Automated Technician Task Assignment System - Dataset List and Analysis Report

Sue Byeon
October 19, 2025
MSDS 498 Capstone

Dataset List and Initial Analysis

Dataset List

1. combined_dataset
    a. Source: Kaggle - Technician Scheduling Dataset
    b. Description: This dataset contains task assignments and completion records for field technicians. Each row represents a unique task assigned to a technician, capturing key attributes related to both the task and the technician's suitability for it.
2. enhanced_combined_dataset
    a. Source: Kaggle - Technician Scheduling Dataset
    b. Description: This dataset captures detailed information about completed task assignments and their outcomes, focusing on operational logistics, task requirements, and customer satisfaction. Each row represents a single task, including attributes that describe the context of the assignment, resource needs, and the results as perceived by the customer.

Initial Analysis

- Small Sample Size
    - The *combined_dataset* contains only 50 rows, while the *enhanced_dataset* contains 200 rows. The limited number of rows in the training dataset poses a challenge for building robust predictive models. With such a small training sample, the model will have fewer patterns to learn from, increasing the risk of overfitting and reducing the overall accuracy and generalizability of the results. This limitation should be considered when evaluating model performance and selecting modeling techniques.

- Lack of Contextual Data
    - Both datasets focus mainly on task and technician-related attributes but lack additional contextual or operational variables (e.g., time of day, location type, technician skill level, customer demographics, or task category). The absence of this information may limit the depth of insights that can be drawn from the analysis and could also restrict the model's ability to learn meaningful relationships that exist in real-world scenarios.
- Column Inconsistencies Between Datasets
    - There are naming discrepancies and structural differences between the two datasets, including missing columns and limited overlap. Only two columns are shared between the datasets, which makes it challenging to join them for integrated analysis or to perform direct comparisons. These inconsistencies require additional preprocessing steps, such as standardizing column names,

creating composite keys, or engineering new linking variables to ensure proper data alignment and integrity prior to analysis.

- No Missing Data
    - Both datasets are complete, with no missing values detected. This simplifies the data cleaning process and removes the need for imputation strategies, allowing the focus to shift toward feature engineering and exploratory analysis.
- Data Quality and Format
    - The datasets are very clean, with most columns being numeric. This minimizes the need for extensive data standardization or transformation. The numeric nature of the data facilitates straightforward application of descriptive statistics, correlation analysis, and machine learning algorithms without requiring significant preprocessing.

Data Management and Transformation/Engineering/Architecture Document

Dataset Inventory

1. *combined_dataset*
    a. Purpose: Serves as the historical record of tasks that have been assigned to technicians. This dataset will be primarily used for training models, as it contains outcomes (e.g., task completion) that can inform predictive modeling.
    b. Key Variables:
        i. *TechnicanID* - Unique identifier for each technician.
        ii. *TaskID* – Unique identifier for each task.
        iii. *Expertise Match* – Indicates whether the task was assigned to a technician with matching skill or complexity level (1 = Yes, 0 = No).
        iv. *Task Priority* - Priority level of the task on a 1–5 scale, where a higher value indicates greater urgency.
        v. *Task Completed* - Indicates whether the task was successfully completed (1 = Yes, 0 = No).
        vi. *Distance to Task* - Distance (in kilometers) from the technician's location to the task's location.
2. *enhanced_combined_dataset*
    a. Purpose: Represents incoming task requests that technicians must be assigned to. This dataset is intended for testing and evaluation, as it reflects future or operational data without historical completion outcomes.
    b. Key Variables:
        i. *Distance to Task (km)* - Distance (in kilometers) from the technician's location to the task's location.
        ii. *Priority* - Task priority on a scale of 1 to 5, where a higher value indicates greater urgency.
        iii. *Task Complexity* - Complexity rating of the task on a scale of 1 to 10.
        iv. *Equipment Required* - Binary indicator showing whether equipment is needed for the task (1 = equipment required, 0 = no equipment required).
        v. *Customer Rating* - Customer satisfaction rating on a scale of 1 to 5.
        vi. *Penalty for Delay ($)* - Monetary penalty based on task priority, which increases with higher priority tasks. (Will refer as *Penalty Cost* for the rest of the paper.)

Data Processing

1. Remove Unnecessary Columns
2. Standardize Column Names

3. One-Hot Encode *TechnicianID*
    a. Apply one-hot encoding to the *TechnicianID* variable to enable the model to learn technician-specific patterns from historical data
4. Calculate *Penalty Cost*
    a. Create *Penalty Cost* column in *combined_dataset* using formula defined in *enhanced_combined_dataset* to reflect the actual penalty incurred for uncompleted tasks.
5. Build Technician Profile Dataset
    a. Derive a separate technician dataset from *combined_dataset* to capture technician characteristics based on past performance.
        i. New Derived Columns:
            1. *Complexity Level* – Numeric variable derived from patterns in Expertise Match to reflect the maximum task complexity a technician can handle reliably.

            2. *Eqpt Trained* – Binary variable derived from Expertise Match to indicate whether the technician has experience with tasks requiring equipment.

Data Usage Plan

- Two datasets will be merged by *Task Priority* and *Distance to Task (km)* to align historical performance with incoming task requests.
- Prediction model will evaluate task variables like *Task Priority* and *Distance to Task* and skills variables like *Complexity Level* and *Eqpt Trained* to predict the likelihood that a technician will complete the task successfully.
- To calculate success metrics:
    - Total Penalty Cost
        - If task is completed, then *Penalty for Delay ($)* accrued = 0.
        - Otherwise, the penalty is based on the priority level of the task.
    - Customer Rating
        - If task is completed, then rating increases by 1 but no greater than 5.
    - Expertise Match Rate
        - If *Task Complexity* & *Equipment Required* are aligned with tech's skills, then *Expertise Match* = 1; otherwise, it is 0.
    - Task Completion Rate
        - *Task Completed* is the target variable of the algorithm. The overall completion rate will be measured as the mean of the predicted *Task Completed* values across all test records.

Technical Architecture for Deployment Considerations

1. Initial Exploration Stage
    - Local CSV storage, Python-based EDA, visualization in VS Code.
2. Data Preparation & Model Development
    - PostgreSQL for cleaned data, XGBoost for modeling, Dockerized deployment on AWS, FastAPI backend, Postman for API testing.
3. Model Serving & Inference
    - Real-time prediction through FastAPI REST endpoints; potential for batch processing. Auto-scaling for demand.
4. Model Monitoring & Logging
    - Prometheus/Grafana for metrics, logging requests and outputs, monitoring drift and latency.
5. Model Versioning & CI/CD
    - Git for code versioning, automated pipelines for testing, building, deploying, and rollback strategies.

In [1]:
```
# %pip install nbconvert
```

Insights and Trends Analysis Report

The objective of this project is to improve operational efficiency by increasing task completion rates and expertise match rates, while reducing total penalty costs and enhancing customer satisfaction. This section examines the current state and interrelationships among these success metrics to determine whether their observed patterns align with the projected benefits for the company.

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
```

In [3]:
```
train = pd.read_csv('combined_dataset.csv',sep=',')
train.info()

test = pd.read_csv('enhanced_combined_dataset.csv',sep=',')
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 7 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   Technician ID          50 non-null     object
 1   Task ID                50 non-null     object
 2   Expertise Match        50 non-null     int64 
 3   Task Priority          50 non-null     int64 
 4   Task Duration          50 non-null     int64 
 5   Distance to Task (km)  50 non-null     int64 
 6   Task Completed         50 non-null     int64 
dtypes: int64(5), object(2)
memory usage: 2.9+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 9 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Distance to Task (km)  200 non-null    int64  
 1   Priority               200 non-null    int64  
 2   Task Complexity        200 non-null    int64  
 3   Max Working Hours      200 non-null    int64  
 4   Travel Time (minutes)  200 non-null    float64
 5   Overtime Cost ($)      200 non-null    float64
 6   Equipment Required     200 non-null    int64  
 7   Customer Rating        200 non-null    int64  
 8   Penalty for Delay ($)  200 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 14.2 KB
```

Expertise Match vs Task Completion

In [4]:
```
pd.crosstab(train['Expertise Match'], train['Task Completed'], margins=True)
```

Out[4]:

Task Completed	0	1	All
Expertise Match			
0	9	12	21
1	12	17	29
All	21	29	50

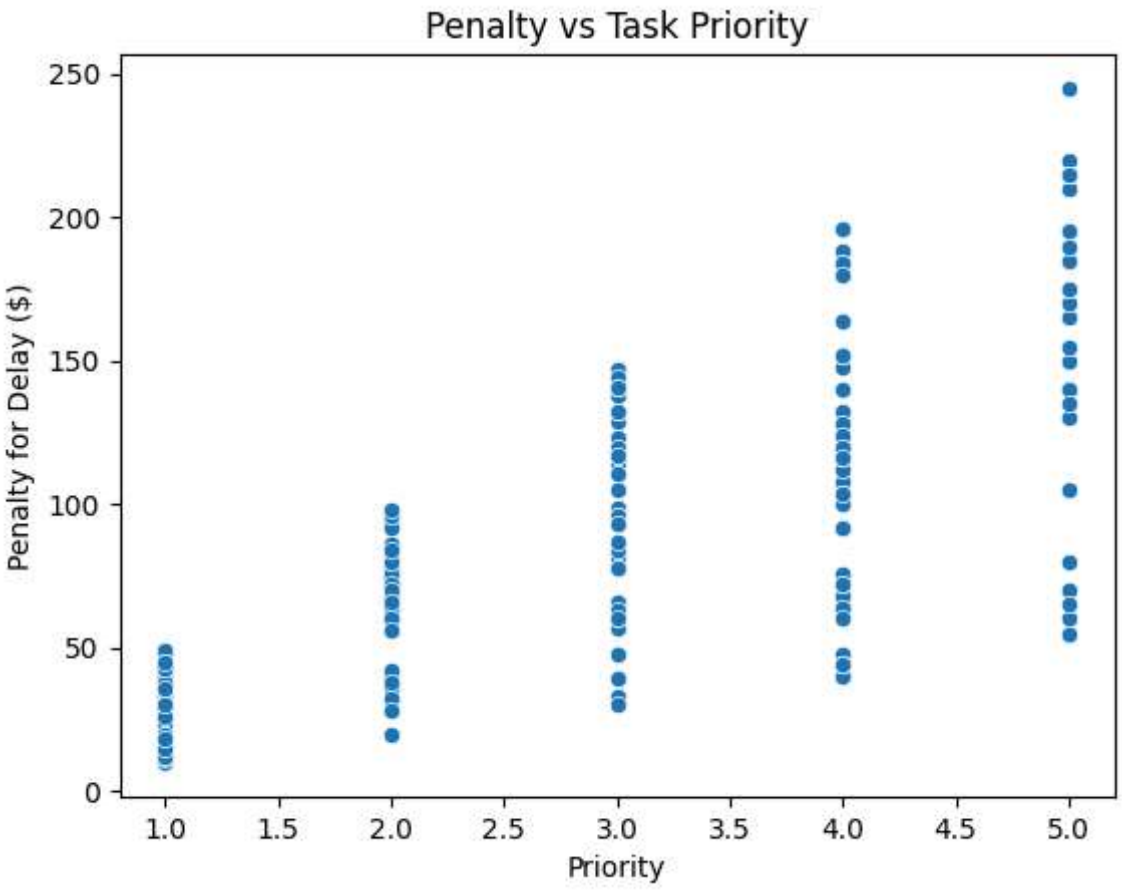
This table shows that when expertise is matched, a higher number of tasks are completed successfully, supporting the hypothesis that aligning technician expertise with task requirements increases the likelihood of successful task completion. Interestingly, among the incomplete tasks, there are still more instances where expertise was matched than not. This may suggest that technicians are sometimes assigned tasks that are either too easy or too complex relative to their actual capabilities. Due to the lack of contextual data, it is difficult to draw definitive conclusions. Nonetheless, the overall trend indicates that expertise

matching remains a positive factor, even if its impact alone is not strongly significant in this dataset.

Expertise Match vs Penalty Cost

In this analysis, the goal is to demonstrate that when assigned tasks are not matched to technician expertise, the resulting penalty costs are higher. Since Penalty Cost is not originally included in the same dataset as Expertise Match, it was necessary to derive a penalty cost variable by identifying its calculation pattern from the second dataset. According to the data source, Penalty Cost is determined by Task Priority, so a formula based on priority levels was applied to the primary dataset to create a corresponding Penalty Cost column. This allowed for a direct comparison between expertise matching and associated penalty costs within a single dataset.

In [5]:
```
plot = sns.scatterplot(
    data=test,
    x='Priority',
    y='Penalty for Delay ($)'
)
t = plt.title('Penalty vs Task Priority')
plt.show()
```



This graph shows each Priority level are in set ranges of cost.

In [6]:
```
test.groupby('Priority')['Penalty for Delay ($)'].describe()
```

Out[6]:

Priority	count	mean	std	min	25%	50%	75%	max
1	51.0	30.921569	11.629004	10.0	19.50	33.0	42.00	49.0
2	36.0	64.888889	20.951342	20.0	59.00	68.0	78.50	98.0
3	42.0	94.714286	35.706996	30.0	63.75	96.0	123.00	147.0
4	43.0	110.976744	47.610267	40.0	70.00	108.0	150.00	196.0
5	28.0	150.000000	55.226805	55.0	123.75	152.5	191.25	245.0

I looked for the min and max for each Priority to create a formula that randomly picks a number between those ranges and apply the formula to the combined_dataset to recreate the Penalty Cost column.

In [7]:
```
penalty_ranges = {
    1: (10, 50),
    2: (20, 100),
    3: (30, 150),
    4: (40, 200),
    5: (50, 250)
}

def calculate_penalty(priority):
    low, high = penalty_ranges[priority]
    return np.random.randint(low, high)

train['Penalty Cost'] = train['Task Priority'].apply(calculate_penalty)

train.head()
```

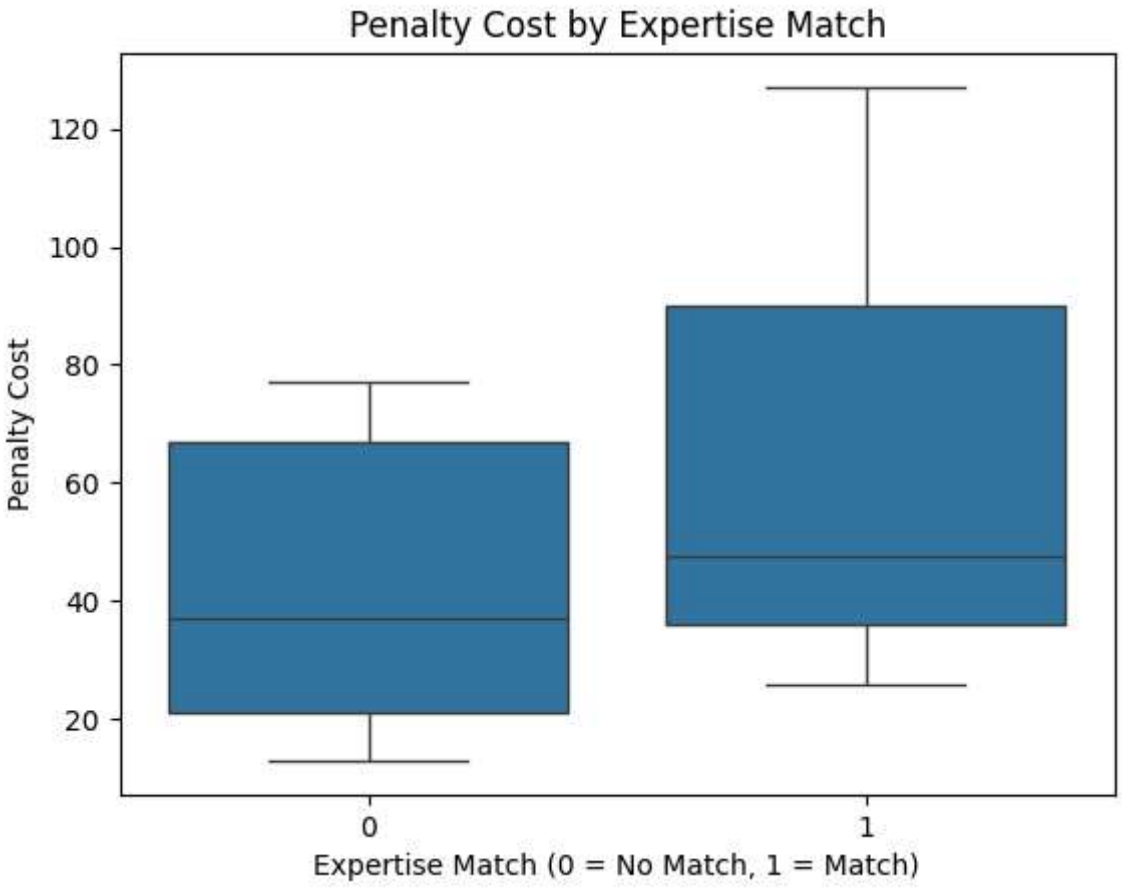
Out[7]:

	Technician ID	Task ID	Expertise Match	Task Priority	Task Duration	Distance to Task (km)	Task Completed	Penalty Cost
0	T005	J001	0	2	3	1	1	23
1	T006	J002	1	1	2	17	0	26
2	T005	J003	0	2	3	19	1	71
3	T009	J004	1	3	3	15	1	116
4	T003	J005	1	3	2	14	1	125

Of the incomplete tasks, I wanted to see the distribution of the penalty amount by Expertise Match.

In [8]:
```
incomplete_tasks = train[train['Task Completed'] == 0]


plot = sns.boxplot(
    data=incomplete_tasks,
    x='Expertise Match',
    y='Penalty Cost'
)
t = plt.title('Penalty Cost by Expertise Match')
x = plt.xlabel('Expertise Match (0 = No Match, 1 = Match)')
y = plt.ylabel('Penalty Cost')
plt.show()
```



The graph shows that higher penalty costs are associated with tasks matched by expertise, which reflects the system's practice of assigning high-priority tasks to technicians with matching expertise to reduce operational risk. Because penalty cost is directly driven by task priority, expertise-matched tasks naturally exhibit higher averages and wider spreads in penalty costs.

This indicates that penalty cost and expertise match capture different success dimensions. Expertise Match measures the quality of task assignment, while Penalty Cost measures operational efficiency in task execution.

While some correlation may exist, the two metrics are not directly dependent. A more meaningful analysis would be to compare penalty costs and completion rates within the

same priority levels to isolate the impact of expertise matching.

Expertise Match Rate by Priority and Penalty Cost Distribution by Task Priority & Expertise Match

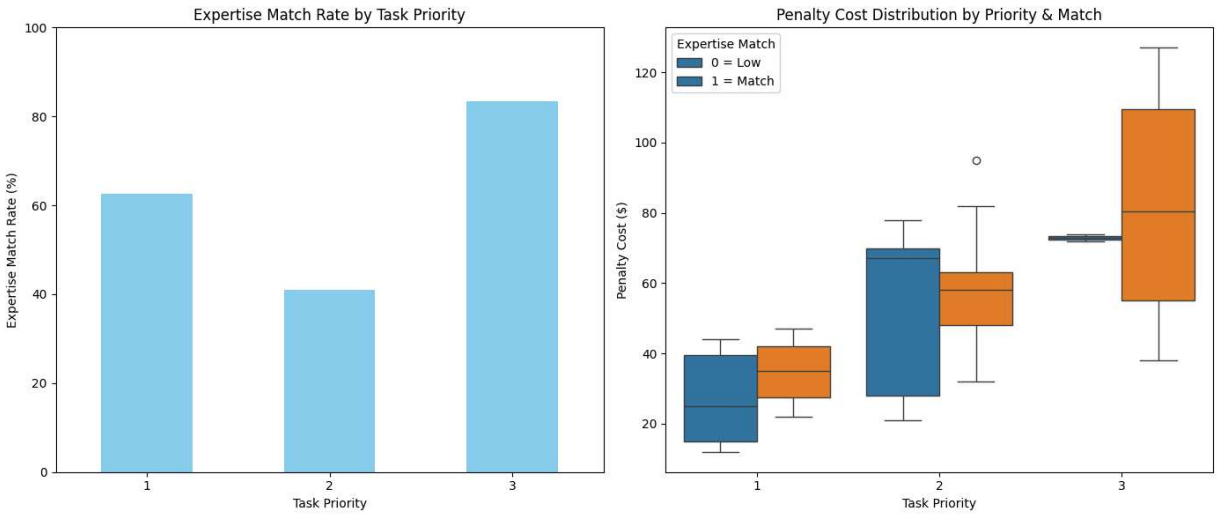
In [9]:
```
match_rate_by_priority = (
    train
    .groupby('Task Priority')['Expertise Match']
    .mean() * 100
)

figure = plt.figure(figsize=(14, 6))

# ---------- Panel A: Expertise Match Rate ----------
subplot = plt.subplot(1, 2, 1)
plot = match_rate_by_priority.plot(kind='bar', color='skyblue')
t = plt.title('Expertise Match Rate by Task Priority')
x = plt.xlabel('Task Priority')
y = plt.ylabel('Expertise Match Rate (%)')
ylim = plt.ylim(0, 100)
xticks = plt.xticks(rotation=0)

# ---------- Panel B: Penalty Cost Distribution ----------
subplot = plt.subplot(1, 2, 2)
plot = sns.boxplot(
    data=train,
    x='Task Priority',
    y='Penalty Cost',
    hue='Expertise Match'
)
t = plt.title('Penalty Cost Distribution by Priority & Match')
x = plt.xlabel('Task Priority')
y = plt.ylabel('Penalty Cost ($)')
l = plt.legend(title='Expertise Match', labels=['0 = Low', '1 = Match'])

plt.tight_layout()
plt.show()
```



Priority 1 tasks are likely more routine, allowing unmatched technicians to handle them effectively, though expertise matching still provides a slight advantage. In contrast, Priority 2–3 tasks are more complex, so the company tends to assign them to matched technicians by default.

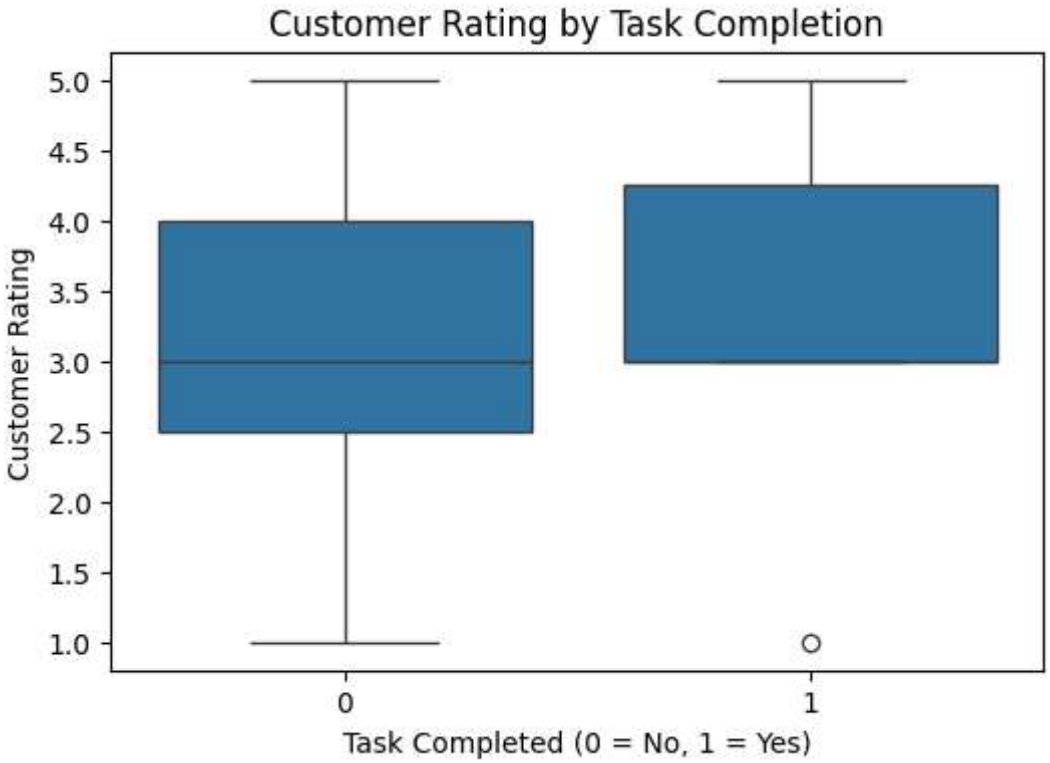
For lower-priority tasks (Priority 1), expertise match appears to lower penalty costs, indicating a positive impact on operational efficiency. However, for mid-priority tasks (Priority 2–3), penalty costs are higher for matched technicians, likely because these tasks are inherently more complex or costly, not because of technician mismatch.

This pattern reinforces that expertise match and penalty cost measure different dimensions of success—expertise match reflects the quality of task assignment, while penalty cost reflects task risk and execution efficiency.

Customer Rating vs Task Completed

In [10]:
```
merged_df = pd.merge(
    train[['Task Priority', 'Distance to Task (km)', 'Task Completed']],
    test[['Priority', 'Distance to Task (km)',  'Customer Rating']],
    left_on=['Task Priority', 'Distance to Task (km)'],
    right_on=['Priority', 'Distance to Task (km)'],
    how='inner'
)

figure = plt.figure(figsize=(6,4))
plot = sns.boxplot(
    data=merged_df,
    x='Task Completed',
    y='Customer Rating'
)
t = plt.title('Customer Rating by Task Completion')
x = plt.xlabel('Task Completed (0 = No, 1 = Yes)')
y = plt.ylabel('Customer Rating')
plt.show()
```



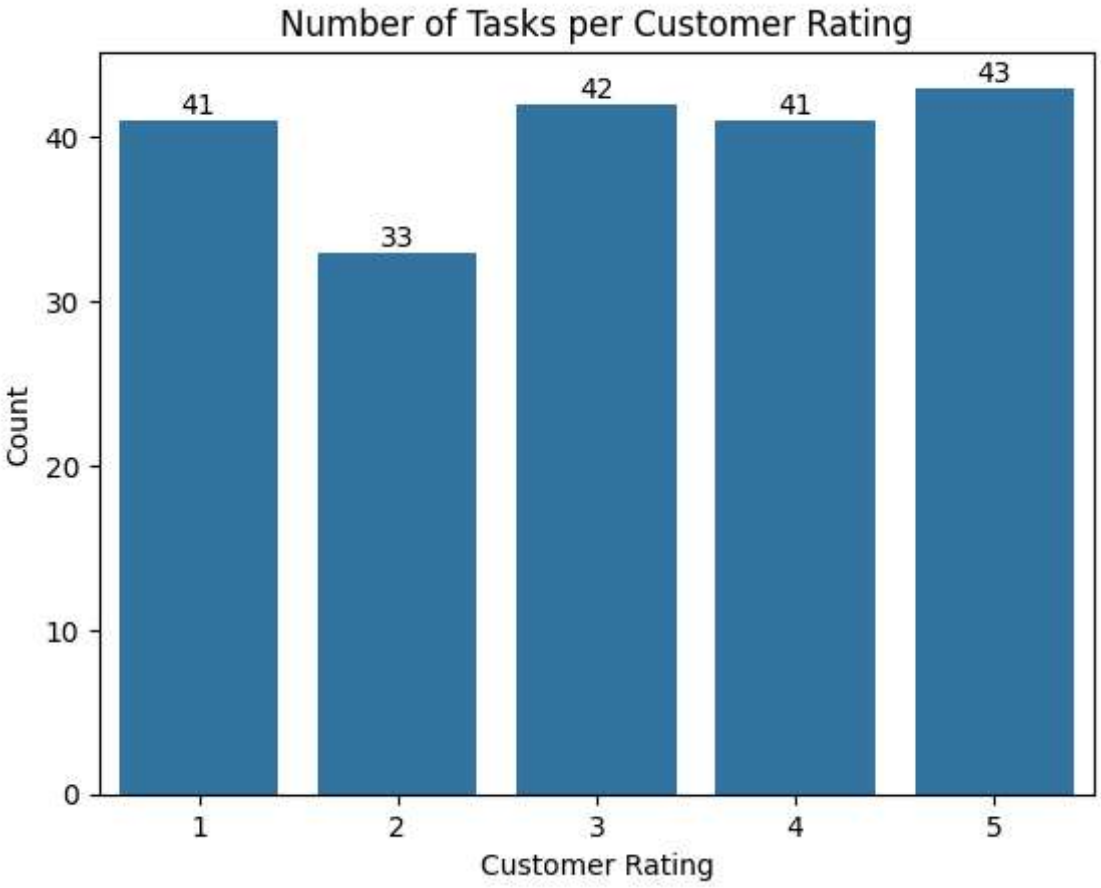
The spread of customer ratings is higher for completed tasks compared to those not completed, indicating that customers have more positive experiences when tasks are successfully completed. This suggests that the company should prioritize improving task completion rates, as it directly contributes to higher customer satisfaction.

Number of Tasks in Customer Rating

In [11]:
```
ax = sns.countplot(data=test, x='Customer Rating')
t = plt.title('Number of Tasks per Customer Rating')
x = plt.xlabel('Customer Rating')
y = plt.ylabel('Count')

for p in ax.patches:
    an = ax.annotate(f'{int(p.get_height())}',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='bottom')

plt.show()
```



The distribution of Customer Ratings is relatively even across all levels, indicating that there is significant room for improvement in customer satisfaction. Specifically, the company has the opportunity to convert lower ratings (1s and 2s) into mid-to-high ratings (3 and above) through improved task execution and service quality.