SLA for S2L Trend - Module

SLA Determination Function

Contents

Abstract

In the fiercely competitive world of sales, every minute counts. The ability to connect with potential customers swiftly can make or break a sale. This is why the concept of lead response time, or the duration it takes to contact a lead after it's generated and available in the pipeline, has become a critical metric for companies in the sales industry. In this article, we delve into the significance of lead response time and discuss a specially designed function that calculates the time difference for sale and non-sale categories of generated leads, shedding light on how this can optimize sales processes.

To facilitate the measurement of lead response time in the sales industry, a specialized function has been designed. This function calculates the time difference between sales and non-sale categories of generated leads, offering invaluable insights into a company's performance.

Objective of the Function

Lead response time is a metric that quantifies the speed at which sales teams reach out to potential leads. It encompasses the entire process from lead generation to initial contact. In today's fast-paced digital landscape, where consumers expect quick and personalized responses, lead response time has never been more crucial.

The formula for calculating lead response time is relatively straightforward: it's the time elapsed between the moment a lead is generated and made available in the sales pipeline, and the time the sales team initiates contact with that lead. This is measured in minutes, giving businesses a clear picture of their efficiency in capitalizing on potential opportunities.

In a perfect world, sales teams would be able to contact every lead the moment they are generated. However, in reality, this is unattainable. With a finite number of resources and leads pouring into the pipeline, it's essential to prioritize and allocate these resources effectively.

Regularly analyzing data to determine if the sweet spot in the lead response strategy needs to be adjusted based on changing market conditions or lead behavior.

Introduction

In a typical sales environment, it's a given that not all leads can be contacted simultaneously. The number of sales representatives available, the diversity of lead sources, and the potential variations in lead quality all contribute to the need for prioritization. Therefore, it becomes essential to strategize the allocation of resources to achieve the maximum impact.

Lead response time, the time taken to initiate contact with a lead after it's generated, plays a crucial role in sales outcomes. The primary goal is to respond quickly to all leads, but as mentioned earlier, this isn't always feasible. This is where the concept of significant difference comes into play.

The significant difference refers to the point at which the lead response times for sale and non-sale categories diverge most significantly when plotted on two different line graphs. In other words, it's the percentage of leads contacted where the response time for potential sales opportunities starts to outshine the response time for non-sales leads, regularly analyzing data to identify the inflection point where the lead response times for sale and non-sale leads begin to significantly diverge is a critical step in optimizing your sales strategy

Problem Statement

<u>The key challenges faced by sales management to identify the Optimal Point for Lead Response Time:</u>

Sales management faces various challenges in identifying the optimal point for lead response time. Balancing the need for rapid response with resource limitations is a common hurdle.

Additionally, it's a complex task due to varying lead quality and customer behavior. Gathering and analyzing relevant data accurately can be challenging, and achieving consensus on the optimal point within the organization is often met with resistance. Continuous adaptation as market dynamics change further complicates the process. Finally, striking the right balance between automated responses and personalization requires careful consideration. These challenges highlight the multifaceted nature of pinpointing the ideal lead response time in sales management.

Data Overview

The data used to demonstrate this module is basically a sales data having time taken to contact lead for the very first time units in minutes(Lead2FirstIntr_datedifference_Minute), Sale non-sale column(saleflag) with 0 referring to non-sale and 1 referring sale, date of order(order_date), lead generation date(lead_date). The columns are described below with their data types

Dataset column names	Description	Data type
Lead2FirstIntr_datedifference_Minute	time taken to contact lead for the very first time units in minutes	Continuous
saleflag	'0' stands for unsold and '1' for sold entries	Binary discrete
order_date	Sales Date	Date
lead_date	Date on which lead was generated	Date

	Lead2FirstIntr_datedifference_Minute	saleflag	lead_date	order_date
0	5	0	2021-08-27 17:01:46.000	NaN
1	4	0	2021-07-14 11:39:48.000	NaN
2	1007	0	2021-09-03 21:42:04.000	NaN
3	5	0	2021-11-09 14:53:31.000	NaN
4	23	0	2021-10-01 08:22:57.000	NaN
...	...	...	...	...
95720	1	0	2021-09-27 13:27:06.000	NaN
95721	456	0	2021-03-31 01:45:41.000	NaN
95722	5	0	2021-11-28 16:43:51.000	NaN
95723	2747	0	2021-04-24 17:09:55.000	NaN
95724	347	0	2021-06-19 03:19:53.000	NaN

95725 rows × 4 columns

INSTALLATION

Installation of the package is very simple just typing the code referred bellow we can install our package into our python environment

```
pip install businessmodels==0.0.9
```

```
pip install businessmodels==0.0.7

Collecting businessmodels==0.0.7
  Downloading businessmodels-0.0.7-py3-none-any.whl (6.4 kB)
Collecting business-models-initial (from businessmodels==0.0.7)
  Downloading business_models_initial-0.0.2-py3-none-any.whl (4.0 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from business-models-initial->businessmodels==0.0.7) (1.5.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from business-models-initial->businessmodels==0.0.7) (1.11.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas->business-models-initial->businessmodels==0.0.7) (2.
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->business-models-initial->businessmodels==0.0.7) (2023.3.post1
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas->business-models-initial->businessmodels==0.0.7) (1.23.5)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas->business-models-initial->businessmodels==
Installing collected packages: business-models-initial, businessmodels
Successfully installed business-models-initial-0.0.2 businessmodels-0.0.7
```

The module needs to be imported from the businessmodels package using the following line of code

```
from businessmodels import sla_determine
```

```
column = "Lead2FirstIntr_datedifference_Minute"
sla_check = sla_determine.SLA_Determination(df, column)
```

In this line, the code is creating an instance of the "SLA_Determination" class from the "sla_determibne" module and assigning it to a variable named "sla_check". The instance of the "SLA_Determination" class is assigned to the variable "sla_check," which can be used later in the code to interact with the functionalities provided by the "SLA_Determination" class. We will use this instance to call the all functions within the class.

The DataFrame df is passed as an argument to the "SLADetermination" class constructor and column as an attribute from the dataframe. This suggests that the "SLADetermination" class expects a DataFrame and a column which is basically the first lead response time or S2L as input.

```
a, b, c, d = sla_check.calculate_sla(st, et)
```

The code a, b, c, d = sla_determination.categorize_lead_to_sale_calculate() is assigning the results of a method call to two variables, a, b, c and d.

CODES

```
import pandas as pd

class SLA_Determination:
    def __init__(self, df_selected, column1):
        self.df_selected = df_selected
        self.column1=column1
```

```
    def calculate_sla(self, st, et):
        df_sale = self.df_selected[(self.df_selected.saleflag == 1) &
((self.df_selected.order_date >= st) & (self.df_selected.order_date < et))]
        df_sale_perc = pd.DataFrame(data=df_sale[self.column1].quantile(q=[i/100 for i in
range(75, 96)]))
```

```
        df_nosale = self.df_selected[(self.df_selected.saleflag == 0) &
((self.df_selected.lead_date >= st) & (self.df_selected.lead_date < et))]
        df_nosale_perc = pd.DataFrame(data=df_nosale[self.column1].quantile(q=[i/100 for i in
range(75, 96)]))
```

```
        df_perc_diff = df_nosale_perc.Lead2FirstIntr_datedifference_Minute -
df_sale_perc.Lead2FirstIntr_datedifference_Minute
        diff = pd.DataFrame(df_perc_diff)
```

```
        df_describe_sale = df_sale.describe()[[self.column1]].round(0)
        df_describe_sale = pd.concat([df_describe_sale.loc[['min', 'max', 'mean'], :],
df_sale_perc.loc[[0.75, 0.80, 0.85, 0.90, 0.95], :]])
        df_describe_sale.index = ['Minimum', 'Maximum', 'Mean', '75th Perc', '80th Perc',
'85th Perc', '90th Perc', '95th Perc']
        df_describe_sale.rename(columns={'Lead2FirstIntr_datedifference_Minute': 'Sale'},
inplace=True)

        df_describe_nosale = df_nosale.describe()[[self.column1]].round(0)
        df_describe_nosale = pd.concat([df_describe_nosale.loc[['min', 'max', 'mean'], :],
df_nosale_perc.loc[[0.75, 0.80, 0.85, 0.90, 0.95], :]])
        df_describe_nosale.index = ['Minimum', 'Maximum', 'Mean', '75th Perc', '80th Perc',
'85th Perc', '90th Perc', '95th Perc']
        df_describe_nosale.rename(columns={'Lead2FirstIntr_datedifference_Minute': 'Lead'},
inplace=True)
```

```
        df_table = pd.merge(df_describe_sale, df_describe_nosale, left_index=True,
right_index=True)
        df_table = df_table.T.round(0)
        df_table.insert(0, 'Category', ['Sale', 'Non-Sale'])
```

```
        return df_sale_perc, df_nosale_perc, diff, df_table
```

IMPLEMENTATION

Import Libraries:

The code starts by importing the Pandas library, a popular Python library for data manipulation and analysis.

Define the SLA_Determination Class:

A class named SLA_Determination is defined. This class is designed to calculate and compare Service Level Agreement (SLA) metrics for sale and non-sale leads.

Constructor (__init__):

The class constructor takes two arguments: df_selected and column1.
df_selected: This is the DataFrame containing the data for analysis.
column1: This is the name of the column within the DataFrame that represents the lead response time in minutes.

calculate_sla Method:

This method calculates SLA metrics and returns the results. It takes two arguments, st (start time) and et (end time), which define a time range for analysis.

Data Filtering:

The method starts by filtering the DataFrame df_selected to separate sale leads (those with saleflag == 1) and non-sale leads (those with saleflag == 0) within the specified time range.

Calculate Percentiles:

For both sale and non-sale leads, the code calculates percentiles for the lead response times. Specifically, it calculates the 75th, 80th, 85th, 90th, and 95th percentiles for each category.

Percentile Difference Calculation:

The code calculates the difference in percentiles between sale and non-sale leads for the specified time range. This provides insights into how response times differ between the two categories.

Generate Summary Statistics:

For both sale and non-sale leads, the code generates summary statistics, including the minimum, maximum, and mean response times. These statistics are rounded to the nearest integer.

Create Comparison Tables:

The summary statistics and percentile data are organized into tables for both sale and non-sale leads. These tables are structured to provide an overview of the data, including minimum, maximum, mean, and specific percentiles.

Merge and Format Tables:

The code then merges the sale and non-sale tables to create a single comparison table. It also adds a "Category" column to identify whether the data is for sale or non-sale leads.

Return Results:

The method returns several objects, including DataFrames representing the percentiles for sale and non-sale leads, the difference in percentiles, and the comparison table.

```
df_sale_perc.head(10)
```
✓ 0.0s

	Lead2FirstIntr_datedifference_Minute
0.75	58.00
0.76	67.12
0.77	82.00
0.78	98.00
0.79	115.23
0.80	140.60
0.81	167.00
0.82	215.34
0.83	287.42
0.84	359.24

```
df_nosale_perc.head(10)
```
✓ 0.0s

	Lead2FirstIntr_datedifference_Minute
0.75	237.0
0.76	286.0
0.77	344.0
0.78	415.0
0.79	468.0
0.80	513.0
0.81	551.0
0.82	582.0
0.83	607.0
0.84	632.0

```
diff.head(10)
```
[16] ✓ 0.0s

	Lead2FirstIntr_datedifference_Minute
0.75	179.00
0.76	218.88
0.77	262.00
0.78	317.00
0.79	352.77
0.80	372.40
0.81	384.00
0.82	366.66
0.83	319.58
0.84	272.76

PROBLEMS 7 TERMINAL SEARCH TERMINAL OUTPUT JUPYTER COMMENT

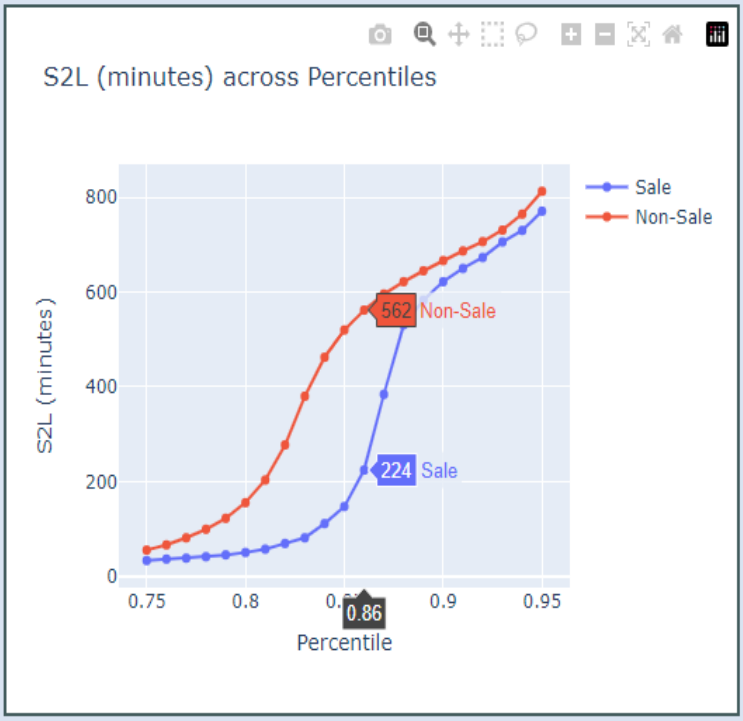
```
df_table
```

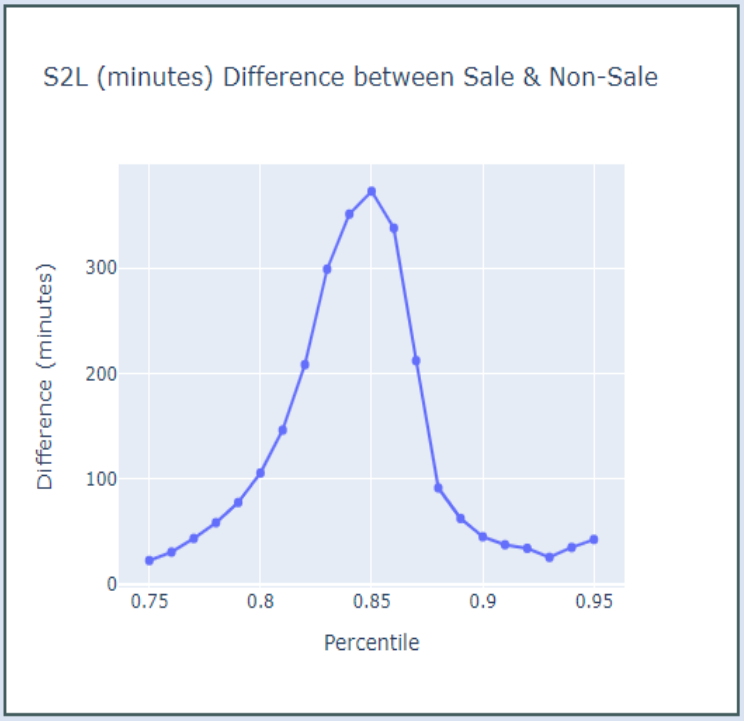
0.0s

	Category	Minimum	Maximum	Mean	75th Perc	80th Perc	85th Perc	90th Perc	95th Perc
Sale	Sale	0.0	311295.0	372.0	58.0	141.0	437.0	669.0	909.0
Lead	Non-Sale	0.0	364267.0	304.0	237.0	513.0	653.0	779.0	1091.0

+ Code + Markdown

S2L Trends





CONCLUSION

The module named "sla_determination" published under python package "businessmodels" makes it easy to get interpreted result to a powerful tool for sales management to analyze and compare lead response times between sale and non-sale categories. By calculating percentiles, summary statistics, and percentile differences, it offers valuable insights into the efficiency of lead response strategies. The generated comparison tables enable data-driven decision-making, allowing sales teams to optimize their approaches based on real-time data and tailored to different lead categories. This approach can ultimately enhance sales outcomes and help organizations meet their Service Level Agreement (SLA) goals.

PACKAGE REFERENCE

For your reference you can visit the https://pypi.org/project/packageimportant/#description page where this python package is available.