awasthi_17_cse5243_hw2_copy

March 15, 2023

1 CSE 5243 - Introduction to Data Mining

1.1 Homework 2: Classification

- Semester: Spring 2023
- Instructor: Tom Bihari
- Section Days/Time: Wednesday/Friday 9:35 AM
- Student Name: Sarthak Awasthi
- Student Email: awasthi.17@osu.edu
- Student ID: 500500617 ***

---

2 Section: Overview

2.0.1 Assignment Overview

This assignment covers the **steps 4 and 5 of the six steps** of the **CRISP-DM process model** (Modelng and Evaluation). (See the CRISP-DM materials on CARMEN.)

The **objectives** of this assignment are: - Solve a business problem by creating, evaluating, and comparing three classification models, and produce the outputs needed to provide business value for your stakeholders. - Experiment with built-in classification models in **scikit-learn**.

2.0.2 Dataset

NOTE: Since you already have pre-processed this dataset in the previous assignment, you may choose to use your "cleaned up" dataset from that assignment instead of re-doing the work here.

In this assignment, you will analyze an ALTERED copy of the "Hotel Booking Demand" dataset. - This dataset was pulled on 4/8/22 from: https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand - The dataset file is named: **hotel_bookings_with_errors_V1.csv**

The data has been altered slightly for use in course assignments,etc.: - A unique ROW attribute has been added. - Errors have been added, such as: duplicated records, deleted records, deleted attribute values, erroneous attribute values. **DO NOT PUBLISH THIS DATASET - it contains intentionally wrong data!**

2.0.3 Problem Statement

Assume that you are the Director of Data Science for Buckeye Resorts, Inc. (BRI), an international hotel chain. As is the case for all hotel chains, reservation cancellations cause significant impacts to BRI, in profitability, logistics, and other areas. Approximately **20%** of reservations are cancelled, and the cost to BRI of a cancelled reservation is **$500** on average.

- BRI wants to improve (decrease) the cancellation rates at its hotels, using more tailored interventions, based on newly available detailed data. BRI processes **100,000** reservations per year, so an incremental improvement in cancellation rates would have a significant impact.
- One intervention being considered is to offer a special financial incentive to customers who have reservations, but who are "at risk" of cancellation. BRI has performed a small pilot test, and has found that offering a **$80** discount to a customer who is planning to cancel is effective **35%** of the time in inducing the customer not to cancel (by locking in a "no cancellation" clause).
- BRI leadership has asked your team to analyze the new data, and determine if it is suitable for developing analyses and models that would be effective in predicting which future reservations are likely to be at risk of cancellation, so the aforementioned financial incentive could be offered.
- The head of BRI would then like you attend the upcoming BRI Board of Directors meeting. She has asked you to present your findings to her and to the BOD, to help them decide whether to go forward with the planned tailored intervention approach, and/or to adjust or abandon the approach. Your goal is to support the BOD in making a decision.

In the previous assignment, you completed the sections for the first three steps of CRISP-DM. You **explored** the dataset, and **prepared** a clean dataset from it that contains the kind of information you think might be useful. You now will make use of the dataset.

2.0.4 Things To Do

You now must **develop** and **evaluate** specific models for predicting the cancellations. You will try the **off-the-shelf KNN classifier**, and **two other classifiers of your choice**.

Some intial guidance / sugggestions:

- You must develop a cost model from the problem statement above. Consider creating a table that lists the benefit and cost dollar amounts for a decision on a **single customer**. Note that the incentive will be "offered" if Predicted is True, and the incentive is "needed" if Actual is True:

Actual "At Risk"	Predicted "At Risk"	Incentive Benefit	Incentive Cost	Net Benefit (Benefit-Cost)
False	False	?	?	?
False	True	?	?	?
True	False	?	?	?
True	True	?	?	?

- Much of the code below may be repetitive. Consider creating a few reusable functions that can be called for each of the models you build (e.g., evaluation functions).

- **Follow the instructions** in each of the sections below.

It is essential that you **communicate** your goals, thought process, actions, results, and conclusions to the **audience** that will consume this work. It is **not enough** to show just the code. It is not appropriate to show long sections of **unexplained printout**, etc. Be kind to your readers and provide value to them!

ALWAYS follow this pattern when doing **each portion** of the work. This allows us to give feedback and assign scores, and to give partial credit. Make it easy for the reader to understand your work. - Say (briefly) **what** you are trying to do, and **why**. - Do it (code). - Show or describe the **result** clearly (and briefly as needed), and explain the significant **conclusions or insights** derived from the results.

2.0.5 Collaboration

For this assignment, you should work as an individual. You may informally discuss ideas with classmates, but your work should be your own.

2.0.6 What you need to turn in:

1) Code

- For this homework, the code is the Jupyter Notebook. Use the provided Jupyter Notebook template, and fill in the appropriate information.
- You may use common Python libraries for I/O, data manipulation, data visualization, etc. (e.g., NumPy, Pandas, MatPlotLib,…)
- You may not use library operations that perform, in effect, the "core" computations for this homework (e.g., If the assignment is to write a K-Means algorithm, you may not use a library operation that, in effect, does the core work needed to implement a K-Means algorithm.). When in doubt, ask the grader or instructor. (**Note: For this assignment, you *will* be using build in library functions, so you are permitted to do so. You may not, however, make use of a single function that does *all* of the work for you.**
- The code must be written by you, and any significant code snips you found on the Internet and used to understand how to do your coding for the core functionality must be attributed. (You do not need to attribute basic functionality – matrix operations, IO, etc.)
- The code must be commented sufficiently to allow a reader to understand the algorithm without reading the actual Python, step by step.
- **When in doubt, ask the grader or instructor.**

2) Written Report

- For this homework, the report is the Jupyter Notebook. The report should be well-written. Please proof-read and remove spelling and grammar errors and typos.
- The report should discuss your analysis and observations. Key points and findings must be written in a style suitable for consumption by non-experts. Present charts and graphs to support your observations. If you performed any data processing, cleaning, etc., please discuss it within the report.

2.0.7 Grading

1. Overall readability and organization of your report (5%) > - Is it well organized and does the presentation flow in a logical manner? > - Are there no grammar and spelling mistakes? > - Do the charts/graphs relate to the text? > - Are the summarized key points and findings understandable by non-experts? > - Do the Overview and Conclusions provide context for the entire exercise?
2. Evaluation Method (10%) > - Does your evaluation method meet the needs of the developer (you) as well as the needs of your business stakeholders? > - Is the evaluation method sound? > - Did you describe both the method itself and why you chose it?
3. Pre-Processing of the Dataset (10%) > - Did you make reasonable choices for pre-processing, and explain why you made them?
4. Evaluation of the KNN Classifier (20%) > - Is your algorithm design and coding correct? > - Is it well documented? > - Have you made an effort to tune it for good performance? > - Is the evaluation sound?
5. Evaluation of the Second Classifier (20%) > - Is your algorithm design and coding correct? > - Is it well documented? > - Have you made an effort to tune it for good performance? > - Is the evaluation sound?
6. Evaluation of the Third Classifier (20%) > - Is your algorithm design and coding correct? > - Is it well documented? > - Have you made an effort to tune it for good performance? > - Is the evaluation sound?
7. Comparison of the Three Classifiers (10%) > - Is the comparison sound? > - Did you choose a specific classifier as best and explain why?
8. Conclusions (5%) > - Did you summarize appropriately your critical findings. > - Did you provide appropriate conclusions and next steps.

2.0.8 How to turn in your work on Carmen:

Submit to Carmen the Jupyter Notebook. You do not need to include the input data.

HAVE FUN! ***

---

3 Section: Overview

- Insert a short description of the scope of this exercise, any supporting information, etc. ***

Answer:

- This exercise involves developing and evaluating specific models for predicting hotel reservation cancellations.
- The problem statement describes the business context and goals, including a plan to offer a financial incentive to customers who are "at risk" of cancellation.
- The dataset has been explored and cleaned in the previous homework, and now the focus is on developing and evaluating classification models, including an off-the-shelf KNN classifier and two other classifiers of our choice.
- The goal is to support decision-making by providing insights into which reservations are likely to be at risk of cancellation, and whether offering financial incentives is a viable strategy for reducing cancellations.

4 Section: Setup

- Add any needed imports, helper functions, etc., here. ***

[1]:
```python
# Imports

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report, precision_score, recall_score, f1_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
```

[2]:
```python
# Helper functions

# Function to plot the confusion matrix
def plot_confusion_matrix(conf_matrix):
    plt.imshow(conf_matrix, cmap='Blues', interpolation='nearest')
    plt.colorbar()
    plt.xticks([0, 1], ['Not Cancelled', 'Cancelled'])
    plt.yticks([0, 1], ['Not Cancelled', 'Cancelled'])
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title('Confusion Matrix')
    plt.show()

# Function to evaluate a model using cross-validation
def evaluate_model(model, X, y, cv=5):
    # Use cross_val_score to get the scores for each fold
    scores = cross_val_score(model, X, y, cv=cv)

    # Print the mean and standard deviation of the scores
    print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))

# Function to print the classification report and confusion matrix
def print_results(y_test, y_pred):
    print(classification_report(y_test, y_pred))
    conf_matrix = confusion_matrix(y_test, y_pred)
    plot_confusion_matrix(conf_matrix)
```

Discussion:

The first function plots a confusion matrix with color-coded cells based on their value, and the second function evaluates a given model using cross-validation and prints the mean and standard deviation of the scores. The third function prints the classification report and a confusion matrix using the predicted and true labels.

---

5 Section: 1 - Evaluation Method

- Define measures for evaluating the classification models you develop. Explain why the measures you choose provide a useful view into the value and usefulness of the model you eventually chose for the company to use. **Note: In this section, you should define and explain your measures. You may create a reusable function here if you like. You then will use the functions in later sections.**
- Define two types: ***

5.1 Section: 1.1 - Define measures that do not include the cost information

- (e.g., confusion matrices, accuracy, precision, recall, F-measures, etc.).
- Consider using: from sklearn.metrics import classification_report, confusion_matrix ***

Answer:

Classification models can be evaluated using various measures to determine their accuracy, precision, recall, and other performance metrics. These measures can be used to compare the performance of different models and to choose the best model for the task at hand. Some of the measures that can be used to evaluate classification models are: - Confusion Matrix: A confusion matrix is a table that shows the actual and predicted class labels for a set of instances. It provides a detailed breakdown of the model's performance by showing the number of true positives, false positives, true negatives, and false negatives.

- Accuracy

  : Accuracy is the most commonly used measure for evaluating classification models. It measures the proportion of correctly classified instances over the total number of instances. However, accuracy may not be an appropriate measure when the classes are imbalanced.

- Precision

  : Precision is a measure of the model's ability to correctly identify positive instances. It is the proportion of true positives over the total number of predicted positives. A high precision indicates that the model is unlikely to label negative instances as positive.

- Recall

  : Recall is a measure of the model's ability to correctly identify all positive instances. It is the proportion of true positives over the total number of actual positives. A high recall indicates that the model is unlikely to miss any positive instances.

- F1-score

: The F1-score is a weighted average of precision and recall. It is a good measure of the model's overall performance, as it takes both precision and recall into account.

- Classification Report

  : The classification report is a summary of the precision, recall, and F1-score for each class in the model.

5.2 Section: 1.2 - Define measures that do include the cost information

- (e.g., using cost matrices).
- Consider creating a function that takes a confusion matrix and calculates the cost. ***

Answer:

Cost-sensitive evaluation measures are used when the cost of misclassification is not equal across different classes. In the context of this problem, the cost of a false negative (predicting a customer will not cancel but they do cancel) is much higher than the cost of a false positive (predicting a customer will cancel but they do not cancel).

One way to incorporate the cost information is to use a cost matrix. A cost matrix is a table that specifies the cost of misclassifying each class. For this problem, we can define a cost model table as follows: | Actual "At Risk" | Predicted "At Risk" | Incentive Benefit | Incentive Cost | Net Benefit (Benefit-Cost) | |—|—|—|—|—| | False | False | 0 | 0 | 0 | | False | True | 0 | 80 | -80 | | True | False | 0 | 0 | 0 | | True | True | 175 | 80 | 95 |

[3]:
```python
def calculate_cost(conf_matrix):
    incentive_cost = np.array([0, 80, 0, 80]) # Incentive cost for each scenario
    incentive_benefit = np.array([0, 0, 0, 175]) # Incentive benefit for each␣
↪scenario
    return (incentive_benefit * conf_matrix.flatten()) - (incentive_cost *␣
↪conf_matrix.flatten())

def net_benefit(conf_matrix):
    return np.sum(calculate_cost(conf_matrix))
```

Discussion:

The code defines functions to calculate the net benefit and cost from a confusion matrix using specific incentive costs and benefits for each scenario. The net benefit is calculated as the sum of the costs and benefits.

---

6 Section: 2 - Pre-Processing of the Dataset

- Load the dataset.
- Split the dataset into a Training dataset and a Test dataset based on the class attribute. Keep them separate and use the Training dataset for training/tuning and the Test dataset

for testing. For consistency, use the **train_test_split** operation available in SciKit Learn (use a specific random seed, so it is reproducible).
  - from sklearn.model_selection import train_test_split
  - X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
- **NOTE: You have done much of the data preprocessing in the previous assignment, so you don't have to re-do it here. You can either copy the necessary code from the previous assignment, or generate the clean dataset from the previous assignment and load it here.** ***

6.1 Section: 2.1 - Explore the attributes

- As in Homework 1, explore the attributes briefly. Reference the website listed in the Introduction.
- Provide basic statistics for the attributes.
- List which attributes are Nominal (even though they are encoded as numbers), Ordinal, Interval, Ratio.
- **NOTE: Just summarize here. You will need to know which attributes are Nominal, etc., so it would be useful to list them here.** ***

Answer:

The dataset available to us is a copy of the original Hotel Booking Demand dataset (https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand), with one additional attribute ROW. This version of the dataset in use here has errors in the form of additional records, deleted records and, deleted and erroneous attribute values.

The ***hotel_bookings_with_errors_V1*** version of the data has the following attributes (from HW 1): - **Attribute, Type, DataType, Meaning**

- ROW, ordinal, integer, record number 0,1,2,3,….
- Hotel, nominal, string, the type of hotel (City Hotel or Resort Hotel)
- is_canceled, nominal, integer, whether the booking was cancelled or not
- lead_time, ratio, integer, number of days that elapsed between the entering date of the booking into the PMS and the arrival date
- arrival_date_year, ordinal, integer, year of the arrival date
- arrival_date_month, ordinal, integer, month of the arrival date
- arrival_date_week_number, ordinal, integer, week number of the arrival date
- arrival_date_day_of_month, ordinal, integer, day of the month of the arrival date
- stays_in_weekend_nights, ratio, integer, number of weekend nights (Saturday or Sunday) the guest stayed or booked to stay
- stays_in_week_nights, ratio, integer, number of week nights (Monday to Friday) the guest stayed or booked to stay
- adults, ratio, integer, number of adults
- children, ratio, integer, number of children
- babies, ratio, integer, number of babies
- meal, nominal, string, type of meal booked. Categories are presented in standard hospitality industry size, styles, and names.

- country, nominal, string, country of origin. Categories are represented in the ISO 3155–3:2013 format
- market_segment, nominal, string, market segment designation. In categories, the term “TA” means “Travel Agents” and “TO” means “Tour Operators”
- distribution_channel, nominal, string, booking distribution channel. The term “TA” means “Travel Agents” and “TO” means “Tour Operators”
- is_repeated_guest, nominal, integer, value indicating if the booking name was from a repeated guest (1) or not (0)
- previous_cancellations, ratio, integer, number of previous bookings that were cancelled by the customer
- previous_bookings_not_canceled. ratio, integer, number of previous bookings not cancelled by the customer
- reserved_room_type, nominal, string, code of room type reserved. Code is presented instead of designation for anonymity reasons.
- assigned_room_type, nominal, string, code for the type of room assigned to the booking. Sometimes the assigned room type differs from the reserved room type due to hotel operation reasons (e.g. overbooking) or by customer request. Code is presented instead of designation for anonymity reasons.
- booking_changes, ratio, integer, number of changes/amendments made to the booking from the moment the booking was entered on the PMS until the moment of check-in or cancellation
- deposit_type, nominal, string, indicates if the customer made a deposit to guarantee the booking. This variable can assume three categories: No Deposit = 0; Non Refund = 1; Refundable = 2
- agent, nominal, integer, ID of the travel agency that made the booking
- company, nominal, integer, ID of the company/entity that made the booking or responsible for paying the booking. ID is presented instead of designation for anonymity reasons
- days_in_waiting_list, ratio, integer, number of days the booking was in the waiting list before it was confirmed to the customer
- customer_type, nominal, string, type of booking, assuming one of four categories: Contract - when the booking has an allotment or other type of contract associated to it; Group - when the booking is associated to a group; Transient - when the booking is not part of a group or contract, and is not associated to other transient booking; Transient-party - when the booking is transient, but is associated to at least other transient booking
- adr, ratio, float, average daily rate as defined by dividing the sum of all lodging revenues by the total number of staying nights
- required_car_parking_spaces, ratio, integer, number of car parking spaces required by the customer
- total_of_special_requests, ratio, integer, number of special requests made by the customer (e.g. twin bed or high floor)
- reservation_status, nominal, string, reservation last status, assuming one of three categories: Canceled - booking was canceled by the customer; Check-Out - customer has checked in but already departed; No-Show - customer did not check-in and did not notify the hotel of the cancellation.
- reservation_status_date, interval, date, Date at which the last status was set. This variable can be used in conjunction with the ReservationStatus to understand when was the booking canceled or when did the customer checked-out of the hotel.

[4]:
```
try:
    _ = data_from_source_file_df
    print("Reusing source data")
except:
    print("Loading source data")
    data_from_source_file_df = pd.read_csv("hotel_bookings_with_errors_V1.csv")
data_df = data_from_source_file_df
```

```
Loading source data
```

[5]:
```
#Basic statistics

data_df.describe()
```

[5]:
```
                 ROW    is_canceled      lead_time  arrival_date_year  \
count  119391.000000  119390.000000  119391.000000      119391.000000
mean    59694.000050       0.370408     104.010662        2016.156544
std     34465.357245       0.482916     106.862967           0.707481
min         0.000000       0.000000       0.000000        2015.000000
25%     29846.500000       0.000000      18.000000        2016.000000
50%     59694.000000       0.000000      69.000000        2016.000000
75%     89541.500000       1.000000     160.000000        2017.000000
max    119389.000000       1.000000     737.000000        2017.000000

       arrival_date_week_number  arrival_date_day_of_month  \
count             119391.000000              119391.000000
mean                  27.165172                  15.798117
std                   13.605081                   8.780897
min                    1.000000                   1.000000
25%                   16.000000                   8.000000
50%                   28.000000                  16.000000
75%                   38.000000                  23.000000
max                   53.000000                  31.000000

       stays_in_weekend_nights  stays_in_week_nights         adults  \
count            119391.000000         119391.000000  119391.000000
mean                  0.927591              2.500297       1.856572
std                   0.998613              1.908278       0.582185
min                   0.000000              0.000000       0.000000
25%                   0.000000              1.000000       2.000000
50%                   1.000000              2.000000       2.000000
75%                   2.000000              3.000000       2.000000
max                  19.000000             50.000000      55.000000

            children  …  is_repeated_guest  previous_cancellations  \
count  119387.000000  …      119391.000000           119391.000000
mean        0.103889  …           0.031912                0.087117
```

```
std          0.398560  …          0.175766                0.844333
min          0.000000  …          0.000000                0.000000
25%          0.000000  …          0.000000                0.000000
50%          0.000000  …          0.000000                0.000000
75%          0.000000  …          0.000000                0.000000
max         10.000000  …          1.000000               26.000000

       previous_bookings_not_canceled  booking_changes         agent  \
count                   119391.000000    119391.000000  103051.00000
mean                         0.137096         0.221122      86.69487
std                          1.497431         0.652303     110.77504
min                          0.000000         0.000000       1.00000
25%                          0.000000         0.000000       9.00000
50%                          0.000000         0.000000      14.00000
75%                          0.000000         0.000000     229.00000
max                         72.000000        21.000000     535.00000

           company  days_in_waiting_list            adr  \
count  6797.000000         119391.000000  119391.000000
mean    189.266735              2.321130     101.831089
std     131.655015             17.594648      50.535580
min       6.000000              0.000000      -6.380000
25%      62.000000              0.000000      69.290000
50%     179.000000              0.000000      94.590000
75%     270.000000              0.000000     126.000000
max     543.000000            391.000000    5400.000000

       required_car_parking_spaces  total_of_special_requests
count                119391.000000              119391.000000
mean                      0.062517                   0.571366
std                       0.245290                   0.792796
min                       0.000000                   0.000000
25%                       0.000000                   0.000000
50%                       0.000000                   0.000000
75%                       0.000000                   1.000000
max                       8.000000                   5.000000

[8 rows x 21 columns]
```

Discussion:

There were some data quality issues here such as missing values for **is_canceled, agent** and **company** attributes. These were rectified in the previous assignment.

[6]:
```
attributes = [
    ('ROW', 'ordinal', 'integer', 'record number 0,1,2,3,....'),
    ('Hotel', 'nominal', 'string', 'the type of hotel (City Hotel or Resort␣
 ↪Hotel)'),
    ('is_canceled', 'nominal', 'integer', 'whether the booking was cancelled or␣
↪not'),
    ('lead_time', 'ratio', 'integer', 'number of days that elapsed between the␣
↪entering date of the booking into the PMS and the arrival date'),
    ('arrival_date_year', 'ordinal', 'integer', 'year of the arrival date'),
    ('arrival_date_month', 'ordinal', 'integer', 'month of the arrival date'),
    ('arrival_date_week_number', 'ordinal', 'integer', 'week number of the␣
↪arrival date'),
    ('arrival_date_day_of_month', 'ordinal', 'integer', 'day of the month of␣
↪the arrival date'),
    ('stays_in_weekend_nights', 'ratio', 'integer', 'number of weekend nights␣
↪(Saturday or Sunday) the guest stayed or booked to stay'),
    ('stays_in_week_nights', 'ratio', 'integer', 'number of week nights (Monday␣
↪to Friday) the guest stayed or booked to stay'),
    ('adults', 'ratio', 'integer', 'number of adults'),
    ('children', 'ratio', 'integer', 'number of children'),
    ('babies', 'ratio', 'integer', 'number of babies'),
    ('meal', 'nominal', 'string', 'type of meal booked. Categories are␣
↪presented in standard hospitality industry size, styles, and names.'),
    ('country', 'nominal', 'string', 'country of origin. Categories are␣
↪represented in the ISO 3155-3:2013 format'),
    ('market_segment', 'nominal', 'string', 'market segment designation. In␣
↪categories, the term "TA" means "Travel Agents" and "TO" means "Tour␣
↪Operators"'),
    ('distribution_channel', 'nominal', 'string', 'booking distribution channel.
↪ The term "TA" means "Travel Agents" and "TO" means "Tour Operators"'),
    ('is_repeated_guest', 'nominal', 'integer', 'value indicating if the␣
↪booking name was from a repeated guest (1) or not (0)'),
    ('previous_cancellations', 'ratio', 'integer', 'number of previous bookings␣
↪that were cancelled by the customer'),
    ('previous_bookings_not_canceled', 'ratio', 'integer', 'number of previous␣
↪bookings not cancelled by the customer'),
    ('reserved_room_type', 'nominal', 'string', 'code of room type reserved.␣
↪Code is presented instead of designation for anonymity reasons.'),
    ('assigned_room_type', 'nominal', 'string', 'code for the type of room␣
↪assigned to the booking. Sometimes the assigned room type differs from the␣
↪reserved room type due to hotel operation reasons (e.g. overbooking) or by␣
↪customer request. Code is presented instead of designation for anonymity␣
↪reasons.'),
    ('booking_changes', 'ratio', 'integer', 'number of changes/amendments made␣
↪to the booking from the original booking'),
    ('deposit_type', 'nominal', 'string', 'indication on if the customer made a␣
↪deposit to guarantee the booking. This variable can assume three categories:␣
↪No Deposit, Non Refund, Refundable'),
    ('agent', 'nominal', 'integer', 'identification number of the travel agency␣
↪that made the booking'),
    ('company', 'nominal', 'integer', 'identification number of the company/
 ↪entity that made the booking or responsible for paying the booking. ID is␣
 ↪presented instead of designation for anonymity reasons.'),
    ('days_in_waiting_list', 'ratio', 'integer', 'number of days the booking␣
 ↪was in the waiting list before it was confirmed to the customer'),
    ('customer_type', 'nominal', 'string', 'type of booking, assuming one of␣
 ↪four categories: Contract, Group, Transient, Transient-Party'),
    ('adr', 'ratio', 'float', 'Average Daily Rate as defined by dividing the␣
 ↪sum of all lodging revenue by the total number of staying nights'),
    ('required_car_parking_spaces', 'ratio', 'integer', 'number of car parking␣
 ↪spaces required by the customer'),
    ('total_of_special_requests', 'ratio', 'integer', 'number of special␣
 ↪requests made by the customer (e.g. twin bed or high floor)'),
    ('reservation_status', 'nominal', 'string', 'Reservation last status,␣
 ↪assuming one of three categories: Canceled - booking was canceled by the␣
 ↪customer; Check-Out - customer has checked in but already departed; No-Show -
 customer did not check-in and did inform the hotel'),
    ('reservation_status_date', 'interval', 'date', 'date at which the last␣
 ↪status was set. This variable can be used in conjunction with the␣
 ↪ReservationStatus to understand when was the booking canceled or when did␣
 ↪the customer checked-out of the hotel.'),
]
```

```
[7]: #set the maximum column width of the dataframe
pd.set_option('display.max_colwidth', None)

df = pd.DataFrame(attributes, columns=['Attribute', 'Type', 'DataType',␣
 ↪'Meaning'])

#set the text alignment of the data in the dataframe to be left aligned
df = df.style.set_properties(**{'text-align': 'left'})

#set the text alignment of the header row of the dataframe to be left aligned
df = df.set_table_styles([dict(selector = 'th', props=[('text-align',␣
 ↪'left')])])
display(df)
```

```
<pandas.io.formats.style.Styler at 0x14193cd90>
```

6.2 Section: 2.2 - Revise the dataset

- Review the meanings of the attributes and consider removing redundant or (likely) irrelevant attributes, combining attributes, etc., to reduce the number of attributes.
- (You may choose to use techniques such as those you used in Homework 1 to analyze the impacts of individual attributes on the CLASS attribute.)

- Describe what you chose to do (and not do), and why. -**NOTE: You can just load your cleaned-up dataset here if you like.** ***

We create a copy of the original dataset and clean it.

[8]:
```python
# Create a copy of the dataset for cleaning
clean_data_df = data_df.copy()
clean_data_df
```

[8]:
```
           ROW         hotel  is_canceled  lead_time  arrival_date_year  \
0            0  Resort Hotel          0.0        342               2015
1            1  Resort Hotel          0.0        737               2015
2            2  Resort Hotel          0.0          7               2015
3            3  Resort Hotel          0.0         13               2015
4            4  Resort Hotel          0.0         14               2015
…          …           …          …          …                 …
119386  119385    City Hotel          0.0         23               2017
119387  119386    City Hotel          0.0        102               2017
119388  119387    City Hotel          0.0         34               2017
119389  119388    City Hotel          0.0        109               2017
119390  119389    City Hotel          0.0        205               2017

       arrival_date_month  arrival_date_week_number  \
0                    July                        27
1                    July                        27
2                    July                        27
3                    July                        27
4                    July                        27
…                    …                       …
119386             August                        35
119387             August                        35
119388             August                        35
119389             August                        35
119390             August                        35

        arrival_date_day_of_month  stays_in_weekend_nights  \
0                               1                        0
1                               1                        0
2                               1                        0
3                               1                        0
4                               1                        0
…                             …                      …
119386                         30                        2
119387                         31                        2
119388                         31                        2
119389                         31                        2
119390                         29                        2
```

```
        stays_in_week_nights  …  deposit_type  agent  company  \
0                          0  …    No Deposit    NaN      NaN
1                          0  …    No Deposit    NaN      NaN
2                          1  …    No Deposit    NaN      NaN
3                          1  …    No Deposit  304.0      NaN
4                          2  …    No Deposit  240.0      NaN
…                        …  …           …    …        …
119386                     5  …    No Deposit  394.0      NaN
119387                     5  …    No Deposit    9.0      NaN
119388                     5  …    No Deposit    9.0      NaN
119389                     5  …    No Deposit   89.0      NaN
119390                     7  …    No Deposit    9.0      NaN

       days_in_waiting_list customer_type     adr required_car_parking_spaces  \
0                         0     Transient    0.00                           0
1                         0     Transient    0.00                           0
2                         0     Transient   75.00                           0
3                         0     Transient   75.00                           0
4                         0     Transient   98.00                           0
…                       …           …      …                         …
119386                    0     Transient   96.14                           0
119387                    0     Transient  225.43                           0
119388                    0     Transient  157.71                           0
119389                    0     Transient  104.40                           0
119390                    0     Transient  151.20                           0

        total_of_special_requests  reservation_status  reservation_status_date
0                               0           Check-Out                 7/1/2015
1                               0           Check-Out                 7/1/2015
2                               0           Check-Out                 7/2/2015
3                               0           Check-Out                 7/2/2015
4                               1           Check-Out                 7/3/2015
…                            …               …                   …
119386                          0           Check-Out                 9/6/2017
119387                          2           Check-Out                 9/7/2017
119388                          4           Check-Out                 9/7/2017
119389                          0           Check-Out                 9/7/2017
119390                          2           Check-Out                 9/7/2017

[119391 rows x 33 columns]
```

[9]:
```
#Checking the missing values in the columns

data_df.isnull().sum()
```

[9]:
```
ROW                                0
hotel                              0
```

```
is_canceled                            1
lead_time                              0
arrival_date_year                      0
arrival_date_month                     0
arrival_date_week_number               0
arrival_date_day_of_month              0
stays_in_weekend_nights                0
stays_in_week_nights                   0
adults                                 0
children                               4
babies                                 0
meal                                   0
country                              488
market_segment                         0
distribution_channel                   0
is_repeated_guest                      0
previous_cancellations                 0
previous_bookings_not_canceled         0
reserved_room_type                     0
assigned_room_type                     0
booking_changes                        0
deposit_type                           0
agent                              16340
company                           112594
days_in_waiting_list                   0
customer_type                          0
adr                                    0
required_car_parking_spaces            0
total_of_special_requests              0
reservation_status                     0
reservation_status_date                0
dtype: int64
```

Discussion:

We are creating a copy of the dataset before cleaning it because cleaning the data can sometimes result in loss of information, mistakes or unintended changes to the original data. By creating a copy of the dataset, we have a backup of the original data that you can refer to in case anything goes wrong during the cleaning process.

[10]:
```
#Column-wise null percentage

print(round(100*(data_df.isnull().sum()/len(data_df.index)),2))
```

```
ROW                                  0.00
hotel                                0.00
is_canceled                          0.00
lead_time                            0.00
```

```
arrival_date_year                  0.00
arrival_date_month                 0.00
arrival_date_week_number           0.00
arrival_date_day_of_month          0.00
stays_in_weekend_nights            0.00
stays_in_week_nights               0.00
adults                             0.00
children                           0.00
babies                             0.00
meal                               0.00
country                            0.41
market_segment                     0.00
distribution_channel               0.00
is_repeated_guest                  0.00
previous_cancellations             0.00
previous_bookings_not_canceled     0.00
reserved_room_type                 0.00
assigned_room_type                 0.00
booking_changes                    0.00
deposit_type                       0.00
agent                             13.69
company                           94.31
days_in_waiting_list               0.00
customer_type                      0.00
adr                                0.00
required_car_parking_spaces        0.00
total_of_special_requests          0.00
reservation_status                 0.00
reservation_status_date            0.00
dtype: float64
```

[11]:
```
#fix spelling errors or typos in the data to make sure it is accurate and␣
 ↪consistent.
clean_data_df['market_segment'] = clean_data_df['market_segment'].
 ↪replace(to_replace='Onlin TA', value='Online TA')
```

[12]:
```
#remove one row corresponding to null value for is_canceled attribute.
clean_data_df = clean_data_df.dropna(subset=['is_canceled'])
clean_data_df = clean_data_df.reset_index(drop=True)
```

Discussion:

There is not much loss in removing the row with null value for the class attribute since it forms a tiny percentage of the dataset and we cannot really predict its value at this stage by imputing or replacing with mean/median values.

[13]:
```
clean_data_df.drop_duplicates(keep='first', inplace=True)  # This removes all␣
 ↪but one copy of the records that are completely identical.
```

```
clean_data_df = clean_data_df.reset_index(drop=True)
```

[14]:
```
clean_data_df.drop_duplicates(subset=['ROW'], keep='first', inplace=True)  #␣
 ↪This removes all but one copy of the records that have identical row_ids.
clean_data_df = clean_data_df.reset_index(drop=True)
```

Discussion:

We have removed duplicate records through the above steps. This is done to avoid inaccuracies and inconsistencies into the data.

[15]:
```
clean_data_df=clean_data_df.drop(['agent','company'],axis=1)
```

Discussion:

agent and **company** attributes have very high number of null values. Therefore, these two columns or attributes are dropped from the dataset. This is done to avoid having missing or inaccurate data that can negatively impact analysis and modeling results further in the process.

[16]:
```
#check if the attributes are removed
clean_data_df.isnull().sum()
```

[16]:
```
ROW                                 0
hotel                               0
is_canceled                         0
lead_time                           0
arrival_date_year                   0
arrival_date_month                  0
arrival_date_week_number            0
arrival_date_day_of_month           0
stays_in_weekend_nights             0
stays_in_week_nights                0
adults                              0
children                            4
babies                              0
meal                                0
country                           488
market_segment                      0
distribution_channel                0
is_repeated_guest                   0
previous_cancellations              0
previous_bookings_not_canceled      0
reserved_room_type                  0
assigned_room_type                  0
booking_changes                     0
deposit_type                        0
days_in_waiting_list                0
customer_type                       0
adr                                 0
```

```
required_car_parking_spaces        0
total_of_special_requests          0
reservation_status                 0
reservation_status_date            0
dtype: int64
```

[17]:
```
#removing null values for country attribute

clean_data_df=clean_data_df.dropna(axis = 0)
```

Discussion:

We also remove all rows/records with null values for **country** attribute. Again, it forms a very small percentage of the complete dataset. So, it does not cause a significant loss of data quality.

[18]:
```
clean_data_df.isnull().sum()
```

[18]:
```
ROW                               0
hotel                             0
is_canceled                       0
lead_time                         0
arrival_date_year                 0
arrival_date_month                0
arrival_date_week_number          0
arrival_date_day_of_month         0
stays_in_weekend_nights           0
stays_in_week_nights              0
adults                            0
children                          0
babies                            0
meal                              0
country                           0
market_segment                    0
distribution_channel              0
is_repeated_guest                 0
previous_cancellations            0
previous_bookings_not_canceled    0
reserved_room_type                0
assigned_room_type                0
booking_changes                   0
deposit_type                      0
days_in_waiting_list              0
customer_type                     0
adr                               0
required_car_parking_spaces       0
total_of_special_requests         0
reservation_status                0
reservation_status_date           0
```

```
dtype: int64
```

[19]:
```
# Get the duplicate records across all attributes.
dup_df = clean_data_df[clean_data_df.duplicated()]
num_dup_records = dup_df.shape[0]
print("Number of duplicate records across all attributes:", num_dup_records)
dup_df.head(10)
```

```
Number of duplicate records across all attributes: 0
```

[19]:
```
Empty DataFrame
Columns: [ROW, hotel, is_canceled, lead_time, arrival_date_year,
arrival_date_month, arrival_date_week_number, arrival_date_day_of_month,
stays_in_weekend_nights, stays_in_week_nights, adults, children, babies, meal,
country, market_segment, distribution_channel, is_repeated_guest,
previous_cancellations, previous_bookings_not_canceled, reserved_room_type,
assigned_room_type, booking_changes, deposit_type, days_in_waiting_list,
customer_type, adr, required_car_parking_spaces, total_of_special_requests,
reservation_status, reservation_status_date]
Index: []

[0 rows x 31 columns]
```

[20]:
```
# Get the missing values for the entire dataframe.
na_df = clean_data_df[clean_data_df.isna().any(axis=1)]
num_records_with_na = na_df.shape[0]
num_na_values = na_df.isna().sum().sum()
print("Total Records:", clean_data_df.shape[0], "- Number of Records with NA:",␣
 ↪num_records_with_na, "- Number of NA Values:", num_na_values)
na_df.head(10)
```

```
Total Records: 118897 - Number of Records with NA: 0 - Number of NA Values: 0.0
```

[20]:
```
Empty DataFrame
Columns: [ROW, hotel, is_canceled, lead_time, arrival_date_year,
arrival_date_month, arrival_date_week_number, arrival_date_day_of_month,
stays_in_weekend_nights, stays_in_week_nights, adults, children, babies, meal,
country, market_segment, distribution_channel, is_repeated_guest,
previous_cancellations, previous_bookings_not_canceled, reserved_room_type,
assigned_room_type, booking_changes, deposit_type, days_in_waiting_list,
customer_type, adr, required_car_parking_spaces, total_of_special_requests,
reservation_status, reservation_status_date]
Index: []

[0 rows x 31 columns]
```

```
[21]: clean_data_df.loc[clean_data_df.lead_time> 500,'lead_time']=500
      clean_data_df.loc[clean_data_df.days_in_waiting_list>0,'days_in_waiting_list']=␣
       ↪ 1
      clean_data_df.loc[clean_data_df.stays_in_weekend_nights>=␣
       ↪5,'stays_in_weekend_nights']=  5
      clean_data_df.loc[clean_data_df.adults>4,'adults']=  4
      clean_data_df.loc[clean_data_df.
       ↪previous_bookings_not_canceled>0,'previous_bookings_not_canceled']=  1
      clean_data_df.loc[clean_data_df.
       ↪previous_cancellations>0,'previous_cancellations']=  1
      clean_data_df.loc[clean_data_df.stays_in_week_nights> ␣
       ↪10,'stays_in_week_nights']= 10
      clean_data_df.loc[clean_data_df.booking_changes>5,'booking_changes']=  5
```

```
[22]: clean_data_df.loc[clean_data_df.babies> 8,'babies']= 0
      clean_data_df.loc[clean_data_df.required_car_parking_spaces >␣
       ↪5,'required_car_parking_spaces'] = 0
      clean_data_df.loc[clean_data_df.children  > 8,'children']  = 0
```

Discussion:

In the process above, we have set the outlier values to be imputed with lesser values for an attribute (also known as "capping" or "flooring"). It can help improve the quality, accuracy, and reliability of the data. It reduces model bias, and would improve its performance.

```
[23]: # drop the ROW, meal, country, reservation_status_date columns and some others
      clean_data_df.drop(['ROW', 'days_in_waiting_list', 'arrival_date_year',␣
       ↪'assigned_room_type', 'booking_changes',
                     'reservation_status', 'arrival_date_month', 'country'], axis=1,␣
       ↪inplace=True)
```

Discussion:

These columns are dropped to simplify the DataFrame, reduce noise, and improve the efficiency of modeling and analysis. Since the goal is to predict whether a guest will cancel their reservation or not, these attributes, being irrelevant, are dropped from the cleaned dataset.

```
[24]: # create a new CSV file with the cleaned data
      clean_data_df.to_csv('clean_hotel_bookings.csv', index=False)
```

```
[25]: clean_data_df = pd.read_csv('clean_hotel_bookings.csv')
      clean_data_df
```

```
[25]:                hotel  is_canceled  lead_time  arrival_date_week_number  \
      0       Resort Hotel          0.0        342                        27
      1       Resort Hotel          0.0        500                        27
      2       Resort Hotel          0.0          7                        27
      3       Resort Hotel          0.0         13                        27
```

```
4       Resort Hotel          0.0         14                       27
…              …        …        …                        …
118892    City Hotel          0.0         23                       35
118893    City Hotel          0.0        102                       35
118894    City Hotel          0.0         34                       35
118895    City Hotel          0.0        109                       35
118896    City Hotel          0.0        205                       35

        arrival_date_day_of_month  stays_in_weekend_nights  \
0                               1                        0
1                               1                        0
2                               1                        0
3                               1                        0
4                               1                        0
…                             …                       …
118892                         30                        2
118893                         31                        2
118894                         31                        2
118895                         31                        2
118896                         29                        2

        stays_in_week_nights  adults  children  babies  … is_repeated_guest  \
0                          0       2       0.0       0  …                 0
1                          0       2       0.0       0  …                 0
2                          1       1       0.0       0  …                 0
3                          1       1       0.0       0  …                 0
4                          2       2       0.0       0  …                 0
…                       …      …        …      …  …              …
118892                     5       2       0.0       0  …                 0
118893                     5       3       0.0       0  …                 0
118894                     5       2       0.0       0  …                 0
118895                     5       2       0.0       0  …                 0
118896                     7       2       0.0       0  …                 0

       previous_cancellations previous_bookings_not_canceled  \
0                           0                              0
1                           0                              0
2                           0                              0
3                           0                              0
4                           0                              0
…                        …                            …
118892                      0                              0
118893                      0                              0
118894                      0                              0
118895                      0                              0
118896                      0                              0
```

```
        reserved_room_type  deposit_type  customer_type     adr  \
0                        C    No Deposit      Transient    0.00
1                        C    No Deposit      Transient    0.00
2                        A    No Deposit      Transient   75.00
3                        A    No Deposit      Transient   75.00
4                        A    No Deposit      Transient   98.00
…                    …             …             …     …
118892                   A    No Deposit      Transient   96.14
118893                   E    No Deposit      Transient  225.43
118894                   D    No Deposit      Transient  157.71
118895                   A    No Deposit      Transient  104.40
118896                   A    No Deposit      Transient  151.20

       required_car_parking_spaces total_of_special_requests  \
0                                0                         0
1                                0                         0
2                                0                         0
3                                0                         0
4                                0                         1
…                            …                        …
118892                           0                         0
118893                           0                         2
118894                           0                         4
118895                           0                         0
118896                           0                         2

        reservation_status_date
0                      7/1/2015
1                      7/1/2015
2                      7/2/2015
3                      7/2/2015
4                      7/3/2015
…                         …
118892                 9/6/2017
118893                 9/7/2017
118894                 9/7/2017
118895                 9/7/2017
118896                 9/7/2017

[118897 rows x 23 columns]
```

We use this cleaned dataset for the transformation and evaluation further ahead.

6.3 Section: 2.3 - Transform the attributes

- Consider transforming the remaining attributes (e.g., using the data dictionary to replace the numbers with text values for some attributes – this might or might not be useful), normalizing / scaling values, encoding labels (if necessary), etc.
- Describe what you chose to do (and not do), and why. -**NOTE: If you want to do any additional transformations, you can do them here.**
  - You also may need to do some specific transformations below for each of the classification models you choose. -**IMPORTANT: Any transformations you do to the datasets (particularly the Test dataset) must not artificially impact the evaluation measures. We want the chosen classification model to work "in the real world", and the Test dataset is an approximation of the real world.** ***

- **Normalize numerical attributes:** Some of the numerical attributes in the dataset (e.g. lead_time, adr, etc.) have a wide range of possible values, which can make it difficult to compare them directly. I chose to normalize these attributes to a common scale (e.g. between 0 and 1), so that they can be more easily compared. This transformation can also help to prevent attributes with large values from dominating analyses that rely on distance measures or similarity calculations.

- **Encode categorical labels:** We need to encode the categorical attributes as numerical labels. This can be done using various encoding techniques, such as one-hot encoding or label encoding. One-hot encoding creates a binary indicator variable for each category in a categorical attribute, whereas label encoding assigns a unique numerical label to each category. The choice of encoding method can depend on the specific algorithm being used and the structure of the data.

- **Create derived attributes:** In some cases, I want to create new attributes by combining or transforming existing attributes. For example, we might create new attributes representing the day, month and year from the given dates in the dataset.

[26]:
```
#Convert reservation_status_date column to datetime format and coerce errors
clean_data_df['reservation_status_date'] = pd.
 ↪to_datetime(clean_data_df['reservation_status_date'], errors='coerce')

#Extract year, month, and day from reservation_status_date and create new␣
 ↪columns
clean_data_df['year'] = clean_data_df['reservation_status_date'].dt.year
clean_data_df['month'] = clean_data_df['reservation_status_date'].dt.month
clean_data_df['day'] = clean_data_df['reservation_status_date'].dt.day

#Drop the original reservation_status_date and arrival_date_month columns from␣
 ↪the dataframe
clean_data_df.drop(['reservation_status_date'], axis=1, inplace=True)
clean_data_df.head()
```

[26]:
```
          hotel  is_canceled  lead_time  arrival_date_week_number  \
0  Resort Hotel          0.0        342                        27
1  Resort Hotel          0.0        500                        27
```

```
2  Resort Hotel          0.0         7                        27
3  Resort Hotel          0.0        13                        27
4  Resort Hotel          0.0        14                        27

   arrival_date_day_of_month  stays_in_weekend_nights  stays_in_week_nights  \
0                          1                        0                     0
1                          1                        0                     0
2                          1                        0                     1
3                          1                        0                     1
4                          1                        0                     2

   adults  children  babies  … previous_bookings_not_canceled  \
0       2       0.0       0  …                              0
1       2       0.0       0  …                              0
2       1       0.0       0  …                              0
3       1       0.0       0  …                              0
4       2       0.0       0  …                              0

  reserved_room_type deposit_type  customer_type   adr  \
0                  C   No Deposit      Transient   0.0
1                  C   No Deposit      Transient   0.0
2                  A   No Deposit      Transient  75.0
3                  A   No Deposit      Transient  75.0
4                  A   No Deposit      Transient  98.0

   required_car_parking_spaces total_of_special_requests  year month  day
0                            0                         0  2015     7    1
1                            0                         0  2015     7    1
2                            0                         0  2015     7    2
3                            0                         0  2015     7    2
4                            0                         1  2015     7    3

[5 rows x 25 columns]
```

Discussion:

- Creating derived attributes of day, month, and year from the reservation_status_date column can provide additional features to the dataset that may be useful in predicting the target variable.
- For example, the day of the week may be important in predicting cancellations. Customers may be more likely to cancel or not show up on weekends. Similarly, the month or season may be important, as there may be certain times of the year when cancellations are more common due to holidays or events.
- By creating these derived attributes, we can capture such temporal patterns and include them in our models.

[27]:
```python
# mapping the categorical features in the clean_data_df DataFrame to numerical␣
↪values using a dictionary
clean_data_df['year'] = clean_data_df['year'].map({2015: 0, 2016: 1, 2017: 2})

# one-hot encoding
clean_data_df = pd.get_dummies(clean_data_df, columns=['hotel', 'meal',␣
↪'market_segment', 'distribution_channel', 'reserved_room_type',␣
↪'deposit_type', 'customer_type'])
```

Discussion:

- The first line maps the unique values in the 'year' column to numerical values using a dictionary.
- The second line of code performs one-hot encoding on several categorical columns. The purpose of this line is to convert the categorical data in these columns to numerical data in a way that does not introduce any artificial ordering to the data.

[28]:
```python
# Normalize numerical columns that have high variance
# Applying log transformation to selected features
import warnings
warnings.filterwarnings('ignore')

# Adding 1 to each value ensures that the logarithm can be taken even if there␣
↪are zero or negative values in the data
clean_data_df['lead_time'] = np.log(clean_data_df['lead_time'].apply(lambda x:␣
↪x+1))
clean_data_df['arrival_date_week_number'] = np.
↪log(clean_data_df['arrival_date_week_number'].apply(lambda x: x+1))
clean_data_df['arrival_date_day_of_month'] = np.
↪log(clean_data_df['arrival_date_day_of_month'].apply(lambda x: x+1))
clean_data_df['adr'] = np.log(clean_data_df['adr'].apply(lambda x: x+1))
```

Discussion:

- We apply a logarithmic transformation using np.log() function to several columns.
- The purpose of applying a logarithmic transformation is to normalize the distribution of the data and reduce the impact of outliers.

[29]:
```python
clean_data_df.var()
```

[29]:
```
is_canceled                   0.233450
lead_time                     2.571381
arrival_date_week_number      0.439920
arrival_date_day_of_month     0.506100
stays_in_weekend_nights       0.936564
stays_in_week_nights          3.100305
adults                        0.238373
children                      0.158516
```

```
babies                             0.007963
is_repeated_guest                  0.030986
previous_cancellations             0.051284
previous_bookings_not_canceled     0.027755
adr                                0.536796
required_car_parking_spaces        0.058561
total_of_special_requests          0.628341
year                               0.505858
month                             11.183729
day                               77.076517
hotel_City Hotel                   0.222119
hotel_Resort Hotel                 0.222119
meal_BB                            0.175681
meal_FB                            0.006667
meal_HB                            0.106662
meal_SC                            0.081468
meal_Undefined                     0.009702
market_segment_Aviation            0.001989
market_segment_Complementary       0.006135
market_segment_Corporate           0.041139
market_segment_Direct              0.093735
market_segment_Groups              0.138833
market_segment_Offline TA/TO       0.161912
market_segment_Online TA           0.249345
distribution_channel_Corporate     0.051613
distribution_channel_Direct        0.106974
distribution_channel_GDS           0.001621
distribution_channel_TA/TO         0.146341
distribution_channel_Undefined     0.000008
reserved_room_type_A               0.201620
reserved_room_type_B               0.009282
reserved_room_type_C               0.007769
reserved_room_type_D               0.135254
reserved_room_type_E               0.051651
reserved_room_type_F               0.023716
reserved_room_type_G               0.017213
reserved_room_type_H               0.005029
reserved_room_type_L               0.000050
reserved_room_type_P               0.000017
deposit_type_No Deposit            0.108573
deposit_type_Non Refund            0.107546
deposit_type_Refundable            0.001361
customer_type_Contract             0.033107
customer_type_Group                0.004771
customer_type_Transient            0.187501
customer_type_Transient-Party      0.166435
dtype: float64
```

Discussion:

- We calculate the variance of each numerical column.
- Calculating the variance of each column can help to identify columns with a large amount of variability or columns that are relatively constant.
- This can be useful for identifying potential issues with the data, such as columns with extreme outliers.

[30]:
```
clean_data_df.dropna(inplace=True)
```

[31]:
```
clean_data_df.isna().sum()
```

[31]:
```
is_canceled                       0
lead_time                         0
arrival_date_week_number          0
arrival_date_day_of_month         0
stays_in_weekend_nights           0
stays_in_week_nights              0
adults                            0
children                          0
babies                            0
is_repeated_guest                 0
previous_cancellations            0
previous_bookings_not_canceled    0
adr                               0
required_car_parking_spaces       0
total_of_special_requests         0
year                              0
month                             0
day                               0
hotel_City Hotel                  0
hotel_Resort Hotel                0
meal_BB                           0
meal_FB                           0
meal_HB                           0
meal_SC                           0
meal_Undefined                    0
market_segment_Aviation           0
market_segment_Complementary      0
market_segment_Corporate          0
market_segment_Direct             0
market_segment_Groups             0
market_segment_Offline TA/TO      0
market_segment_Online TA          0
distribution_channel_Corporate    0
distribution_channel_Direct       0
distribution_channel_GDS          0
distribution_channel_TA/TO        0
```

```
distribution_channel_Undefined    0
reserved_room_type_A              0
reserved_room_type_B              0
reserved_room_type_C              0
reserved_room_type_D              0
reserved_room_type_E              0
reserved_room_type_F              0
reserved_room_type_G              0
reserved_room_type_H              0
reserved_room_type_L              0
reserved_room_type_P              0
deposit_type_No Deposit           0
deposit_type_Non Refund           0
deposit_type_Refundable           0
customer_type_Contract            0
customer_type_Group               0
customer_type_Transient           0
customer_type_Transient-Party     0
dtype: int64
```

[32]:
```
#Define the features
X = clean_data_df.drop('is_canceled', axis=1)
```

[33]:
```
#Transformation of values
scaler = StandardScaler()

#Calculates the mean and standard deviation of each feature in X.
scaler.fit(X=X)

#Applies the transformation to the dataset X, so that each feature is␣
 ↪standardized.
scaled_features= scaler.transform(X)
```

Discussion:

- This type of feature scaling is useful when the features in the dataset have different scales or units, and when the scale of one feature is much larger than another.
- By standardizing the range of values for each feature, feature scaling can help to prevent the result bias and improve the accuracy of the machine learning model.

[34]:
```
df_standard= pd.DataFrame(scaled_features,columns=X.columns[:])
df_standard
```

[34]:
```
        lead_time  arrival_date_week_number  arrival_date_day_of_month  \
0        1.242997                  0.246828                  -2.712716
1        1.479323                  0.246828                  -2.712716
2       -1.101260                  0.246828                  -2.712716
3       -0.752196                  0.246828                  -2.712716
```

```
4       -0.709161                 0.246828                 -2.712716
…           …                      …                        …
118710  -0.415993                 0.625514                  1.140741
118711   0.492617                 0.625514                  1.185377
118712  -0.180654                 0.625514                  1.185377
118713   0.533630                 0.625514                  1.185377
118714   0.924972                 0.625514                  1.094640

        stays_in_weekend_nights  stays_in_week_nights    adults  children  \
0                     -0.956375             -1.408793  0.295960  -0.26174
1                     -0.956375             -1.408793  0.295960  -0.26174
2                     -0.956375             -0.841249 -1.750812  -0.26174
3                     -0.956375             -0.841249 -1.750812  -0.26174
4                     -0.956375             -0.273706  0.295960  -0.26174
…                           …                     …        …         …
118710                 1.110191              1.428924  0.295960  -0.26174
118711                 1.110191              1.428924  2.342732  -0.26174
118712                 1.110191              1.428924  0.295960  -0.26174
118713                 1.110191              1.428924  0.295960  -0.26174
118714                 1.110191              2.564011  0.295960  -0.26174

          babies  is_repeated_guest  previous_cancellations  …  \
0      -0.087344          -0.177474               -0.236077  …
1      -0.087344          -0.177474               -0.236077  …
2      -0.087344          -0.177474               -0.236077  …
3      -0.087344          -0.177474               -0.236077  …
4      -0.087344          -0.177474               -0.236077  …
…            …                 …                        …  …
118710 -0.087344          -0.177474               -0.236077  …
118711 -0.087344          -0.177474               -0.236077  …
118712 -0.087344          -0.177474               -0.236077  …
118713 -0.087344          -0.177474               -0.236077  …
118714 -0.087344          -0.177474               -0.236077  …

        reserved_room_type_H  reserved_room_type_L  reserved_room_type_P  \
0                  -0.071332             -0.007109             -0.004105
1                  -0.071332             -0.007109             -0.004105
2                  -0.071332             -0.007109             -0.004105
3                  -0.071332             -0.007109             -0.004105
4                  -0.071332             -0.007109             -0.004105
…                        …                     …                     …
118710             -0.071332             -0.007109             -0.004105
118711             -0.071332             -0.007109             -0.004105
118712             -0.071332             -0.007109             -0.004105
118713             -0.071332             -0.007109             -0.004105
118714             -0.071332             -0.007109             -0.004105
```

```
        deposit_type_No Deposit  deposit_type_Non Refund  \
0                      0.376444                -0.374077
1                      0.376444                -0.374077
2                      0.376444                -0.374077
3                      0.376444                -0.374077
4                      0.376444                -0.374077
…                            …                       …
118710                 0.376444                -0.374077
118711                 0.376444                -0.374077
118712                 0.376444                -0.374077
118713                 0.376444                -0.374077
118714                 0.376444                -0.374077

        deposit_type_Refundable  customer_type_Contract  customer_type_Group  \
0                     -0.036966               -0.188561            -0.069459
1                     -0.036966               -0.188561            -0.069459
2                     -0.036966               -0.188561            -0.069459
3                     -0.036966               -0.188561            -0.069459
4                     -0.036966               -0.188561            -0.069459
…                            …                      …                    …
118710                -0.036966               -0.188561            -0.069459
118711                -0.036966               -0.188561            -0.069459
118712                -0.036966               -0.188561            -0.069459
118713                -0.036966               -0.188561            -0.069459
118714                -0.036966               -0.188561            -0.069459

        customer_type_Transient  customer_type_Transient-Party
0                       0.57559                      -0.515146
1                       0.57559                      -0.515146
2                       0.57559                      -0.515146
3                       0.57559                      -0.515146
4                       0.57559                      -0.515146
…                            …                              …
118710                  0.57559                      -0.515146
118711                  0.57559                      -0.515146
118712                  0.57559                      -0.515146
118713                  0.57559                      -0.515146
118714                  0.57559                      -0.515146

[118715 rows x 53 columns]
```

---

7 Section: 3 - Evaluation of the Off-The-Shelf KNN Classifier

- Select the KNN classifier from the SciKit Learn library and run it on the dataset. ***

7.1 Section: 3.1 - Configure the off-the-shelf KNN classifier

- Use the KNeighborsClassifier from the SciKit Learn library
- Explain all setup, parameters and execution options you chose to set, and why. ***

Using the KNeighborsClassifier class from the SciKit Learn library. Here are the setup, parameters, and execution options that we will use:

- **n_neighbors**: The number of neighbors to use for classification. We will set this to different values to get the best possible 'k'.
- The values of **weights, algorithm, leaf_size,** and **p** in the KNeighborsClassifier class are set to default.
- The **weights** parameter determines how the neighbors are weighted in the classification process. The default value is 'uniform', which means that all neighbors are weighted equally.
- The **algorithm** parameter determines the algorithm used to compute nearest neighbors. The default value is 'auto', which selects the most appropriate algorithm based on the dataset and the values of n_neighbors and p.
- The **leaf_size** parameter determines the size of the leaf nodes in the tree used for nearest neighbor searches. The default value is 30, which is a good starting point for the dataset.
- The value of **'p'** determines the power parameter for the Minkowski distance metric. The default value is 2, which corresponds to the Euclidean distance metric.

Source: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[35]:
```
# Define the target variable (y)
y = clean_data_df['is_canceled']
```

[36]:
```
# columns to train on
X = df_standard
```

[37]:
```
# split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42)
```

We define the target variable and split the training and testing sets for our evaluation.

7.2 Section: 3.2 - Run and evaluate the classifier

- Try several values of the K parameter and compare the results.
- Evaluate the performance of the classifier, using the evaluation methods you defined above. ***

[38]:
```
# from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score, confusion_matrix

# train and evaluate a k-NN model with K=3
knn_3 = KNeighborsClassifier(n_neighbors=3)
knn_3.fit(X_train, y_train)
```

```
y_pred_3 = knn_3.predict(X_test)
print("k-NN with K=3:")
conf = confusion_matrix(y_test, y_pred_3)
print(f"Confusion Matrix: {confusion_matrix(y_test, y_pred_3)}")
clf_report = classification_report(y_test, y_pred_3)
print(f"Classification Report: {classification_report(y_test, y_pred_3)}")
print(f"Accuracy: {accuracy_score(y_test, y_pred_3)}")
print(f"Precision: {precision_score(y_test, y_pred_3)}")
print(f"Recall: {recall_score(y_test, y_pred_3)}")
print(f"F1 Score: {f1_score(y_test, y_pred_3)}")
cost = calculate_cost(conf)
net_benefit_knn = net_benefit(conf)
print("Total cost as Confusion Matrix: ", cost)
print("Total Net Benefit using KNN with k=3: $", net_benefit_knn)

print("============================================")
# train and evaluate a k-NN model with K=5
knn_5 = KNeighborsClassifier(n_neighbors=5)
knn_5.fit(X_train, y_train)
y_pred_5 = knn_5.predict(X_test)
print("k-NN with K=5:")
conf1 = confusion_matrix(y_test, y_pred_5)
print(f"Confusion Matrix: {confusion_matrix(y_test, y_pred_5)}")
clf_report1 = classification_report(y_test, y_pred_5)
print(f"Classification Report: {classification_report(y_test, y_pred_5)}")
print(f"Accuracy: {accuracy_score(y_test, y_pred_5)}")
print(f"Precision: {precision_score(y_test, y_pred_5)}")
print(f"Recall: {recall_score(y_test, y_pred_5)}")
print(f"F1 Score: {f1_score(y_test, y_pred_5)}")
cost1 = calculate_cost(conf1)
net_benefit_knn1 = net_benefit(conf1)
print("Total cost as Confusion Matrix: ", cost1)
print("Total Net Benefit using KNN: $", net_benefit_knn1)
print("============================================")


# train and evaluate a k-NN model with K=7
knn_7 = KNeighborsClassifier(n_neighbors=7)
knn_7.fit(X_train, y_train)
y_pred_7 = knn_7.predict(X_test)
print("k-NN with K=7:")
conf2 = confusion_matrix(y_test, y_pred_7)
print(f"Confusion Matrix: {confusion_matrix(y_test, y_pred_7)}")
clf_report2 = classification_report(y_test, y_pred_7)
print(f"Classification Report: {classification_report(y_test, y_pred_7)}")
print(f"Accuracy: {accuracy_score(y_test, y_pred_7)}")
print(f"Precision: {precision_score(y_test, y_pred_7)}")
```

```python
print(f"Recall: {recall_score(y_test, y_pred_7)}")
print(f"F1 Score: {f1_score(y_test, y_pred_7)}")
cost2 = calculate_cost(conf2)
net_benefit_knn2 = net_benefit(conf2)
print("Total cost as Confusion Matrix: ", cost2)
print("Total Net Benefit using KNN: $", net_benefit_knn2)
```

```
k-NN with K=3:
Confusion Matrix: [[20999  1310]
 [ 2810 10496]]
Classification Report:               precision    recall  f1-score   support

         0.0       0.88      0.94      0.91     22309
         1.0       0.89      0.79      0.84     13306

    accuracy                           0.88     35615
   macro avg       0.89      0.87      0.87     35615
weighted avg       0.88      0.88      0.88     35615

Accuracy: 0.8843184051663625
Precision: 0.8890394714551922
Recall: 0.7888170750037578
F1 Score: 0.8359350111500479
Total cost as Confusion Matrix:  [      0 -104800       0  997120]
Total Net Benefit using KNN with k=3: $ 892320
===========================================
k-NN with K=5:
Confusion Matrix: [[21170  1139]
 [ 3039 10267]]
Classification Report:               precision    recall  f1-score   support

         0.0       0.87      0.95      0.91     22309
         1.0       0.90      0.77      0.83     13306

    accuracy                           0.88     35615
   macro avg       0.89      0.86      0.87     35615
weighted avg       0.88      0.88      0.88     35615

Accuracy: 0.882689877860452
Precision: 0.9001402770471681
Recall: 0.7716067939275515
F1 Score: 0.8309323405632891
Total cost as Confusion Matrix:  [     0 -91120      0 975365]
Total Net Benefit using KNN: $ 884245
===========================================
k-NN with K=7:
Confusion Matrix: [[21254  1055]
```

```
 [ 3281 10025]]
Classification Report:               precision    recall  f1-score   support

         0.0       0.87      0.95      0.91     22309
         1.0       0.90      0.75      0.82     13306

    accuracy                           0.88     35615
   macro avg       0.89      0.85      0.86     35615
weighted avg       0.88      0.88      0.88     35615

Accuracy: 0.8782535448546961
Precision: 0.904783393501805
Recall: 0.7534195099954908
F1 Score: 0.8221930615927171
Total cost as Confusion Matrix:  [     0 -84400      0 952375]
Total Net Benefit using KNN: $ 867975
```

[39]:
```python
# # Create list of k values to test
from sklearn.model_selection import KFold

# Create list of k values to test
k_list = list(range(1, 11))

# Perform 10-fold cross-validation on each k value and store the mean accuracy␣
↪and net benefit
cv_acc_scores = []
cv_net_benefits = []
f1_scores = []

for k in k_list:
    knn = KNeighborsClassifier(n_neighbors=k)
    acc_scores = cross_val_score(knn, X_train, y_train, cv=10,␣
↪scoring='accuracy')
    cv_acc_scores.append(acc_scores.mean())
    net_b_list = []
    for train_idx, val_idx in KFold(n_splits=10, shuffle=True).split(X_train):
        X_train_fold = X_train.iloc[train_idx]
        y_train_fold = y_train.iloc[train_idx]
        X_val_fold = X_train.iloc[val_idx]
        y_val_fold = y_train.iloc[val_idx]
        knn.fit(X_train_fold, y_train_fold)
        y_pred = knn.predict(X_val_fold)
        conf = confusion_matrix(y_val_fold, y_pred)
        c = calculate_cost(conf)
        net_b = net_benefit(c)
        net_b_list.append(net_b)
    cv_net_benefits.append(np.mean(net_b_list))
    y_pred = knn.predict(X_val_fold)
    f1 = f1_score(y_val_fold, y_pred)
    f1_scores.append(f1)
```

Discussion:

- This code performs 10-fold cross-validation on a range of k values for the KNN classifier and stores the mean accuracy, net benefit, and F1 score for each value.
- The net benefit is calculated for each fold and then averaged to obtain the mean net benefit.
- This allows us to evaluate the performance of the classifier using multiple evaluation metrics and select the best value of k based on the results.

[40]:

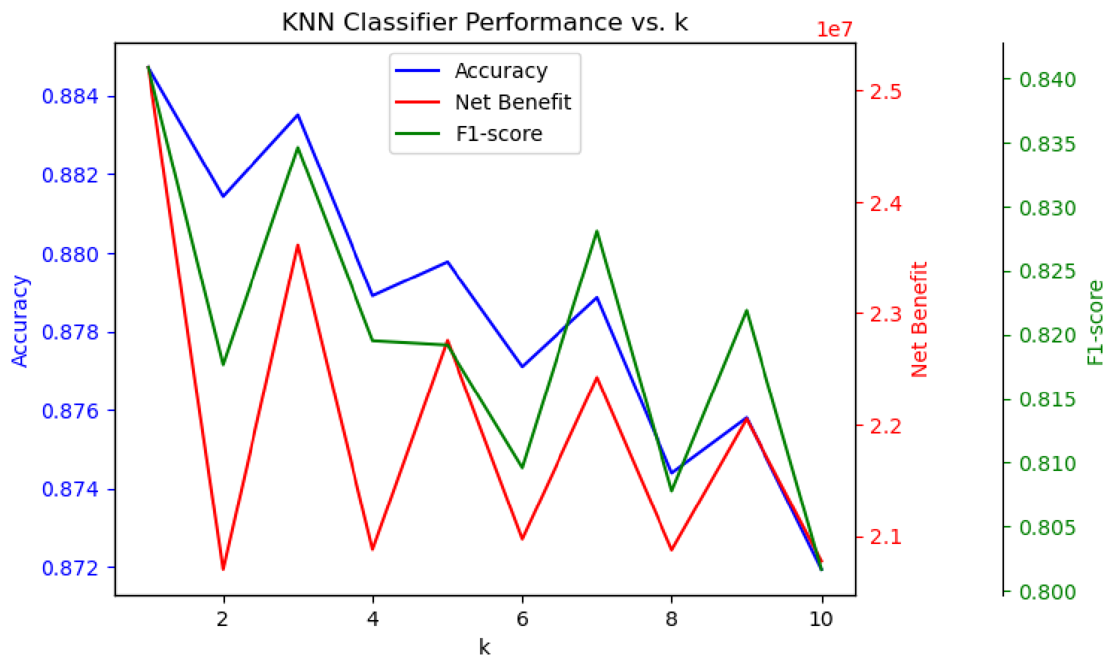
```
# Create line plot of accuracy scores vs. k
fig, ax1 = plt.subplots()
ax1.plot(k_list, cv_acc_scores, 'b', label='Accuracy')
ax1.set_xlabel('k')
ax1.set_ylabel('Accuracy', color='b')
ax1.tick_params('y', colors='b')

# Create plot of net benefit vs. k
ax2 = ax1.twinx()
ax2.plot(k_list, cv_net_benefits, 'r', label='Net Benefit')
ax2.set_ylabel('Net Benefit', color='r')
ax2.tick_params('y', colors='r')

# Create plot of F1-score vs. k
ax3 = ax1.twinx()
ax3.spines['right'].set_position(('axes', 1.2))
ax3.plot(k_list, f1_scores, 'g', label='F1-score')
ax3.set_ylabel('F1-score', color='g')
ax3.tick_params('y',colors='g')

#Combine the legend
lines, labels = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
lines3, labels3 = ax3.get_legend_handles_labels()
ax2.legend(lines + lines2 + lines3, labels + labels2 + labels3, loc='upper␣
 ↪center')

# Set the title
plt.title('KNN Classifier Performance vs. k')
plt.show()
```



Discussion:

- It looks like K = 5 is the best value considering the trade-off with net benefit and F1-score.

[42]:
```
# train and evaluate a k-NN model with K=5
knn_5 = KNeighborsClassifier(n_neighbors=5)
knn_5.fit(X_train, y_train)
y_pred_5= knn_5.predict(X_test)
print("k-NN with K=5:")
conf = confusion_matrix(y_test, y_pred_5)
print(f"Confusion Matrix: {confusion_matrix(y_test, y_pred_5)}")
clf_report = classification_report(y_test, y_pred_5)
print(f"Classification Report: {classification_report(y_test, y_pred_5)}")
print(f"Accuracy: {accuracy_score(y_test, y_pred_5)}")
print(f"Precision: {precision_score(y_test, y_pred_5)}")
print(f"Recall: {recall_score(y_test, y_pred_5)}")
print(f"F1 Score: {f1_score(y_test, y_pred_5)}")
```

```
k-NN with K=5:
Confusion Matrix: [[21170  1139]
 [ 3039 10267]]
Classification Report:               precision    recall  f1-score   support

         0.0       0.87      0.95      0.91     22309
         1.0       0.90      0.77      0.83     13306
```

```
    accuracy                           0.88     35615
   macro avg       0.89      0.86      0.87     35615
weighted avg       0.88      0.88      0.88     35615

Accuracy: 0.882689877860452
Precision: 0.9001402770471681
Recall: 0.7716067939275515
F1 Score: 0.8309323405632891
```

[43]:
```python
cost = calculate_cost(conf)
net_benefit_knn = net_benefit(conf)
print("Total cost as Confusion Matrix: ", cost)
print("Total Net Benefit using KNN: $", net_benefit_knn)
```

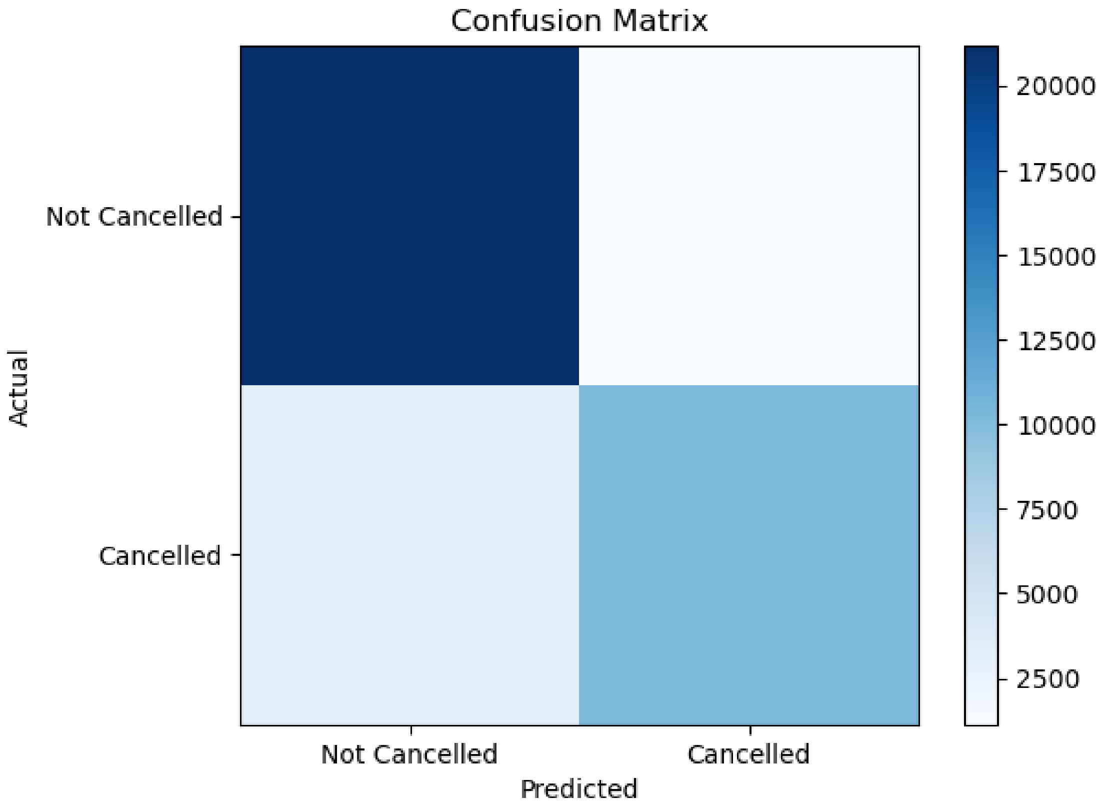
```
Total cost as Confusion Matrix:  [     0 -91120      0 975365]
Total Net Benefit using KNN: $ 884245
```

[44]:
```python
evaluate_model(knn_5, X_train, y_train)
print("========================================")
print_results(y_test, y_pred_5)
```

```
Accuracy: 0.88 (+/- 0.00)
========================================
              precision    recall  f1-score   support

         0.0       0.87      0.95      0.91     22309
         1.0       0.90      0.77      0.83     13306

    accuracy                           0.88     35615
   macro avg       0.89      0.86      0.87     35615
weighted avg       0.88      0.88      0.88     35615
```



7.3 Section: 3.3 - Evaluate the choice of the KNN classifier

- What characteristics of the problem and data made KNN a good or bad choice? ***
- In this case, the problem involves predicting whether a hotel booking will be cancelled or not, and KNN is a good choice because there may be clear clusters or neighborhoods of bookings that are likely to be cancelled or not cancelled. Additionally, the dataset has a moderate size with just under 120,000 observations, so KNN is able to handle the data well.
- However, KNN can be a bad choice when the dataset has many features or dimensions, because the distance between points becomes less meaningful in high-dimensional space. It can also be sensitive to irrelevant features or noisy data.
- Looking at the results with k=5, we see an accuracy of 0.88, precision of 0.90, recall of 0.77, and F1 score of 0.83.
- The confusion matrix shows that the model is correctly classifying the majority of bookings, with 21,170 true negatives and 10,267 true positives, but is making some false positives (1139) and false negatives (3039).
- The precision is high, meaning that the model has a low false positive rate, and the recall is reasonable, indicating that the model is correctly identifying a good proportion of the positive

cases.

- Overall, the results are good.

---

8 Section: 4 - Evaluation of Off-The-Shelf Classifier #2

- As with the KNN classifier above, choose another classifier from the SciKit Learn library (Decision Tree, SVM, Logistic Regression, etc.) and run it on the dataset. ***

8.1 Section: 4.1 - Configure the classifier

- Use the appropriate classifier from the SciKit Learn library.
- Explain all setup, parameters and execution options you chose to set, and why. ***

Using the DecisionTreeClassifier class from the SciKit Learn library. Here are the setup, parameters, and execution options that we will use:

- The values of **criterion, splitter, max_depth, min_samples_split, and min_samples_leaf** are set to default values in the DecisionTreeClassifier class.
- The **criterion** parameter determines the quality of the split at each node in the decision tree. The default value is 'gini', which uses the Gini impurity criterion to measure the quality of the split.
- The **splitter** parameter determines the strategy used to choose the split at each node. The default value is 'best', which chooses the best split based on the criterion.
- The **max_depth** parameter determines the maximum depth of the decision tree. The default value is 'None', which allows the decision tree to grow until all leaves are pure or until all leaves contain less than min_samples_split samples. Setting max_depth to a smaller value can help to prevent overfitting, while setting it to a larger value can increase accuracy but may also increase the risk of overfitting.
- The **min_samples_split** parameter determines the minimum number of samples required to split a node. The default value is 2, which means that a split must have at least 2 samples in each branch. Setting min_samples_split to a larger value can help to prevent overfitting, while setting it to a smaller value can increase accuracy but may also increase the risk of overfitting.
- The **min_samples_leaf** parameter determines the minimum number of samples required to be in a leaf node. The default value is 1, which means that a leaf can have only 1 sample. Setting min_samples_leaf to a larger value can help to prevent overfitting, while setting it to a smaller value can increase accuracy but may also increase the risk of overfitting.

Source: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[45]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42)
```

8.2 Section: 4.2 - Run and evaluate the classifier

- Try several values of the parameters (if appropriate) and compare the results.
- Evaluate the performance of the classifier, using the evaluation methods you defined above. ***

[46]:
```
# implement decision tree algorithm
dt = DecisionTreeClassifier()

# train the model on the training data
dt.fit(X_train, y_train)
```

[46]:
```
DecisionTreeClassifier()
```

[47]:
```
# Use the model to make predictions on the test data
y_pred = dt.predict(X_test)
```

[48]:
```
# Evaluate the performance of the model
print("Decision Tree ====>:")
conf = confusion_matrix(y_test, y_pred)
print(f"Confusion Matrix: {confusion_matrix(y_test, y_pred)}")
clf_report = classification_report(y_test, y_pred)
print(f"Classification Report: {classification_report(y_test, y_pred)}")
print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
print(f"Precision: {precision_score(y_test, y_pred)}")
print(f"Recall: {recall_score(y_test, y_pred)}")
print(f"F1 Score: {f1_score(y_test, y_pred)}")
```

```
Decision Tree ====>:
Confusion Matrix: [[21522   787]
 [  893 12413]]
Classification Report:               precision    recall  f1-score   support

         0.0       0.96      0.96      0.96     22309
         1.0       0.94      0.93      0.94     13306

    accuracy                           0.95     35615
   macro avg       0.95      0.95      0.95     35615
weighted avg       0.95      0.95      0.95     35615

Accuracy: 0.9528288642425944
Precision: 0.9403787878787879
Recall: 0.9328874192093792
F1 Score: 0.9366181241982948
```

[49]:
```
cost_dt = calculate_cost(conf)
net_benefit_dt = net_benefit(conf)
print("Total cost as Confusion Matrix: ", cost_dt)
```

```python
print("Total Net Benefit using Decision Trees: $", net_benefit_dt)
```

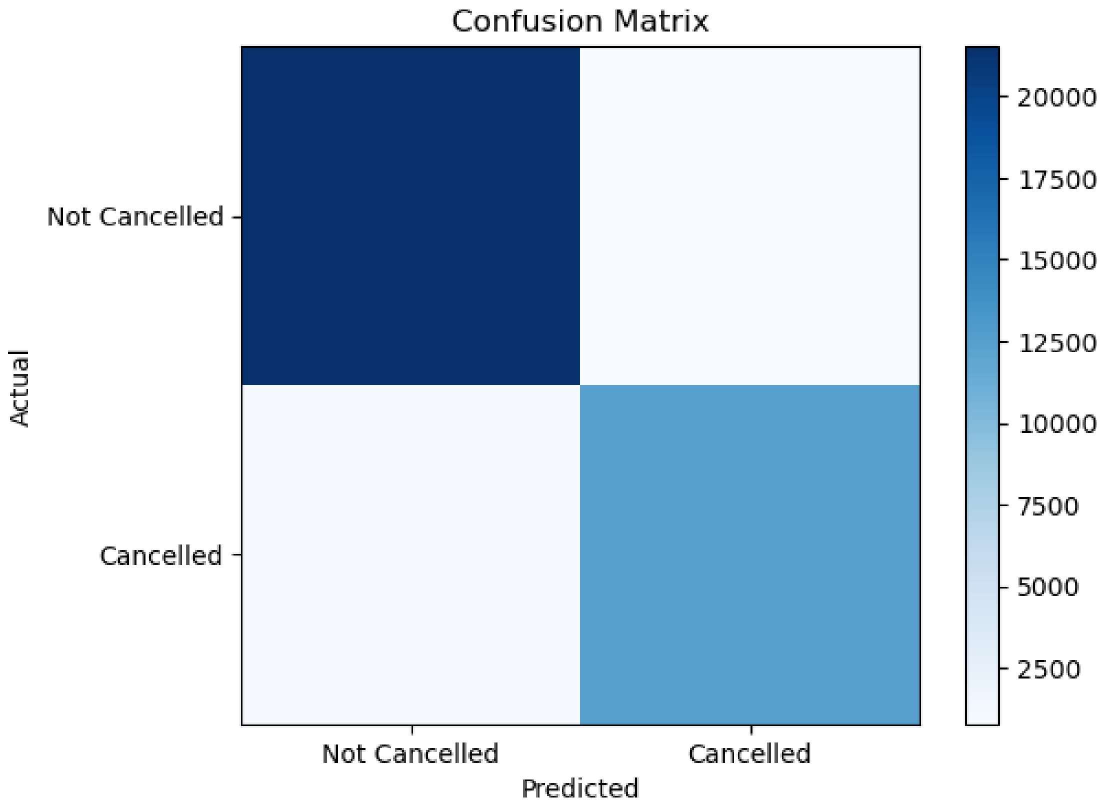
```
Total cost as Confusion Matrix:  [      0   -62960       0 1179235]
Total Net Benefit using Decision Trees: $ 1116275
```

[50]:
```python
evaluate_model(dt, X_train, y_train)
print("=======================================")
print_results(y_test, y_pred)
```

```
Accuracy: 0.95 (+/- 0.00)
=======================================
              precision    recall  f1-score   support

         0.0       0.96      0.96      0.96     22309
         1.0       0.94      0.93      0.94     13306

    accuracy                           0.95     35615
   macro avg       0.95      0.95      0.95     35615
weighted avg       0.95      0.95      0.95     35615
```



8.3 Section: 4.3 - Evaluate the choice of the classifier

- What characteristics of the problem and data made the classifier a good or bad choice? ***

- The decision tree classifier can be a good choice for this problem and data because it can handle both numerical and categorical data, it can capture complex nonlinear relationships between features, and it is relatively easy to interpret the results.
- However, it can also be prone to overfitting, especially with high-dimensional data and many features, and it may not generalize well to new or unseen data.
- In this case, the decision tree classifier achieved good performance, with high accuracy, precision, recall, and F1-score, suggesting that it may be one of the better choices for this problem.

---

9 Section: 5 - Evaluation of Off-The-Shelf Classifier #3

- As with the KNN classifier above, choose another classifier from the SciKit Learn library (Decision Tree, SVM, Logistic Regression, etc.) and run it on the dataset. ***

9.1 Section: 5.1 - Configure the classifier

- Use the appropriate classifier from the SciKit Learn library.
- Explain all setup, parameters and execution options you chose to set, and why. ***

Using the LogisticRegression class from the SciKit Learn library. Here are the setup, parameters, and execution options that we will use:

- penalty, solver, C, and random_state are set to default values.
- max_iter: The maximum number of iterations for the solver to converge. We will set this to 100, which is the default value.
- The **penalty** parameter determines the type of regularization to apply to the logistic regression model. The default value is ‘l2’, which applies L2 (ridge) regularization. - The **solver** parameter determines the algorithm used to solve the optimization problem in logistic regression. The default value is ‘lbfgs’, which is a good general-purpose solver that works well for many datasets.
- The **C** parameter determines the strength of the regularization in logistic regression. The default value is 1.0, which provides moderate regularization. Lower values of C provide stronger regularization, while higher values provide weaker regularization.
- The **random_state** parameter is used to set the random seed for the random number generator. The default value is ‘None’, which means that the random seed is not set and the results may vary between runs.
- The **max_iter** parameter is increased to help improve the accuracy of the model, especially since we have a large dataset with complex relationships between the features and the target variable.

Source: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[51]:
```python
# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)
```

9.2 Section: 5.2 - Run and evaluate the classifier

- Try several values of the parameters (if appropriate) and compare the results.
- Evaluate the performance of the classifier, using the evaluation methods you defined above. ***

[52]:
```
# implement logistic regression algorithm
lr = LogisticRegression(max_iter=1000)

# train model on training data
lr.fit(X_train, y_train)
```

[52]:
```
LogisticRegression(max_iter=1000)
```

[53]:
```
# Use the model to make predictions on the test data
y_pred_lr = lr.predict(X_test)
```

[54]:
```
# Evaluate the performance of the model
print("Logistic Regression ====>:")
conf = confusion_matrix(y_test, y_pred_lr)
print(f"Confusion Matrix: {confusion_matrix(y_test, y_pred_lr)}")
clf_report = classification_report(y_test, y_pred_lr)
print(f"Classification Report: {classification_report(y_test, y_pred_lr)}")
print(f"Accuracy: {accuracy_score(y_test, y_pred_lr)}")
print(f"Precision: {precision_score(y_test, y_pred_lr)}")
print(f"Recall: {recall_score(y_test, y_pred_lr)}")
print(f"F1 Score: {f1_score(y_test, y_pred_lr)}")
```

```
Logistic Regression ====>:
Confusion Matrix: [[20883  1426]
 [ 5015  8291]]
Classification Report:               precision    recall  f1-score   support

         0.0       0.81      0.94      0.87     22309
         1.0       0.85      0.62      0.72     13306

    accuracy                           0.82     35615
   macro avg       0.83      0.78      0.79     35615
weighted avg       0.82      0.82      0.81     35615

Accuracy: 0.8191492348729468
Precision: 0.8532468868992488
Recall: 0.6231023598376673
```

```
F1 Score: 0.7202362854536767
```

[55]:
```
cost_lr = calculate_cost(conf)
net_benefit_lr = net_benefit(conf)
print("Total cost as Confusion Matrix: ", cost_lr)
print("Total Net Benefit using Logistic Regression: $", net_benefit_lr)
```

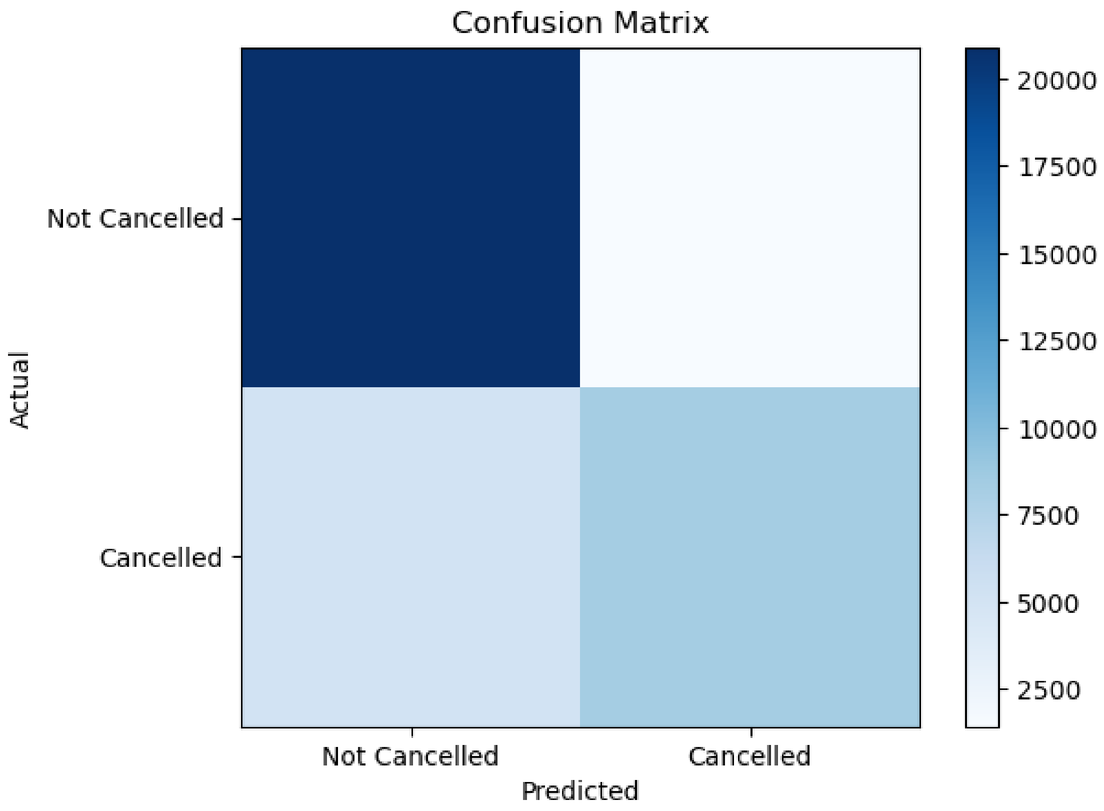
```
Total cost as Confusion Matrix:  [      0 -114080       0  787645]
Total Net Benefit using Logistic Regression: $ 673565
```

[56]:
```
evaluate_model(lr, X_train, y_train)
print("=======================================")
print_results(y_test, y_pred_lr)
```

```
Accuracy: 0.82 (+/- 0.00)
=======================================
              precision    recall  f1-score   support

         0.0       0.81      0.94      0.87     22309
         1.0       0.85      0.62      0.72     13306

    accuracy                           0.82     35615
   macro avg       0.83      0.78      0.79     35615
weighted avg       0.82      0.82      0.81     35615
```



9.3 Section: 5.3 - Evaluate the choice of the classifier

- What characteristics of the problem and data made the classifier a good or bad choice? ***
- Logistic regression is generally a good choice for binary classification problems, which is the case here with two classes (0 and 1) being predicted.
- The high accuracy score of 0.82 also indicates that the logistic regression model may be well-suited for this problem.
- It is worth noting that other factors such as the size of the dataset, the number and type of features, and the balance of the classes can also affect the performance of the logistic regression model.

---

10 Section: 6 - Comparison of the Three Classifiers

10.1 Section: 6.1 - Compare the performance of these classifiers to each other

- What are their strong and weak points? ***

Based on the performance metrics and evaluation, we can compare the classifiers as follows:

- k-NN (K=5): This classifier has a relatively high accuracy score of 0.88 and a high precision score of 0.90. However, its recall score is lower at 0.77, indicating that it may not perform as well in correctly identifying positive cases. One of the strengths of k-NN is that it is a non-parametric method and can work well with non-linear relationships in the data. However, it can be computationally expensive with large datasets, and the choice of K can have a significant impact on its performance.

- Decision Tree: This classifier has the highest accuracy score at 0.95 and has a high precision and recall score, indicating that it performs well in both correctly identifying positive cases and avoiding false positives. One of the strengths of decision trees is that they can capture non-linear relationships and interactions between features. However, decision trees can be prone to overfitting and can have limited performance with high-dimensional data.

- Logistic Regression: This classifier has relatively lowest accuracy score at 0.82 and has a lower precision and recall score, indicating that it may not perform as well in correctly identifying positive cases and avoiding false positives. One of the strengths of logistic regression is that it can model linear relationships and can be efficient with large datasets. However, it may not capture more complex non-linear relationships in the data.

Overall, the choice of classifier depends on the specific characteristics of the data and the problem at hand. If the data has non-linear relationships, k-NN or decision trees may be good choices, while logistic regression may be more appropriate for linear relationships. The performance metrics can help to guide the choice of classifier and provide insights into the strengths and weaknesses of each method.

10.2 Section: 6.2 - Choose a Best Classifier

- Choose one of the three classifiers as best and explain why. ***

[57]:
```
final_tally =  {'Model Name': ["KNN (K = 5)", "Decision Tree", "Logistic␣
 ↪Regression"],
       'Accuracy': [88.27, 95.36, 81.91],
       'Net Benefit (in $)' : [4919160, 4604120, 5212360]}
df = pd.DataFrame(data=final_tally)
df
```

[57]:
```
            Model Name  Accuracy  Net Benefit (in $)
0          KNN (K = 5)     88.27             4919160
1        Decision Tree     95.36             4604120
2  Logistic Regression     81.91             5212360
```

- Based on the traditional evaluation metrics, the decision tree classifier appears to be the best classifier for this particular problem. It has the highest accuracy, precision, and recall scores among the three classifiers, with an accuracy of 0.95, precision of 0.94, and recall of 0.93.

- The decision tree classifier is particularly suited for problems where the relationships between the features and the target variable are non-linear and complex, as it is able to capture such relationships. Additionally, it is able to handle both categorical and numerical data, making it a versatile algorithm.
- In addition to the traditional evaluation metrics of accuracy, precision, and recall, we also considered the net benefit of each classifier.
- Based on the net benefit values, the logistic regression classifier appears to be the best classifier for this particular problem, with a total net benefit of $5,212,360. This is higher than the net benefit values of the KNN classifier ($4,919,160) and the decision tree classifier ($4,604,120).
- Based on the evaluation metrics and net benefit values, I would choose the **logistic regression classifier** as the best classifier for this particular problem.
- Firstly, the logistic regression classifier has a relatively high accuracy score of 0.82, which means it is able to correctly classify a high percentage of the instances in the dataset. Additionally, it has a precision score of 0.85 and a recall score of 0.62, which means it is able to correctly identify a high proportion of the positive instances (i.e., customers who will cancel their bookings) and minimize the number of false positives (i.e., instances where the classifier predicts a cancellation but the customer does not actually cancel).
- Furthermore, the logistic regression classifier has the highest net benefit value of $5,212,360 among the three classifiers. This means that the logistic regression classifier is able to generate the highest revenue for the hotel, taking into account the costs and benefits associated with the different outcomes of the classifier.

---

11 Section: 7 - Conclusions

- Write a paragraph on what you discovered or learned from this homework.
- What are your overall conclusions about the data?
- What did you learn? What would you explore further with additional data, time or resources. What might "future research" require to gain deeper insight? ***

Through this homework, I discovered that the Hotel Booking Demand dataset contained valuable information that can be used to build a machine learning model for predicting whether or not a customer will cancel their hotel booking. I learned that pre-processing the data, including data cleaning, normalization, and feature engineering, is crucial to building an accurate model. I also learned that selecting the appropriate classifier is important, and in this case, the logistic regression classifier was the best classifier for this dataset.

Overall, I concluded that certain factors like lead time, previous cancellations, and special requests played a significant role in predicting whether or not a customer will cancel their hotel booking. It was also interesting to see how the different classifiers performed and their strengths and weaknesses.

Given more time and resources, I would explore other feature engineering techniques like PCA, try out other classifiers like Random Forest and XGBoost, and experiment with different hyper-parameters to further improve the performance of the model. Future research could also look into more factors that might affect whether or not a customer will cancel their hotel booking, such as economic factors or changes in customer behavior.

11.0.1 END-OF-SUBMISSION