicating that the model is very effective at separating large fleets from all other businesses, even if the final precision is impacted by the class's rarity.

5.3. Further Experimentation: Ensemble Methods

Subsequent to achieving these results with the RandomForest model, we explored more complex ensemble techniques, including **Voting** and **Stacking classifiers**. These methods combine the predictions of multiple different models in an attempt to produce a more robust and accurate final prediction. However, in our tests, these advanced ensemble methods did not yield any significant improvement over the performance of the optimized RandomForest model. This solidifies our current model as the most effective and efficient solution for this business problem.

6. Conclusion and Next Steps

The development of the fleet classification model has met the project's primary objective: to create a reliable, data-driven proxy for fleet size that can be used to qualify and prioritize prospective leads. The final RandomForest model is not only statistically robust but also transparent and aligned with business intuition. The true value of this model is realized when it is applied to score unlabeled data, directly fueling the sales and marketing pipeline.

6.1. Generating Actionable Lead Data

The final step in the `fleet_classification.py` pipeline is to apply the trained model to the `unlabeled` dataset—that is, the universe of prospective companies for which we have no internal fleet size data. The output is a CSV file containing the original company data enriched with three key fields:

1. **`pred_fleet_bucket`**: The predicted fleet size category (`<= 10`, `11-50`, or `> 50`).
2. **`pred_fleet_size_median`**: A concrete numerical estimate of fleet size, derived from the median value of the training data within the predicted bucket. This serves as a critical input for downstream models.
3. **`pred_confidence`**: The model's confidence (probability) in its prediction, allowing for further prioritization of high-certainty leads.

This enriched dataset is the primary deliverable of this project, providing an actionable list for strategic outreach.

6.2. Recommendations and Future Work

To maximize the impact of this work, we recommend the following next steps:

1. **Operational Integration:** The output predictions should be integrated into our CRM and sales platforms, making the predicted fleet size and confidence scores directly available to the sales team to inform their prospecting efforts.
2. **Downstream Model Development:** The `pred_fleet_size_median` should be utilized as a key feature in the next stage of modeling to predict `outstanding_card_count` or a similar business value metric.
3. **Implement a Feedback Loop:** A process should be established for the sales team to provide feedback on the quality of the predicted leads. This real-world data is invaluable and should be used to periodically retrain and refine the model to adapt to changing market dynamics and improve its accuracy over time.

Source code:
https://github.com/wexinc/aips-northstar/tree/northstar-dev