

Final Project

[advanced data science](#)

Linguo Ren
May 4th, 2025

Part 0: Github Repo & Data

https://github.com/LINGUOREN369/Adv_ds_meal_swipe

Part 1: Problem Identification

Predicting the waiting lines at the dining halls (Moulton and Thorne), focusing on dinner time. Seating capacity is not considered, as we are primarily concerned with the waiting time to get food

Part 2: Data Planning

Variable Name	Justification	Sample Values	Data Type	Collection Method
Number of Swipes	Dependent variable. Measures dining hall traffic—i.e., how many people swipe in during each interval.	10, 50, 100	Integer	Bowdoin swipe data
Dining Hall	Encodes dining location: Moulton (0) or Thorne (1).	0, 1	Binary	Bowdoin swipe data
Time Interval (15 min blocks from 17:00–20:00)	Tracks time in 15-minute intervals to model traffic patterns across the dinner period.	1, 2, 3, 4...	Integer	Swipe data timestamps
Month of the Year	Seasonality affects dining patterns. For example, in fall, football players may dine together post-practice. Warm weather may drive students to Moulton for outdoor seating.	1–12	Integer	Bowdoin swipe data
Dining Menu Quality	Menu offerings influence turnout. Popular cuisines (e.g., Korean or Indian night) attract more students. Rank menus based on expected appeal.	1–4	Integer	Dining service records
Exam/Reading Period or Weekend	During exams and weekends, students follow less structured routines, often affecting dining behavior.	0, 1	Binary	Academic calendar
Number of Students on Campus	Total student population fluctuates due to study abroad or class size differences. Affects total possible diners.	1600–2000	Integer	Bowdoin enrollment data
Campus Events with Takeout Food (e.g.,	Special events offering alternative meals reduce dining hall attendance.	0, 1	Binary	Student Activities

Variable Name	Justification	Sample Values	Data Type	Collection Method
food truck)				Office
Number of Students on Declining Balance	Students without meal swipes are more likely to rely on dining halls. Useful for modeling dining hall dependence.	0, 1	Binary	Registrar or dining database

Part 3: Model Selection and Justification

I plan to use a Negative Binomial Regression model to predict the number of dining hall swipes. Although I have not yet accessed the data, I anticipate that the outcome variable (count of swipes per 15-minute interval) will show overdispersion, where the variance exceeds the mean. This will happen because the number of swipes is influenced by factors like time of day, day of week, and menu quality. A Negative Binomial model is the Poisson model but introducing a dispersion parameter, making it more flexible for count data with variable rates.

The model aligns with the expected structure of the dataset, which includes a mix of binary, categorical, and continuous predictors such as dining location, meal type, campus events, and academic calendar context. My primary goal is predicting and forecasting attendance patterns to inform students who does not wait in line for dinner and also staffing at the dinning hall.

A key limitation is that the model assumes independence across observations and does not account for temporal dependencies. If time-series patterns emerge in the data, this may not perform as well as desired.

Part 4: Synthetic Data Generation

Below is my prompt. My plan was to work with a large dataset, so I ended up asking for code that could generate data based on logical assumptions. The data looks reasonable overall because I instructed the code to generate it using information it had learned about Bowdoin College. My sample size is also quite large, since it spans an entire year. That said, the model might be more accurate with data from multiple years. However, due to the complications introduced by COVID in recent years, I decided to focus solely on 2024.

Prompt

```
try to be as close to reality as possible for bowdoin college. Give me the code to
generate data for the past 5 years and turn them into csv files. Following the
guideline of the workbook below: | **Variable Name**
**Justification**
|
**Sample Values** | **Data Type** | **Collection Method**
|
| ----- | -----
| -----
| ----- | ----- | -----
-- | ----- |
```

Number of Swipes	**Dependent variable**. Measures dining hall traffic—i.e., how many people swipe in during each interval.
	10, 50, 100 Integer
Bowdoin swipe data	
Dining Hall (0) or Thorne (1).	Encodes dining location: Moulton
	0, 1 Binary
Bowdoin swipe data	
Time Interval (15 min blocks from 17:00–20:00)	Tracks time in 15-minute intervals to model traffic patterns across the dinner period.
	1, 2, 3, 4...
Integer Swipe data timestamps	
Month of the Year	Seasonality affects dining patterns. For example, in fall, football players may dine together post-practice. Warm weather may drive students to Moulton for outdoor seating.
1–12	
Integer Bowdoin swipe data	
Dining Menu Quality	Menu offerings influence turnout. Popular cuisines (e.g., Korean or Indian night) attract more students. Rank menus based on expected appeal.
1–4	
Integer Dining service records	
Exam/Reading Period or Weekend	During exams and weekends, students follow less structured routines, often affecting dining behavior.
	0, 1 Binary
Academic calendar	
Number of Students on Campus	Total student population fluctuates due to study abroad or class size differences. Affects total possible diners.
1600–2000	
Integer Bowdoin enrollment data	
Campus Events with Takeout Food (e.g., food truck)	Special events offering alternative meals reduce dining hall attendance.
	0, 1
Binary Student Activities Office	
Number of Students on Declining Balance	Students without meal swipes are more likely to rely on dining halls. Useful for modeling dining hall dependence.
0, 1	Binary
Registrar or dining database	

Response

```

import pandas as pd
import numpy as np
from datetime import datetime, timedelta

def generate_bowdoin_2024_data():
    np.random.seed(2024) # Fixed seed for reproducibility

    # Bowdoin-specific constants
    BASE_ENROLLMENT = 1881
    MEAL_PLAN_STUDENTS = 1600
    THORNE_CAPACITY = 350
    MOULTON_CAPACITY = 250

    # 2024 Academic Calendar
    fall_start, fall_end = datetime(2024,9,3), datetime(2024,12,20)
    spring_start, spring_end = datetime(2025,1,21), datetime(2025,5,23)

    def generate_semester_dates(start, end):
        dates = []
        current = start
        while current <= end:
            if current.weekday() < 5: # Mon-Fri only
                dates.append(current)
                current += timedelta(days=1)
        return dates

    dates = generate_semester_dates(fall_start, fall_end) +
generate_semester_dates(spring_start, spring_end)

    data = []

    for date in dates:
        # Enrollment adjustments
        enrollment = BASE_ENROLLMENT - 170 if date.month in [1,2,9,10] else
BASE_ENROLLMENT
        month = date.month # Extract month

        # Campus events
        is_thursday = date.weekday() == 3
        has_event = is_thursday and (np.random.random() < 0.3)

        # Exam periods
        is_exam = ((date.month == 12 and date.day >= 7) or
                    (date.month == 5 and date.day >= 10))

        # Menu quality (Bowdoin's typical distribution)

```

```

menu_quality = np.random.choice([1,2,3,4], p=[0.1,0.25,0.5,0.15])

# Generate dinner swipes
for interval in range(1, 13):
    time_str = (datetime.min + timedelta(hours=17, minutes=15*(interval-
1))).strftime('%H:%M')

    # Base traffic patterns
    if interval in [5,6,7]: # Peak dinner
        base_thorne = np.random.normal(240, 20)
        base_moulton = np.random.normal(200, 15)
    else:
        base_thorne = np.random.normal(130, 15)
        base_moulton = np.random.normal(100, 10)

    # Apply modifiers
    menu_factor = 1 + (menu_quality-1)*0.12
    event_factor = 0.65 if has_event else 1.0
    exam_factor = 0.75 if is_exam else 1.0

    thorne_swipes = min(THORNE_CAPACITY,
                        int(base_thorne * menu_factor * event_factor *
exam_factor * np.random.uniform(0.95,1.05)))
    moulton_swipes = min(MOULTON_CAPACITY,
                        int(base_moulton * menu_factor * (0.85 if has_event
else 1.0) * exam_factor * np.random.uniform(0.95,1.05)))

# Append data with month column
data.append([
    date.strftime('%Y-%m-%d'),
    time_str,
    interval,
    thorne_swipes,
    1, # Thorne
    month, # Month as separate column
    menu_quality,
    int(is_exam),
    enrollment,
    int(has_event),
    MEAL_PLAN_STUDENTS
])
data.append([
    date.strftime('%Y-%m-%d'),
    time_str,
    interval,
    moulton_swipes,

```

```

        0, # Moulton
        month,
        menu_quality,
        int(is_exam),
        enrollment,
        int(has_event),
        MEAL_PLAN_STUDENTS
    ])

# Create DataFrame with Month column
df = pd.DataFrame(data, columns=[
    "Date",
    "Time",
    "Time Interval",
    "Number of Swipes",
    "Dining Hall",
    "Month of the Year", # New month column
    "Dining Menu Quality",
    "Exam/Reading Period or Weekend",
    "Number of Students on Campus",
    "Campus Events with Takeout Food",
    "Number of Students on Declining Balance"
])

return df.sort_values(['Date', 'Time Interval']).reset_index(drop=True)

# Generate and save data
bowdoin_2024 = generate_bowdoin_2024_data()
bowdoin_2024.to_csv("bowdoin_dining_swipes_2024_with_month.csv", index=False)
print(f"Generated {len(bowdoin_2024)} records")
print("Month distribution:")
print(bowdoin_2024['Month of the Year'].value_counts().sort_index())

```

Part 5: Analysis Implementation

The data cleaning was essentially already taken care of, as the data generation code was well-constructed. I ran some exploratory data analysis (EDA) and didn't observe any major issues. I chose to run two models: the first was a Negative Binomial model, and the second was an OLS regression model. I included OLS as a benchmark for comparison, since it's a commonly used baseline and helps highlight the strengths of the Negative Binomial approach.

Metric	Negative Binomial	OLS Linear Regression
Mean Absolute Error (MAE)	43.59	43.73 (slightly worse)
RMSE	53.70	53.83 (slightly worse)

- Both models agree on key significant predictors: DiningHall, MenuQuality, ExamPeriod, TakeoutEvent, DecliningBalance.
- The Negative Binomial model is more appropriate for this count-based dataset because it captures the relationship between predictors and swipes more reliably because there is also over dispersion in our dependent variable number of Swipes: Mean: 159.70486111111111, Variance: 3714.9667042228284.
- However, the model is not doing really well because of the limitation to take care of non-linear data such as the meal data.
- The data is also not the real world data, and the way how I generate the data may also introduce bias that makes the model inaccurate.

Negative Binomial GLM Results:

Generalized Linear Model Regression Results

Dep. Variable:	Swipes	No. Observations:	3225
Model:	GLM	Df Residuals:	3217
Model Family:	NegativeBinomial	Df Model:	7
Link Function:	Log	Scale:	1.0000
Method:	IRLS	Log-Likelihood:	-19537.
Date:	Thu, 01 May 2025	Deviance:	313.95
Time:	22:47:02	Pearson chi2:	362.
No. Iterations:	7	Pseudo R-squ. (CS):	0.03262
Covariance Type:	nonrobust		

	coef	std err	z	P> z	[0.025	0.975]
Intercept	1.848e-06	1.61e-07	11.510	0.000	1.53e-06	2.16e-06
TimeInterval	-0.0096	0.005	-1.870	0.061	-0.020	0.000
DiningHall	0.2191	0.035	6.200	0.000	0.150	0.288
Month	8.19e-05	0.005	0.017	0.986	-0.009	0.009
MenuQuality	0.0949	0.022	4.381	0.000	0.052	0.137
ExamPeriod	-0.2830	0.059	-4.831	0.000	-0.398	-0.168
StudentsOnCampus	4.558e-05	0.000	0.205	0.837	-0.000	0.000
TakeoutEvent	-0.2865	0.065	-4.414	0.000	-0.414	-0.159
DecliningBalance	0.0030	0.000	11.510	0.000	0.002	0.003

Evaluation on Test Data:

Mean Absolute Error (MAE): 43.59

Root Mean Squared Error (RMSE): 53.70

McFadden's Pseudo R²: 0.0027

Linear Regression Results:

OLS Regression Results

```

=====
Dep. Variable:          Swipes    R-squared:                0.214
Model:                  OLS       Adj. R-squared:           0.212
Method:                 Least Squares   F-statistic:             125.2
Date:                  Thu, 01 May 2025   Prob (F-statistic):      3.09e-163
Time:                  22:47:45    Log-Likelihood:          -17443.
No. Observations:      3225       AIC:                     3.490e+04
Df Residuals:          3217       BIC:                     3.495e+04
Df Model:              7
Covariance Type:       nonrobust
=====

```

```

=====
              coef      std err          t      P>|t|      [0.025      0.975]
-----
Intercept    4.325e-05   8.66e-06     4.994     0.000    2.63e-05    6.02e-05
TimeInterval -1.2786        0.276    -4.627     0.000    -1.820    -0.737
DiningHall   35.9035        1.906    18.841     0.000    32.167    39.640
Month        -0.0591        0.258    -0.229     0.819    -0.565     0.447
MenuQuality  14.6096         1.168    12.505     0.000    12.319    16.900
ExamPeriod   -39.5296         3.158   -12.519     0.000   -45.721   -33.339
StudentsOnCampus  0.0050         0.012     0.416     0.677    -0.019     0.028
TakeoutEvent -42.0101         3.497   -12.012     0.000   -48.868   -35.153
DecliningBalance  0.0692         0.014     4.994     0.000     0.042     0.096
=====

```

```

=====
Omnibus:          444.471   Durbin-Watson:           2.011
Prob(Omnibus):    0.000   Jarque-Bera (JB):        654.148
Skew:             1.102   Prob(JB):                8.99e-143
Kurtosis:         2.900   Cond. No.                1.17e+18
=====

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The smallest eigenvalue is 1.38e-26. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.

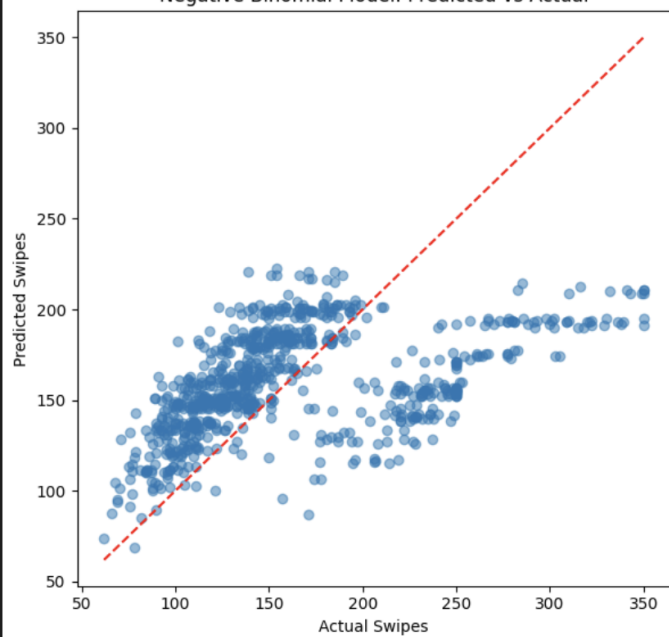
Evaluation on Test Data (Linear Regression):

Mean Absolute Error (MAE): 43.73

Root Mean Squared Error (RMSE): 53.83

McFadden's Pseudo R² (Linear Regression): 0.0218

Negative Binomial Model: Predicted vs Actual



Linear Regression Model: Predicted vs Actual

