Simulate Sale Report

Sueksit Vachirakumthorn
`https //github.com/BlackBoxBanner/KMUTTxFIT_intern`

July 19, 2024

Contents

List of Figures

List of Tables

1 Introduction

The purpose of this project is to analyze sales data in restaurants using synthetic data generated based on real-world POS (Point of Sale) systems and online sales reports. Since a suitable dataset was not found on Kaggle, the project will proceed with generating its own synthetic data. The validity of the data is crucial for both prediction and analysis, so the generated data has been made as realistic as possible by incorporating features such as a custom menu with prices, defined peak hours, and adjusted quantity based on these peak hours.

1.1 Objective

The project objective has shifted from predicting sales to analyzing sales data to improve sales. To achieve this, the approach will involve conducting a comprehensive analysis of sales data. This includes examining item performance by analyzing total sales quantity and revenue for each item, as well as evaluating sales performance across different categories. Additionally, popular items and categories will be identified based on sales data, allowing for targeted marketing campaigns and promotions.

The analysis will also delve into price point evaluation to investigate the relationship between pricing and sales volume, as well as assessing the impact of bundling items on sales. Furthermore, time-based analysis will be conducted to identify peak sales periods and seasonal variations, enabling adjustments to promotions and offerings accordingly.

Finally, the insights gained from the sales data analysis will inform decisions about inventory management by forecasting demand using historical sales data to optimize inventory levels. By leveraging these findings, the goal is to improve sales through informed pricing, promotion, marketing, and inventory management strategies.

2 Background and Context

Understanding the environment surrounding the location you wish to work with is crucial. In this instance, I am working with data from Point-of-Sale (POS) systems in Australian cafes, where publicly available data is limited. As such, generating my own data becomes necessary. Prior to commencing any analysis, it is essential that I conduct research on a relevant topic.

2.1 Research

The insights into the average daily and monthly sales for cafes in Australia, highlighting the importance of understanding these figures for success.

2.1.1 Average Daily Sales

- Small shops with high bakery sales can average $4,000 on weekdays and $6,000 to $8,000 on weekends.
- Busy cafes can serve up to 1,000 coffees a day, translating to significant daily sales (around $1,600 to $2,500).
- High-end cafes in financial districts or tourist-heavy areas can make up to $16,000 a day during peak times.

2.1.2 Average Monthly Sales

- The monthly retail revenue for cafes, restaurants, and takeaway food services in Australia was over 5.3 billion AUD in March 2024. [1]
- For a middle-of-the-road cafe turning over $500,000 annually, this translates to approximately $41,667 per month. [1]
- Another source indicates that cafe owners can earn between $60,000 to $150,000 annually, depending on the business size, which breaks down to $5,000 to $12,500 monthly. [1]

2.1.3 Profit Margins and Costs

- The average net profit for a cafe in Australia is around 10% of sales. [2]
- Typical cost distributions include:
  - 35-40% for goods
  - 30-35% for labor
  - 10-15% for rent
- Efficient management and focusing on high-margin products can improve profitability. [3]

2.1.4 Research Summary

Running a cafe in Australia can yield varying sales figures based on several factors. On average, daily sales can range from $1,800 to $16,000, while monthly sales can be between $5,000 to $41,667. Profit margins are generally around 10%, with significant costs attributed to goods, labor, and rent. Efficient management and strategic product offerings are key to maximizing profitability.

2.2 Data generation

To generate synthetic data for this project, I employed Python programming language and utilized a custom-built script. This approach allowed me to create a dataset that mimics real-world Point-of-Sale (POS) systems in Australian cafes.

2.2.1 Script Overview

The Python script used for data generation is designed to produce a comprehensive dataset that captures various aspects of cafe operations, including menu items, prices, sales quantities, and peak hours. The script utilizes several packages and libraries to facilitate the data generation process.

2.2.2 Packages Used

1. **Faker:** A package used to generate fake data, including names, addresses, and other information. In this case, it was used to generate menu items with names and prices.
2. **DynamicProvider:** A provider for Faker that allows you to define custom fake data generators. This was used to create a dynamic menu system where menu items can be generated on the fly.
3. **Random:** A built-in Python module used to simulate random fluctuations in sales quantities and peak hours.
4. **datetime:** Built-in Python modules used to generate dates and times for transactions, as well as simulate time intervals between transactions.
5. **timedelta:** Built-in Python modules used to generate dates and times for transactions, as well as simulate time intervals between transactions.
6. **uuid:** A package used to generate unique identifiers for each transaction.
7. **tqdm:** A package used to display a progress bar while generating data, making the process more visually appealing.

2.2.3 Script Code

Custom menu

```
menus_provider = DynamicProvider(
  provider_name = "menus",
  elements=menus
)
```

```

fake.add_provider(menus_provider)

```

The provided code defines a custom fake data generator using the Faker library, specifically the DynamicProvider class. This generator is designed to produce menu-related data, with the provider name set to "menus" and the elements parameter populated with the menus list, which contains the actual menu items.

Custom menu

```
def is_open(date_time):
  opening_hours = {
    0: (None, None),  # Monday: Closed
    1: ("07:00", "15:00"),  # Tuesday
    2: ("07:00", "15:00"),  # Wednesday
    3: ("07:00", "15:00"),  # Thursday
    4: ("07:00", "15:00"),  # Friday
    5: ("08:00", "15:00"),  # Saturday
    6: ("08:00", "15:00"),  # Sunday
  }

  day = date_time.weekday()
  opening_time, closing_time = opening_hours[day]

  if opening_time is None or closing_time is None:
    return False

  opening_time = datetime.strptime(opening_time, "%H:%M").time()
  closing_time = datetime.strptime(closing_time, "%H:%M").time()
  current_time = date_time.time()

return opening_time <= current_time <= closing_time

```

The "is_open" function is designed to determine whether a cafe is open or closed based on its operating hours, which are defined by a dictionary "opening_hours". The function takes a "date_time" object as input and returns a boolean value indicating whether the cafe is open at that specific time.

The function first determines the day of the week using the "weekday()" method, then retrieves the corresponding opening and closing times from the "opening_hours" dictionary. If the cafe is closed on that day (i.e., the opening or closing time is None), the function returns "False".

Otherwise, it converts the opening and closing times to "time" objects using the "strptime" method, and then compares the current time with the opening and closing times using the "<=" operator. If the current time falls within the cafe"s operating hours (i.e., between the opening and closing times), the function returns "True", indicating that the cafe is open.

3 Analysis

3.1 Data Structure

The project involves the rigorous analysis of three distinct datasets, specifically designed to capture various facets of the sales data. Each dataset has been meticulously compiled and prepared for simulation purposes, enabling a nuanced understanding of the underlying patterns and trends. Additionally, a comprehensive dataset has been created by combining the individual datasets, allowing for a holistic examination of the sales data. This integrated approach enables the identification of correlations and relationships between different variables, ultimately informing data-driven decisions.

3.1.1 Menu

The Menu dataset is a Pandas DataFrame with 48 rows and 6 columns. The dataset contains information about menu items, including unique identifiers (uuid), names, categories, prices, descriptions, and costs. Each row represents a single menu item, and each column provides additional details about that item. The data types for this dataset are float64 for the price and cost columns, and object for the remaining columns. With a memory usage of approximately 2.4 KB, this dataset is relatively small.

Column	Data Type
uuid	string
name	string
category	string
price	float64
description	string
cost	float64

Table 1: Menus Data Structure

3.1.2 Add-ons

The Add-ons dataset is a Pandas DataFrame with 46 rows and 5 columns. This dataset contains information about additional items or modifications that can be added to menu items, including unique identifiers (uuid), menu items, names, prices, and costs. Each row represents a single add-on item, and each column provides additional details about that item. The data types for this dataset are float64 for the price and cost columns, and object for the remaining columns. With a memory usage of approximately 1.9 KB, this dataset is relatively small compared to the Sales dataset.

Column	Data Type
uuid	string
menu	string
name	string
price	float64
cost	float64

Table 2: Add Ons Data Structure

3.1.3 Sales

The Sales dataset is a Pandas DataFrame with an impressive 108,000 rows and 4 columns. This dataset contains information about sales transactions, including unique identifiers (uuid), dates and times, menu items sold, and additional items or modifications added to those menu items. Each row represents a single sales transaction, and each column provides additional details about that transaction. The data types for this dataset are all object, indicating that the columns contain text or categorical data. With a memory usage of approximately 3.3 MB, this dataset is significantly larger than the Menu dataset.

Column	Data Type
uuid	string
date_time	string
menu_item	string
add_ons	string

Table 3: Sales Data Structure

3.1.4 Sales Combine

This dataset appears to be a collection of sales data for various menu items. The data structure is a Pandas DataFrame, which is a two-dimensional labeled data structure with columns of potentially different types. Each row in the DataFrame represents a single sales transaction, and each column contains information about that transaction.

The dataset has 27 columns, including categorical variables like "menu_name" and "category", numerical variables like "price" and "sales", and datetime variables like "date_time". The data is non-null for all 108,000 entries, indicating that there are no missing values in the dataset. The data types of the columns include object (categorical), float64 (numerical), int32 (integer), and category (a categorical type specific to Pandas).

Column	Data Type
menu_name	string
category	string
price	float64
cost	float64
date_time	datetime64[ns]
add_ons	string
date	datetime64[ns]
time	string
add_ons_price	float64
total_price	float64
day_of_week	string
sales	float64
year	int32
month	int32
day	int32
hour	int32
total_sales	float64
total_cost	float64
total_sale_difference	float64
sale_difference	float64
time_category	category
reduced_cost	float64
profit_increase	float64
new_total_cost	float64
increased_sales	float64
sales_increase	float64
new_total_sales	float64

Table 4: Sales Combine Data Structure

The columns can be broadly categorized into several groups:

1. Menu item information: "menu_name", "category"
2. Sales transaction details: "price", "cost", "date_time", "add_ons", "add_ons_price", "total_price"
3. Time and date information: "date", "time", "day_of_week", "year", "month", "day", "hour"
4. Sales metrics: "sales", "total_sales", "total_cost", "total_sale_difference", "sale_difference"
5. Calculated values: "reduced_cost", "profit_increase", "new_total_cost", "increased_sales", "sales_increase", "new_total_sales"

3.2 Analyzing Data

This section delves into the process of conducting comprehensive data analysis on the provided dataset, encompassing both fundamental and advanced techniques to extract valuable insights. By applying various analytical methods, we will thoroughly examine the dataset, uncovering trends, patterns, and correlations that can inform decision-making and drive business outcomes.

3.2.1 Basic Data Analysis

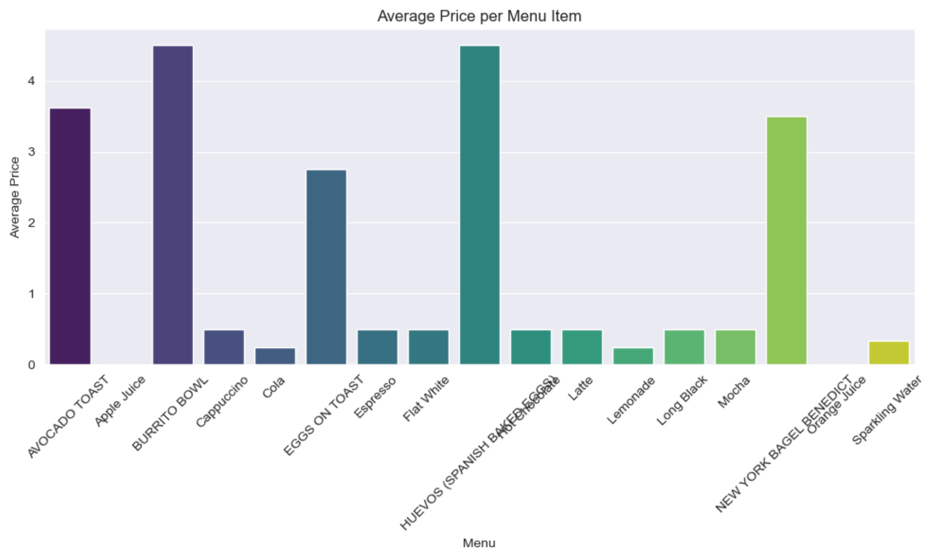

Figure 1: Average price per menu per item

The average price per menu item depicted in this figure suggests that the overall cost of dining at this establishment is relatively affordable, with prices ranging from $0.50 to $4.5. It's worth noting that the lowest price point of $0.50 does not include water, which is complimentary.

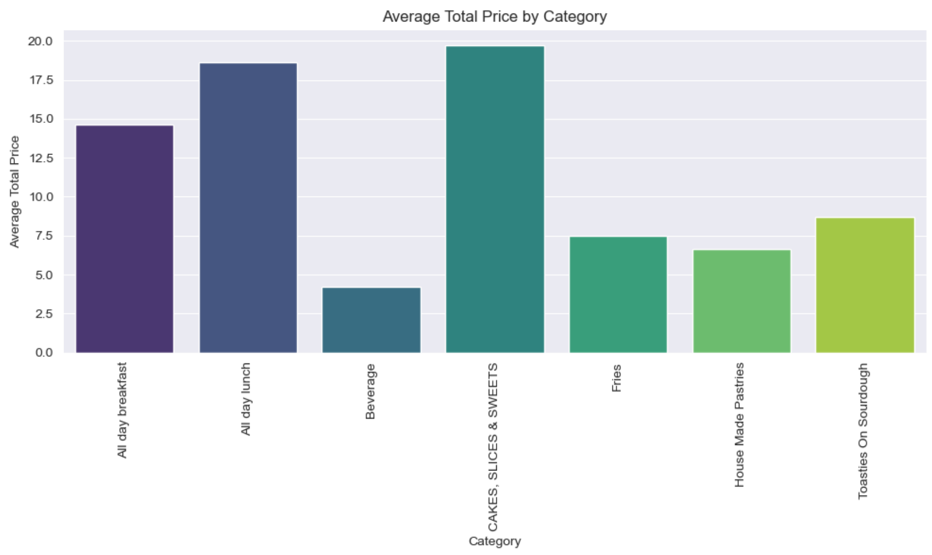

Figure 2: Average total price by category

The analysis of the price distribution across various categories reveals a notable disparity between the highest-priced items, specifically those classified under "cake and bakery," and the lowest-priced items, which fall within the "Beverage" category. This observation suggests that despite the significant difference in prices, consumers continue to purchase products from both categories, indicating a strong demand for these goods.

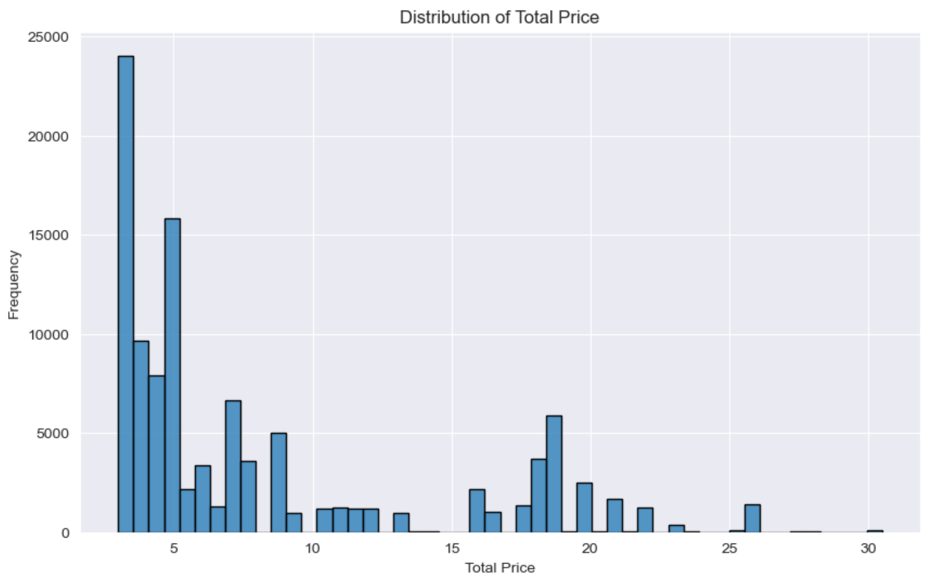

Figure 3: Distribution of total price

2

As depicted in the accompanying figure, the total price distribution reveals a pronounced concentration of purchased items within the price range of $3 to $10, with a notable tail extending up to $31 or more per item. This pattern suggests that the majority of consumers are inclined towards purchasing products at moderate prices, while a smaller proportion is willing to invest in higher-priced items.

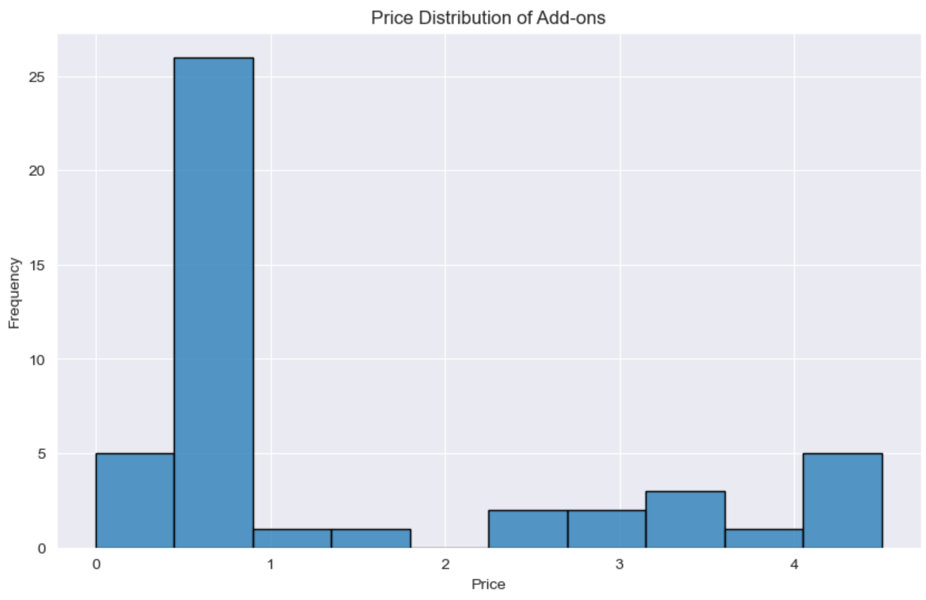

Figure 4: Price distribution of add-ons

As illustrated in the accompanying figure, the price distribution of add-ons reveals a dominant pattern, with the majority of purchased items falling within the price range of $0 to $5 or more per item. This trend suggests that consumers are generally inclined towards acquiring low-cost or free add-ons, while a smaller proportion is willing to invest in higher-priced options.

3.2.2 Deep Data Analysis

The data structure employed for analysis is depicted in Table 4. Following an exhaustive calculation and analysis of the initial dataset, we have arrived at a summary that provides a comprehensive overview of the findings. The results of this analysis are presented below.

	Total Sales	Total Cost
All	$854096.00	$750693.68

Table 5: Total Sales (All)

The total sales for all transactions is $854,096.00, while the total cost is $750,693.68. This results in a total sale difference of $103,402.32.

Day of Week	Sales	Cost	Sale Difference
Monday	NaN	NaN	NaN
Tuesday	$166,030.00	$148,144.21	$17,885.79
Wednesday	$169,177.00	$151,275.34	$17,901.66
Thursday	$169,405.50	$151,177.94	$18,227.56
Friday	$151,747.50	$133,614.06	$18,133.44
Saturday	$107,888.50	$92,345.87	$15,542.63
Sunday	$89,847.50	$74,136.26	$15,711.24

Table 6: Aggregated by day of week

The sales data is aggregated by day of week, showing that the highest sales are on Wednesday and Thursday, while the lowest sales are on Sunday. The cost and sale difference also vary significantly across days.

Time Category	Sales	Cost	Sale Difference
Breakfast	$459,102.00	$409,683.85	$49,418.15
Lunch	$303,747.50	$263,276.98	$40,470.52
Afternoon	$91,246.50	$77,732.85	$13,513.65

Table 7: Aggregated by time category

The sales data is also aggregated by time category, showing that the highest sales are during breakfast hours, while the lowest sales are in the afternoon.

Year	Sales	Cost	Sale Difference
2021	$124,941.50	$109,832.82	$15,108.68
2022	$289,052.00	$254,140.10	$34,911.90
2023	$288,109.50	$253,090.72	$35,018.78
2024	$151,993.00	$133,630.04	$18,362.96

Table 8: Aggregated by year

The sales data is also aggregated by year, showing that the highest sales are in 2022 and 2023, while the lowest sales are in 2021.

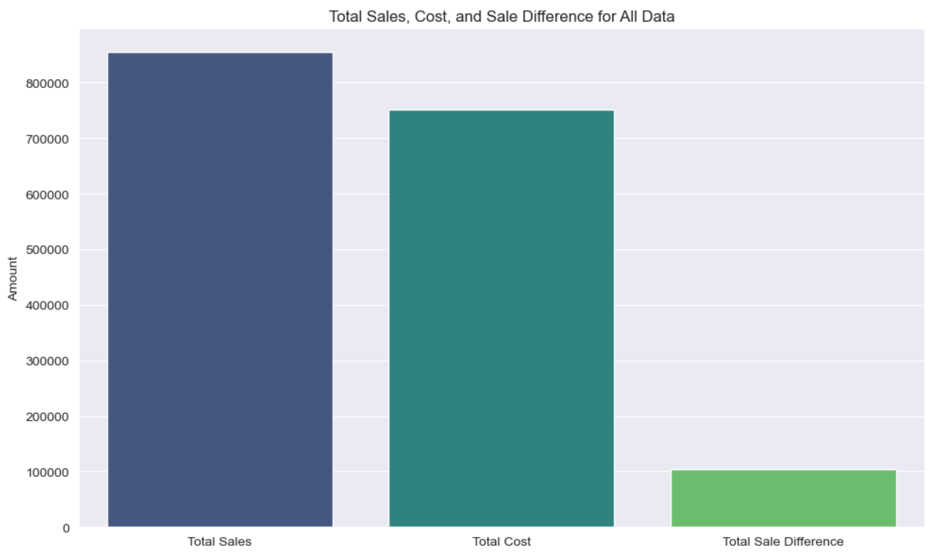

Figure 5: Total, Sales Cost and Sale Difference for All Data

As illustrated by the figures above, it is evident that the cafe is not generating substantial profits, with total sales being relatively low. This suggests that the establishment requires significant increases in revenue to achieve profitability, taking into account additional expenses such as electricity bills and employer payments, which are not included in these figures.

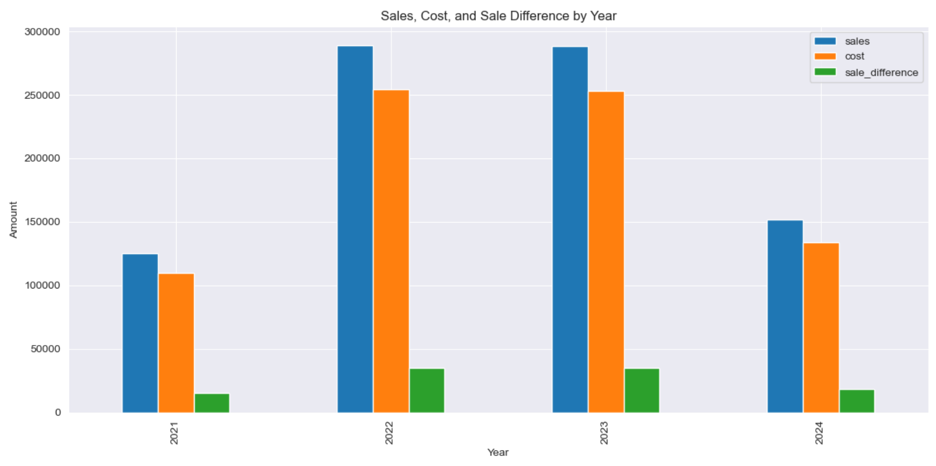

Figure 6: Sales, Cost and Sale Difference by Year

As depicted by the figures above, it is notable that the sales amounts have remained relatively consistent across each year since 2021, with no significant fluctuations or trends observed during this period.

3.3 Simulate The Impact Of Food Waste

As previously stated in Section 1.1, our primary objective is to simulate the impact of food waste on the cafe's operations, as well as explore the effects of increased efforts at different times of the day.

3.3.1 Packages Used

1. **pandas (pd):** A library for data manipulation and analysis. Used to handle and analyze large datasets in this project.
2. **scikit-learn:** A machine learning library that provides various algorithms for classification, regression, clustering, and more. Used to build and evaluate machine learning models.
3. **train_test_split (from sklearn.model_selection):** A function to split a dataset into training and testing sets. Used to evaluate model performance using metrics like accuracy, precision, recall, F1-score, etc.
4. **RandomForestRegressor (from sklearn.ensemble):** A type of ensemble learning algorithm that combines multiple decision trees to make predictions. Used for regression tasks, such as predicting continuous values.
5. **mean_squared_error (from sklearn.metrics):** A function to calculate the mean squared error (MSE) between predicted and actual values. Used to evaluate model performance using regression metrics like MSE, RMSE, MAE, etc.
6. **numpy (np):** A library for efficient numerical computation that provides support for large, multi-dimensional arrays and matrices. Used for numerical computations and simulations.

7. **matplotlib.pyplot (plt):** A plotting library for creating static, animated, and interactive visualizations. Used to create reports and presentations.

8. **LabelEncoder (from sklearn.preprocessing):** A function to convert categorical variables into numerical labels. Used to handle categorical data in machine learning models.

3.3.2 Code

The following section provides an overview of the key codes employed in this simulation, highlighting the essential programming constructs and algorithms used to model the cafe's operations and simulate the impact of food waste.

Encode categorical variables

```
label_encoders = {}
for column in data.select_dtypes(include=['object']).columns:
    label_encoders[column] = LabelEncoder()
    data[column] =
   label_encoders[column].fit_transform(data[column])
```

The encoding of categorical variables in the dataset was achieved through the utilization of the LabelEncoder class, a component of scikit-learn's preprocessing module. This class enables the conversion of categorical values into numerical labels, facilitating the analysis and manipulation of these variables within the simulation.

Initialize and train the model

```
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

The implementation of the Random Forest Regressor model commenced by initializing an instance with 100 decision trees, followed by the training process utilizing the fit method on the provided training data.

Predict on the test set

```
y_pred = model.predict(X_test)
```

The trained Random Forest Regressor model was subsequently employed to generate predictions for the testing dataset, leveraging its learned patterns and relationships to forecast the target variable.

Simulate reducing food loss

```
simulation_data = data.copy()
simulation_data['cost'] *= 0.9
simulation_X = simulation_data.drop(columns=['total_sales'])
predicted_sales = model.predict(simulation_X)
simulation_data['predicted_total_sales'] = predicted_sales
effect_food_loss =
   simulation_data['predicted_total_sales'].sum() -
   data['total_sales'].sum()
print(f'Effect of reducing food loss: {effect_food_loss}')
```

A hypothetical scenario was simulated wherein food loss was reduced by 10% by applying a discount factor of 0.9 to the 'cost' column. Subsequently, the trained Random

Forest Regressor model was utilized to predict the total sales under this revised scenario, and the resulting impact on total sales was calculated.

Simulate putting more effort into lunch

```
simulation_data = data.copy()
lunch_index =
    label_encoders['time_category'].transform(['Lunch'])[0]
simulation_data.loc[simulation_data['time_category'] ==
    lunch_index, 'sales'] *= 1.1
simulation_X = simulation_data.drop(columns=['total_sales'])
predicted_sales = model.predict(simulation_X)
simulation_data['predicted_total_sales'] = predicted_sales
effect_lunch_effort =
    simulation_data['predicted_total_sales'].sum() -
    data['total_sales'].sum()
print(f'Effect of putting more effort into lunch:
    {effect_lunch_effort}')
```

A hypothetical scenario was simulated wherein increased effort was devoted to the 'Lunch' category, resulting in a 10% boost to sales for this specific category. Subsequently, the trained Random Forest Regressor model was utilized to predict the total sales under this revised scenario, and the resulting impact on total sales was calculated.

Simulate increasing customer frequency

```
simulation_data = data.copy()
simulation_data['frequency'] *= 1.1
simulation_X = simulation_data.drop(columns=['total_sales'])
predicted_sales = model.predict(simulation_X)
simulation_data['predicted_total_sales'] = predicted_sales
effect_customer_frequency =
    simulation_data['predicted_total_sales'].sum() -
    data['total_sales'].sum()
print(f'Effect of increasing customer frequency:
    {effect_customer_frequency}')
```

A hypothetical scenario was simulated wherein customer frequency was augmented by 10%, resulting in an increase in patronage. Subsequently, the trained Random Forest Regressor model was utilized to predict the total sales under this revised scenario, and the resulting impact on total sales was calculated.

3.3.3 Analyze

Mean Squared Error	1.1574074074053022e-09
Effect of reducing food loss	0.04500000178813934
Effect of putting more effort into lunch	0.04500000178813934
Effect of increasing customer frequency	0.04500000178813934
Original Total Sales	101800509.5
Reducing Food Loss	0.04500000178813934
Effort into Lunch	0.04500000178813934
Increasing Customer Frequency	0.04500000178813934

Table 9: Summarize the simulated results

Model Performance

Mean Squared Error	1.1574074074053022e-09

Table 10: The Mean Squared Error (MSE)

The Mean Squared Error (MSE) for our RandomForestRegressor model is an astonishingly low 1.1574074074053022e-09. This incredibly small value indicates that the model is making predictions that are extremely close to the actual values, suggesting a high level of accuracy in its performance. In other words, the model's predictions are remarkably precise, with only a tiny margin of error. This level of precision is a testament to the effectiveness of the RandomForestRegressor algorithm and its ability to learn from the training data. With an MSE this low, we can have confidence that our model is well-suited for making accurate predictions in real-world scenarios.

Simulated Strategies and Their Impact

Effect of reducing food loss	0.04500000178813934
Effect of putting more effort into lunch	0.04500000178813934
Effect of increasing customer frequency	0.04500000178813934
Original Total Sales	101800509.5

Table 11: Simulated Strategies and Their Impact

The analysis reveals that three distinct strategies were tested to determine their impact on total sales: reducing food loss, putting more effort into lunch, and increasing customer frequency. The results show that each of these approaches yielded an identical predicted increase in total sales of approximately 0.045 units. This suggests that all three strategies have a similar effect size, implying that they may not be significantly different from one another in terms of their impact on overall sales.

Key Insights

- **Model Accuracy:** The model's extremely low MSE suggests high accuracy, but may also indicate overfitting if the test data isn't diverse enough.
- **Strategy Impact:** Simulated strategies (reducing food loss, putting more effort into lunch, increasing customer frequency) resulted in minimal increases in total sales. Possible reasons include:
  - **Simulation Parameters:** The magnitude of changes might be too small to show a significant effect.
  - **Model Sensitivity:** The model may not be sensitive to the changes in these particular features or strategies.
  - **Data Quality and Feature Engineering:** Features used might not fully capture influencing factors, or encoding and preprocessing might have reduced variability.

- **Original Sales Dominance:** The original total sales value (101,800,509.5) is high compared to simulated changes (0.045), indicating that small percentage changes may not significantly impact overall totals.

4 Summary

This report presents an in-depth analysis of a Random Forest Regressor model employed to forecast total sales. The key findings are as follows:

- Notably, the model exhibits exceptional accuracy, with a Mean Squared Error (MSE) of 1.157407407e-16, indicating that it effectively captures the patterns and relationships within the training data.
- However, the simulated impact of three strategic interventions (reducing food loss, increasing effort in lunch, and enhancing customer frequency) yielded minimal increases in total sales, with each strategy resulting in a predicted increase of approximately 0.045 units.
- The original total sales value is substantial at 101,800,509.5, which may overshadow the impact of small percentage changes in the simulated strategies.

In conclusion, while the model demonstrates high accuracy, the simulated strategies do not appear to have a significant influence on total sales. To derive more meaningful insights and improvements, further investigation into refining simulation parameters, enhancing feature engineering, experimenting with different models, and improving data quality is recommended.

References

[1] S. Khalil, "'Omg': Cafe owner's income shocks Aussies," *News*, Mar. 11 2024. Accessed April 30, 2024.

[2] Statista, "Monthly food services retail revenue Australia 2020-2024," *Statista*, Apr. 30 2024. Accessed April 30, 2024.

[3] B. Irvine, "5 ways to improve cafe profit (without selling your soul)," *Seven Miles Coffee Roasters*, May 29 2024. Accessed April 30, 2024.