Vitale_Ana500

October 13, 2025

1 ANA 500 Week 1: Ddata Organization & Analysis

1.0.1 Vincent Vitale

1.0.2 Instructor: Professor Ghaznavy

1.0.3 Date: October 5, 2025

1.0.4 Problem Statement

Airline companies aim to enhance customer satisfaction and reduce delays. Using passenger and operational flight data, this project will identify which flight-related and service-related factors most influence overall passenger satisfaction and on-time performance.

H (Satisfaction): Flight- and service-related variables have no effect on passenger satisfaction.

H : At least one flight or service variable (e.g., class, type of travel, seat comfort) has a significant effect on satisfaction.

H (On-time performance): Operational characteristics in this dataset (e.g., flight distance) have no effect on departure delay.

H : At least one operational characteristic (e.g., distance groups) shows a systematic difference in average departure delay.

[1]:
```
# Import packages and data
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.widgets import Slider, RadioButtons, CheckButtons


# Airline data:
AIRLINE_DATA = r"C:\Users\vince\Desktop\School\ANA500\airline.csv"
```

2 Organize Data

[2]:
```
# View data first to ensure its loaded correctly:
airlines = pd.read_csv(AIRLINE_DATA)

airlines.head()
```

[2]:

	Unnamed: 0	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	…	Inflight entertainment	On-board service	Leg room service	Baggage handling	Checkin service	Inflight service	Cleanliness	Departure Delay in Minutes	Arrival Delay in Minutes	satisfaction
0	0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	…	5	4	3	4	4	5	5	25	18.0	neutral or dissatisfied
1	1	5047	Male	disloyal Customer	25	Business travel	Business	235	3	2	…	1	1	5	3	1	4	1	1	6.0	neutral or dissatisfied
2	2	110028	Female	Loyal Customer	26	Business travel	Business	1142	2	2	…	5	4	3	4	4	4	5	0	0.0	satisfied
3	3	24026	Female	Loyal Customer	25	Business travel	Business	562	2	5	…	2	2	5	3	1	4	2	11	9.0	neutral or dissatisfied
4	4	119299	Male	Loyal Customer	61	Business travel	Business	214	3	3	…	3	3	4	4	3	3	3	0		

```
4                         0.0                 satisfied

[5 rows x 25 columns]
```

[3]:
```
# Inspect the structure of the data
print("Shape:", airlines.shape)
airlines.info()
```

```
Shape: (129880, 25)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129880 entries, 0 to 129879
Data columns (total 25 columns):
 #   Column                            Non-Null Count   Dtype
---  ------                            --------------   -----
 0   Unnamed: 0                        129880 non-null  int64
 1   id                                129880 non-null  int64
 2   Gender                            129880 non-null  object
 3   Customer Type                     129880 non-null  object
 4   Age                               129880 non-null  int64
 5   Type of Travel                    129880 non-null  object
 6   Class                             129880 non-null  object
 7   Flight Distance                   129880 non-null  int64
 8   Inflight wifi service             129880 non-null  int64
 9   Departure/Arrival time convenient 129880 non-null  int64
 10  Ease of Online booking            129880 non-null  int64
 11  Gate location                     129880 non-null  int64
 12  Food and drink                    129880 non-null  int64
 13  Online boarding                   129880 non-null  int64
 14  Seat comfort                      129880 non-null  int64
 15  Inflight entertainment            129880 non-null  int64
 16  On-board service                  129880 non-null  int64
 17  Leg room service                  129880 non-null  int64
 18  Baggage handling                  129880 non-null  int64
 19  Checkin service                   129880 non-null  int64
 20  Inflight service                  129880 non-null  int64
 21  Cleanliness                       129880 non-null  int64
 22  Departure Delay in Minutes        129880 non-null  int64
 23  Arrival Delay in Minutes          129487 non-null  float64
 24  satisfaction                      129880 non-null  object
dtypes: float64(1), int64(19), object(5)
memory usage: 24.8+ MB
```

[4]:
```
# First thing I want to do is change all the string value types to either␣
 ↪binary or categorical
# Gender: Male = 1, Female = 0
airlines['Gender'] = airlines['Gender'].map({'Male': 1, 'Female': 0})
```

```
# Customer Type: Loyal = 1, Disloyal = 0
airlines['Customer Type'] = airlines['Customer Type'].map({
    'Loyal Customer': 1,
    'disloyal Customer': 0
})

# Type of Travel: Business = 1, Personal = 0
airlines['Type of Travel'] = airlines['Type of Travel'].map({
    'Business travel': 1,
    'Personal Travel': 0
})

# Class: Business = 2, Eco Plus = 1, Eco = 0 (or adjust as you prefer)
airlines['Class'] = airlines['Class'].map({
    'Business': 2,
    'Eco Plus': 1,
    'Eco': 0
})

# Satisfaction: satisfied = 1, neutral or dissatisfied = 0
airlines['satisfaction'] = airlines['satisfaction'].map({
    'satisfied': 1,
    'neutral or dissatisfied': 0
})

distance_col = 'Flight Distance_Adjusted' if 'Flight Distance_Adjusted' in␣
 ↪airlines.columns else 'Flight Distance'

# Validate the changes:
airlines[['Gender', 'Customer Type', 'Type of Travel', 'Class',␣
 ↪'satisfaction']].head()
```

[4]:

	Gender	Customer Type	Type of Travel	Class	satisfaction
0	1	1	0	1	0
1	1	0	1	2	0
2	0	1	1	2	1
3	0	1	1	2	0
4	1	1	1	2	1

[5]:
```
# Values look good, lets check for missing data now
airlines.isnull().sum()
```

[5]:
```
Unnamed: 0                         0
id                                 0
Gender                             0
Customer Type                      0
Age                                0
```

```
Type of Travel                          0
Class                                   0
Flight Distance                         0
Inflight wifi service                   0
Departure/Arrival time convenient       0
Ease of Online booking                  0
Gate location                           0
Food and drink                          0
Online boarding                         0
Seat comfort                            0
Inflight entertainment                  0
On-board service                        0
Leg room service                        0
Baggage handling                        0
Checkin service                         0
Inflight service                        0
Cleanliness                             0
Departure Delay in Minutes              0
Arrival Delay in Minutes              393
satisfaction                            0
dtype: int64
```

[6]:
```
# Since only 1 column has null values, need to find out why its null and if we need this data or not.
# Compare missing vs non-missing delay values
missing_arrival = airlines[airlines['Arrival Delay in Minutes'].isnull()]
non_missing_arrival = airlines[airlines['Arrival Delay in Minutes'].notnull()]

print("Rows with missing Arrival Delay:", len(missing_arrival))
print("Rows without missing Arrival Delay:", len(non_missing_arrival))

# Average departure delay in both groups
print("Avg Departure Delay (missing):", missing_arrival['Departure Delay in Minutes'].mean())
print("Avg Departure Delay (non-missing):", non_missing_arrival['Departure Delay in Minutes'].mean())
```

```
Rows with missing Arrival Delay: 393
Rows without missing Arrival Delay: 129487
Avg Departure Delay (missing): 37.88549618320611
Avg Departure Delay (non-missing): 14.643385050236704
```

[7]:
```
# Should check if there is a relationship between the missing values, and on time flights
airlines[airlines['Arrival Delay in Minutes'].isnull()][['Departure Delay in Minutes']].value_counts()
```

[7]:
```
Departure Delay in Minutes
0                             147
4                              11
1                              11
2                              10
16                              6
                               …
116                             1
118                             1
119                             1
121                             1
530                             1
Name: count, Length: 121, dtype: int64
```

[8]:
```
# Visualize the missing data too
airlines[['Departure Delay in Minutes', 'Arrival Delay in Minutes']].describe()
```

[8]:

	Departure Delay in Minutes	Arrival Delay in Minutes
count	129880.000000	129487.000000
mean	14.713713	15.091129
std	38.071126	38.465650
min	0.000000	0.000000
25%	0.000000	0.000000
50%	0.000000	0.000000
75%	12.000000	13.000000
max	1592.000000	1584.000000

[9]:
```
# Since the missing data is a small amount of the data, and its very closely␣
 ↪related to on time departures im making
# the call to input the missing data with 0 values
airlines['Arrival Delay in Minutes'] = airlines['Arrival Delay in Minutes'].
 ↪fillna(0)

# Verify all null values have been replaced
airlines['Arrival Delay in Minutes'].isnull().sum()
```

[9]:
```
0
```

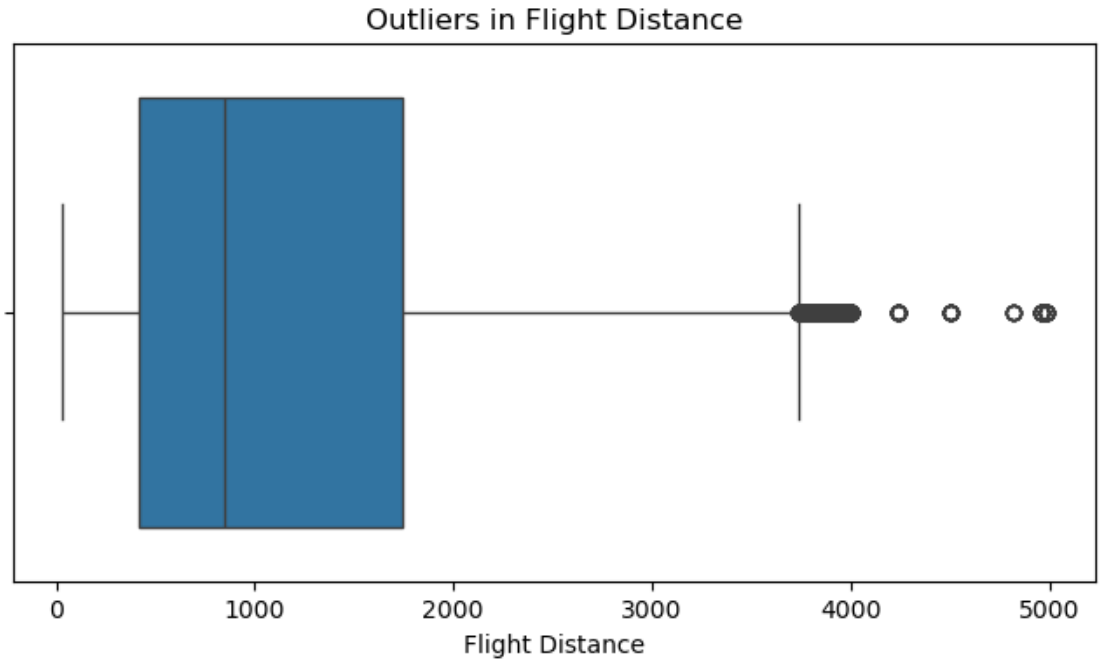
2.0.1 Detecting Outliers on Flight Distance

[10]:
```
q1, q3 = airlines["Flight Distance"].quantile([0.25, 0.75])
iqr = q3 - q1
low, high = q1 - 1.5*iqr, q3 + 1.5*iqr
airlines['Flight Distance_Capped'] = airlines["Flight Distance"].
 ↪clip(lower=low, upper=high)

airlines[["Flight Distance", "Flight Distance_Capped"]].describe()
```

[10]:

	Flight Distance	Flight Distance_Capped
count	129880.000000	129880.000000
mean	1190.316392	1186.995681
std	997.452477	988.394696
min	31.000000	31.000000
25%	414.000000	414.000000
50%	844.000000	844.000000
75%	1744.000000	1744.000000
max	4983.000000	3739.000000

[11]:
```
plt.figure(figsize=(8,4))
sns.boxplot(x=airlines["Flight Distance"])
plt.title("Outliers in Flight Distance")
plt.show()
```

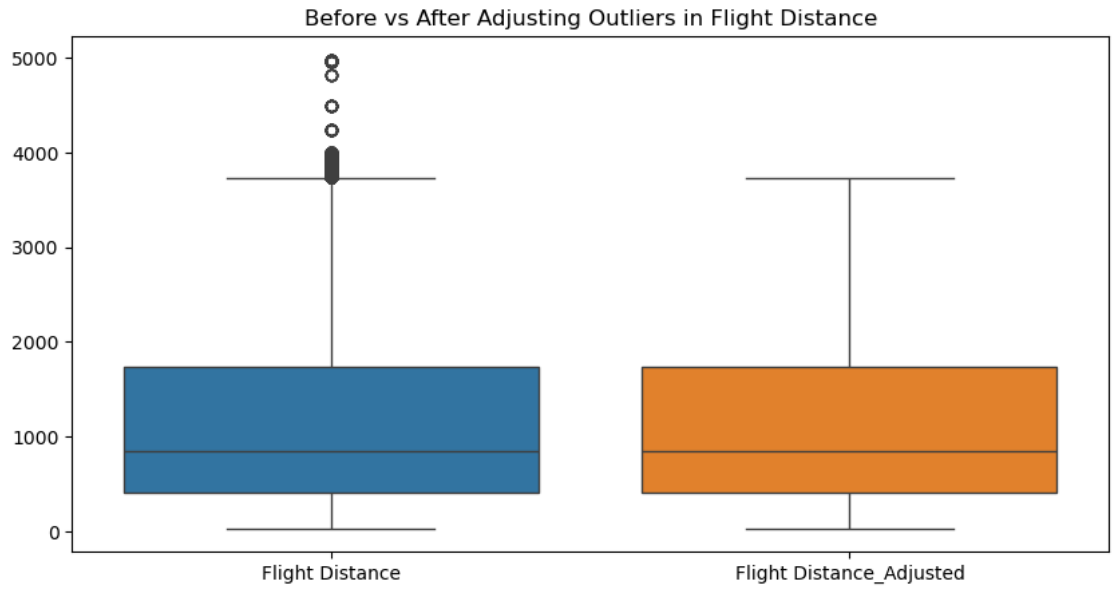


[12]:
```
# I do have some outlier data, I have opted to not remove the data and instead␣
 ↪just reassign it to the highest value of q3 of the iqr.
airlines["Flight Distance_Adjusted"] = np.where(
    airlines["Flight Distance"] > high,
    high,
    airlines["Flight Distance"]
)

airlines[["Flight Distance", "Flight Distance_Adjusted"]].describe()
```

[12]:

	Flight Distance	Flight Distance_Adjusted
count	129880.000000	129880.000000
mean	1190.316392	1186.995681
std	997.452477	988.394696
min	31.000000	31.000000
25%	414.000000	414.000000
50%	844.000000	844.000000
75%	1744.000000	1744.000000
max	4983.000000	3739.000000

[13]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=airlines[["Flight Distance", "Flight Distance_Adjusted"]])
plt.title("Before vs After Adjusting Outliers in Flight Distance")
plt.show()
```



3 Analyzing Data

3.0.1 Questions to answer now that our data is cleaned and organized:

3.0.2 1. Average Satisfaction by Class

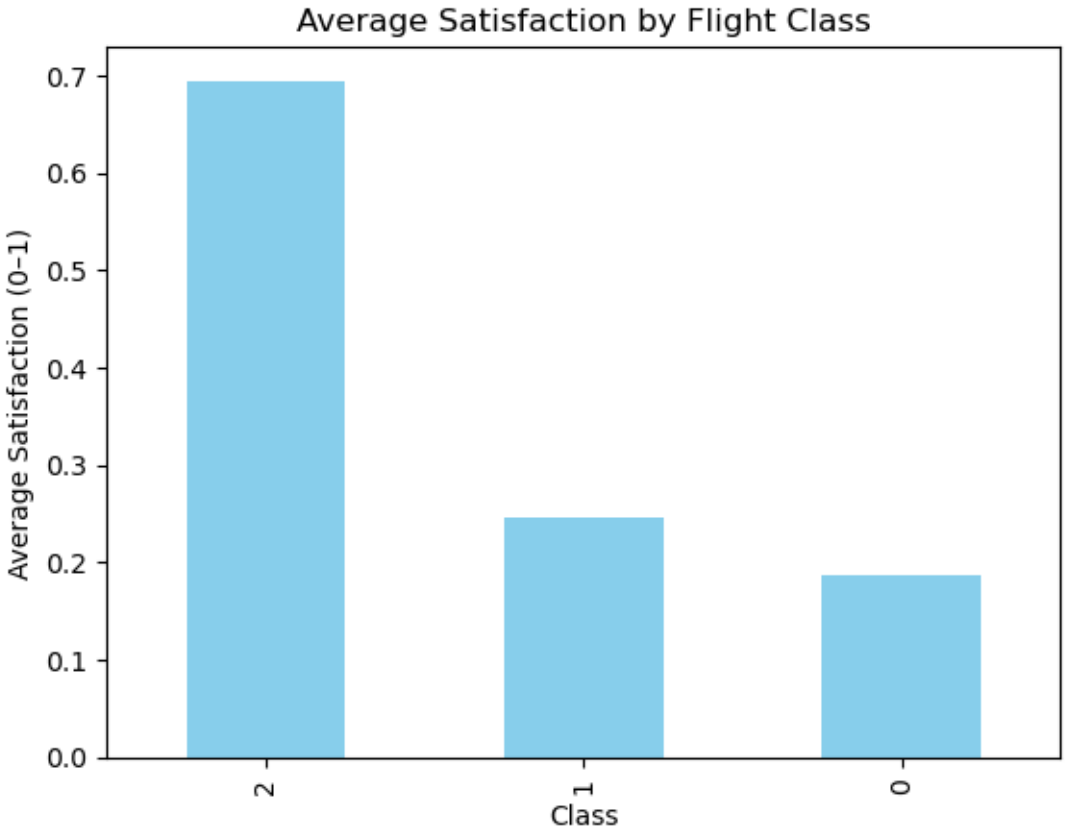
3.0.3 2. Average Flight Distance by Type of Travel

3.0.4 3. Satisfaction by Customer Type

3.0.5 4. Pivot Table (Class vs Type of Travel)

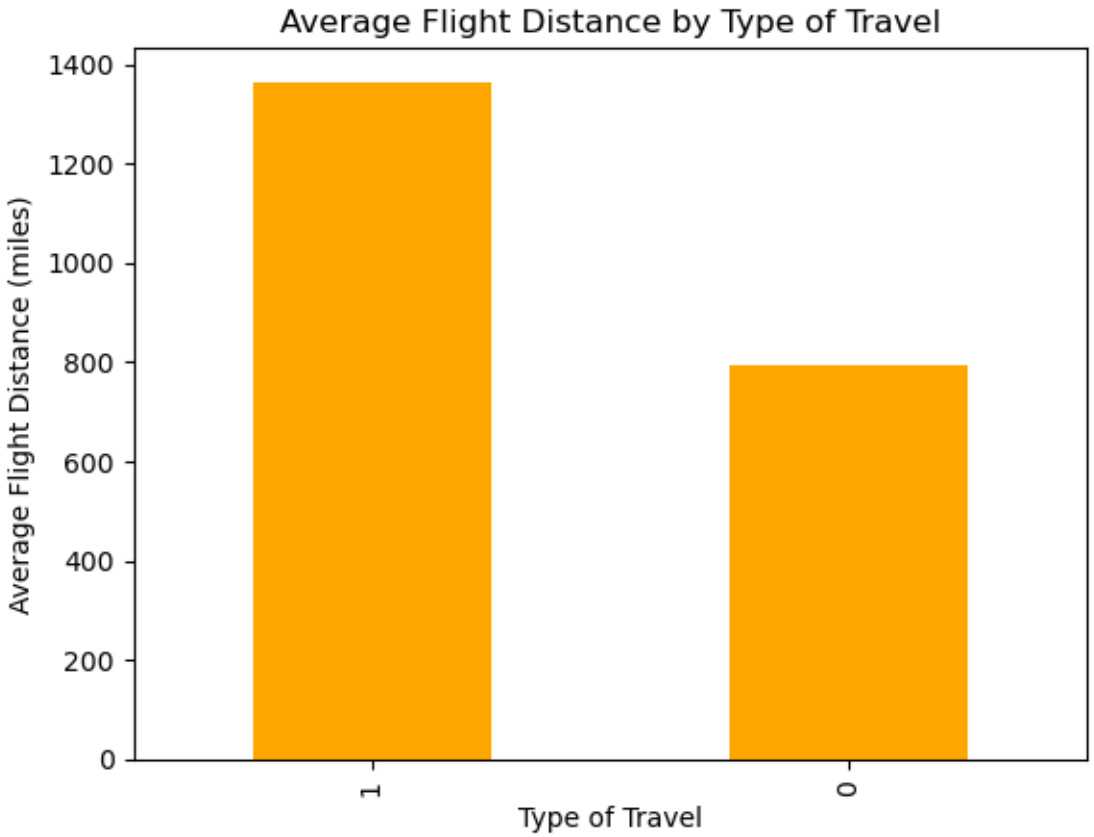
3.0.6 5. Aggregate Average Departure Delay by Distance Range

[14]:
```
# Average satisfaction score by class
satisfaction_by_class = airlines.groupby("Class")["satisfaction"].mean().
 ↪sort_values(ascending=False)
satisfaction_by_class

satisfaction_by_class.plot(kind="bar", title="Average Satisfaction by Flight␣
 ↪Class", color="skyblue")
plt.xlabel("Class")
plt.ylabel("Average Satisfaction (0-1)")
plt.show()
```



Avg Satisfaction by Class: Business class shows the highest average satisfaction, indicating service tier is a strong driver of perceived quality.

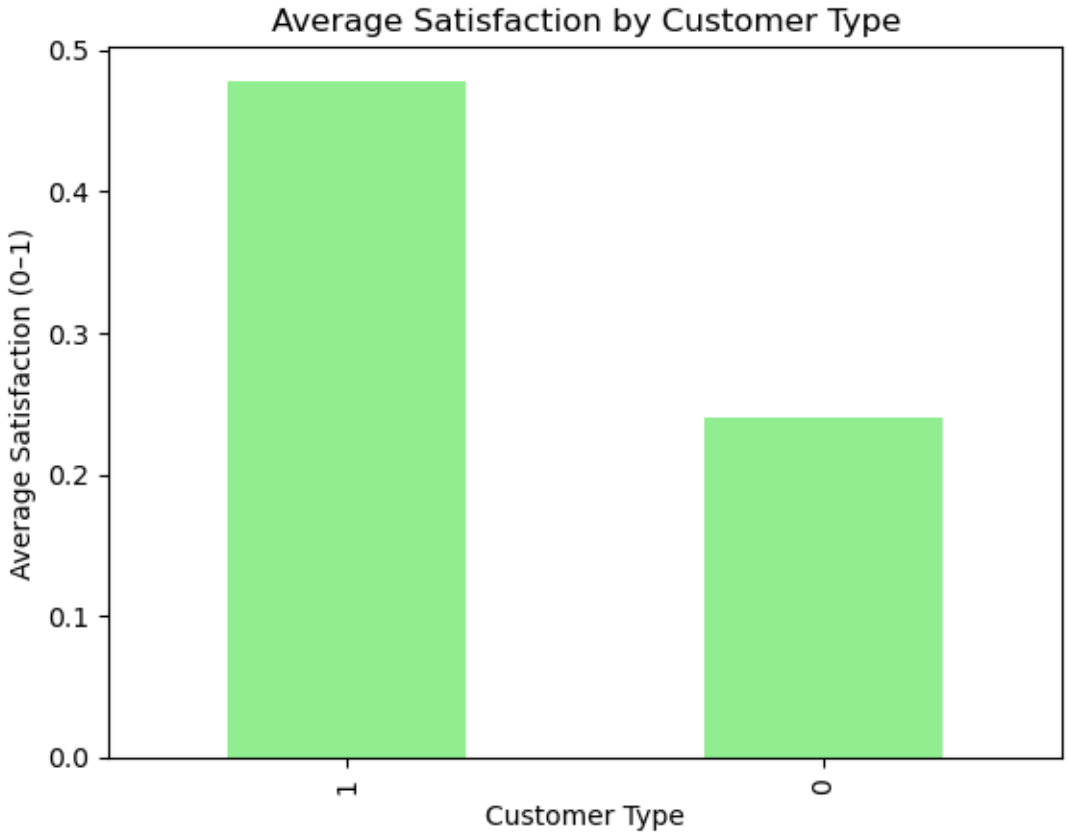
[15]:
```
# Average Flight Distance by Type of Travel
distance_by_travel = airlines.groupby("Type of Travel")["Flight␣
 ↪Distance_Adjusted"].mean().sort_values(ascending=False)
distance_by_travel

distance_by_travel.plot(kind="bar", title="Average Flight Distance by Type of␣
 ↪Travel", color="orange")
plt.xlabel("Type of Travel")
plt.ylabel("Average Flight Distance (miles)")
plt.show()
```



Avg Distance by Type of Travel: Business travel tends to cover longer routes, which may correlate with different expectations and satisfaction profiles.

[16]:
```
# Satisfaction by Customer Type
```

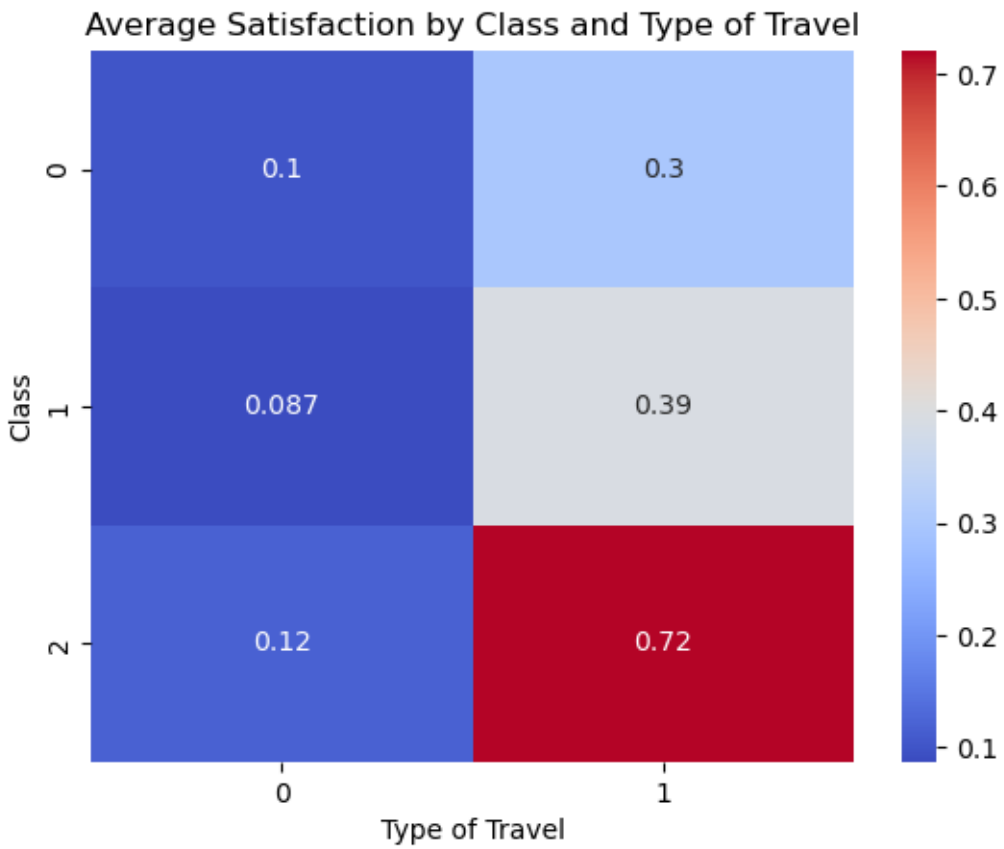
```
satisfaction_by_customer = airlines.groupby("Customer Type")["satisfaction"].
  ↪mean().sort_values(ascending=False)
satisfaction_by_customer.plot(kind="bar", title="Average Satisfaction by␣
  ↪Customer Type", color="lightgreen")

plt.xlabel("Customer Type")
plt.ylabel("Average Satisfaction (0-1)")
plt.show()
```



Satisfaction by Customer Type: Loyal customers score higher on average, consistent with selection/experience effects.

[17]:
```
# Pivot Table (Class vs Type of Travel)
pivot_table = airlines.pivot_table(
    values="satisfaction",
    index="Class",
    columns="Type of Travel",
    aggfunc="mean"
)
pivot_table
```

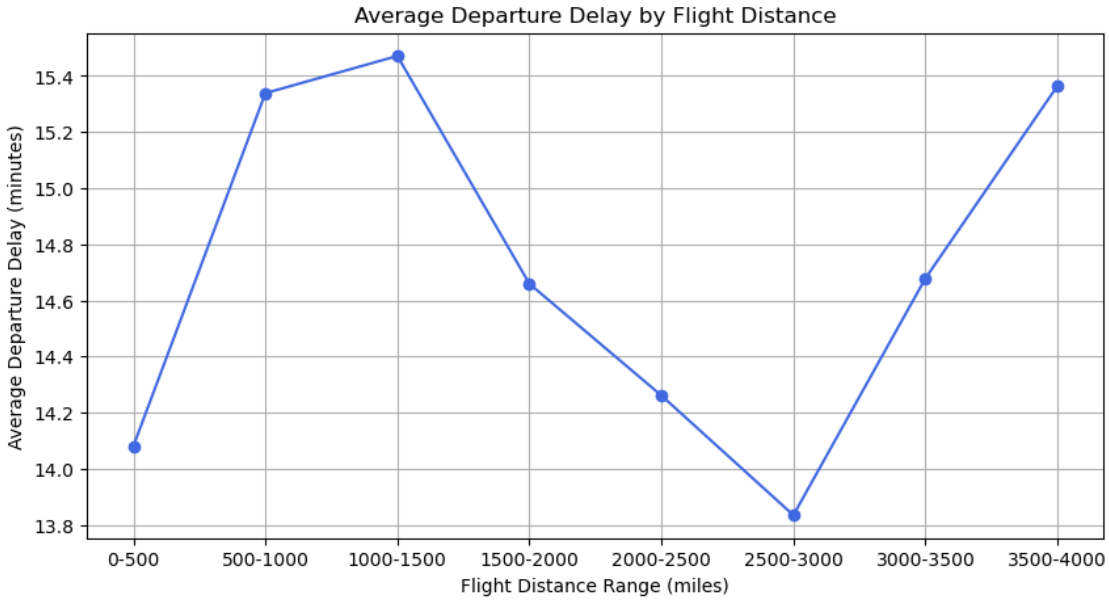
```
sns.heatmap(pivot_table, annot=True, cmap="coolwarm")
plt.title("Average Satisfaction by Class and Type of Travel")
plt.show()
```



Satisfaction is strongest where Business class intersects Business travel, highlighting joint effects of service tier and trip purpose.

[18]:
```
# Aggregate Average Departure Delay by Distance Range
# Create distance bins
bins = [0, 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000]
labels = ['0-500', '500-1000', '1000-1500', '1500-2000',
          '2000-2500', '2500-3000', '3000-3500', '3500-4000']

airlines['DistanceGroup'] = pd.cut(
    airlines['Flight Distance_Adjusted'],
    bins=bins, labels=labels, include_lowest=True
)

# Calculate average departure delay per distance group
```

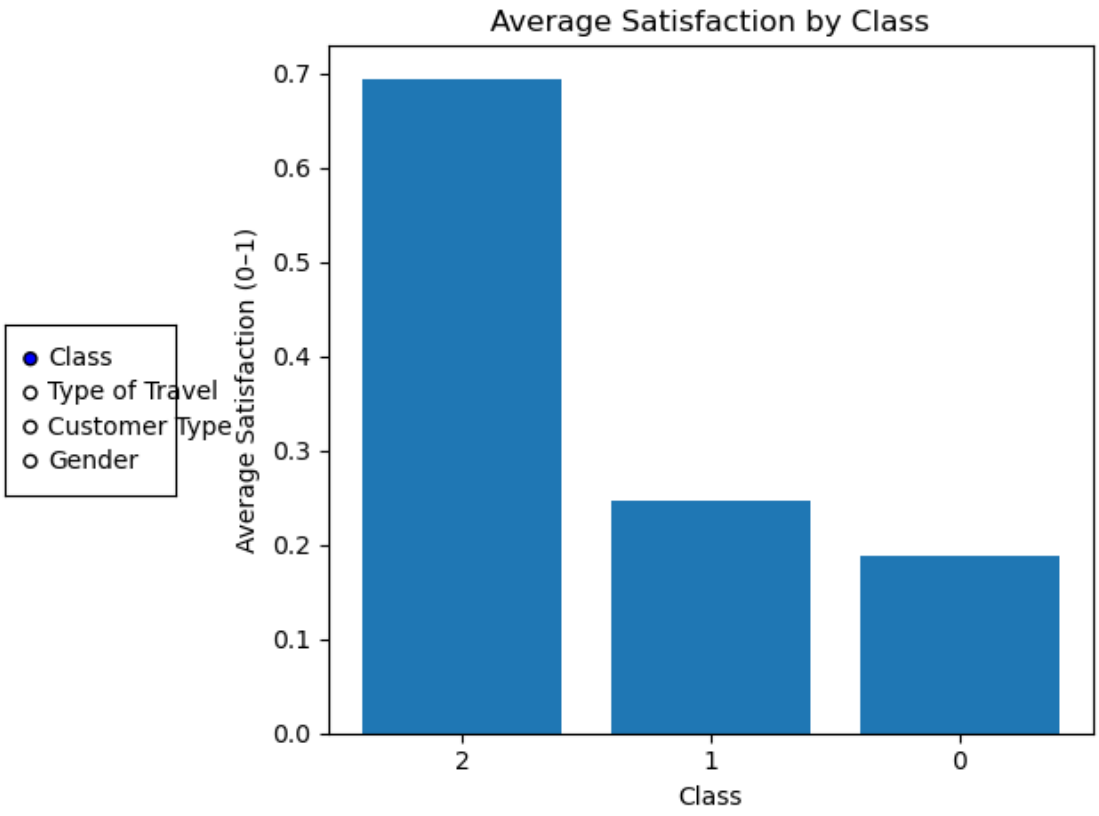
```
avg_delay_by_distance = airlines.groupby("DistanceGroup",␣
 ↪observed=True)["Departure Delay in Minutes"].mean()
avg_delay_by_distance

plt.figure(figsize=(10,5))
avg_delay_by_distance.plot(kind='line', marker='o', color='royalblue')
plt.title("Average Departure Delay by Flight Distance")
plt.xlabel("Flight Distance Range (miles)")
plt.ylabel("Average Departure Delay (minutes)")
plt.grid(True)
plt.show()
```



Delays vary across distance bins (non-monotonic); mid-range bins appear elevated, suggesting operational constraints by route length.

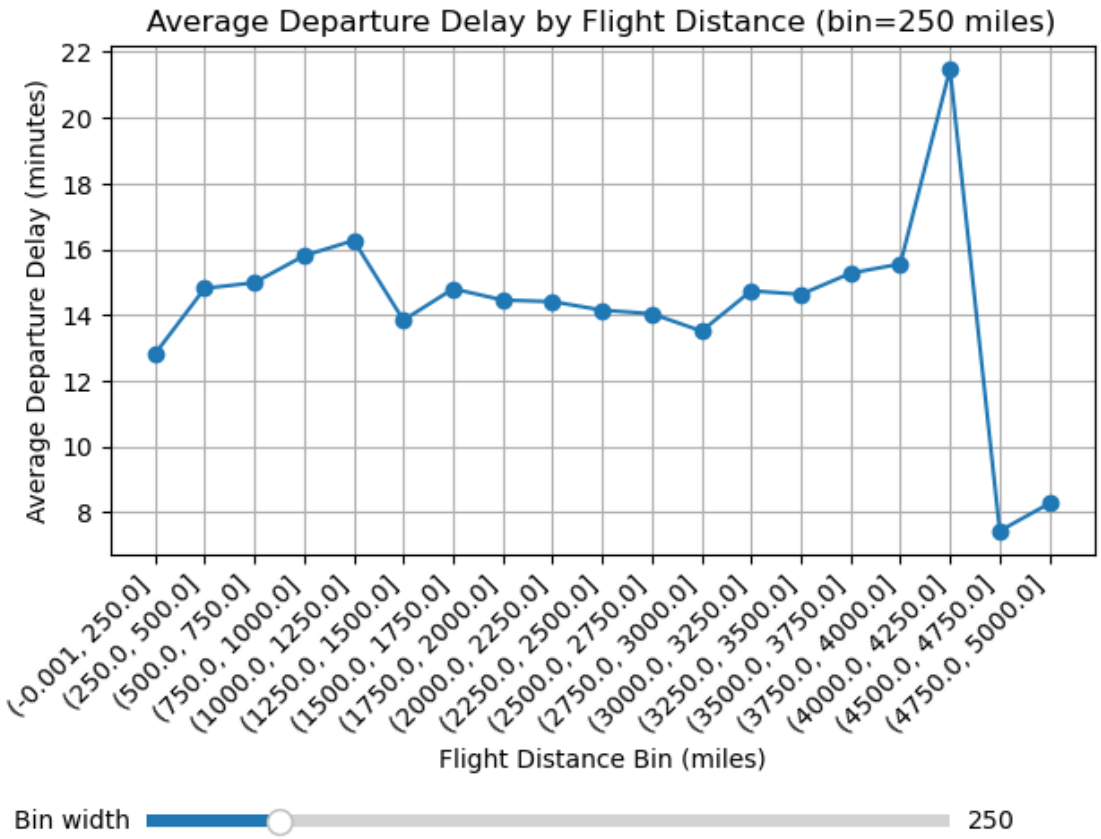
[19]:
```
cat_candidates = [c for c in ['Class', 'Type of Travel', 'Customer Type',␣
 ↪'Gender'] if c in airlines.columns]
if not cat_candidates:
    raise ValueError("Expected categorical columns not found.")

# Initial grouping var
current_group = cat_candidates[0]

# Compute initial means
means = airlines.groupby(current_group)['satisfaction'].mean().
 ↪sort_values(ascending=False)
```

```
fig, ax = plt.subplots()
bars = ax.bar(range(len(means)), means.values)
ax.set_xticks(range(len(means)))
ax.set_xticklabels(means.index.astype(str), rotation=0)
ax.set_title(f"Average Satisfaction by {current_group}")
ax.set_xlabel(current_group)
ax.set_ylabel("Average Satisfaction (0-1)")
plt.tight_layout(rect=[0.2, 0, 1, 1])  # leave space on the left for controls

# RadioButtons on the left
rax = plt.axes([0.02, 0.4, 0.15, 0.2])
radio = RadioButtons(rax, cat_candidates, active=0)

def update_group(label):
    global current_group
    current_group = label
    new_means = airlines.groupby(current_group)['satisfaction'].mean().
 ↪sort_values(ascending=False)
    ax.clear()
    ax.bar(range(len(new_means)), new_means.values)
    ax.set_xticks(range(len(new_means)))
    ax.set_xticklabels(new_means.index.astype(str), rotation=0)
    ax.set_title(f"Average Satisfaction by {current_group}")
    ax.set_xlabel(current_group)
    ax.set_ylabel("Average Satisfaction (0-1)")
    fig.canvas.draw_idle()

radio.on_clicked(update_group)
plt.show()
```



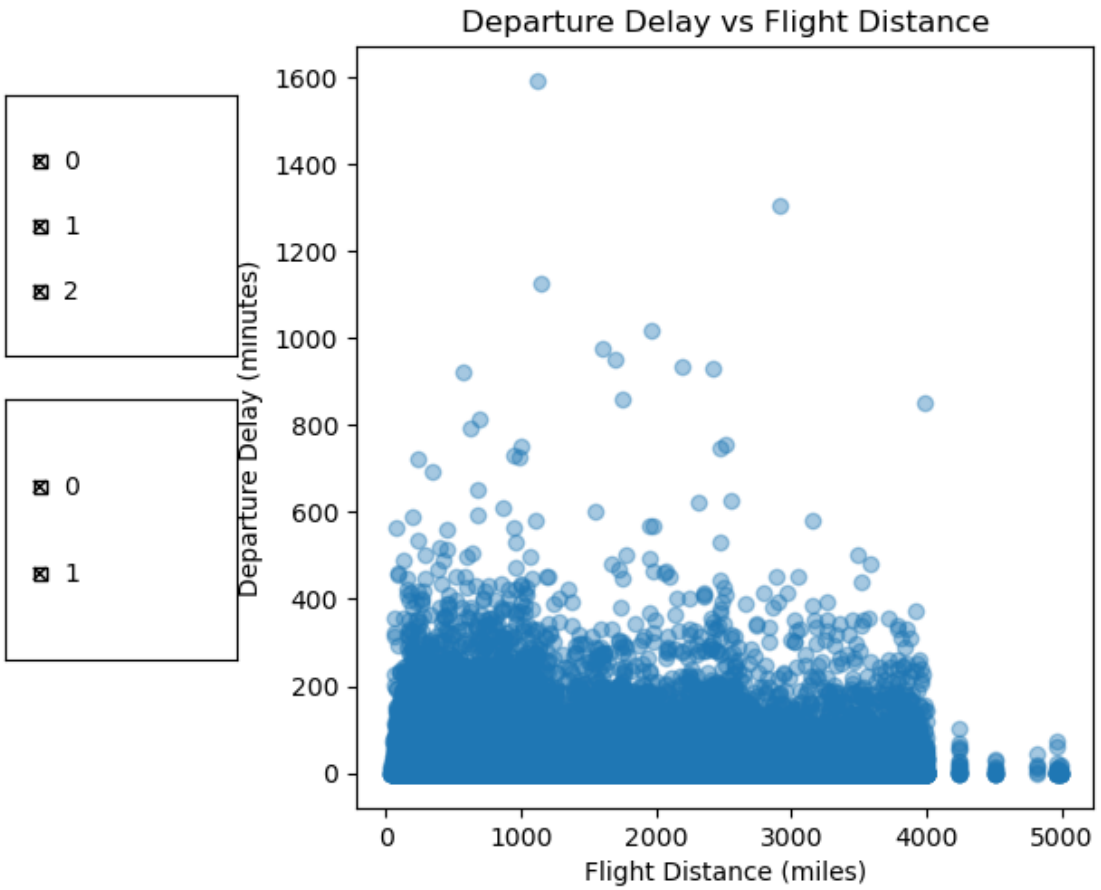
[20]:
```
init_step = 250

dmax = int(max(1, airlines[distance_col].max()))
edges = list(range(0, dmax + init_step, init_step))
grp = pd.cut(airlines[distance_col], bins=edges, include_lowest=True)
avg_delay = airlines.groupby(grp, observed=True)['Departure Delay in Minutes'].
 ↪mean()

fig2, ax2 = plt.subplots()
(line2,) = ax2.plot(range(len(avg_delay)), avg_delay.values, marker='o')
ax2.set_xticks(range(len(avg_delay)))
ax2.set_xticklabels([str(i) for i in avg_delay.index.astype(str)], rotation=45,␣
 ↪ha='right')
ax2.set_title(f"Average Departure Delay by Flight Distance (bin={init_step}␣
 ↪miles)")
ax2.set_xlabel("Flight Distance Bin (miles)")
ax2.set_ylabel("Average Departure Delay (minutes)")
ax2.grid(True)
plt.tight_layout(rect=[0, 0.05, 1, 1])
```

```
# Slider
ax_step = plt.axes([0.15, 0.01, 0.7, 0.03])
step_slider = Slider(ax_step, 'Bin width', 100, 1000, valinit=init_step,␣
 ↪valstep=50)

def on_step_change(val):
    step = int(step_slider.val)
    dmax2 = int(max(1, airlines[distance_col].max()))
    edges2 = list(range(0, dmax2 + step, step))
    grp2 = pd.cut(airlines[distance_col], bins=edges2, include_lowest=True)
    avg2 = airlines.groupby(grp2, observed=True)['Departure Delay in Minutes'].
 ↪mean()
    ax2.clear()
    ax2.plot(range(len(avg2)), avg2.values, marker='o')
    ax2.set_xticks(range(len(avg2)))
    ax2.set_xticklabels([str(i) for i in avg2.index.astype(str)], rotation=45,␣
 ↪ha='right')
    ax2.set_title(f"Average Departure Delay by Flight Distance (bin={step}␣
 ↪miles)")
    ax2.set_xlabel("Flight Distance Bin (miles)")
    ax2.set_ylabel("Average Departure Delay (minutes)")
    ax2.grid(True)
    fig2.canvas.draw_idle()

step_slider.on_changed(on_step_change)
plt.show()
```



[21]:
```
class_vals = sorted(airlines['Class'].dropna().unique().tolist()) if 'Class' in␣
 ↪airlines.columns else []
travel_vals = sorted(airlines['Type of Travel'].dropna().unique().tolist()) if␣
 ↪'Type of Travel' in airlines.columns else []

fig3, ax3 = plt.subplots()
plt.tight_layout(rect=[0.25, 0, 1, 1])  # space on left for controls

# Initial mask: all selected
selected_class = {v: True for v in class_vals}
selected_travel = {v: True for v in travel_vals}

def current_mask():
    df = airlines.copy()
    if class_vals:
        df = df[df['Class'].isin([k for k,v in selected_class.items() if v])]
    if travel_vals:
        df = df[df['Type of Travel'].isin([k for k,v in selected_travel.items()␣
 ↪if v])]
    return df
```

```
def redraw():
    df = current_mask()
    ax3.clear()
    ax3.scatter(df[distance_col], df['Departure Delay in Minutes'], alpha=0.4)
    ax3.set_title("Departure Delay vs Flight Distance")
    ax3.set_xlabel("Flight Distance (miles)")
    ax3.set_ylabel("Departure Delay (minutes)")
    fig3.canvas.draw_idle()

redraw()

# Controls: two groups of CheckButtons
y0 = 0.6
if class_vals:
    rax_class = plt.axes([0.02, y0, 0.2, 0.3])
    labels_class = [str(v) for v in class_vals]
    checks_class = CheckButtons(rax_class, labels_class, [True]*len(class_vals))
    def on_class(label):
        key = int(label) if label.isdigit() else label
        selected_class[key] = not selected_class[key]
        redraw()
    checks_class.on_clicked(on_class)
    y0 -= 0.35

if travel_vals:
    rax_travel = plt.axes([0.02, y0, 0.2, 0.3])
    labels_travel = [str(v) for v in travel_vals]
    checks_travel = CheckButtons(rax_travel, labels_travel,␣
 ↪[True]*len(travel_vals))
    def on_travel(label):
        key = int(label) if label.isdigit() else label
        selected_travel[key] = not selected_travel[key]
        redraw()
    checks_travel.on_clicked(on_travel)

plt.show()
```



3.0.7 Results

Across ~130k records, service/flight factors relate meaningfully to outcomes. Business class yields the highest satisfaction; loyal customers rate experiences higher than disloyal customers. Satisfaction peaks for Business class on Business trips, indicating a joint effect of cabin and purpose. Average departure delays vary by distance group, with mid-range distances showing elevated means relative to the shortest and longest groups. These patterns support H and H and suggest airlines can improve satisfaction by aligning service tier to trip purpose and address delay hot-spots in specific distance bands. Limitations: the dataset lacks dates/airports (no external drivers like weather or traffic), so causality can't be inferred; results describe associations within the available variables.

[ ]: