

✓ **Project Name - IndiGo Airline Passenger Referral Prediction Analysis**

Contribution - Individual

Project Type - Classification

Github Link -

<https://github.com/codecat840/Airline-Passenger-Prediction-Analysis>

✓ **Project Summary**

Machine learning models are indispensable tools that allow for the classification of airlines based on different criteria. This document outlines the development and implementation of an airline classification machine learning model. In the dynamic and highly competitive airline industry, companies like IndiGo are continuously striving to enhance customer experience and build lasting loyalty. A critical component of this effort is understanding and predicting passenger referrals, which can significantly influence the airline's market position and reputation.

This insight enables IndiGo to:

- Develop a classification model to categorize airlines based on the likelihood of customers
- Identify specific areas needing improvement, whether it be in-flight comfort, customer service, or overall value for money.
- Understanding referral patterns and trends to follow marketing strategies that will leverage positive word-of-mouth and foster a strong brand reputation.
- Enable airlines to strategically utilize customer referral information for codeshare agreements, pricing strategies, and market analysis.
- Assess the model's capability to provide actionable insights for airlines to tailor services, improve customer satisfaction as well as enhance brand reputation.

General Guidelines

1. Well-structured, formatted, and commented code is required.
2. Exception Handling, Production Grade Code & Deployment Ready Code will be a plus.
Those students will be awarded some additional credits.

The additional credits will have advantages over other students during Star Student selection.

[Note: - Deployment Ready Code is defined as, the whole .ipynb notebook should be executed without a single error logged.]



3. Each and every logic should have proper comments.
4. You may add as many number of charts you want. Make Sure for each and every chart the following format should be answered.

Chart visualization code

Why did you pick the specific chart? What is/are the insight(s) found from the chart? Will the gained insights help creating a positive business impact? Are there any insights that lead to negative growth? Justify with specific reason. You have to create at least 20 logical & meaningful charts having important insights. [Hints : - Do the Vizualization in a structured way while following "UBM" Rule.

U - Univariate Analysis,

B - Bivariate Analysis (Numerical - Categorical, Numerical - Numerical, Categorical - Categorical)

M - Multivariate Analysis]

Let's Begin !

✓ 1. Know Your Data

```
!pip install category_encoders
```



Collecting category_encoders

Downloading category_encoders-2.8.1-py3-none-any.whl.metadata (7.9 kB)

Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.11/dist-packag

Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.11/dist-packag

Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.11/dist-package

Requirement already satisfied: scikit-learn>=1.6.0 in /usr/local/lib/python3.11/dist-

```
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/di
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (f
Downloading category_encoders-2.8.1-py3-none-any.whl (85 kB)
85.7/85.7 kB 2.0 MB/s eta 0:00:00
Installing collected packages: category_encoders
Successfully installed category_encoders-2.8.1
```

✓ Import Libraries

```
# Import Libraries
```

```
import pandas as pd
import numpy as np
import seaborn as sns
import datetime as dt
import missingno as msno
import matplotlib.pyplot as plt
from scipy import stats
from scipy.stats import chi2_contingency
from scipy.stats import f_oneway
from sklearn.preprocessing import LabelEncoder
import category_encoders as ce
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score, accuracy_score, precision_score, recall_score, f
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV, cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
```

✓ Dataset Loading

```
# Load the dataset
df = pd.read_excel("/content/data_airline_reviews.xlsx")
```

✓ Dataset First View

```
df.head()
```



	airline	overall	author	review_date	customer_review	aircraft	traveller_type
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	Turkish Airlines	7.0	Christopher Hackley	8th May 2019	âœ… Trip Verified London to Izmir via Istanb...	NaN	Business
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	Turkish Airlines	2.0	Adriana PISOI	7th May 2019	âœ… Trip Verified Istanbul to Bucharest. We ...	NaN	Family Leisure
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN

✓ Dataset Rows & Columns count

```
df.shape
```



```
(131895, 17)
```

✓ Dataset Information

```
df.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 131895 entries, 0 to 131894
Data columns (total 17 columns):
#   Column                Non-Null Count  Dtype
---  -
0   airline                65947 non-null  object
1   overall                64017 non-null  float64
2   author                 65947 non-null  object
3   review_date            65947 non-null  object
4   customer_review        65947 non-null  object
5   aircraft               19718 non-null  object
6   traveller_type         39755 non-null  object
7   cabin                  63303 non-null  object
8   route                  39726 non-null  object
9   date_flown             39633 non-null  object
10  seat_comfort            60681 non-null  float64
11  cabin_service           60715 non-null  float64
12  food_bev                52608 non-null  float64
```

```

13  entertainment    44193 non-null    float64
14  ground_service   39358 non-null    float64
15  value_for_money   63975 non-null    float64
16  recommended      64440 non-null    object
dtypes: float64(7), object(10)
memory usage: 17.1+ MB

```

✓ Duplicate Values

```
df.duplicated().sum()
```

```
np.int64(70711)
```

✓ Missing Values/Null Values

```
df.isnull().sum()
```

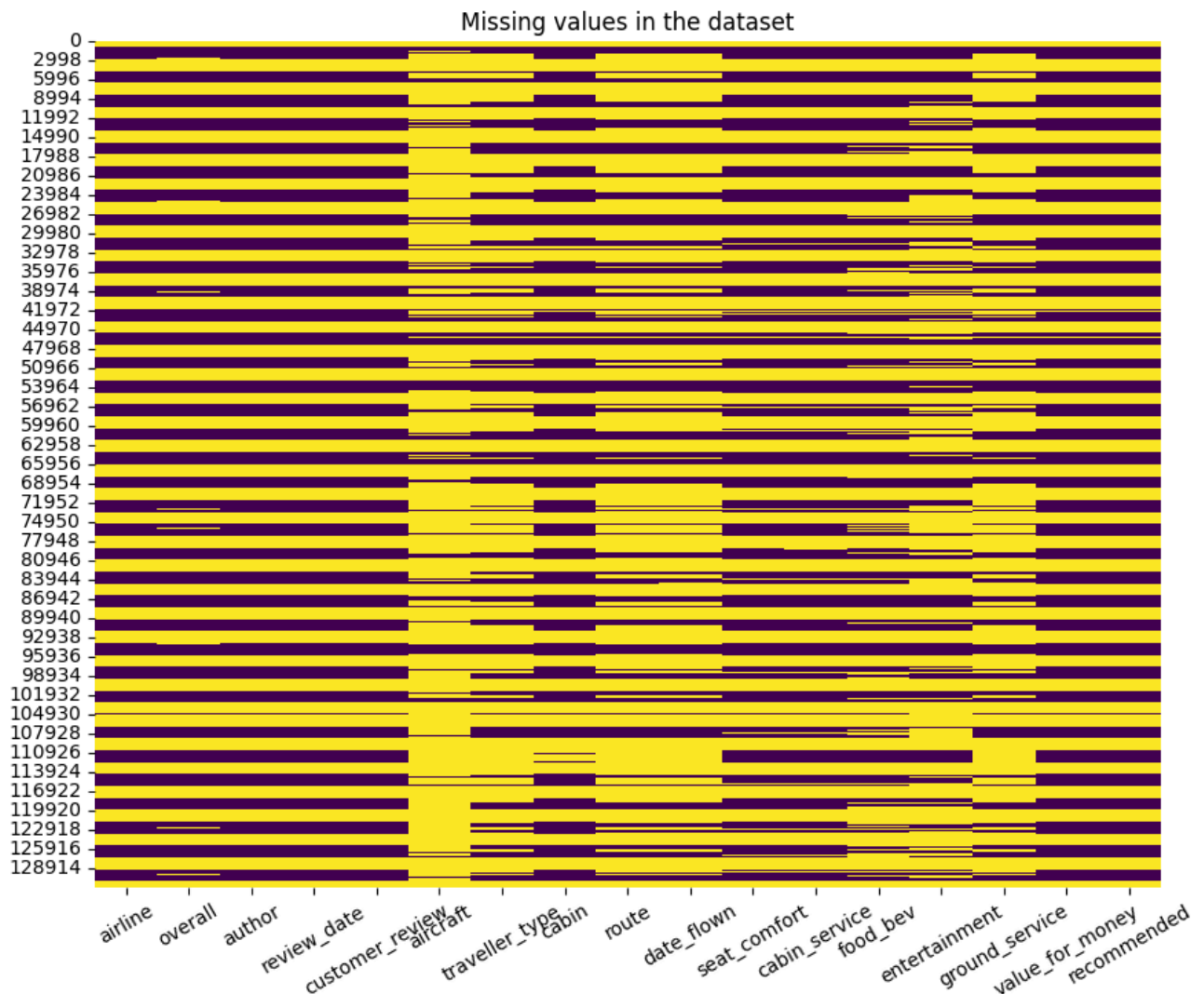
```

0
airline      65948
overall      67878
author       65948
review_date  65948
customer_review 65948
aircraft     112177
traveller_type 92140
cabin        68592
route        92169
date_flown   92262
seat_comfort 71214
cabin_service 71180
food_bev     79287
entertainment 87702
ground_service 92537
value_for_money 67920
recommended  67455

```

```
dtype: int64
```

```
plt.figure(figsize=(10,8))
sns.heatmap(df.isnull(),cbar=False,cmap="viridis")
plt.title('Missing values in the dataset')
plt.xticks(rotation=30)
plt.show()
```



✓ What did you know about your dataset?

Data includes airline reviews from 2006 to 2019 for popular airlines around the world with user feedback ratings and reviews based on their travel experience.

rows = 131895 columns = 17

Feature descriptions including the parameters are as follows:

airline - Airline name overall - Overall score Author - Author information review_date - Customer Review posted date Customer_review - Actual customer review(Textual) aircraft - Type of aircraft traveller_type - Type of traveller cabin- Cabin type chosen by traveller (Economy, Business,Premium economy,First class) route - Route flown by flyer date_flown - Date of travel seat_comfort - Rating provided towards seat comfort cabin_service - Rating provided towards cabin service. food_bev - Rating provided towards food and beverages supplied during travel. entertainment - Rating provided towards on board flight entertainment ground_service - Rating provided towards ground service staff. value_for_money - Rating provided towards value for money. recommended - Airline service Recommended by flyer (Yes/No)

2. Understanding Your Variables

Dataset Columns

df.columns

```
Index(['airline', 'overall', 'author', 'review_date', 'customer_review',
      'aircraft', 'traveller_type', 'cabin', 'route', 'date_flown',
      'seat_comfort', 'cabin_service', 'food_bev', 'entertainment',
      'ground_service', 'value_for_money', 'recommended'],
      dtype='object')
```

Dataset Describe

df.describe()

	overall	seat_comfort	cabin_service	food_bev	entertainment	ground_
count	64017.000000	60681.000000	60715.000000	52608.000000	44193.000000	3935
mean	5.145430	2.952160	3.191814	2.908170	2.863372	
std	3.477532	1.441362	1.565789	1.481893	1.507262	
min	1.000000	1.000000	1.000000	1.000000	1.000000	
25%	1.000000	1.000000	2.000000	1.000000	1.000000	
50%	5.000000	3.000000	3.000000	3.000000	3.000000	
75%	9.000000	4.000000	5.000000	4.000000	4.000000	
max	10.000000	5.000000	5.000000	5.000000	5.000000	

Variables Description

1. airline - object type
2. overall - Overall rating defined by customer. float type
3. Author - object type
4. Customer_review - object type
5. aircraft - Type of aircraft. object type
6. traveller_type - Type of traveller. object type
7. cabin- Cabin type chosen by traveller. object type
8. route - object type
9. date_flown - object type
10. seat_comfort - Seat Rating. float type
11. cabin_service - float type
12. food_bev - float type
13. entertainment - float type
14. review_date - object type
15. ground_service - float type
16. value_for_money - float type
17. recommended - object type

--- This data descriptions include in a lot of blank rows with many null values.

✓ Check Unique Values for each variable.

```
# Check Unique Values for each variable.
```

```
dict_uniq_value = {}
```

```
dict_uniq_cnt = {}
```

```
for i in df.columns:
```

```
    dict_uniq_value[i] = df[i].unique()
```

```
    dict_uniq_cnt[i] = len(df[i].unique())
```

```
print(dict_uniq_value['airline'])
```

```
print(dict_uniq_cnt['airline'])
```

```

[nan 'Turkish Airlines' 'Qatar Airways' 'Emirates' 'Lufthansa'
 'KLM Royal Dutch Airlines' 'Virgin America' 'American Airlines'
 'Delta Air Lines' 'Southwest Airlines' 'United Airlines'
 'Jetblue Airways' 'Aegean Airlines' 'Aeroflot Russian Airlines'
 'Aeromexico' 'Air Canada' 'Air New Zealand' 'Alitalia' 'AirAsia'
 'Asiana Airlines' 'Avianca' 'Austrian Airlines' 'British Airways'
 'Brussels Airlines' 'China Eastern Airlines' 'China Southern Airlines'
 'Copa Airlines' 'Ethiopian Airlines' 'Egyptair' 'Finnair' 'Iberia'
 'ANA All Nippon Airways' 'easyJet' 'Korean Air' 'LATAM Airlines'
 'LOT Polish Airlines' 'Qantas Airways' 'Air France' 'Etihad Airways'
 'Pegasus Airlines' 'Royal Jordanian Airlines' 'Ryanair'
 'South African Airways' 'Saudi Arabian Airlines' 'TAP Portugal'
 'Eurowings' 'EVA Air' 'Royal Air Maroc' 'Singapore Airlines'
 'SAS Scandinavian' 'Swiss Intl Air Lines' 'Thai Airways' 'Air India'

```



```
'Air Europa' 'Air Canada rouge' 'airBaltic' 'Air China'
'Cathay Pacific Airways' 'Wizz Air' 'Spirit Airlines' 'TAROM Romanian'
'Vueling Airlines' 'Sunwing Airlines' 'QantasLink' 'Bangkok Airways'
'flydubai' 'Garuda Indonesia' 'Germanwings' 'Frontier Airlines'
'Icelandair' 'IndiGo' 'Aer Lingus' 'Adria Airways' 'Air Arabia'
'Alaska Airlines' 'Tunisair' 'Norwegian' 'Thai Smile Airways' 'Gulf Air'
'Kuwait Airways' 'WOW air' 'Ukraine International']
```

82

```
for i in df.columns.tolist():
    print("Unique values in", i, df[i].nunique())
```

```
⇒ Unique values in airline 81
Unique values in overall 10
Unique values in author 44069
Unique values in review_date 3015
Unique values in customer_review 61172
Unique values in aircraft 2088
Unique values in traveller_type 4
Unique values in cabin 4
Unique values in route 24549
Unique values in date_flown 63
Unique values in seat_comfort 5
Unique values in cabin_service 5
Unique values in food_bev 5
Unique values in entertainment 5
Unique values in ground_service 5
Unique values in value_for_money 5
Unique values in recommended 2
```

3. Data Wrangling

✓ Data Wrangling Code

```
#Create a copy of the dataset
df.copy()
```



	airline	overall	author	review_date	customer_review	aircraft	travel
0	NaN	NaN	NaN	NaN	NaN	NaN	
1	Turkish Airlines	7.0	Christopher Hackley	8th May 2019	âœ... Trip Verified London to Izmir via Istanb...	NaN	
2	NaN	NaN	NaN	NaN	NaN	NaN	
3	Turkish Airlines	2.0	Adriana PISOI	7th May 2019	âœ... Trip Verified Istanbul to Bucharest. We ...	NaN	Fai
4	NaN	NaN	NaN	NaN	NaN	NaN	
...	
131890	Ukraine International	NaN	Andriy Yesypenko	19th May 2006	Kiev - London (Gatwick) in business class (in ...	NaN	
131891	NaN	NaN	NaN	NaN	NaN	NaN	
131892	Ukraine International	NaN	Volodya Bilotkach	29th April 2006	Several flights - KBP to AMS (3 times one way)...	NaN	
131893	NaN	NaN	NaN	NaN	NaN	NaN	
131894	Ukraine International	NaN	Kasper Hettinga	10th February 2006	KBP-AMS with UIA. Although it was a relatively...	NaN	

131895 rows × 17 columns

```
df.drop_duplicates(inplace=True)
```

```
df.reset_index(drop=True,inplace=True)
```

```
df.shape
```



(61184, 17)

```
df.isnull().sum()
```



0

airline	1
overall	1783
author	1
review_date	1
customer_review	1
aircraft	42696
traveller_type	23644
cabin	2479
route	23671
date_flown	23750
seat_comfort	4973
cabin_service	4944
food_bev	12843
entertainment	20954
ground_service	24015
value_for_money	1857
recommended	1423

dtype: int64

```
#Dropping unqanted columns like author, route, customer_review  
df.drop(columns=['author', 'route', 'customer_review'], axis=1, inplace=True)
```

```
df.drop(columns=['aircraft'], axis=1, inplace=True)
```

```
df.isnull().sum().sort_values(ascending=False)
```



0

ground_service	24015
date_flown	23750
traveller_type	23644
entertainment	20954
food_bev	12843
seat_comfort	4973
cabin_service	4944
cabin	2479
value_for_money	1857
overall	1783
recommended	1423
review_date	1
airline	1

dtype: int64

```
#Dropping nan values for ground_service and entertainment  
df.dropna(subset=['ground_service', 'entertainment'], inplace=True)
```

```
df.isnull().sum().sort_values(ascending=False)
```



0

food_bev	782
cabin	13
date_flown	10
traveller_type	2
overall	1
seat_comfort	1
cabin_service	1
review_date	0
airline	0
entertainment	0
ground_service	0
value_for_money	0
recommended	0

dtype: int64

```
#Imputing nan values for food_bev
df['food_bev'].fillna(df['food_bev'].mean(), inplace=True)
```



<ipython-input-27-1921402457>:2: FutureWarning: A value is trying to be set on a copy
The behavior will change in pandas 3.0. This inplace method will never work because t

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({

```
df['food_bev'].fillna(df['food_bev'].mean(), inplace=True)
```



```
df.dropna(inplace=True)
```

```
#Final check for null values
df.isnull().sum()
```



	0
airline	0
overall	0
review_date	0
traveller_type	0
cabin	0
date_flown	0
seat_comfort	0
cabin_service	0
food_bev	0
entertainment	0
ground_service	0
value_for_money	0
recommended	0

dtype: int64

df.shape



(23606, 13)

```
#Resetting index after data cleaning
df.reset_index(drop=True, inplace=True)
```

df.head()



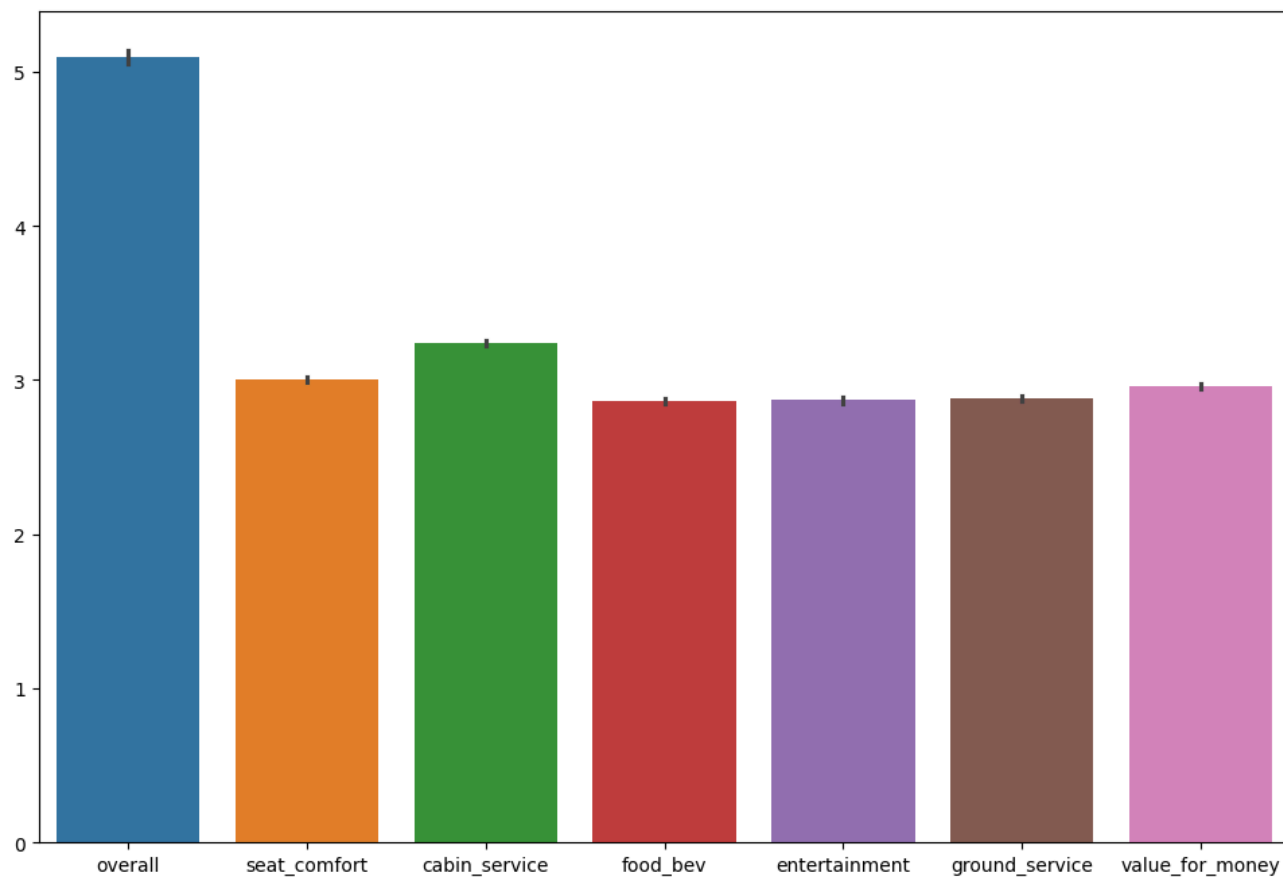
	airline	overall	review_date	traveller_type	cabin	date_flown	seat_comfort
0	Turkish Airlines	7.0	8th May 2019	Business	Economy Class	2019-05-01 00:00:00	4.0
1	Turkish Airlines	2.0	7th May 2019	Family Leisure	Economy Class	2019-05-01 00:00:00	4.0
2	Turkish Airlines	3.0	7th May 2019	Business	Economy Class	2019-05-01 00:00:00	1.0



```
#Checking for outliers
plt.figure(figsize=(12,8))
sns.barplot(df)
```



<Axes: >



Data manipulation

```
df.info()
```



```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 23606 entries, 0 to 23605
```

```
Data columns (total 13 columns):
```

#	Column	Non-Null Count	Dtype
0	airline	23606 non-null	object
1	overall	23606 non-null	float64
2	review_date	23606 non-null	object
3	traveller_type	23606 non-null	object
4	cabin	23606 non-null	object
5	date_flown	23606 non-null	object
6	seat_comfort	23606 non-null	float64
7	cabin_service	23606 non-null	float64

```

8   food_bev          23606 non-null   float64
9   entertainment    23606 non-null   float64
10  ground_service    23606 non-null   float64
11  value_for_money   23606 non-null   float64
12  recommended       23606 non-null   object
dtypes: float64(7), object(6)
memory usage: 2.3+ MB

```

```

d_type = {'overall': 'int8', 'review_date': 'datetime64[ns]',
          'seat_comfort': 'int8', 'cabin_service': 'int8',
          'food_bev': 'int8', 'entertainment': 'int8',
          'ground_service': 'int8', 'value_for_money': 'int8'}
for i, j in d_type.items():
    df[i] = df[i].astype(j)

```

```
df.info()
```

```

➡ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 23606 entries, 0 to 23605
Data columns (total 13 columns):
#   Column                Non-Null Count  Dtype
---  -
0   airline                23606 non-null  object
1   overall                23606 non-null  int8
2   review_date            23606 non-null  datetime64[ns]
3   traveller_type         23606 non-null  object
4   cabin                  23606 non-null  object
5   date_flown             23606 non-null  object
6   seat_comfort           23606 non-null  int8
7   cabin_service          23606 non-null  int8
8   food_bev               23606 non-null  int8
9   entertainment          23606 non-null  int8
10  ground_service         23606 non-null  int8
11  value_for_money        23606 non-null  int8
12  recommended            23606 non-null  object
dtypes: datetime64[ns](1), int8(7), object(5)
memory usage: 1.2+ MB

```

```

#Converting date_flown column in a proper date format by removing timestamp and changed t
df['date_flown'] = pd.to_datetime(df['date_flown'], errors='coerce')


```

```


#Renamong overall_rating and date_flown to department_date
rename_col = {'overall': 'overall_rating', 'date_flown': 'department_date'}
df.rename(columns=rename_col, inplace=True)

```

```
df.head()
```

	airline	overall_rating	review_date	traveller_type	cabin	department_date	se
0	Turkish Airlines	7	2019-05-08	Business	Economy Class	2019-05-01	
1	Turkish Airlines	2	2019-05-07	Family Leisure	Economy Class	2019-05-01	
2	Turkish Airlines	3	2019-05-07	Business	Economy Class	2019-05-01	




What all manipulations have you done and insights you found?

1. Converted Date columns to datetime format as they were in object datatype and converted various rating columns from float to int as all ratings are only in integers.
2. date_flown column was not in proper date format it also contained Timestamp so we changed to a proper date format by removing timestamp and converted it into datetime datatype.
3. Renamed overall to overall_rating and date_flown to departure_date for better understanding.

4. Data Vizualization, Storytelling & Experimenting with charts : Understand the relationships between variables

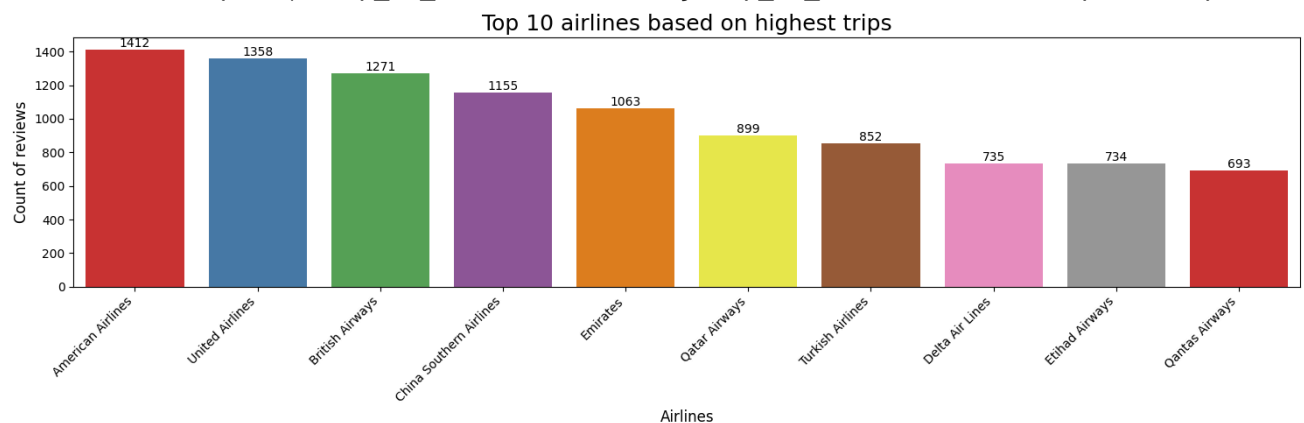
Chart - 1

```
plt.figure(figsize=(15,5))
air_cnt = df['airline'].value_counts()
palette = sns.color_palette("Set1", 10)
# Select top 10 airlines
top_10_airlines = air_cnt.head(10)
ax = sns.barplot(x=top_10_airlines.index, y=top_10_airlines.values, palette=palette)
plt.xlabel('Airlines', fontsize=12)
plt.ylabel('Count of reviews', fontsize=12)
plt.title('Top 10 airlines based on highest trips', fontsize=18)
plt.xticks(rotation=45, ha='right') # Rotate x-axis labels for better readability
for num in ax.containers:
    ax.bar_label(num)
plt.tight_layout() # Adjust layout to prevent labels from overlapping
plt.show()
```

 <ipython-input-40-3172439272>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

```
ax = sns.barplot(x=top_10_airlines.index, y=top_10_airlines.values, palette=palette)
```



1. Why did you pick the specific chart?

Bar graph is commonly used when we have to represent categorical values with numerical values and here it suits well as we are to show airlines with its count of reviews

2. What is/are the insight(s) found from the chart?

It has been shown that the top 10 airlines in terms of their reviews count and can understand that American Airlines has the most number of review followed by United Airlines and British Airways.

3. Will the gained insights help creating a positive business impact? Are there any insights that lead to negative growth? Justify with specific reason.

It's essential to consider the sentiments expressed by customers, understanding what customers appreciate about an airline, whether it's excellent service, punctuality, or other

positive aspects, can help the company leverage and enhance these strengths and Identifying common issues or complaints allows the airline to address and rectify problems, leading to an improved customer experience.

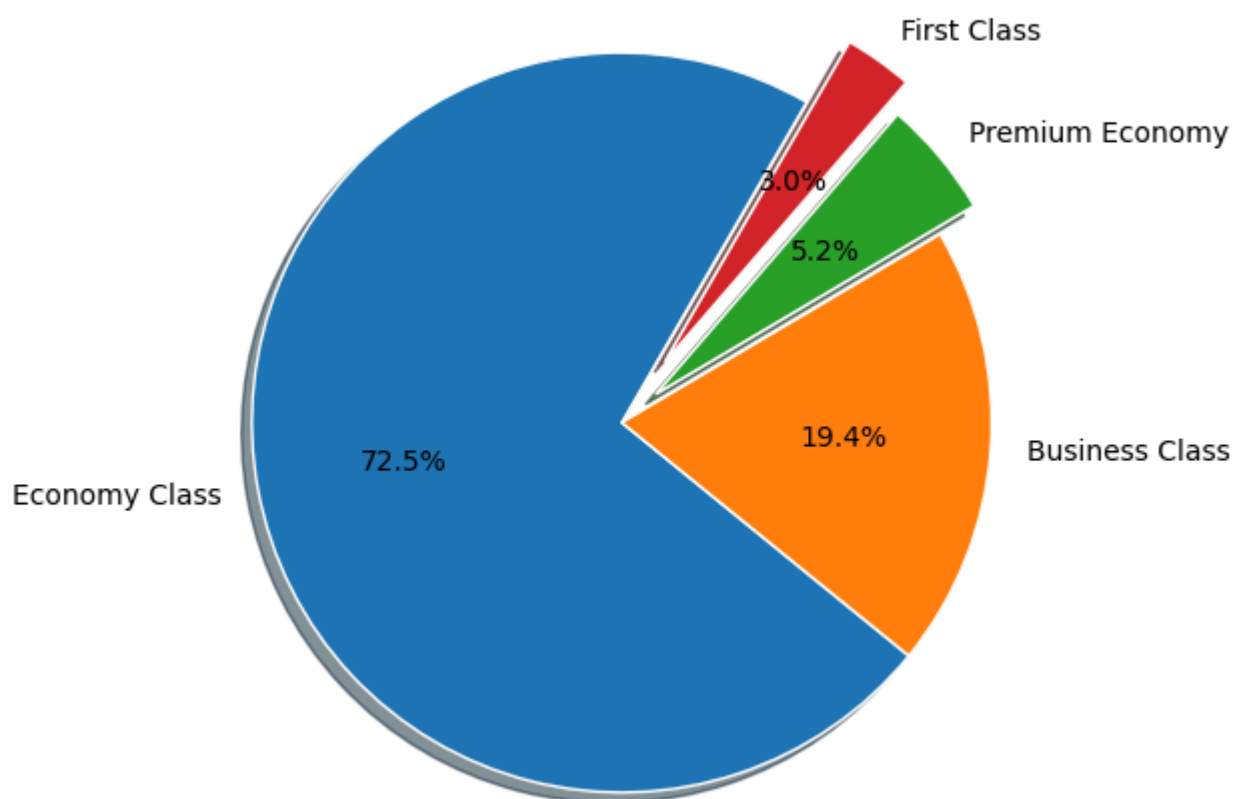
Chart - 2

```
cab_cnt = df['cabin'].value_counts().reset_index()
cab_cnt.columns = ['cabin', 'count']

# Fixing the KeyError and plotting the pie chart
plt.figure(figsize=(12,6))
plt.pie(cab_cnt['count'], labels=cab_cnt['cabin'], autopct='%1.1f%%', explode = [0, 0, 0.
plt.title('Distribution of Different Cabin classes preffered by Passengers', y=1.08, fonts
plt.show()
```



Distribution of Different Cabin classes preffered by Passengers



1. Why did you pick the specific chart?

A pie chart is a circular statistical graphic that is divided into slices to illustrate numerical proportions. It's primarily used to show the relationship of parts to a whole.

2. What is/are the insight(s) found from the chart?

From the above chart we can see that economy class constitutes the largest part followed by business class and the other two class constitutes very less portion of the chart which tells that mostly people were travelling in economy class .

3. Will the gained insights help creating a positive business impact? Are there any insights that lead to negative growth? Justify with specific reason.

Understanding the popularity of economy class can guide the airline in customizing services to meet the needs and expectations of this larger customer segment, Given that economy class is the most popular, marketing efforts can be targeted towards this segment. Promotions, loyalty programs, and advertising can be tailored to attract and retain economy class travelers.

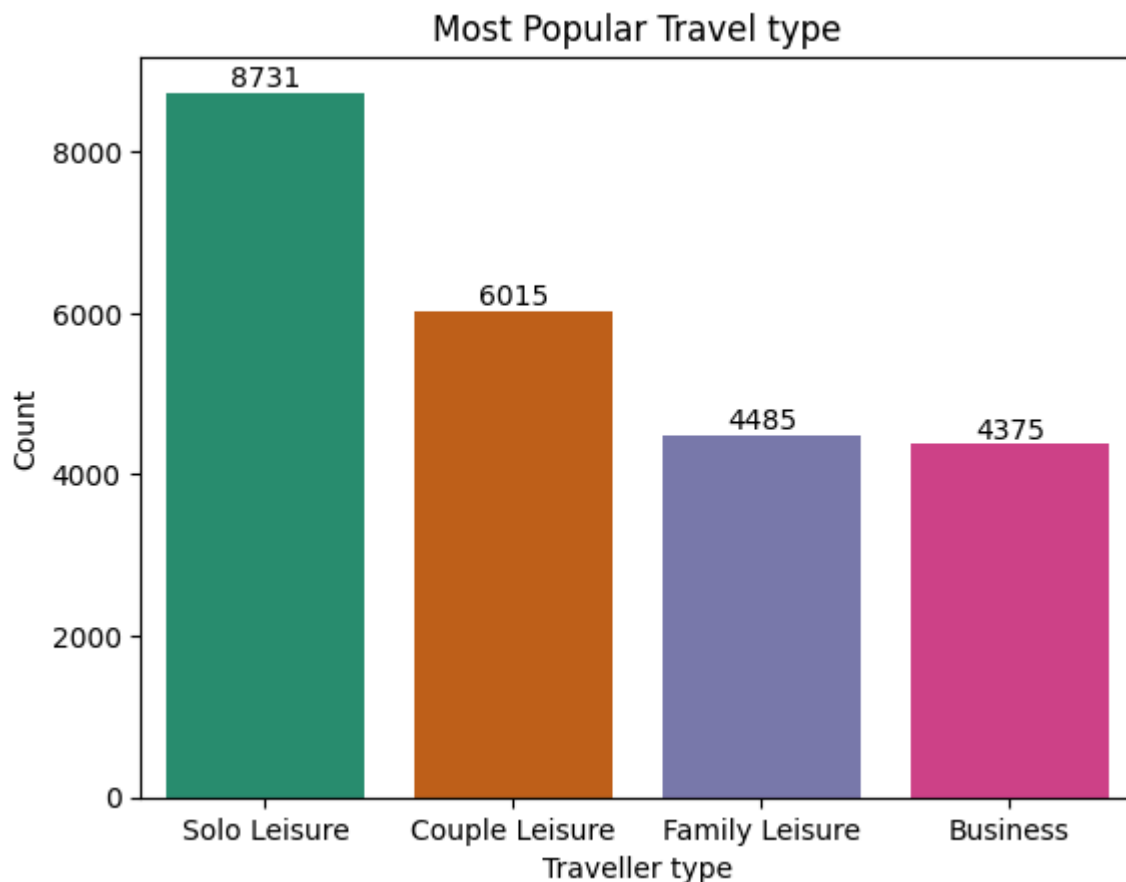
Chart - 3:

```
most_trav = df["traveller_type"].value_counts().reset_index()
most_trav.columns = ['traveller_type', 'count'] # Rename columns
ax = sns.barplot(x=most_trav['traveller_type'], y = most_trav['count'] ,palette = 'Dark2')
plt.ylabel('Count')
plt.xlabel('Traveller type')
plt.title('Most Popular Travel type')
for num in ax.containers:
    ax.bar_label(num)
plt.show()
```

<ipython-input-43-748472124>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

```
ax = sns.barplot(x=most_trav['traveller_type'], y = most_trav['count'], palette = 'D
```



1. Why did you pick the specific chart?

Bar graph is typically used when we have to depict categorical values with numerical values and here it suits well as we are to show air

2. What is/are the insight(s) found from the chart?

Solo Leisure is the most preferred travel_type by passengers while Bussiness is the lowest travel_type.

3. Will the gained insights help creating a positive business impact? Are there any insights that lead to negative growth? Justify with specific reason.

Understanding that solo leisure travel is more popular allows the airline to tailor marketing efforts specifically toward this segment. Promotions, advertising, and loyalty programs can be designed to attract and retain solo leisure travelers, potentially increasing customer acquisition and retention.

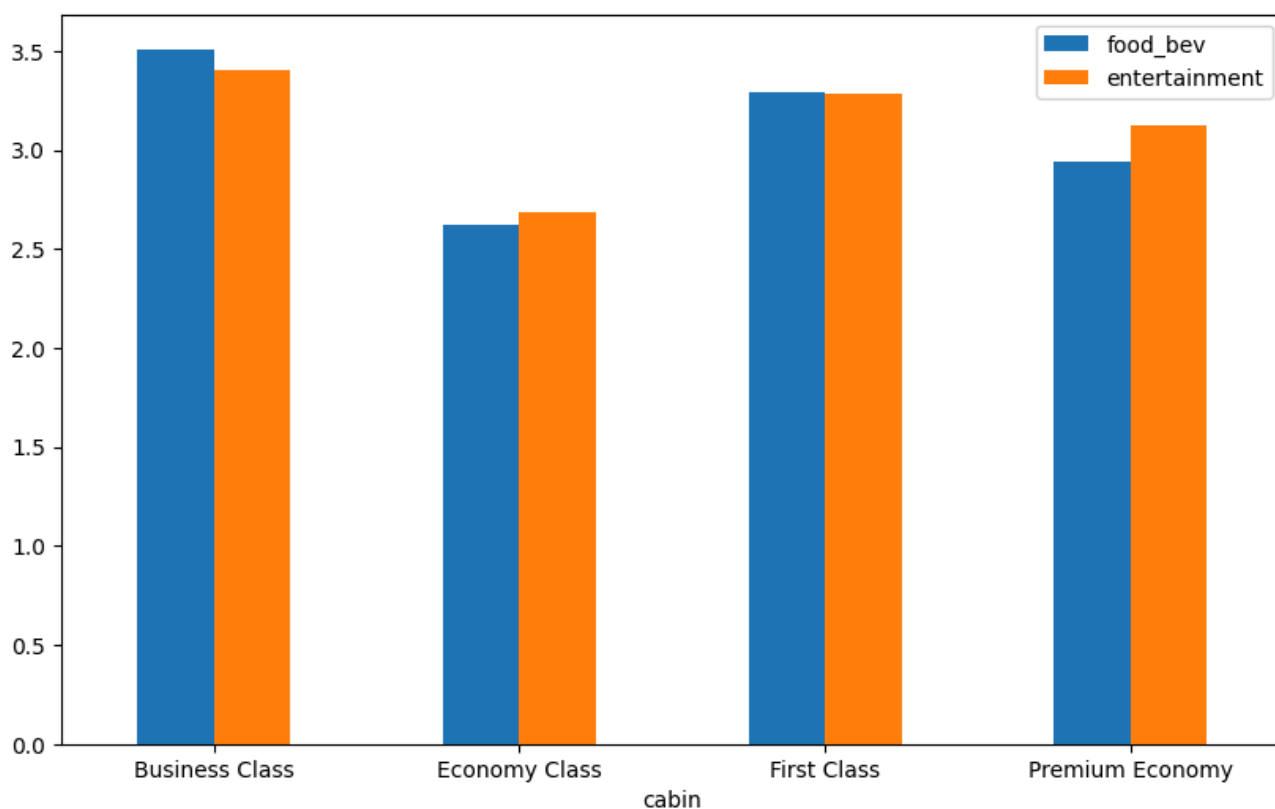
Chart - 4: Comparing Cabin classes based on Food_bev and entertainment ratings

```
eda_4=df.groupby('cabin')[['food_bev', 'entertainment']].mean().reset_index()  
eda_4
```



	cabin	food_bev	entertainment
0	Business Class	3.509517	3.406913
1	Economy Class	2.625146	2.688859
2	First Class	3.295775	3.284507
3	Premium Economy	2.942482	3.123254

```
plt.rcParams['figure.figsize'] = (10, 6)  
eda_4.plot(x='cabin', y=['food_bev', 'entertainment'], kind='bar')  
plt.xticks(rotation=0)  
plt.show()
```



1. Why did you pick the specific chart?

This chart is best suited for showing side by side comparison of various cabin class wrt food_beverages and entertainment.

2. What is/are the insight(s) found from the chart?


There is no significant change in ratings of food_bev and entertainment in Economy and first_class but in premium economy class there is more rating for entertainment as compared to food_bev and vice-versa for Bussiness_class.

3. Will the gained insights help creating a positive business impact? Are there any insights that lead to negative growth? Justify with specific reason.

Knowing that there are different preferences for entertainment and food_bev in Premium Economy and Business Class allows the airline to focus on enhancing services in each class selectively. This could involve improving menu options, upgrading entertainment systems, or introducing new features to align with passenger expectations.

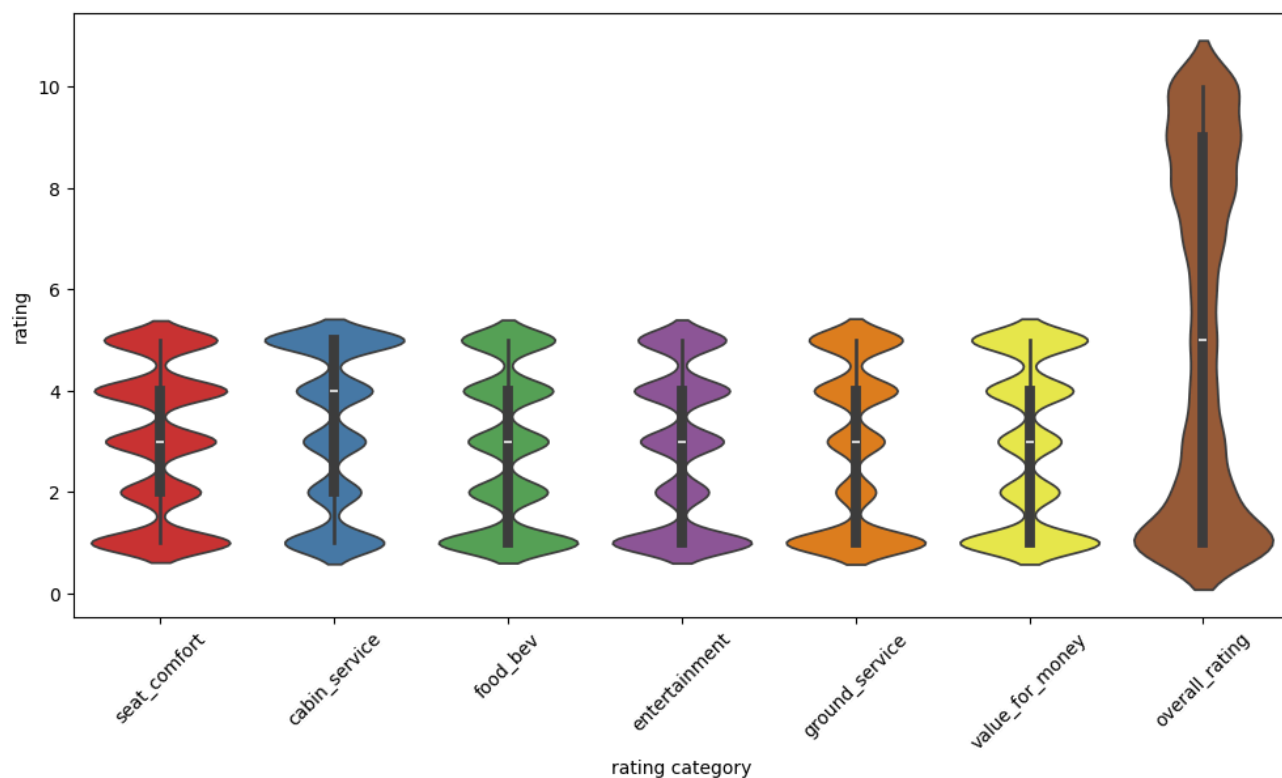
Chart 5: Distribution of different types of ratings

```
columns = ['seat_comfort', 'cabin_service', 'food_bev', 'entertainment', 'ground_service']
df_method = df.melt(value_vars=columns, var_name='rating category', value_name='rating')
plt.figure(figsize=(12,6))
sns.violinplot(x='rating category', y='rating', data=df_method, palette='Set1')
plt.xticks(rotation=45)
plt.show()
```

 <ipython-input-47-1827394392>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

```
sns.violinplot(x='rating category', y='rating', data=df_method, palette='Set1')
```



1. Why did you pick the specific chart?

We chose Violin chart because they are particularly useful for comparing the distribution of data between different groups or categories. This allows us to see not only the average rating for each type but also how ratings are spread out.

2. What is/are the insight(s) found from the chart?


We can see that our mostly rating variables spread between 1-6 and only overall rating is ranging between 1-10 and mostly ratings are either lower range side or upper range side.

- Will the gained insights help creating a positive business impact? Are there any insights that lead to negative growth? Justify with specific reason.

Airlines can use this insight to adjust pricing strategies. If customers rate the value for money lower, airlines can consider offering more competitive pricing or value-added services. Hence improving specific aspects of service that are consistently rated lower can enhance the airline's brand reputation and differentiate it from competitors.

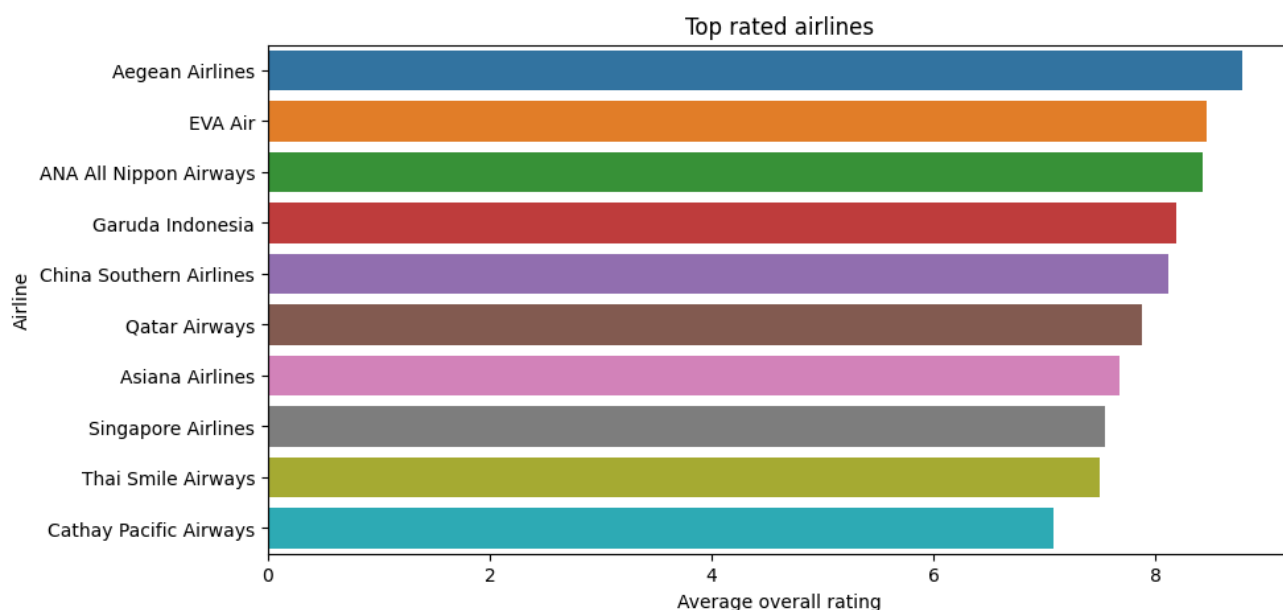
Chart 6: Top Rated Airlines

```
plt.figure(figsize=(10,5))
overall = df.groupby(df['airline'])['overall_rating'].mean().sort_values(ascending = False)
ax = sns.barplot(y=overall['airline'],x = overall['overall_rating'],palette = "tab10" )
plt.xlabel('Average overall rating')
plt.ylabel('Airline')
plt.title('Top rated airlines')
plt.show()
```

 <ipython-input-48-3828159778>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

```
ax = sns.barplot(y=overall['airline'],x = overall['overall_rating'],palette = "tab10"
```



- Why did you pick the specific chart?

This chart, horizontal column chart for comparison of various airlines wrt to Average overall rating.

2. What is/are the insight(s) found from the chart?

From this chart it can be seen Aegean airlines is highest overall rating followed by EVA airlines and ANA all Nippon Airways while Cathay Pacific airways has the lowest rating .


3. Will the gained insights help creating a positive business impact? Are there any insights that lead to negative growth? Justify with specific reason.

Aegean Airlines, EVA Airlines, and ANA All Nippon Airways can continue to focus on providing excellent service to maintain their high ratings. This can include improving in-flight amenities, on-time performance, and customer service. Cathay Pacific Airways, being rated lower, can invest in training and development programs for its staff to enhance customer service and improve overall customer experience.

Chart - 7: Top 10 Airlines wrt to value for money

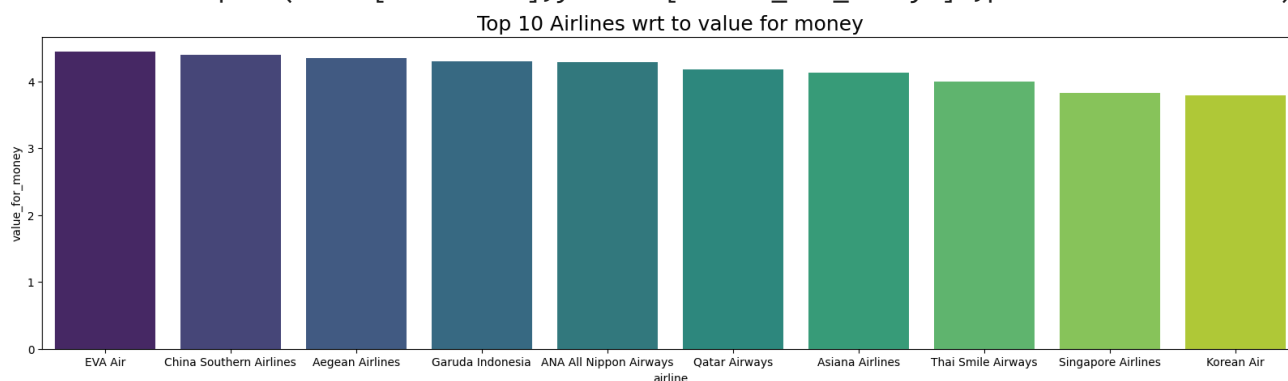
```
plt.figure(figsize= (20,5))
val = df.groupby(df['airline'])['value_for_money'].mean().sort_values(ascending = False).
ax = sns.barplot(x=val['airline'],y = val['value_for_money'] ,palette = 'viridis')

plt.title('Top 10 Airlines wrt to value for money',fontsize = 18)
plt.show()
```

 <ipython-input-49-3951500569>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

```
ax = sns.barplot(x=val['airline'], y = val['value_for_money'], palette = 'viridis')
```



1. Why did you pick the specific chart?

Bar chart is selected for comparison of various airlines wrt to Average rating of value for money.

2. What is/are the insight(s) found from the chart?

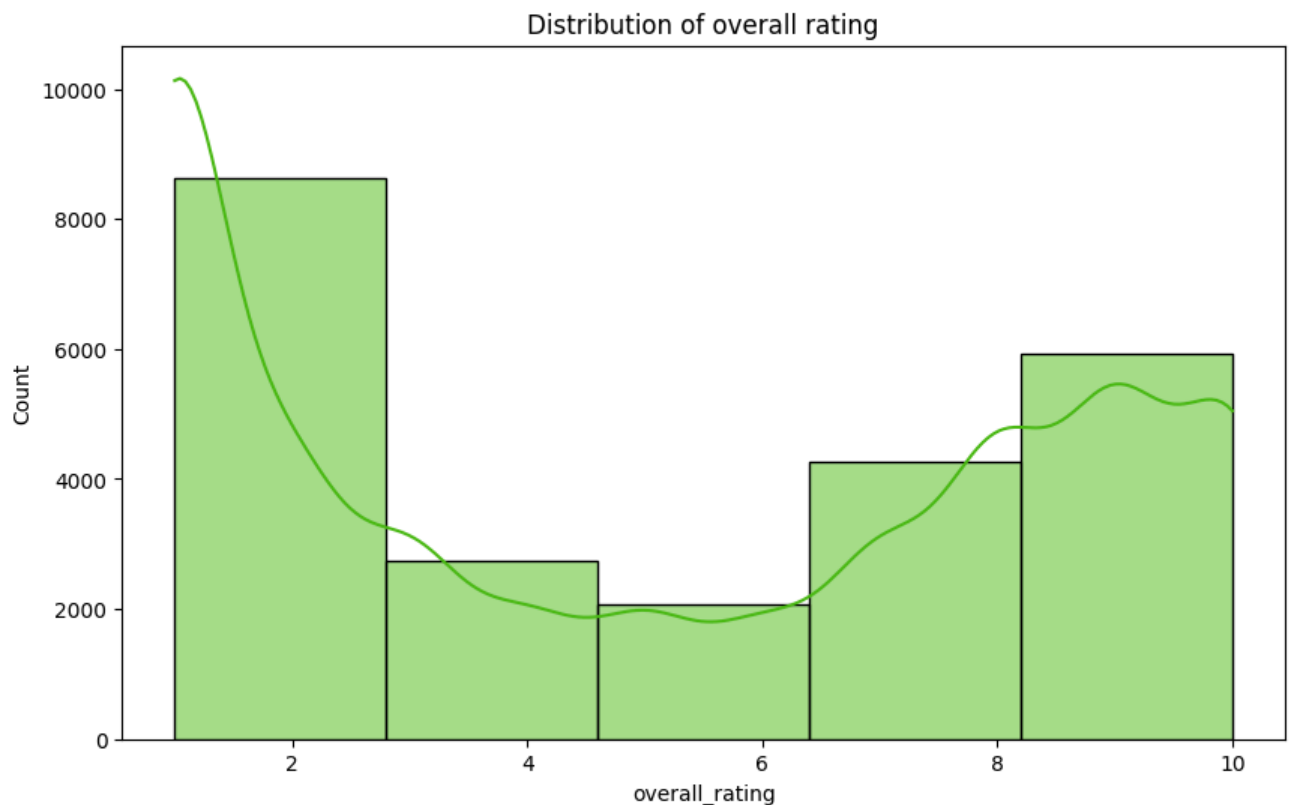
EVA Air is the highest rated followed by China Southern Airlines and Aegean Airlines for value for money .

3. Will the gained insights help creating a positive business impact? Are there any insights that lead to negative growth? Justify with specific reason.

Based on the above insights EVA Air can offer loyalty programs or incentives for frequent flyers to encourage repeat business and enhance customer loyalty.

Chart - 8: Distribution of overall rating

```
sns.histplot(df['overall_rating'], kde = True,bins =5,color='#4CBB17')  
plt.title('Distribution of overall rating')  
plt.show()
```



1. Why did you pick the specific chart?

We selected Histogram for distribution of Overall rating.

2. What is/are the insight(s) found from the chart?

It can be conclude that most people have rated between either 1-2 or 8-10 it shows that passenger have either best or worst experience with airline.

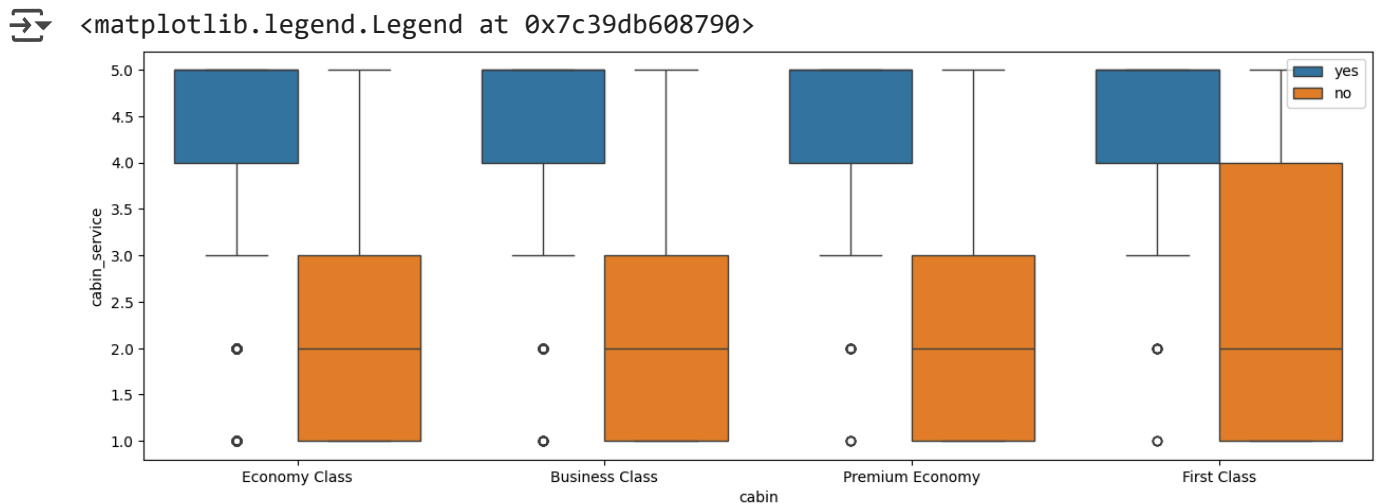
3. Will the gained insights help creating a positive business impact? Are there any insights that lead to negative growth? Justify with specific reason.

Based on above insightes Airlines can focus on addressing the aspects that lead to extreme negative experiences, such as poor customer service, flight delays, or uncomfortable seating, to reduce the number of low ratings and engaging with customers who have provided extreme

ratings (either low or high) can provide valuable feedback for improvement and allow airlines to address specific pain points or areas of excellence.

Chart - 9: Cabin Class Recommendation based on service ratings

```
plt.figure(figsize=(15,5))
sns.boxplot(x=df['cabin'], y=df['cabin_service'], hue = df['recommended'])
plt.legend(loc='upper right')
```



1. Why did you pick the specific chart?

Boxplot is selected to show rating comparison between different cabin classes.

2. What is/are the insight(s) found from the chart?

Every cabin class if the service rating is more than 3 then passenger is more likely to recommend that airline to others.


3. Will the gained insights help creating a positive business impact? Are there any insights that lead to negative growth? Justify with specific reason.

By focusing on enhancing service quality across all cabin classes to ensure ratings exceed 3, airlines can improve overall customer satisfaction, leading to positive recommendations and repeat business

negative-impact from insights: If service ratings for any cabin class consistently fall below 3, it could lead to negative word-of-mouth, lower customer satisfaction, and a decline in recommendations, which could result in a loss of customers and revenue.

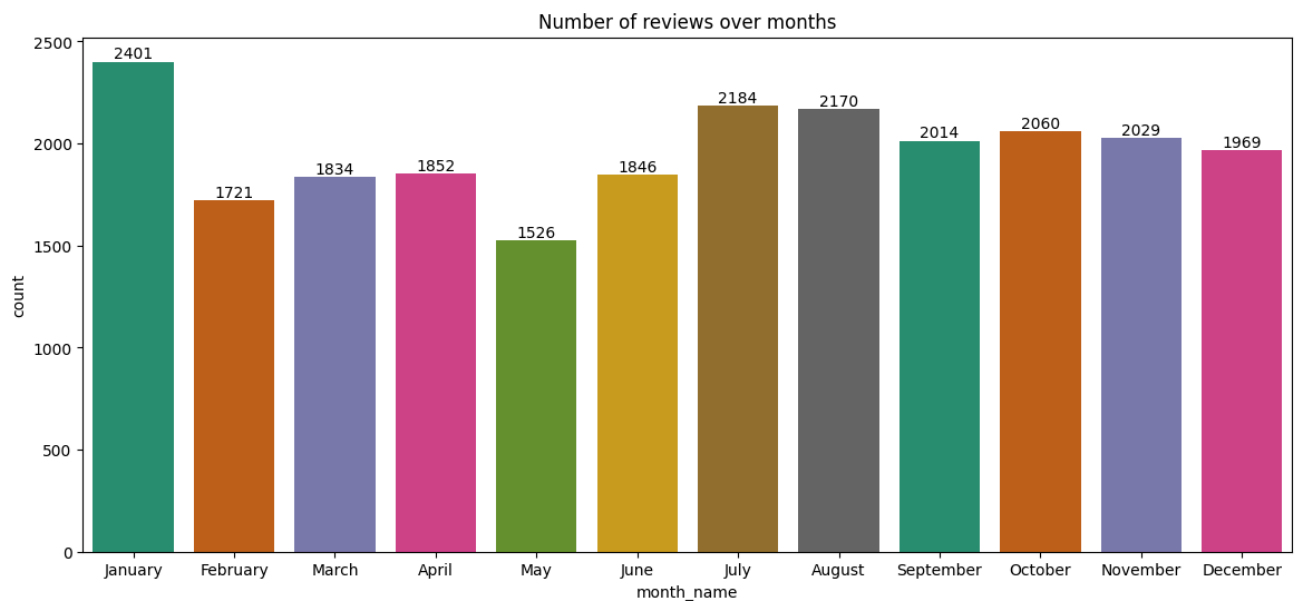
Chart - 10 - Reviews over months

```
plt.figure(figsize= (14,6))
plt.title('Number of reviews over months')
df['month_name'] = df['review_date'].dt.strftime('%B')
df['month'] = df['review_date'].dt.month
df2 = df[['month_name', 'month']].value_counts().reset_index().sort_values(by = 'month')
df2.rename(columns={0: 'count'}, inplace = True)
ax = sns.barplot(x = df2['month_name'], y = df2['count'], palette = 'Dark2')
for num in ax.containers:
    ax.bar_label(num)
```

 <ipython-input-52-3957022254>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

```
ax = sns.barplot(x = df2['month_name'], y = df2['count'],palette = 'Dark2')
```



1. Why did you pick the specific chart?

Here we are plotting in which month how many reviews are submitting so with the help of this we can check is there any pattern or relation of number of reviews with the month so the best suited chart is a bar chart.

2. What is/are the insight(s) found from the chart?

We can analyze from this chart that January month is having a large number of reviews as compared to others followed by July and August while in the month of May we have the least. May being a holiday or vacation month, so more number of travellers are there, so reviews are also there or the staff is not properly managing the services due to this reviews are more because passenger traffic is more.

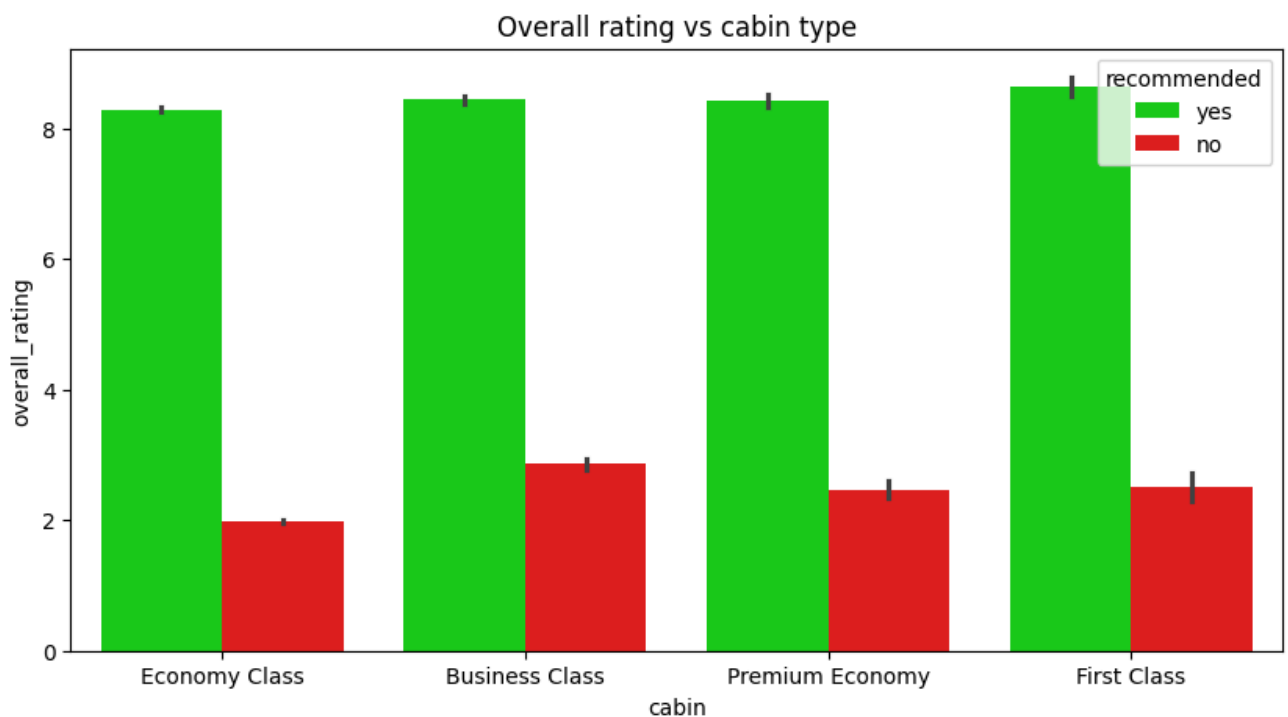
3. Will the gained insights help creating a positive business impact? Are there any insights that lead to negative growth? Justify with specific reason.

For holiday months if we are having more passenger traffic so we should employ the temporary staff to not spoil our services and management if the traffic is the reason.

Chart - 11 - Overall rating vs cabin type

```
plt.figure(figsize=(10,5))
plt.title('Overall rating vs cabin type')
sns.barplot(x = df.cabin, y = df.overall_rating, hue = df['recommended'], palette= ['#00e
```

```
<Axes: title={'center': 'Overall rating vs cabin type'}, xlabel='cabin',
ylabel='overall_rating'>
```



1. Why did you pick the specific chart?

Since we are plotting our categorical value against discrete numerical value so best suited chart is a side bar chart.

2. What is/are the insight(s) found from the chart?

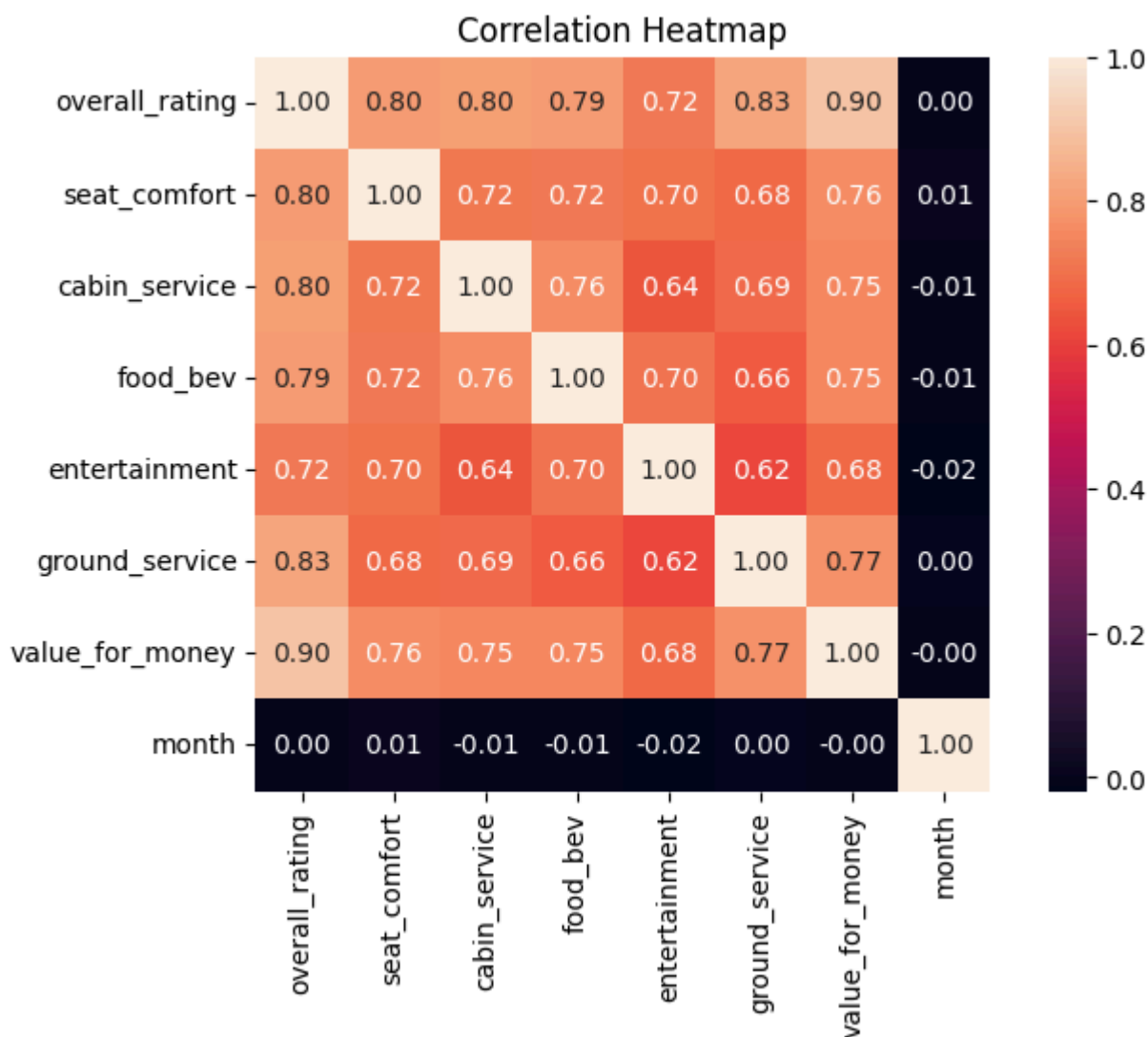
We can clearly see from this that almost in all cabin type we have overall rating more than 8 and the customer recommend the airline to others while for no we have almost 2 rating overall in economy while 3 for rest of all, so there is no much difference between cabin type if we see recommend yes/no .

3. Will the gained insights help creating a positive business impact? Are there any insights that lead to negative growth? Justify with specific reason.

We can say from above insights that if a person give overall rating more than 8 then its 99% sure that he is gonna recommend the airline to others by the help of rating we can request our customer to share their opinions on airline service on some platform for recommendation, while if person is not satisfied we will try to resolve their issue with best possible solution.

Chart - 12 - Correlation Heatmap

```
plt.figure(figsize=(8,5))
plt.title('Correlation Heatmap')
# Select only numerical columns for correlation calculation
numerical_df = df.select_dtypes(include=np.number)
sns.heatmap(numerical_df.corr(), annot = True, fmt='.2f', annot_kws={'size': 10}, vmax=1
plt.show()
```



1. Why did you pick the specific chart?

This particular graph is the most powerful visualisation as it depicts the relationship of all the columns with each other and one another too.

2. What is/are the insight(s) found from the chart?

Here on the graph the positive values tell us about that that particular variable is directly proportional to other one corresponding to it , the negative value indicates that the variable is indirectly proportional to corresponding variable and larger the magnitude , more is the dependency.

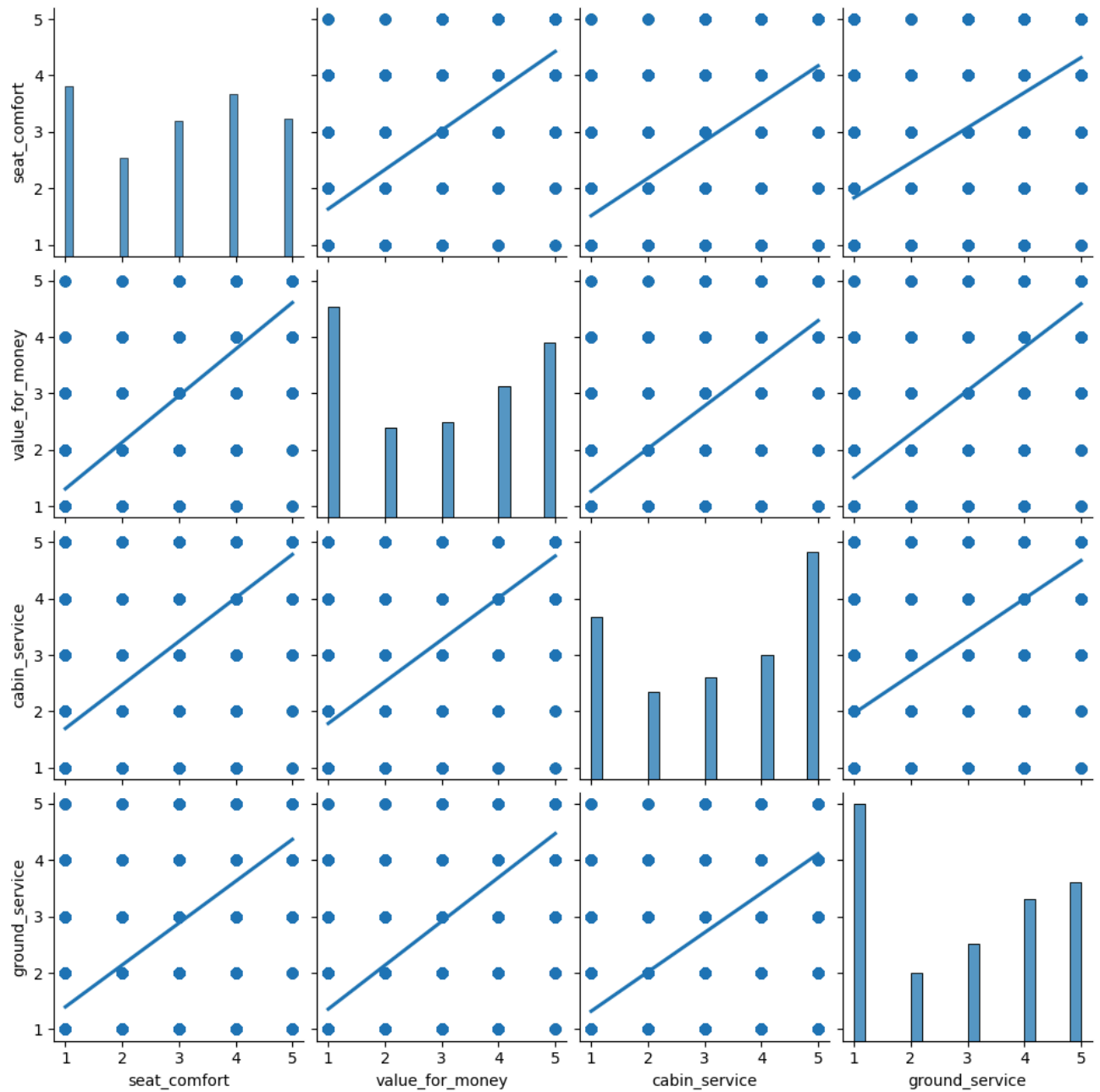
3. Will the gained insights help creating a positive business impact? Are there any insights that lead to negative growth? Justify with specific reason.

Along with understanding the concentration of the distribution of the dataset, it also indicates that if you were looking at a heatmap of a variable like "value_for_money" plotted on both axes,

and you saw a "1" in a cell, it would mean that when the minimum nights is high on the X-axis, it's also high on the Y-axis. When it's low on the X-axis, it's low on the Y-axis, and the relationship between these two instances of value_for_money is very strong and positive.

Chart - 13 - Pair Plot

```
column_name = [ 'seat_comfort', 'value_for_money', 'cabin_service', 'ground_service' ]  
pairplot_data = df[column_name]  
chart15=sns.pairplot(pairplot_data,kind = 'reg')  
plt.show()
```



1. Why did you pick the specific chart?

Here we wanted to have a pairwise visualisation of all the columns in the dataset , hence used pairplot.

2. What is/are the insight(s) found from the chart?

Since we have a discrete type of dataset so the distribution is showing here is not so perfect but by this we can see a positive relationship between all rating related columns.

3. Will the gained insights help creating a positive business impact? Are there any insights that lead to negative growth? Justify with specific reason.

There is a diagonal relationship according to the bar charts whereas other graphs showed a linear change with respect to the columns.

✓ 5.Hypothesis Testing

Based on your chart experiments, define three hypothetical statements from the dataset. In the next three questions, perform hypothesis testing to obtain final conclusion about the statements through your code and statistical testing.

✓ Hypothetical Statement - 1

1. State Your research hypothesis as a null hypothesis and alternate hypothesis.

Null Hypothesis (H0): The overall ratings for two specific airlines are equal.

Alternative Hypothesis (H1): The overall ratings for two specific airlines are not equal.

2. Perform an appropriate statistical test.

```
import numpy as np
from scipy import stats

#Perform statistical test to obtain P value
mean1 = (df[df['airline']=='Singapore Airlines']['overall_rating']).mean()
mean2 = (df[df['airline']=='EVA Air']['overall_rating']).mean()
std1 = (df[df['airline']=='Singapore Airlines']['overall_rating']).std()
std2 = (df[df['airline']=='EVA Air']['overall_rating']).std()

# Calculate the sample sizes
```

```

n1 = (df[df['airline']=='Singapore Airlines']['overall_rating']).count()
n2 = (df[df['airline']=='EVA Air']['overall_rating']).count()

#Calculate the standard error for each airline
se1 = std1/np.sqrt(n1)
se2 = std2/np.sqrt(n2)

# Calculate the standard error of the difference between means
standard_error = np.sqrt((se1**2)+(se2**2))

# Calculate the t_test
t_stat = (mean1 - mean2)/standard_error

alpha = 0.05 # Set the significance level

dodf = n1+n2 - 2 #Degree of freedom

p_value = (1-stats.t.cdf(abs(t_stat),dodf)) * 2 # Calculate the p-value (two-tailed test)
alpha = 0.05 # Set the significance level
print('The p value for 0.05 significance level is {:.5f}'.format(p_value))

if p_value < alpha:
    print('Reject the null hypothesis')
else:
    print('Fail to reject the null hypothesis')

```

Which statistical test have you done to obtain P-Value?

We performed T-Test for this hypothesis testing to obtain P-value.

Why did you choose the specific statistical test?

From the sample dataset and we infer about population and population parameters which are not known to us and to compare the overall ratings of the two selected airlines.

✓ Hypothetical Statement - 2

1. State Your research hypothesis as a null hypothesis and alternate hypothesis.

Null Hypothesis (H0): There is no association between the traveller type and the likelihood of recommending the airline.

Alternative Hypothesis (H1): There is an association between the traveller type and the likelihood of recommending the airline.

2. Perform an appropriate statistical test.

```

contingency_table = pd.crosstab(df['traveller_type'], df['recommended'])
chi2, p_value, dof, expected = chi2_contingency(contingency_table)
print(f"Chi-squared statistic: {chi2}")
print(f"P-value: {p_value}")
print(f"Degrees of freedom: {dof}")

alpha=0.05
if p_value < alpha:
    print("Reject the null hypothesis")
else:
    print("Fail to reject the null hypothesis")

df.head()

```

Which statistical test have you done to obtain P-Value?

We performed Chi-square Test for this hypothesis testing to obtain P-value.

Why did you choose the specific statistical test?

A chi-square test of independence can be used to assess whether there is a significant relationship between the traveller type and the recommended status.

✓ Hypothetical Statement - 3

1. State Your research hypothesis as a null hypothesis and alternate hypothesis.

Null Hypothesis (H0): The seat comfort ratings are the same across different cabin classes.

Alternative Hypothesis (H1): There is a significant difference in seat comfort ratings among different cabin classes.

2. Perform an appropriate statistical test.

```

f_statistic, p_value = f_oneway(df['seat_comfort'][df['cabin'] == 'First Class'],
                                df['seat_comfort'][df['cabin'] == 'Business Class'],
                                df['seat_comfort'][df['cabin'] == 'Premium Economy'],
                                df['seat_comfort'][df['cabin'] == 'Economy Class'])

alpha=0.05

print("\nResults:")
print(f"F-statistic: {f_statistic}")
print(f"P-value: {p_value}")

```

```
if p_value < alpha:
    print("\nResults:Reject the null hypothesis")
else:
    print("\nResults:Fail to reject the null hypothesis")
```

Which statistical test have you done to obtain P-Value?

We performed One-way ANOVA Test for this hypothesis testing to obtain P-value.

Why did you choose the specific statistical test?

A one way ANOVA test is used to compare the means of seat comfort ratings in different cabin classes.

✓ 6. Feature Engineering & Data Pre-processing

1. Handling Missing Values

We have already handled the missing values in exploratory data analysis.

What all missing value imputation techniques have you used and why did you use those techniques?

2. Handling Outliers

What all outlier treatment techniques have you used and why did you use those techniques?

The dataset had no outliers so there was no need of handling as such.

3. Categorical Encoding

```
label_encode = LabelEncoder()
df['recommend'] = label_encode.fit_transform(df['recommend'])

df['cabin'].unique()
```



```
original_encoder=ce.OrdinalEncoder(cols=[{'col':'cabin','mapping' : 'Business Class': 3,
                                         'Premium Economu': 2, 'First Class'}}])
df['cabin']=original_encoder.fit_transform(df['cabin'])
one_hot_encoder = ce.OneHotEncoder(cols='traveller_type')
df = one_hot_encoder.fit_transform(df)
```

What all categorical encoding techniques have you used & why did you use those techniques?

As we can not give categorical values in machine learning model so we need to encode them with numerical values . We have use different techniques of encoding for different columns

For the "Traveller_Type" column, which appears to represent categorical data with different types of travelers (e.g., Solo Leisure), it's appropriate to use one-hot encoding. One-hot encoding is commonly used for categorical variables with multiple levels, where each level is treated as a distinct category.

For "Cabin" column : There is a clear ordinal relationship, where the different cabin classes have a meaningful and consistent order (e.g., Economy < Premium Economy < Business < First Class), then ordinal encoding could be a suitable choice. In this case, each category is assigned a numerical value based on its order.

For "recommended" column : Since we have two categories like "Yes" and "No, we used label encoding. Label encoding involves assigning a numerical label to each unique category. For "Yes" and "No," you could encode them as 1 and 0, respectively.

4. Feature Manipulation & Selection

1. Feature Manipulation

There is no such feature manipulation is being done.

2. Feature Selection

```
df[['overall_rating', 'seat_comfort', 'food_bev', 'cabin_service', 'entertainment', 'grou
```

```
df.drop('overall_rating', axis=1, inplace=True)
```

```
df.drop(columns = ['review_date', 'month_name', 'month', 'departure_date', 'airline'], inplace
```

What all feature selection methods have you used and why?

Feature selection is the process of choosing a subset of relevant features (variables, predictors) for use in model construction

We basically used filter method which evaluate the relevance of features based on statistical measures or scores calculated independently of the machine learning algorithm.

Here we used Correlation coefficient that Measures the linear relationship between two variables. Features with high correlation to the target variable or with other features may be considered redundants and so is the reason we dropped overall_rating column.

Columns like 'review_date' and 'departure_date' represent date or time-related information, which is not directly interpretable by most machine learning algorithms.

Additionally 'month' and 'month_name' are the columns that we made for analysis purpose so model do not require to get trained with these so dropping them too.

Column 'airline' is basically the name of the airline which is again not relevant for the model to get trained on.

5. Data Scaling

Scaling data is used to standardize the range of independent variables or features. It's particularly useful when features have different scales or units of measurement, and here the features are on the same scale so there is no need of scaling.

6. Dimesionality Reduction

Do you think that dimensionality reduction is needed? Explain Why?

Dimensionality reduction resuces computation and overfitting and improve model performance. It also helps to eliminate irrelevant or redundant features, allowing models to focus on the most informative aspects of the data.

```
pca = PCA()
airline_pca = pca.fit_transform(df.iloc[:, :-1])

airline_pca_df = pd.DataFrame(data=airline_pca, columns=[f'PC{i+1}' for i in range(len(d
airline_pca_df.head()

explained_variance = pca.explained_variance_ratio_

# Plotting the explained variance
plt.figure(figsize=(10, 4))
op=sns.barplot(x=range(1, len(explained_variance) + 1), y=explained_variance*100, hue=ran
```

```

plt.ylabel('Variance Percentage')
plt.xlabel('Principal Component')
plt.title('PCA Explained Variance Percentage')
for i, num in enumerate(explained_variance):
    op.text(i, num * 100+1, f'{num*100:.2f}',ha='center')
plt.ylim(0 , 90)
plt.xlim(-1 , 12)
plt.show()

pca.explained_variance_ratio_

for i in range(1,len(explained_variance)):
    print(f'Sum of percentage of variance of {i} columns',round(sum(explained_variance[0:i]

pca_2 = PCA(n_components=6)
airline_pca_2 = pca_2.fit_transform(df.iloc[:, :-1])

# Shape after reduction
print("Original Shape:", df.iloc[:, :-1].shape)
print("Transformed Shape:", airline_pca_2.shape)

airline_pca_2_df = pd.DataFrame(data=airline_pca_2, columns=[f'PC{i+1}' for i in range(1e

airline_pca_2_df.head(3)

```

7. Data Splitting

```

x = airline_pca_2_df
y = df.iloc[:, -1]

```

x

y

8. Handling Imbalanced Dataset

Do you think the dataset is imbalanced? Explain Why.

This dataset is no imbalanced dataset. Because in this dataset, Total samples = 12,338 + 11,268 = 23,606, that means Class 0 = ~52.3% and Class 1 = ~47.7% which is almost balanced.

✓ 7. ML Model Implementation

Data Splitting

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
```

```
#Shape of splitted datasets
print('Shape of X_train',X_train.shape)
print('Shape of y_train',y_train.shape)
print('Shape of X_test',X_test.shape)
print('Shape of y_test',y_test.shape)
```

✓ ML Model - 1 - Decision Tree

```
model = DecisionTreeClassifier()

# Fit the Algorithm

model.fit(X_train, y_train)

# Predict on the model

prediction = model.predict(X_test)
```

Checking the evaluation metrics on test data.

```
print("Accuracy on test data:", accuracy_score(y_test, prediction))
print("Precision on test data:", precision_score(y_test, prediction))
print("Recall on test data:", recall_score(y_test, prediction))
print("F1_score on test data:", f1_score(y_test, prediction))
```

Fit the model on training data and prediction on training data dataset.

```
model1 = DecisionTreeClassifier()
model1.fit(X_train, y_train)

# Predict on the model

prediction1 = model1.predict(X_train)
```

Checking the evaluation metrics on training data.

```
print("Accuracy on training data:", accuracy_score(y_train, prediction1))
print("Precision on training data:", precision_score(y_train, prediction1))
print("Recall on training data:", recall_score(y_train, prediction1))
print("F1_score on training data:", f1_score(y_train, prediction1))
```

Here we are making two variables to get the f1 score of training and test data.

```
f1_dt_test = f1_score(y_test, prediction)
f1_dt_train = f1_score(y_train, prediction1)
```

Making a dataframe on these two variables.

```
dt_df= pd.DataFrame([{'Decision_Tree_Test':f1_dt_test,'Decision_Tree_Train':f1_dt_train}])
```

```
graph=sns.barplot(dt_df)
for bar in graph.containers:
    graph.bar_label(bar)
plt.title('Comparison of F1 score training and testing')
plt.ylim(0.8,1)
plt.yticks([i/100 for i in range(80,102,2)])
plt.xlabel('Model')
plt.ylabel('F1-score')
plt.show()
```

1. Explain the ML Model used and it's performance using Evaluation metric Score Chart.

- Supervised algorithm Decision Tree is used for the classification of airline passenger recommendation for predicting the user will recommend or not recommend the airline according to their flight experience.
- Decision tree is good for explainability but it's not that well suited for prediction but in the end we will compare this model with models so we can get the jist how's it performed on our dataset.
- Plot a visulaization graph for comapring f1 score of training and test.

2. Cross- Validation & Hyperparameter Tuning

```
dt_classifier = DecisionTreeClassifier()

param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [2, 5, 10, 20, 30, 40],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Fit the Algorithm
grid_search = GridSearchCV(dt_classifier, param_grid, cv=5, scoring='f1')
```

```
grid_search.fit(X_train, y_train)

# Predict on the
best_params = grid_search.best_params_

#Check for best parameter
best_params
```

Here we are again fitting the model on training data and check prediction on testing data with the best parameter that we got from CV and hyperparameter tuning.

```
#Model Implementataion
best_dt_classifier = DecisionTreeClassifier(**best_params)
best_dt_classifier.fit(X_train, y_train)

#Predict on the model
prediction2 = best_dt_classifier.predict(X_test)

print("Accuracy on test data:", accuracy_score(y_test, prediction2))
print("Precision on test data:", precision_score(y_test, prediction2))
print("Recall on test data:", recall_score(y_test, prediction2))
print("F1_score on test data:", f1_score(y_test, prediction2))
```

```
#Model Implementataion
best_dt_classifier1 = DecisionTreeClassifier(**best_params)
best_dt_classifier1.fit(X_train, y_train)

#Predict on the model
prediction3 = best_dt_classifier1.predict(X_train)
```

```
#Checking the evaluation metrics on test data
print("Accuracy on test data:", accuracy_score(y_train, prediction3))
print("Precision on test data:", precision_score(y_train, prediction3))
print("Recall on test data:", recall_score(y_train, prediction3))
print("F1_score on test data:", f1_score(y_train, prediction3))
```

Making a confusion matrix to get an idea how well our model predicted

```
#Plot the matrix
cfu = confusion_matrix(y_test, prediction2)
ax= plt.subplot()
labels = ['Recommended', 'Not Recommend']
sns.heatmap(cfu, annot=True, ax = ax) #annot=True to annotate cells

# labels, title and ticks
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix')
```

```
ax.xaxis.set_ticklabels(labels)
ax.yaxis.set_ticklabels(labels)
```

- Here we can see that False negative is approx 230 and False Positive is approx 230.
- Making two variables to get the f1 score of both training data and testing data and make a dataframe.

```
#variables for training and testing f1 score and make a dataframe
f1_dt_test_hyp = f1_score(y_test, prediction2)
f1_dt_train_hyp = f1_score(y_train, prediction3)
dt_df_hyp = pd.DataFrame([{'Decision_Tree_Test_Hyp':f1_dt_test_hyp,'Decision_Tree_Train_Hyp':f1_dt_train_hyp}])

# Visualizing evaluation Metric Score chart
graph1=sns.barplot(dt_df_hyp)
for bar in graph1.containers:
    graph1.bar_label(bar)
plt.title('Comparison of F1 score training and testing after tuning')
plt.ylim(0.8,1)
plt.yticks([i/100 for i in range(80,102,2)])
plt.xlabel('Model')
plt.ylabel('F1-score')
plt.show()
```

Which hyperparameter optimization technique have you used and why?

We used here GridSearchCV to get the best parameter of that model because we want an exhaustive search to get those parameter which works best on this model and also we want to try all possible combinations, despite knowing that it takes more computational time we want to experiment this technique so we pass less number of sample space to each parameter so it can execute fast.

Have you seen any improvement? Note down the improvement with updates Evaluation metric Score Chart.

Yes, we saw the improvement after cross validation and hyperparameter tuning before cross-validation and hyperparameter tuning the F1 score on test data is 0.90 and on training data is 0.98 so clearly this type of values is shown overfitting issue on training data, but after cross-validation and hyperparameter tuning we can see that the F1 score on test data is 0.932 and on training data is 0.938, by this we can see that we are able to minimize the overfitting issue by using cross-validation and hyperparameter tuning.

✓ ML Model - 2 : KNN

```

knn = KNeighborsClassifier()
# Train the classifier
knn.fit(X_train, y_train)

# Make predictions on the test set
y_pred_knn = knn.predict(X_test)

print("Accuracy on test data:", accuracy_score(y_test, y_pred_knn))
print("Precision on test data:", precision_score(y_test, y_pred_knn))
print("Recall on test data:", recall_score(y_test, y_pred_knn))
print("F1_score on test data:", f1_score(y_test,y_pred_knn))

knn1 = KNeighborsClassifier()
# Train the classifier
knn1.fit(X_train, y_train)

# Make predictions on the test set
y_pred_knn1 = knn1.predict(X_train)

print("Accuracy on training data:", accuracy_score(y_train, y_pred_knn1))
print("Precision on training data:", precision_score(y_train, y_pred_knn1))
print("Recall on training data:", recall_score(y_train, y_pred_knn1))
print("F1_score on training data:", f1_score(y_train,y_pred_knn1))

```

1. Explain the ML Model used and it's performance using Evaluation metric Score Chart.

- KNN foundational machine learning algorithm is used here to compare between F1 score of training and testing dataset that can be used for both classification and regression.
- It can be used for both classification (predicting categorical labels) and regression (predicting continuous values).
- It identifies the "K" nearest neighbors from F1 score training data to the nearest F1 score test data.

```

f1_knn_test = f1_score(y_test,y_pred_knn)
f1_knn_train = f1_score(y_train,y_pred_knn1)
knn_df= pd.DataFrame([{'Knn_Test':f1_knn_test,'Knn_Train':f1_knn_train}])
knn_grf=sns.barplot(knn_df)
for bar in knn_grf.containers:
    knn_grf.bar_label(bar)
plt.title('Comparison of F1 score training and testing')
plt.ylim(0.8,1)
plt.yticks([i/100 for i in range(80,102,2)])
plt.show()

```

2. Cross- Validation & Hyperparameter Tuning


```
param_grid = {
    'n_neighbors': [3, 5, 7, 9], # Number of neighbors
    'weights': ['uniform', 'distance'], # Weighting scheme
    'metric': ['euclidean', 'manhattan'] # Distance metric
}

# Instantiate the KNN classifier
knn = KNeighborsClassifier()

# Instantiate GridSearchCV
grid_search1 = GridSearchCV(knn, param_grid, cv=5, scoring='f1')
grid_search1.fit(X_train, y_train)

# Retrieve the best parameters and best score
best_params = grid_search1.best_params_
best_score = grid_search1.best_score_

print("Best Parameters:", best_params)
print("Best Score:", best_score)

# Evaluate the best model on the testing set
best_model = grid_search1.best_estimator_
test_score = best_model.score(X_test, y_test)
print("Test Set Score:", test_score)

curr_parm_knn = best_model.get_params()
curr_parm_knn

knn1_hy = KNeighborsClassifier(**curr_parm_knn)
knn1_hy.fit(X_train, y_train)

# Make predictions on the test set
y_pred_knn1_hy = knn1_hy.predict(X_test)

print("Accuracy on test data:", accuracy_score(y_test, y_pred_knn1_hy))
print("Precision on test data:", precision_score(y_test, y_pred_knn1_hy))
print("Recall on test data:", recall_score(y_test, y_pred_knn1_hy))
print("F1_score on test data:", f1_score(y_test, y_pred_knn1_hy))

knn1_hy1 = KNeighborsClassifier(**curr_parm_knn)

# Train the classifier
knn1_hy1.fit(X_train, y_train)

# Make predictions on the test set
y_pred_knn1_hy1 = knn1_hy1.predict(X_train)

print("Accuracy on training data:", accuracy_score(y_train, y_pred_knn1_hy1))
print("Precision on training data:", precision_score(y_train, y_pred_knn1_hy1))
```

```

print("Recall on training data:", recall_score(y_train, y_pred_knn1_hy1))
print("F1_score on training data:", f1_score(y_train,y_pred_knn1_hy1))

cfu1 = confusion_matrix(y_test, y_pred_knn1_hy)
ax= plt.subplot()
labels = ['Recommended', 'Not Recommend']
sns.heatmap(cfu1, annot=True, ax = ax) #annot=True to annotate cells

# labels, title and ticks
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix')
ax.xaxis.set_ticklabels(labels)
ax.yaxis.set_ticklabels(labels)

f1_knn_test_hyp = f1_score(y_test,y_pred_knn1_hy)
f1_knn_train_hyp = f1_score(y_train,y_pred_knn1_hy1)
knn_df_hyp= pd.DataFrame([{'Knn_Test_hyp':f1_knn_test_hyp, 'Knn_Train_hyp':f1_knn_train_hy
knn_grf_hyp=sns.barplot(knn_df_hyp)
for bar in knn_grf_hyp.containers:
    knn_grf_hyp.bar_label(bar)
plt.title('Comparison of F1 score training and testing after tuning')
plt.ylim(0.8,1)
plt.yticks([i/100 for i in range(80,102,2)])
plt.show()

```

Which hyperparameter optimization technique have you used and why?

Grid Search hyperparameter optimization technique have been used here. It systematically searches for the optimal combination of hyperparameters while improving model accuracy and generalization. It automates the process, ensuring that all combinations within a defined range are tested, leading to better predictions and classifications.

Have you seen any improvement? Note down the improvement with updates Evaluation metric Score Chart.

We can find the best parameters in the dataset, the highest test score values and also can assess the range of test set score in between true labels of recommended and non-recommended values. By using this technique, we can create a confusion matrix that is showed through heatmap.

3. Explain each evaluation metric's indication towards business and the business impact pf the ML model used.

After tuning the dataset, through the comparative analysis of F1 score training and testing dataset, we can infer that there is a very small difference between the two parameters assessing that training hypothesis is more accurate than testing hypothesis. So, in business perspective it can be said that create a training program including dataset will be a more precise decision before going to implement in real-world scenario.

✓ ML Model - 3 : SVM

We are Fitting our model on training data and predict on test data

```
svm_model = SVC(kernel='rbf', gamma = 'auto', random_state=42)
svm_model.fit(X_train,y_train)
```

```
# Predict on the model
pred_svm = svm_model.predict(X_test)
```

Checking evaluation metrics on testing data

```
print("Accuracy on test data:", accuracy_score(y_test, pred_svm))
print("Precision on test data:", precision_score(y_test, pred_svm))
print("Recall on test data:", recall_score(y_test, pred_svm))
print("F1_score on test data:", f1_score(y_test,pred_svm))
```

```
pred_svm1 = svm_model.predict(X_train)
```

Checking evaluation metrics on training data

```
print("Accuracy on training data:", accuracy_score(y_train, pred_svm1))
print("Precision on training data:", precision_score(y_train, pred_svm1))
print("Recall on training data:", recall_score(y_train, pred_svm1))
print("F1_score on training data:", f1_score(y_train,pred_svm1))
```

Explain the ML Model used and its performance using Evaluation metric Score Chart.

We have used SVM (Support Vector Machine) model which is best suitable model for classification of Supervised ML and SVMs tend to generalize well and are less prone to overfitting, especially in high-dimensional spaces.

SVMs provide control over the trade-off between achieving a low training error and minimizing the complexity of the decision boundary, allowing us to avoid overfitting.

Plotting the visualization of matrices:

```

f1_svm_test = f1_score(y_test,pred_svm)
f1_svm_train = f1_score(y_train,pred_svm1)
svm_df= pd.DataFrame([{'Svm_Test':f1_svm_test,'Svm_Train':f1_svm_train}])
svm_grf=sns.barplot(svm_df)
for bar in svm_grf.containers:
    svm_grf.bar_label(bar)
plt.title('Comparison of F1 score training and testing')
plt.xlabel('type of data')
plt.ylabel('F1 Score')
plt.ylim(0.8,1)
plt.yticks([i/100 for i in range(80,102,2)])
plt.show()

```

From above Matrix we can see that our F1 Score is 93% on test data and 94% on train data so it suggests our model is performing very good on both training and testing data.

2. Cross- Validation & Hyperparameter Tuning

Applying RandomSerachCV for getting best hyperparameters

```

param_grid_svm = {'C': [0.1, 1, 10], # Regularization parameter
                  'gamma': [0.001, 0.01, 0.1, 1], # Kernel coefficient (only for RBF kernel)
                  'kernel': ['linear', 'rbf']} # Kernel type

# Create an SVM model
svm_model = SVC()

# Perform grid search with 5-fold cross-validation
random_search = RandomizedSearchCV(svm_model, param_grid_svm,verbose=2,random_state = 42)
random_search.fit(X_train, y_train)

best_params_svm = random_search.best_params_
print("Best Hyperparameters:", best_params)
best_svm_model = random_search.best_estimator_

# Evaluate the best model on the test set
y_pred_svm = best_svm_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred_svm)
print("Accuracy:", accuracy)

curr_param_svm = best_svm_model.get_params()
curr_param_svm

```

Fitting the Model with best parameters we got after RandomSearchCV to predict on Test data

```

svm_model = SVC(**curr_param_svm)
svm_model.fit(X_train,y_train)

```

```
pred_svm1_ = svm_model.predict(X_test)
```

Checking evaluation metrics on testing data after tuning

```
print("Accuracy on test data:", accuracy_score(y_test, pred_svm1_))
print("Precision on test data:", precision_score(y_test, pred_svm1_))
print("Recall on test data:", recall_score(y_test, pred_svm1_))
print("F1_score on test data:", f1_score(y_test, pred_svm1_))
```

#Now fitting the Model with best parameter to predict on training data

```
svm_model1 = SVC(**curr_param_svm)
svm_model1.fit(X_train, y_train)
pred_svm2 = svm_model1.predict(X_train)
```

#Checking evaluation metrics on training data after tuning

```
print("Accuracy on training data:", accuracy_score(y_train, pred_svm2))
print("Precision on training data:", precision_score(y_train, pred_svm2))
print("Recall on training data:", recall_score(y_train, pred_svm2))
print("F1_score on training data:", f1_score(y_train, pred_svm2))
```

Plotting a Confusion matrix to check the model prediction after tuning

```
cfu2 = confusion_matrix(y_test, pred_svm1_)
ax = plt.subplot()
labels = ['Recommended', 'Not Recommend']
sns.heatmap(cfu2, annot=True, ax=ax) #annot=True to annotate cells

# labels, title and ticks
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix')
ax.xaxis.set_ticklabels(labels)
ax.yaxis.set_ticklabels(labels)
```

From the confusion matrix we can say our model is giving 3500 true prediction but 200 False Positive and 230 False Negative.

Which hyperparameter optimization technique have you used and why?

We have used RandomSearchCV for hyperparameter tuning of this model because it gives accurate hyperparameters and Random search is more computationally efficient.

plotting the matrix after Hyperparameter tuning

```

f1_svm_test_hyp = f1_score(y_test, pred_svm1_)
f1_svm_train_hyp = f1_score(y_train, pred_svm2)
svm_df_hyp= pd.DataFrame([{'Svm_Test_Hyp':f1_svm_test_hyp, 'Svm_Train_Hyp':f1_svm_train_hyp}])
svm_grf_hyp=sns.barplot(svm_df_hyp)
for bar in svm_grf_hyp.containers:
    svm_grf_hyp.bar_label(bar)
plt.title('Comparison of F1 score on training and testing data after tuning')
plt.xlabel('type of data')
plt.ylabel('F1 Score')
plt.ylim(0.8,1)
plt.yticks([i/100 for i in range(80,102,2)])
plt.show()

```

Have you seen any improvement? Note down the improvement with updates Evaluation metric Score Chart.

Yes, After hyperparameter tuning of our model we can clearly see that There is some improvement in F1 score and are giving same results on both training and testing data which ranges between 93-94 % .

so we can conclude that our model is generalising even better after RandomSearchCV and giving good results.

✓ ML Model - 4 : Random Forest

```

rf = RandomForestClassifier()

rf.fit(X_train,y_train)

y_pred_rf =rf.predict(X_test)

print("Accuracy on test data:", accuracy_score(y_test, y_pred_rf))
print("Precision on test data:", precision_score(y_test, y_pred_rf))
print("Recall on test data:", recall_score(y_test, y_pred_rf))
print("F1_score on test data:", f1_score(y_test,y_pred_rf))

rf1 = RandomForestClassifier()

# Fit the Algorithm
rf1.fit(X_train,y_train)

```