1. Which are the top three variables in your model which contribute most towards the probability of a lead getting converted?

ANS) import pandas as pd

import numpy as np

from sklearn.linear\_model import LogisticRegression

# Generate imaginary data

np.random.seed(0)

n\_samples = 100

# Simulating features (X) and target variable (y)

data = {

'Lead Source': np.random.choice(['Google', 'Organic Search', 'Direct Traffic'], size=n\_samples),

'TotalVisits': np.random.randint(1, 20, size=n\_samples),

'Specialization': np.random.choice(['Finance', 'Marketing', 'Operations'], size=n\_samples),

'Converted': np.random.choice([0, 1], size=n\_samples) # Binary target variable (0 = not converted, 1 = converted)

}

# Create DataFrame from simulated data

df = pd.DataFrame(data)

# One-hot encode categorical variables

df\_encoded = pd.get\_dummies(df, columns=['Lead Source', 'Specialization'], drop\_first=True)

# Separate features (X) and target variable (y)

X = df\_encoded.drop('Converted', axis=1)

y = df\_encoded['Converted']

# Initialize and fit the logistic regression model

model = LogisticRegression()

model.fit(X, y)

# Retrieve coefficients and feature names

coefficients = model.coef\_[0]

feature\_names = X.columns.tolist()

# Create DataFrame to store coefficients and corresponding feature names

coef\_df = pd.DataFrame({'Feature': feature\_names, 'Coefficient': coefficients})

# Sort coefficients by absolute value to identify top contributing variables

coef\_df['Absolute\_Coefficient'] = coef\_df['Coefficient'].abs()

top\_three\_variables = coef\_df.sort\_values(by='Absolute\_Coefficient', ascending=False).head(3)

# Display the top three variables contributing most towards lead conversion

print("Top Three Variables Contributing Most Towards Lead Conversion:")

print(top\_three\_variables[['Feature', 'Coefficient']])

2What are the top 3 categorical/dummy variables in the model which should be focused the most on in order to increase the probability of lead conversion?

ANS) import pandas as pd

import numpy as np

from sklearn.linear\_model import LogisticRegression

# Generate imaginary data

np.random.seed(0)

n\_samples = 100

# Simulating categorical variables and target variable

data = {

'Lead\_Source': np.random.choice(['Google', 'Organic Search', 'Direct Traffic', 'Referral'], size=n\_samples),

'Lead\_Origin': np.random.choice(['API', 'Landing Page Submission', 'Website'], size=n\_samples),

'Specialization': np.random.choice(['Finance', 'Marketing', 'Operations', 'HR'], size=n\_samples),

'Converted': np.random.choice([0, 1], size=n\_samples) # Simulated target variable (0 = not converted, 1 = converted)

}

# Create DataFrame from simulated data

df = pd.DataFrame(data)

# One-hot encode categorical variables

df\_encoded = pd.get\_dummies(df, columns=['Lead\_Source', 'Lead\_Origin', 'Specialization'], drop\_first=True)

# Separate features (X) and target variable (y)

X = df\_encoded.drop('Converted', axis=1)

y = df\_encoded['Converted']

# Initialize and fit a logistic regression model

model = LogisticRegression()

model.fit(X, y)

# Retrieve coefficients and feature names

coefficients = model.coef\_[0]

feature\_names = X.columns.tolist()

# Create a DataFrame to store coefficients and corresponding feature names

coef\_df = pd.DataFrame({'Feature': feature\_names, 'Coefficient': coefficients})

# Sort coefficients by absolute value to identify top contributing variables

coef\_df['Absolute\_Coefficient'] = coef\_df['Coefficient'].abs()

top\_three\_variables = coef\_df.sort\_values(by='Absolute\_Coefficient', ascending=False).head(3)

# Display the top three categorical variables to focus on for lead conversion

print("Top Three Categorical Variables to Focus on for Lead Conversion:")

print(top\_three\_variables[['Feature', 'Coefficient']])

1. X Education has a period of 2 months every year during which they hire some interns. The sales team, in particular, has around 10 interns allotted to them. So during this phase, they wish to make the lead conversion more aggressive. So they want almost all of the potential leads (i.e. the customers who have been predicted as 1 by the model) to be converted and hence, want to make phone calls to as much of such people as possible. Suggest a good strategy they should employ at this stage.

import pandas as pd

import numpy as np

from sklearn.linear\_model import LogisticRegression

# Generate imaginary data

np.random.seed(0)

n\_samples = 100

# Simulating features and target variable

data = {

'Lead\_Source': np.random.choice(['Google', 'Organic Search', 'Direct Traffic', 'Referral'], size=n\_samples),

'TotalVisits': np.random.randint(1, 20, size=n\_samples),

'Specialization': np.random.choice(['Finance', 'Marketing', 'Operations', 'HR'], size=n\_samples),

'Converted': np.random.choice([0, 1], size=n\_samples) # Simulated target variable (0 = not converted, 1 = converted)

}

# Create DataFrame from simulated data

df = pd.DataFrame(data)

# One-hot encode categorical variables

df\_encoded = pd.get\_dummies(df, columns=['Lead\_Source', 'Specialization'], drop\_first=True)

# Separate features (X) and target variable (y)

X = df\_encoded.drop('Converted', axis=1)

y = df\_encoded['Converted']

# Initialize and fit a logistic regression model

model = LogisticRegression()

model.fit(X, y)

# Predict probabilities of conversion for all leads

conversion\_probabilities = model.predict\_proba(X)[:, 1] # Probabilities of lead conversion

# Identify potential leads to target for aggressive conversion

threshold = 0.8 # Set a threshold probability to classify a lead as potential

potential\_leads = df[conversion\_probabilities >= threshold]

# Determine number of phone calls based on available interns (e.g., 10 interns)

num\_interns = 10

num\_calls\_per\_intern = len(potential\_leads) // num\_interns

# Assign leads to interns for phone calls

for i in range(num\_interns):

start\_idx = i \* num\_calls\_per\_intern

end\_idx = (i + 1) \* num\_calls\_per\_intern

intern\_leads = potential\_leads.iloc[start\_idx:end\_idx]

# Perform phone calls to potential leads assigned to each intern

print(f"Intern {i+1} is assigned to call the following potential leads:")

print(intern\_leads)

print() # Add newline for readability

1. Similarly, at times, the company reaches its target for a quarter before the deadline. During this time, the company wants the sales team to focus on some new work as well. So during this time, the company’s aim is to not make phone calls unless it’s extremely necessary, i.e. they want to minimize the rate of useless phone calls. Suggest a strategy they should employ at this stage.

Ans) import pandas as pd

import numpy as np

from sklearn.linear\_model import LogisticRegression

# Generate imaginary data

np.random.seed(0)

n\_samples = 100

# Simulating features and target variable

data = {

'Lead\_Source': np.random.choice(['Google', 'Organic Search', 'Direct Traffic', 'Referral'], size=n\_samples),

'TotalVisits': np.random.randint(1, 20, size=n\_samples),

'Specialization': np.random.choice(['Finance', 'Marketing', 'Operations', 'HR'], size=n\_samples),

'Converted': np.random.choice([0, 1], size=n\_samples) # Simulated target variable (0 = not converted, 1 = converted)

}

# Create DataFrame from simulated data

df = pd.DataFrame(data)

# One-hot encode categorical variables

df\_encoded = pd.get\_dummies(df, columns=['Lead\_Source', 'Specialization'], drop\_first=True)

# Separate features (X) and target variable (y)

X = df\_encoded.drop('Converted', axis=1)

y = df\_encoded['Converted']

# Initialize and fit a logistic regression model

model = LogisticRegression()

model.fit(X, y)

# Predict probabilities of conversion for all leads

conversion\_probabilities = model.predict\_proba(X)[:, 1] # Probabilities of lead conversion

# Set a threshold for filtering unnecessary phone calls (e.g., avoid calling leads with low conversion probability)

threshold = 0.3 # Adjust threshold based on business requirements

# Filter out leads with low conversion probabilities to minimize useless phone calls

filtered\_leads = df[conversion\_probabilities >= threshold]

# Display the filtered leads for potential outreach (if necessary)

print("Filtered Leads for Potential Outreach (Conversion Probability >= {:.2f}):".format(threshold))

print(filtered\_leads)