## **Державний університет інформаційно-комунікаційних технологій**

**Практична робота №5**

**з дисципліни: Технології штучного інтелекту**

**Тема: Машинні моделі в задачах прогнозування за методами лінійної та логістичної регресії**

Виконав:  
студент ДУІКТ  
Тертишний В.Ю.

група: ШІДМ-51

м.Київ

**Мета:** Ознайомитися з основними поняттями лінійної регресії та методами її застосування для прогнозування. Навчитися будувати та аналізувати модель лінійної регресії, визначати залежність між змінними, проводити оцінку якості моделі та робити прогнозування на основі отриманих результатів. Практично закріпити навички використання інструментів програмування Python та бібліотеки scikit-learn для реалізації моделі лінійної регресії на реальних даних.

**Очищення даних**

Для виконання мого варіанту – 18 було обрано набір даних Hotel Booking Demand.

#setting up

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn

import missingno as msno

import kagglehub

%matplotlib inline

# Download latest version

path = kagglehub.dataset\_download("jessemostipak/hotel-booking-demand")

df = pd.read\_csv(path + '/hotel\_bookings.csv')

df.head(10)

Перевіряємо значення колонок

for col in df.columns:

print(f"{col}: {df[col].nunique()}")

df.info()

Шукаємо відсутні значення

df.isna().sum()

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

agent 16340

company 112593

...

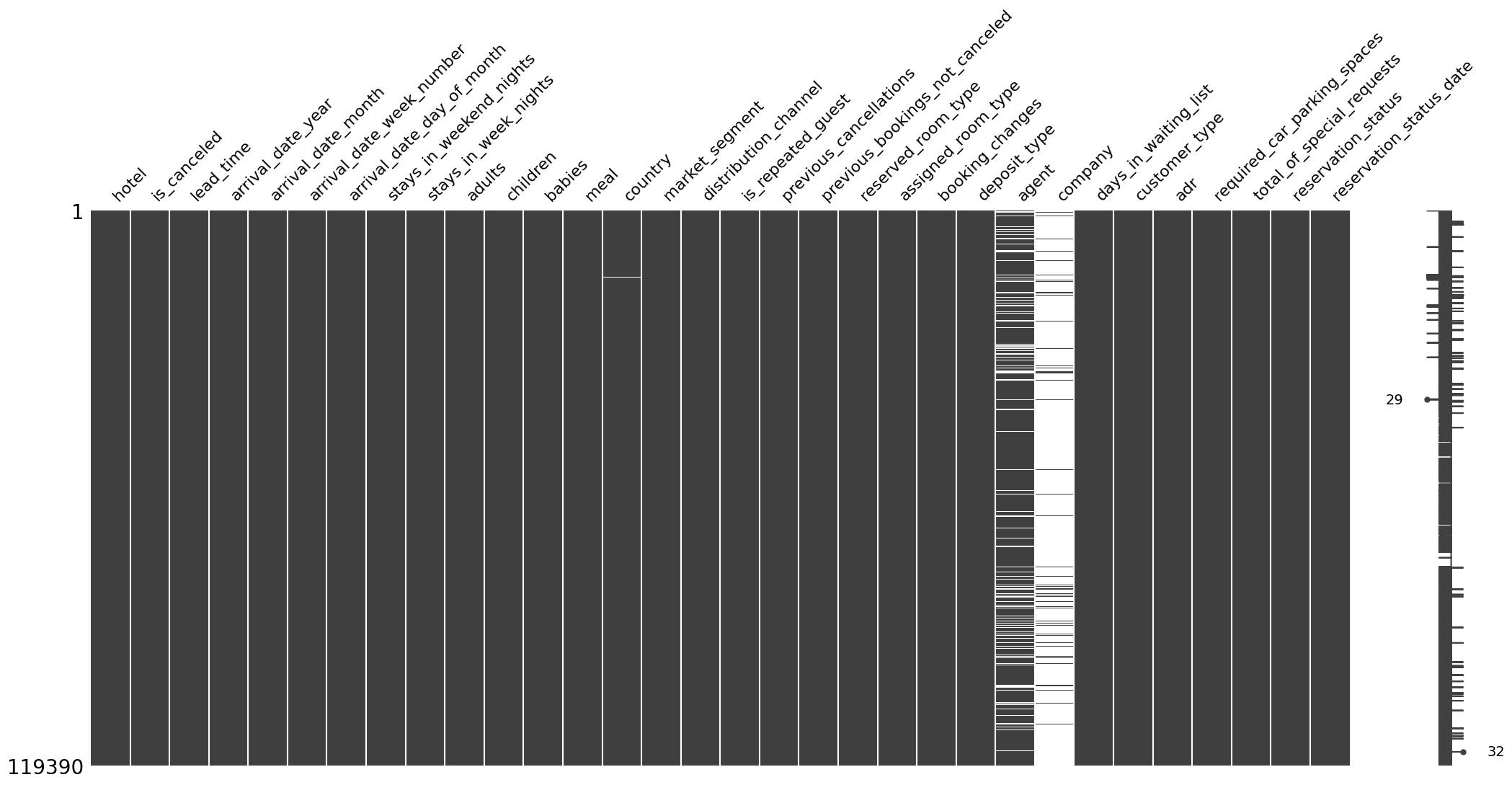
required\_car\_parking\_spaces 0

total\_of\_special\_requests 0

reservation\_status 0

reservation\_status\_date 0

msno.matrix(df)



Прибираємо рядки які не містять інформацію про гостей

df['children'] = df['children'].fillna(0)

# Remove rows where there are no guests

df = df.query('not (children == 0 and adults == 0 and babies == 0)')

Кодуємо категоріальні значення

from sklearn.preprocessing import LabelEncoder

le\_hotel = LabelEncoder()

le\_month = LabelEncoder()

le\_meal = LabelEncoder()

le\_country = LabelEncoder()

le\_market = LabelEncoder()

le\_dist = LabelEncoder()

le\_res = LabelEncoder()

le\_ass = LabelEncoder()

le\_deposit = LabelEncoder()

le\_custt = LabelEncoder()

le\_cancel = LabelEncoder()

le\_reserve = LabelEncoder()

df\_class['hotel'] = le\_hotel.fit\_transform(df\_class['hotel'])

df\_class['arrival\_date\_month'] = le\_month.fit\_transform(df\_class['arrival\_date\_month'])

df\_class['meal'] = le\_meal.fit\_transform(df\_class['meal'])

df\_class['country'] = le\_country.fit\_transform(df\_class['country'])

df\_class['market\_segment'] = le\_market.fit\_transform(df\_class['market\_segment'])

df\_class['distribution\_channel'] = le\_dist.fit\_transform(df\_class['distribution\_channel'])

df\_class['reserved\_room\_type'] = le\_res.fit\_transform(df\_class['reserved\_room\_type'])

df\_class['assigned\_room\_type'] = le\_ass.fit\_transform(df\_class['assigned\_room\_type'])

df\_class['deposit\_type'] = le\_deposit.fit\_transform(df\_class['deposit\_type'])

df\_class['customer\_type'] = le\_custt.fit\_transform(df\_class['customer\_type'])

df\_class['reservation\_status'] = le\_cancel.fit\_transform(df\_class['reservation\_status'])

df\_class['reservation\_status\_date'] = le\_reserve.fit\_transform(df\_class['reservation\_status\_date'])

Використовуємо Класифікатор випадкових лісів для заповнення пропущених значень у колонці agent

#random forest classifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

known\_data = df\_class[df\_class['agent'].notnull()]

miss\_data = df\_class[df\_class['agent'].isnull()]

X = known\_data.drop(columns = ['agent'])

y = known\_data['agent']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 36)

model = RandomForestClassifier(n\_estimators = 100, random\_state=36)

model.fit(X\_train, y\_train)

X\_miss = miss\_data.drop(columns = ['agent'])

agent\_pred = model.predict(X\_miss)

df\_class.loc[miss\_data.index,'agent'] = agent\_pred

df\_class['agent'].isnull().sum()

np.int64(0)

#splitting data into known(notnull) and miss (isnull)

known\_data1 = df\_class[df\_class['company'].notnull()]

miss\_data1 = df\_class[df\_class['company'].isnull()]

#splitting known data in X and y to fit

X = known\_data1.drop(columns = ['company'])

y = known\_data1['company']

#split known data to train and test to fit

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 40)

#create model and fit

from sklearn.ensemble import RandomForestClassifier

model1 = RandomForestClassifier(n\_estimators = 100, random\_state =40)

model1.fit(X\_train, y\_train)

#to predict missing data

x\_miss1 = miss\_data1.drop(columns = ['company'])

comp\_pred = model1.predict(x\_miss1)

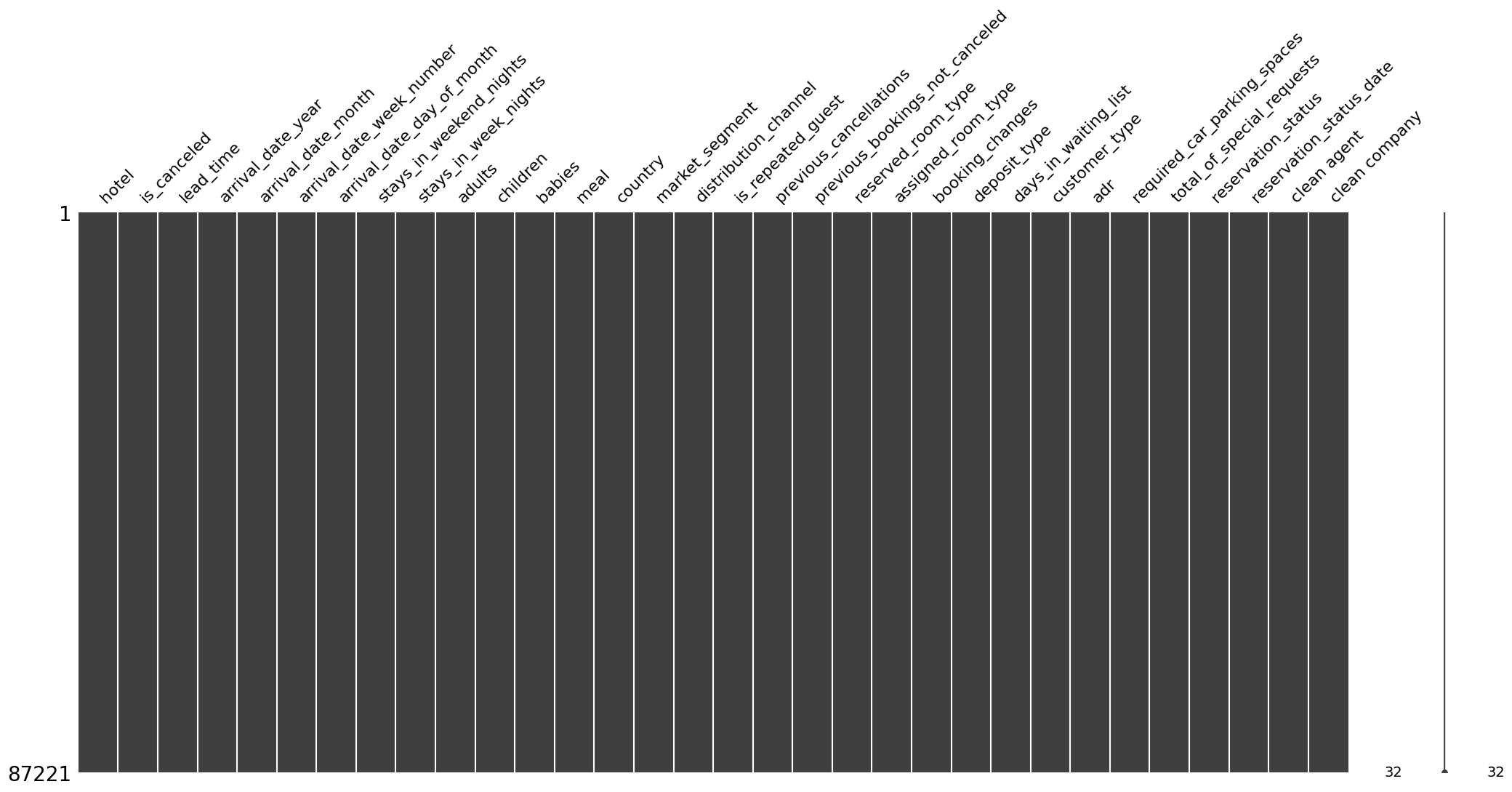
#put it back to df\_class

df\_class.loc[miss\_data1.index,'company'] = comp\_pred

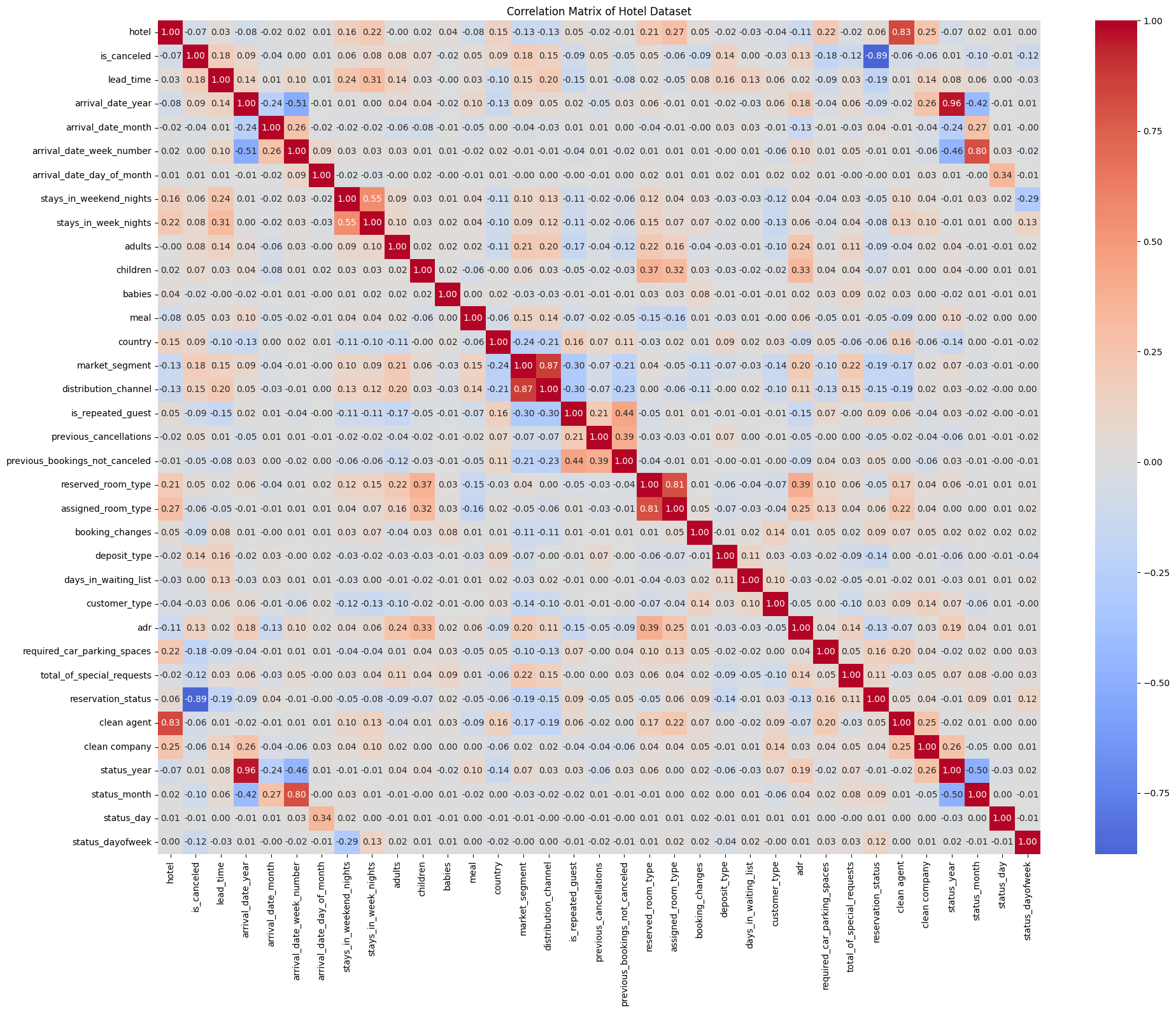
Вирішуємо проблеми з іншими колонками

cleaned\_df['country'] = cleaned\_df['country'].fillna("Unknown")

msno.matrix(cleaned\_df)



Використовуючи кореляційну матрицю визначаємо атрибути, які несуттєво впливають на показник is\_cancelled та прибираємо їх з набору даних.



df = cleaned\_df.copy()

# Binary encoding for hotel (0 for 'City Hotel', 1 for 'Resort Hotel')

df['hotel'] = (df['hotel'] == 'Resort Hotel').astype(int)

# Label encoding for categorical columns

from sklearn.preprocessing import LabelEncoder

categorical\_columns = ['country', 'meal', 'market\_segment', 'distribution\_channel',

'reserved\_room\_type', 'assigned\_room\_type', 'deposit\_type',

'customer\_type', 'reservation\_status', 'arrival\_date\_month']

# Create a dictionary to store the encoders

encoders = {}

# Encode each categorical column

for col in categorical\_columns:

encoders[col] = LabelEncoder()

df[col] = encoders[col].fit\_transform(df[col])

# Convert reservation\_status\_date to datetime and extract useful features

df['reservation\_status\_date'] = pd.to\_datetime(df['reservation\_status\_date'])

df['status\_year'] = df['reservation\_status\_date'].dt.year

df['status\_month'] = df['reservation\_status\_date'].dt.month

df['status\_day'] = df['reservation\_status\_date'].dt.day

df['status\_dayofweek'] = df['reservation\_status\_date'].dt.dayofweek

# Drop the original date column

df = df.drop('reservation\_status\_date', axis=1)

# Now create the correlation matrix

correlation\_matrix = df.corr()

print(correlation\_matrix['is\_canceled'])

# Create a larger figure size for better readability

plt.figure(figsize=(20, 16))

# Create heatmap

seaborn.heatmap(correlation\_matrix,

annot=True,

cmap='coolwarm',

center=0,

fmt='.2f')

plt.title('Correlation Matrix of Hotel Dataset')

plt.tight\_layout()

useless\_col = ['status\_year', 'status\_day', 'arrival\_date\_month', 'arrival\_date\_year', 'arrival\_date\_week\_number',

'reservation\_status', 'country', 'days\_in\_waiting\_list']

df = df.drop(columns=useless\_col)

Завантажуємо підготовлені дані в окремий файл

df.to\_csv('cleaned\_hotel\_data.csv', index=False)

В результаті проведеної роботи було створено якісний набір даних, готовий для подальшого аналізу та моделювання. Дані були очищені від шуму, стандартизовані та приведені до формату, зручного для подальшої обробки.

**Аналіз даних готелів та побудова моделей машинного навчання**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn

import missingno as msno

df = pd.read\_csv('cleaned\_hotel\_data.csv')

df.head()

X = df.drop(columns=['is\_canceled'])

y = df['is\_canceled']

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.30)

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

# Initialize and train the logistic regression model

lr = LogisticRegression(max\_iter=1000) # increased max\_iter to ensure convergence

lr.fit(X\_train, y\_train)

# Make predictions

y\_pred\_lr = lr.predict(X\_test)

# Evaluate the model

acc\_lr = accuracy\_score(y\_test, y\_pred\_lr)

conf = confusion\_matrix(y\_test, y\_pred\_lr)

clf\_report = classification\_report(y\_test, y\_pred\_lr)

print(f"Accuracy Score of Logistic Regression is : {acc\_lr}")

print(f"Confusion Matrix : \n{conf}")

print(f"Classification Report : \n{clf\_report}")

Розділяємо вибірки на навчальну та тестову, в якості цільового атрибута обираємо is\_cancelled. Таким чином наша модель буде прогнозувати ймовірність того чи буде бронювання скасоване клієнтом. Отримані результати:

Accuracy Score of Logistic Regression is : 0.795658654029885

Confusion Matrix :

[[17855 1068]

[ 4279 2965]]

Classification Report :

precision recall f1-score support

0 0.81 0.94 0.87 18923

1 0.74 0.41 0.53 7244

accuracy 0.80 26167

macro avg 0.77 0.68 0.70 26167

weighted avg 0.79 0.80 0.77 26167

Показник точності становить **0.79** що є перспективним, але недостатнім значення для використання моделі у майбутньому.

Пробуємо використати інший вид моделі – ANN

from tensorflow.keras.utils import to\_categorical

y = to\_categorical(df['is\_canceled'])

X = df.drop(columns=['is\_canceled'])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.30)

import keras

from keras.models import Sequential

from keras.layers import Dense

model = Sequential()

model.add(Dense(100, activation = 'relu', input\_shape = (26, )))

model.add(Dense(100, activation = 'relu'))

model.add(Dense(2, activation = 'sigmoid'))

model.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

model\_history = model.fit(X\_train, y\_train, validation\_data = (X\_test, y\_test),

epochs = 100)

В результаті

Epoch 100/100

**1908/1908** ━━━━━━━━━━━━━━━━━━━━ **2s** 869us/step - accuracy: 0.9202 - loss: 0.2041 - val\_accuracy: 0.9025 - val\_loss: 0.2890

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error

# Обчислення MSE та MAE

mse = mean\_squared\_error(y\_test, y\_pred\_lr)

mae = mean\_absolute\_error(y\_test, y\_pred\_lr)

print(f"Mean Squared Error (MSE): {mse:.4f}")

print(f"Mean Absolute Error (MAE): {mae:.4f}")

Було досягнуто точність 0.9025. Що покращує попередній показник, однак залишає нам простір до покращення моделі та підходів.

**Висновки**

В рамках виконання цієї мети було ознайомлено з основними поняттями лінійної регресії, яка є важливим інструментом для аналізу залежностей між змінними та прогнозування результатів. Було вивчено основні етапи побудови моделі лінійної регресії, починаючи від підготовки даних і закінчуючи оцінкою її ефективності. Зокрема, було засвоєно, як визначати залежність між незалежними та залежними змінними за допомогою моделі, а також як оцінювати її точність і придатність для прогнозування на основі метрик, таких як середньоквадратична та середня абсолютна помилка. Крім того, було набуте практичне вміння працювати з інструментами програмування Python та бібліотекою scikit-learn, що дозволяє ефективно будувати, аналізувати та оцінювати моделі лінійної регресії на реальних даних. Важливою частиною процесу стало розуміння якості моделей, їх здатності робити прогнози на основі реальних наборів даних, а також можливості застосування таких моделей для вирішення практичних задач у різних галузях.