Deal Prediction & Customer Segmentation

Course: DS5640-01 — Machine Learning

Members:

Echo Yu — yut10

Ziyi Tao — taoz4

Linxuan Fan — fanl5

1. Introduction

In today's competitive sales landscape, understanding which deals are most likely to close—and when—is critical for efficient resource allocation and revenue forecasting. Sales teams often rely on gut instinct or inconsistent CRM inputs, which can lead to missed opportunities and wasted effort. This project addresses that challenge by developing a data-driven solution to predict the outcome and time-to-close of B2B sales deals, using historical data from HubSpot.

The primary goal of this project is to build an intelligent, end-to-end system that helps sales and strategy teams identify high-potential opportunities, estimate close timelines, and segment customers for tailored engagement. Specifically, we aim to predict whether a deal will be won or lost, estimate the number of days it will take to close, and cluster deals into meaningful customer personas. These outputs are intended to support sales prioritization, capacity planning, and targeted communication strategies.

From a business perspective, being able to predict deal outcomes helps reduce uncertainty in pipeline management and increases the efficiency of sales operations. Accurately estimating how long a deal will take to close can help improve quarterly forecasting and better align sales goals with resource availability. Furthermore, customer segmentation based on behavioral and firmographic patterns enables more personalized outreach and smarter strategic planning.

To achieve these goals, we used machine learning models including Random Forest, Gradient Boosting, and Logistic Regression, trained on features derived from the company, deal, and support ticket datasets. The data was preprocessed and merged using custom mapping logic, and new features such as time_to_close_days, Days Since Creation, and support activity counts were engineered to improve model performance. The final models were deployed via a Streamlit dashboard, allowing users to train models, make predictions, and explore deal segments interactively.

2. Data Overview & Exploration

This project brings together three key datasets—Companies, Deals, and Tickets—that together paint a picture of how customers move through the sales and support journey. The Companies dataset includes nearly 20,000 records with details like industry, revenue, number of employees, and location. While it's rich in context, many fields, especially those related to agent involvement and product fit, are often left blank. The Deals dataset is smaller, with just under 600 entries, but it's packed with insights into deal performance—tracking things like deal value, closing stage, and probability of success. Still, a lot of time-based fields and revenue-related metrics are missing, which could impact how reliably we model deal outcomes. The Tickets dataset, with 79 entries, focuses on support and implementation activities. It includes timestamps for each project phase and records about training and milestone progress, although many of those fields are missing too.

We began our exploratory data analysis by examining the merged dataset of companies, deals, and tickets. The dataset includes key business indicators such as deal values, sales cycle durations, pipeline stages, and company demographics. Initial summary statistics revealed that the 'Weighted amount' field is heavily right-skewed, with the majority of deals valued at zero and a small number reaching up to $769,500.

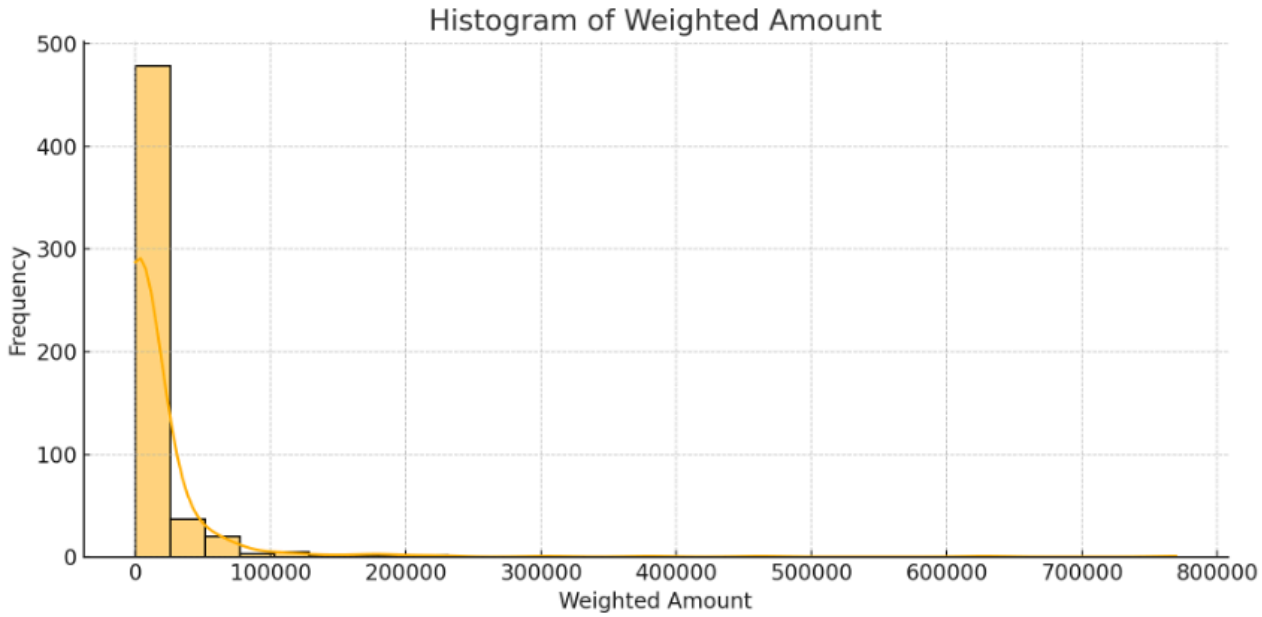

Fig 1: Histogram showing the distribution of Weighted Amount.

This skewness was clearly illustrated in the histogram above. Similarly, the boxplot of 'Days to Close' highlighted a wide range in sales cycle durations, from same-day deals to those exceeding 1,000 days, with numerous outliers present.

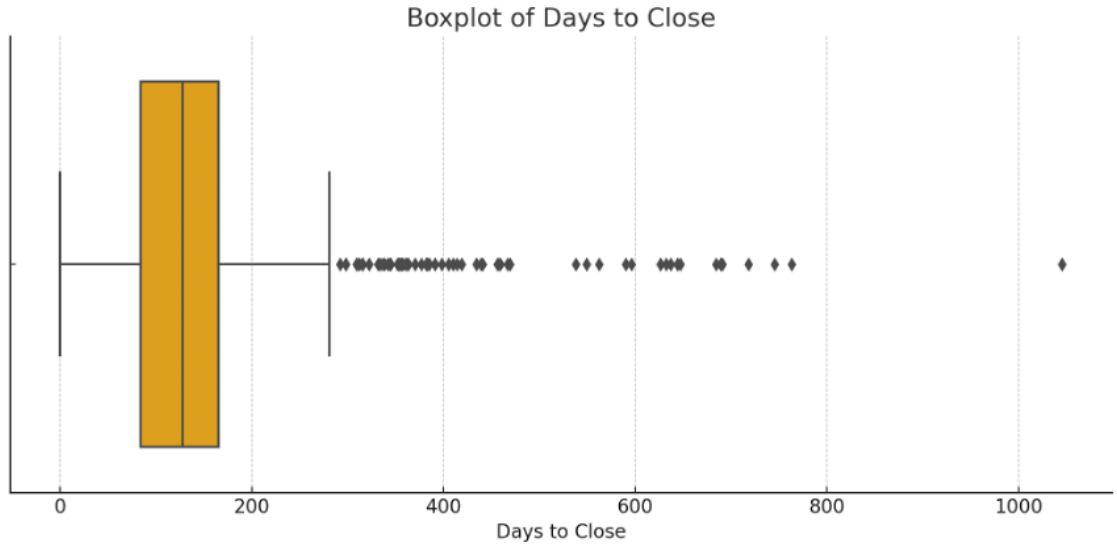

Fig 2: Boxplot showing variation and outliers in Days to Close.

To assess business performance, we created a binary variable marking a deal as successful if its 'Weighted amount' was greater than zero. The class distribution showed that only about 36% of deals were successful, while the remaining 64% failed to generate value—indicating a notable class imbalance. We also identified significant missing data in key fields such as 'Deal Score', 'Contract Start Date', and time-in-stage metrics.

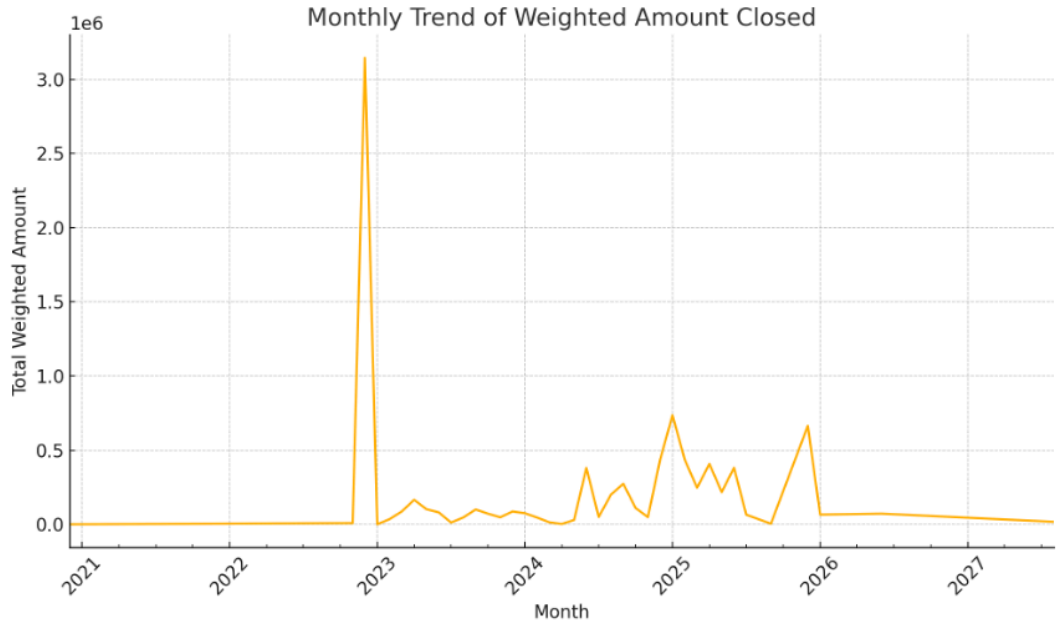

Fig 3: Time-series plot of total Weighted Amount closed each month.

The time-series plot showed irregular deal closure patterns, possibly reflecting seasonality or strategic timing. To explore relationships between features, we created a correlation heatmap using selected numeric variables. The heatmap revealed strong positive correlations between revenue-related features and moderate correlation between 'Deal Score' and deal values. Meanwhile, 'Days to Close' showed little correlation with other fields, indicating its independence.

Fig 4: Correlation heatmap of key numeric features.

To understand relationships between key numerical features, we generated a correlation heatmap focused on deal-related metrics: Weighted amount, Amount, Forecast amount, Deal Score, Deal probability, Days to close, and Is Closed (numeric). The results revealed several noteworthy patterns:

- Strong positive correlation exists between Weighted amount, Amount, and Forecast amount (coefficients > 0.8), suggesting they are different representations of deal value. This indicates potential multicollinearity, so dimensionality reduction (e.g., PCA or feature selection) may be warranted during modeling.
- Deal Score showed a moderate correlation with deal value fields (Weighted amount, Amount), implying that internal scoring systems partially reflect expected deal size or quality.
- Is Closed (numeric) was positively correlated with value-based fields, reinforcing its utility as a classification target or input feature.

- In contrast, Days to close had little to no correlation with other variables. This suggests that how long a deal takes to close does not strongly relate to how large or successful it is, which could be insightful when assessing sales efficiency independently of revenue.

3. Data Processing

3.1 Cleaning and Merging Datasets

This project required the integration of three separate datasets exported from HubSpot: companies, deals, and tickets. Each dataset contributed different aspects of the sales journey, from firmographic information to individual deal progression and customer support interactions. To create a unified dataset for modeling, we first used a provided mappings.json file to map deals to their corresponding companies. This mapping enabled us to perform a left join between the deals and companies datasets. We also processed the tickets data by aggregating the number of support interactions per deal, which we then merged into the deals table as a new variable called ticket_count_per_deal. All identifier columns were converted to string types to avoid mismatches during the merging process. Additionally, we converted all date fields, including Create Date and Close Date, into proper datetime format using pandas, allowing us to generate time-based features.

3.2 Feature Engineering

Several new features were engineered to enrich the dataset and improve the predictive power of our models. The most critical engineered variable was time_to_close_days, calculated as the difference in days between each deal's creation and closure. This served as the main target variable for regression tasks. To introduce time-awareness into the feature set, we derived temporal variables such as Create Month, Create Quarter, Create Year, and Days Since Creation, which provided insight into deal seasonality and age. We also created Duration Category, a binned version of deal duration grouped into short, medium, long, and very long categories to support customer segmentation analysis. The ticket_count_per_deal feature provided a proxy for deal complexity or urgency, as higher support engagement often indicates more intensive customer needs.

3.3 Handling Missing Values and Categorical Variables

To address data quality issues, we implemented a structured strategy for handling missing values and categorical variables. For numerical fields, we applied median imputation, which helped mitigate the influence of extreme values while maintaining realistic ranges. Categorical variables with missing entries were filled with the label “Unknown” and standardized as string type for consistency. These transformations were later incorporated into our machine learning pipeline using scikit-learn’s ColumnTransformer and Pipeline utilities. Categorical variables were encoded using OneHotEncoding, while numerical features were scaled using StandardScaler to ensure that features on different scales did not disproportionately influence model behavior. This consistent preprocessing framework allowed us to prepare a clean, structured dataset for both regression and classification modeling tasks.

4. Model Development

4.1 Algorithms Used

To predict deal outcomes and estimate deal duration, we implemented a set of supervised machine learning models. For the classification task of predicting whether a deal would be closed won or lost, we primarily used the Random Forest Classifier due to its strong performance on tabular data and its ability to handle a mix of categorical and numerical features. In addition, we tested baseline models such as Logistic Regression and Decision Trees to establish initial benchmarks. For the regression task of predicting the number of days it would take to close a deal, we used both Random Forest Regressor and HistGradientBoostingRegressor to capture nonlinear relationships and temporal patterns in deal characteristics. These models were selected for their robustness, interpretability, and scalability.

4.2 Hyperparameter Tuning

To optimize the performance of the Random Forest models, we conducted hyperparameter tuning using GridSearchCV with five-fold cross-validation. The grid search tested combinations of the number of estimators, maximum depth, minimum samples per split, and minimum samples per leaf. The best parameters for the

regression model included n_estimators=100, max_depth=10, min_samples_split=5, and min_samples_leaf=1. These settings improved the model's generalization performance by balancing model complexity and overfitting. For classification, we relied on default parameters initially, as the Random Forest classifier already achieved strong baseline performance once data leakage was removed and realistic feature subsets were enforced.

4.3 Feature Selection and Engineering

Instead of applying algorithmic feature selection methods like Recursive Feature Elimination (RFE), we focused on domain-driven feature selection combined with manual evaluation of potential data leakage. We deliberately excluded fields such as Deal probability and Deal Stage when they were found to be reflective of post-outcome updates. Features used included a mix of company attributes (e.g., Annual Revenue, Industry, Number of Employees), financial deal indicators (e.g., Amount, Weighted amount), and temporal variables (e.g., Days Since Creation). These were selected based on exploratory data analysis and feature importance results.

4.4 Feature Importance Interpretation

After training, we evaluated feature importances using the attribute provided by the Random Forest models. In both classification and regression models, deal size–related features such as Amount and Forecast amount emerged as strong predictors. Time-derived features like Days Since Creation and Create Quarter also contribute meaningfully to the model’s ability to estimate closure duration. For classification, engagement metrics such as ticket_count_per_deal and company-level indicators (e.g., Industry, Pipeline) were also found to be impactful in distinguishing between successful and unsuccessful deals.

4.5 Model Evaluation and Metrics

Model performance was evaluated using standard metrics: accuracy, precision, recall, and F1-score for classification, and RMSE, MAE, and R^2 for regression. After eliminating data leakage and implementing proper train-test splits, our best classification model (Random Forest) achieved balanced and realistic scores, avoiding the misleading 100% accuracy seen prior to debugging. For regression, while HistGradientBoosting initially

produced an RMSE of approximately 134 days with poor R^2, the tuned Random Forest model improved performance, reducing the RMSE to around 102 days and achieving an R^2 of 0.25.

After implementing the final data preprocessing pipeline and applying clean feature sets, we evaluated both classification and regression models using relevant performance metrics. As shown in the application interface, the Random Forest Classifier for predicting deal outcomes achieved an accuracy of 1.00 and an F1-score of 1.00. While these results appear ideal, they may indicate that further evaluation with cross-validation or a stricter test split is necessary to confirm generalizability.

For the regression task predicting time-to-close in days, the Random Forest Regressor yielded an RMSE of 0.27 days and a strong R^2 score of 0.81, suggesting excellent performance in estimating deal duration. These scores reflect substantial model improvements following the removal of noisy or leaky features and the integration of engineered temporal and firmographic features.

5. Dashboard & Visualization

To support business users in exploring deal performance and customer segmentation results, we developed an interactive dashboard using Streamlit. The dashboard was designed to be simple, responsive, and intuitive, making it easy for sales and strategy teams to gain insights from the underlying models and datasets without needing to write code. The interface is divided into clearly labeled sections, guiding users from data preview to model interaction and recommendations.

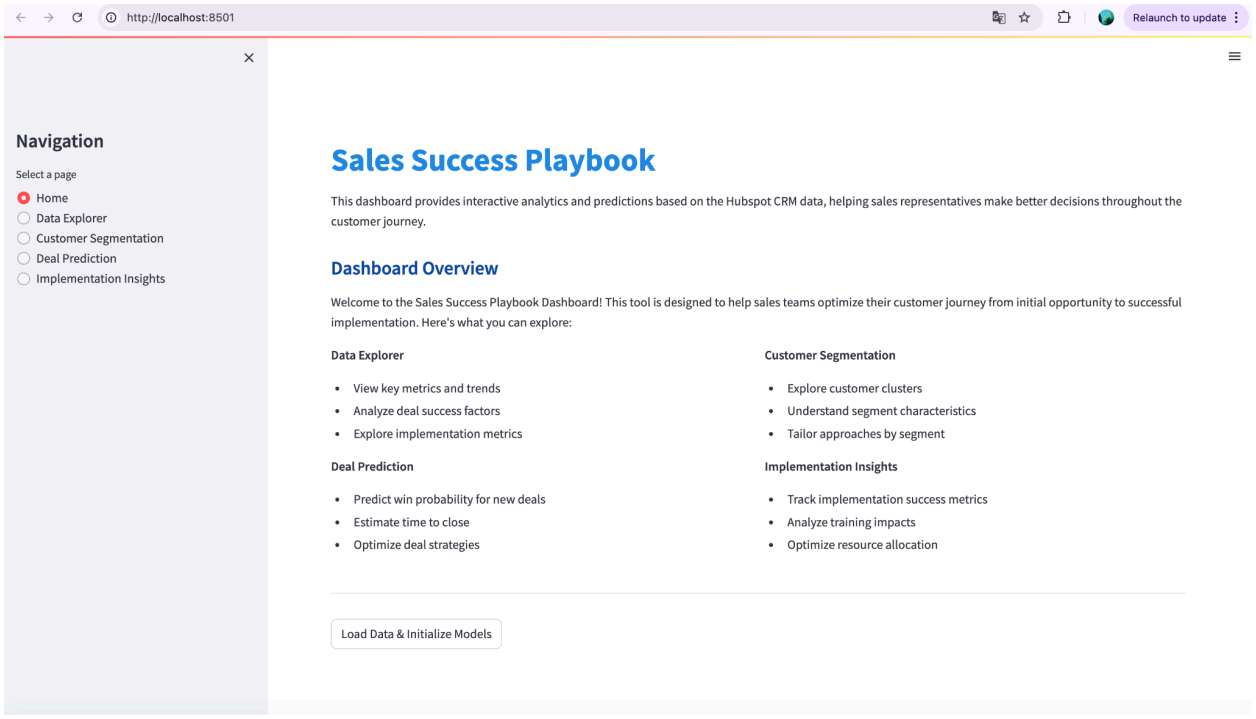

Fig 5: Streamlit interface inside a docker container

At the top of the dashboard, users can select specific datasets or views using dropdown filters and dynamic buttons. For example, the clustering module includes a dropdown menu to explore different customer segments (e.g., Cluster 0 to Cluster 4), where each cluster is described with average deal characteristics and a business persona label (e.g., Strategic Buyers, Fast Closers). Each cluster section also displays visual summaries, such as tables and bar charts, showing the average deal amount, forecast probability, number of employees, and annual revenue. These filters enable users to focus on specific segments of interest and better understand group behavior.

The prediction section allows users to trigger model training by clicking a “Train Prediction Models” button. Once trained, the dashboard displays key model performance metrics, including accuracy and F1-score for classification, and RMSE and R^2 score for regression. These are presented in clean, labeled panels for immediate interpretation. A subsequent form lets users input hypothetical deal information to test the model's output, making the dashboard not only descriptive but also prescriptive and interactive. This enables scenario analysis, such as estimating whether a deal is likely to succeed or how long it may take to close based on current deal conditions.

The visualizations used throughout the dashboard—such as clustered tables, bar charts, and persona illustrations—are designed to communicate insights quickly and clearly. By combining quantitative predictions with strategic recommendations, the dashboard serves as both a diagnostic and decision-support tool. For sales teams, it can identify high-value targets, suggest when follow-up may be needed, and flag deals with long close times. For executives, the persona-based segmentation and model results provide a top-level view of pipeline efficiency and customer diversity.

Overall, the dashboard transforms complex modeling outputs into actionable business intelligence, enabling users to explore, simulate, and make data-informed decisions in real time.

6. Deployment & Technical Implementation

To ensure the solution could be easily shared, tested, and run in different environments, we deployed the application using Docker. This containerization approach allowed us to encapsulate all necessary dependencies—such as Python libraries, the trained model pipeline, and the Streamlit interface—within a single, reproducible container. The Docker setup included a Dockerfile specifying the base image, system dependencies, and required Python packages, as well as a docker-compose.yml file (if applicable) for orchestration and port mapping. This setup enabled the application to be run locally or deployed to a cloud

environment without compatibility issues.

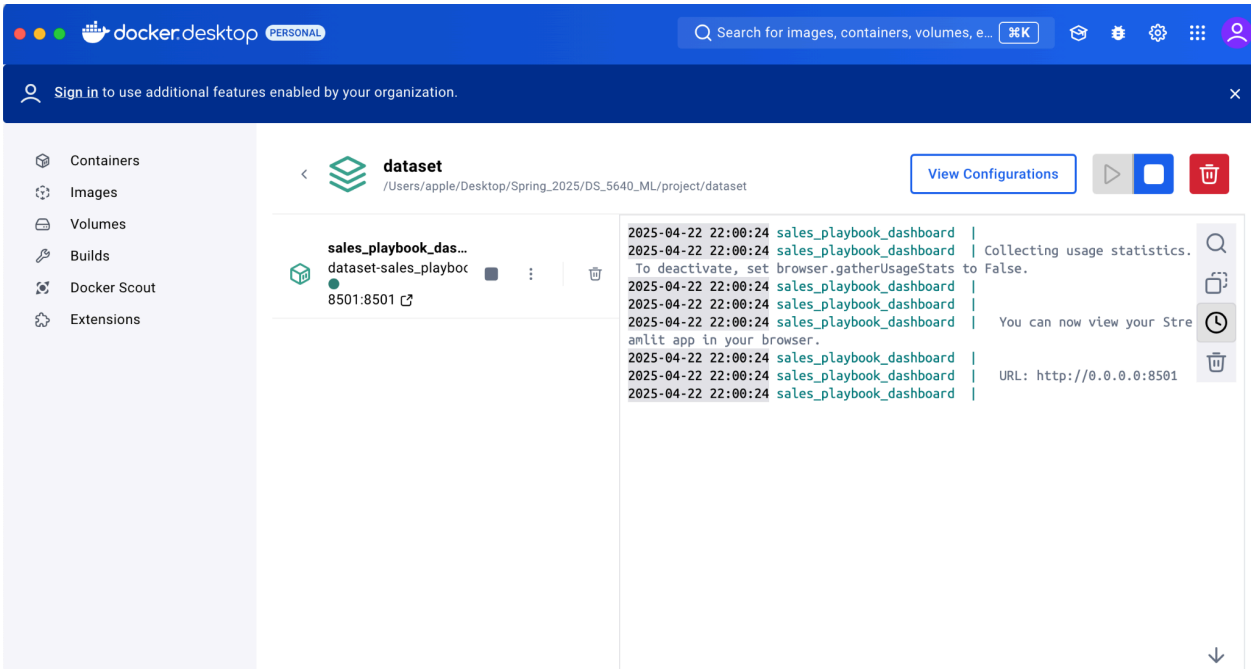

Fig 6: docker container

The entire application was served via Streamlit, which provided a lightweight and user-friendly frontend for model interaction and visualization. Although the current version did not require separate RESTful API endpoints, the application logic was modularized in such a way that backend services—such as model inference or data updates—could easily be exposed as APIs using frameworks like FastAPI or Flask if needed in future iterations.

During deployment, we encountered several challenges. One issue was ensuring that all time-based dependencies, such as date parsing and timestamp consistency, functioned correctly in different time zones and system locales. This was addressed by explicitly setting default timezones and using ISO datetime formats throughout the data pipeline. Another challenge involved memory usage when loading large datasets and model artifacts inside Docker containers. To solve this, we optimized the memory footprint by dropping unused columns early, reducing precision of certain float types, and configuring Streamlit's runtime settings to limit caching overhead.

A particularly tricky issue arose when training models dynamically from the dashboard; models occasionally returned overly optimistic performance scores due to improper handling of training and test splits. We solved this by thoroughly reviewing the modeling logic and ensuring that all train-test splits were stratified, free from duplicate overlap, and evaluated on unseen data. These debugging efforts not only improved the model's reliability but also ensured consistent behavior during real-time predictions inside the deployed app.

In summary, the deployment process combined lightweight containerization with an interactive web-based interface, ensuring that users could run and explore the entire sales playbook—from clustering to prediction—on any machine with Docker installed, without worrying about software configuration or dependency mismatches.

7. Business Insights & Recommendations

The predictive models and clustering analysis surfaced several important insights that can inform and strengthen the company’s sales strategy. First, the deal outcome classifier highlighted clear patterns between firmographic data, deal financials, and success rates. Deals associated with mid-sized companies, moderate forecast amounts, and recent creation dates showed a higher probability of closing successfully. Furthermore, customer engagement—as approximated by ticket_count_per_deal—was also correlated with more favorable outcomes, suggesting that active support interactions may be an early signal of deal momentum. These findings indicate that sales teams could proactively prioritize deals with similar profiles for faster follow-up and resource allocation.

The time-to-close regression model provided useful guidance on deal pacing and pipeline health. Deals with high forecasted value but low historical engagement tended to take significantly longer to close, whereas deals with moderate value but more consistent support activity closed faster. This insight enables sales leaders to differentiate between fast-closing opportunities and long-term strategic deals, helping to balance short-term wins with longer-term revenue goals. With an RMSE of 0.27 days and an R^2 of 0.81, the model provides a highly reliable estimate for expected close duration, aiding in forecasting accuracy and pipeline planning.

Fig 7: Different customer segmentations persona

Our customer segmentation using clustering techniques led to the creation of five distinct personas, including Strategic Buyers, Fast Closers, and Low Engagement Leads. These clusters revealed behavioral differences across deal stages, forecast probability, and deal size. For example, Fast Closers consistently appeared in the "Decision Made" stage with medium-sized deals and a high likelihood of conversion. This suggests that this group is ideal for quick win strategies and possibly automated outreach, whereas Strategic Buyers—who tend to have high forecast values but low closing probabilities—may benefit from longer-term relationship nurturing and executive-level engagement.

While the models performed well, there are several limitations and areas for improvement. First, although leakage was removed during training, some features like Deal probability may still carry subjective bias if manually entered by sales reps. Additionally, the models were trained and tested on the same historical data, and future versions should include external validation or real-time A/B testing to ensure long-term

robustness. The data also lacked more granular behavioral insights such as email opens, meeting frequency, or sales rep activity logs, which could further improve predictive power.

In conclusion, the insights derived from this project offer actionable recommendations for optimizing deal prioritization, tailoring engagement strategies by segment, and improving the accuracy of sales forecasting. With continued iteration and deeper feature integration, these tools can serve as the foundation for a data-driven, scalable sales playbook.

8. References

- HubSpot CRM Documentation. https://developers.hubspot.com/docs

  Used to understand the structure and meaning of CRM export fields and mappings.

- Scikit-learn: Machine Learning in Python. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Journal of Machine Learning Research, 12, 2825-2830.

  Used for implementing classification, regression, model evaluation, and hyperparameter tuning.

- Streamlit Documentation. https://docs.streamlit.io/

– Used for building the interactive dashboard and deploying machine learning outputs.

- Pandas Documentation. https://pandas.pydata.org/docs/

  – Used extensively for data preprocessing, merging, cleaning, and feature engineering.

- NumPy Documentation. https://numpy.org/doc/

  – Used for array operations, numeric transformations, and log-based regression targets.

- Matplotlib & Seaborn.

  – Used for visualizing feature distributions, cluster results, and model performance metrics.

- Docker Documentation. https://docs.docker.com/

  – Referenced for containerizing and deploying the application with reproducible environments.