In [2]:

```
import pandas as pd
import numpy as np
```

Ask AI

In [12]:

```
df = pd.read_csv("C:/Users/donbo/Downloads/marketing_campaign_synthetic_datase
df.head(10)
```

Out[12]:

	age	job	marital	education	balance	housing	loan	duration	campaign	previous
0	56	retired	married	secondary	4213.71	yes	no	960	8	2
1	69	student	divorced	tertiary	781.13	no	yes	670	8	4
2	46	services	divorced	tertiary	4213.27	no	yes	821	6	4
3	32	management	single	primary	-3924.98	yes	no	1366	7	0
4	60	admin.	divorced	primary	4195.71	yes	no	641	6	0
5	25	management	married	secondary	25000.00	no	no	1891	9	0
6	38	NaN	married	tertiary	3304.51	yes	no	838	3	4
7	56	retired	single	tertiary	-2523.84	yes	no	1235	3	4
8	36	retired	married	primary	3537.77	yes	no	715	3	4
9	40	student	married	secondary	1399.76	yes	yes	1881	3	2

In [13]:

```
# Checking missing values

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

Out[13]:

```
age           0
job          10
marital       0
education    10
balance      10
housing       0
loan          0
duration      0
campaign      0
previous      0
poutcome      0
y             0
dtype: int64
```

In [15]:

```
# remove missing values

df = df.dropna()
```

In [16]:

```
# confirm the missing values have been dropped or not

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

Out[16]:

```
age          0
job          0
marital      0
education    0
balance      0
housing      0
loan         0
duration     0
campaign     0
previous     0
poutcome     0
y            0
dtype: int64
```

In [17]:

```
# Check Duplicates

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

Out[17]:

```
5
```

In [18]:

```
# Remove Duplicates

df = df.drop_duplicates()
```

In [19]:

```
# Re-check Duplicates

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

Out[19]:

```
0
```

In [20]:

```
# Convert Target variable

df['y'] = df['y'].map({'yes':1, 'no':0})
```

In [21]:

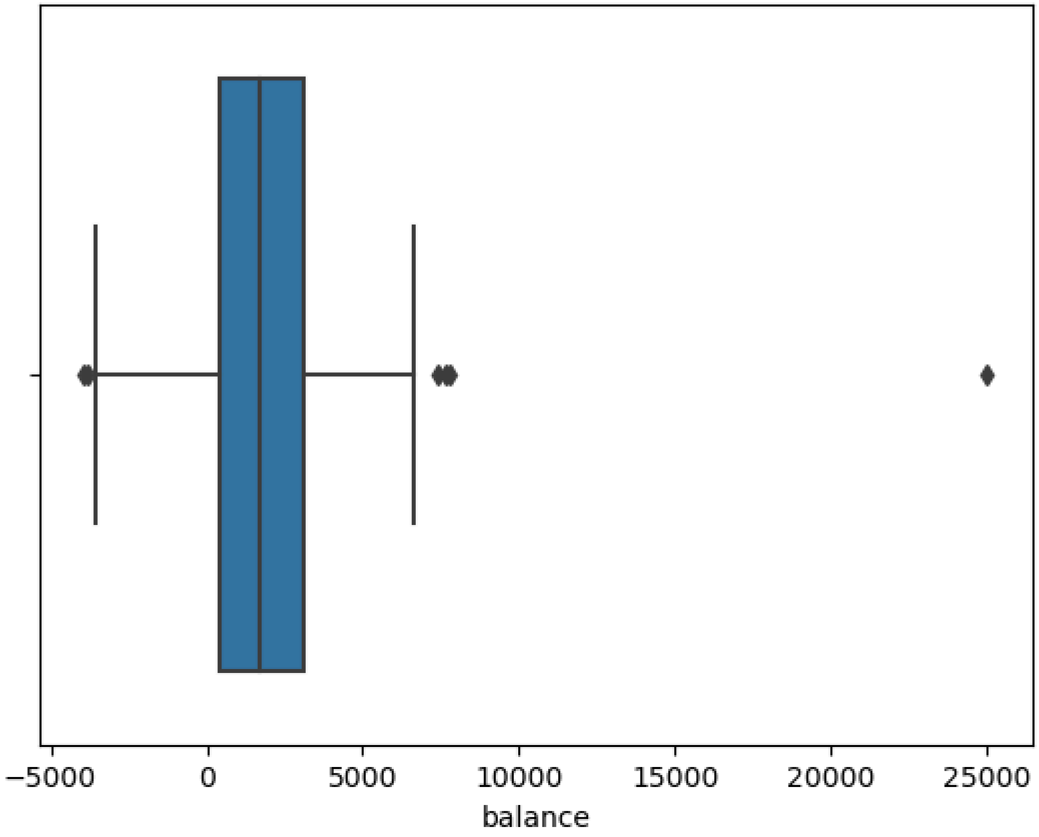
```
# Checking if the conversion was successful or not

df['y'].value_counts()
```

Out[21]:

```
y
0    285
1    105
Name: count, dtype: int64
```

In [22]:

```
# Detect Outlier

import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(x=df['balance'])
plt.show()
```

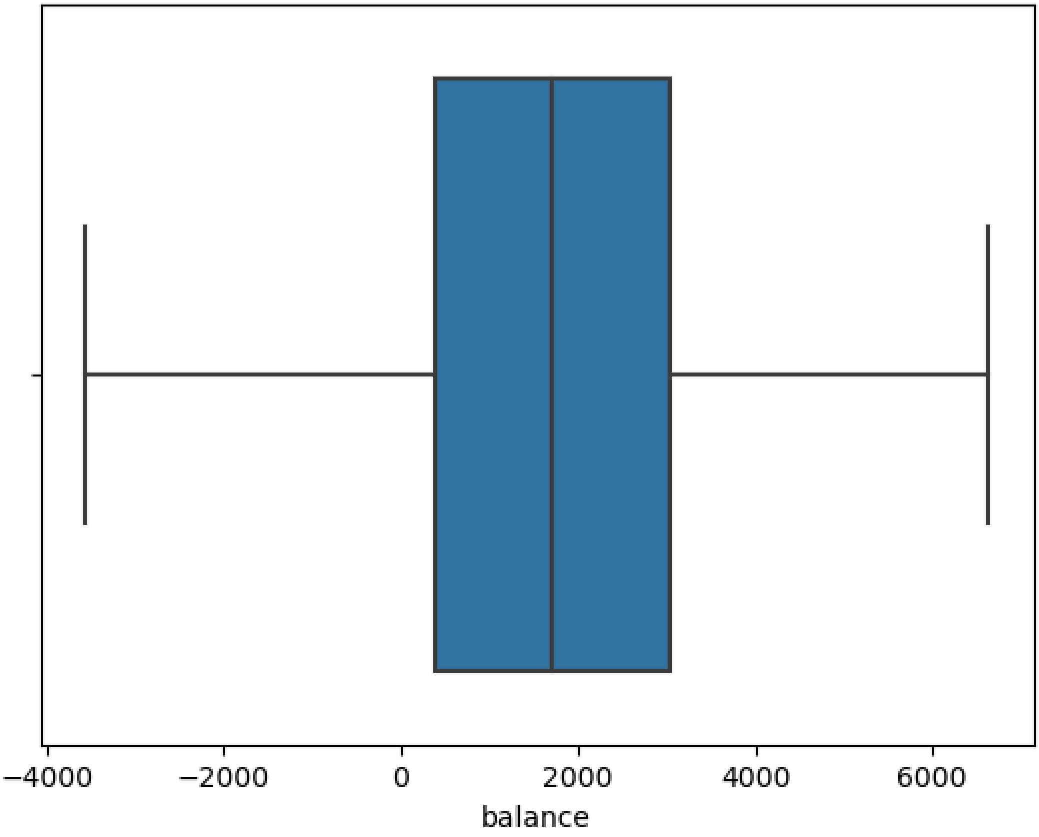


In [23]:

```
# Remove outlier

Q1 = df['balance'].quantile(0.25)
Q3 = df['balance'].quantile(0.75)
IQR = Q3 - Q1

lower = Q1 - 1.5 * IQR
upper = Q3 + 1.5 * IQR

df = df[(df['balance'] >= lower) & (df['balance'] <= upper)]
```

In [24]:

```
# Re-checking outlier

import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(x=df['balance'])
plt.show()
```



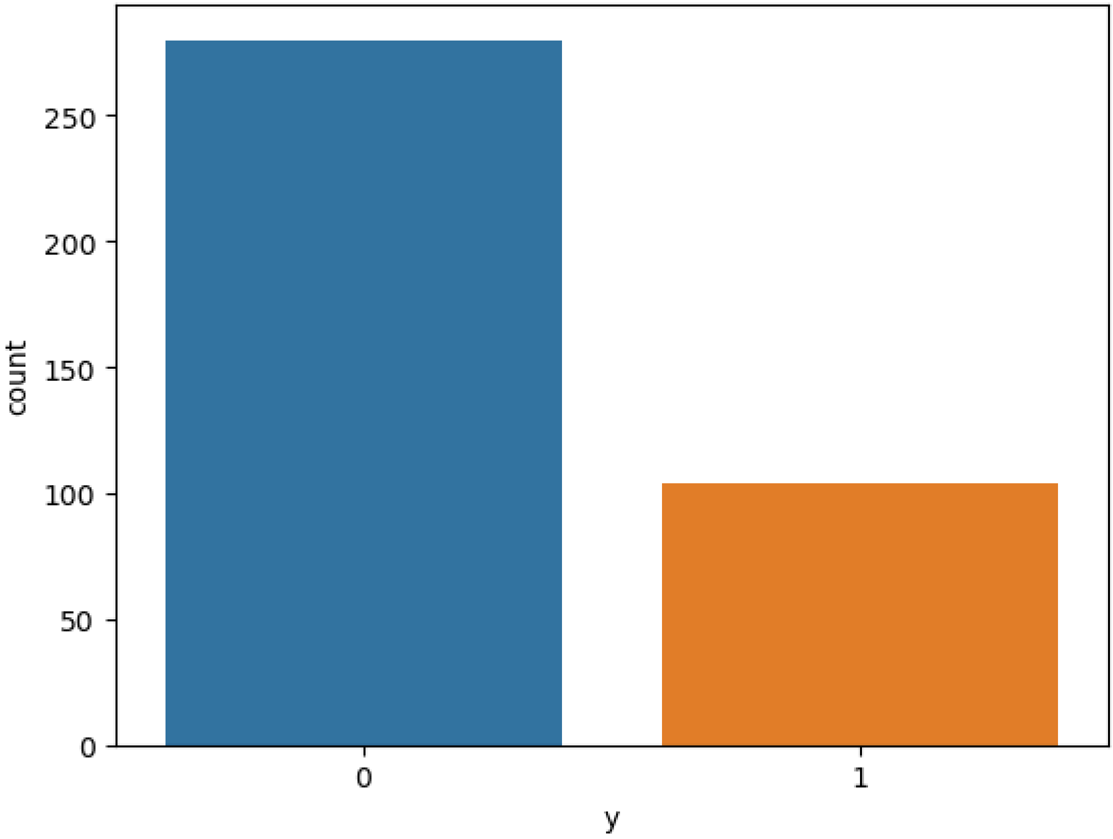
In [25]:

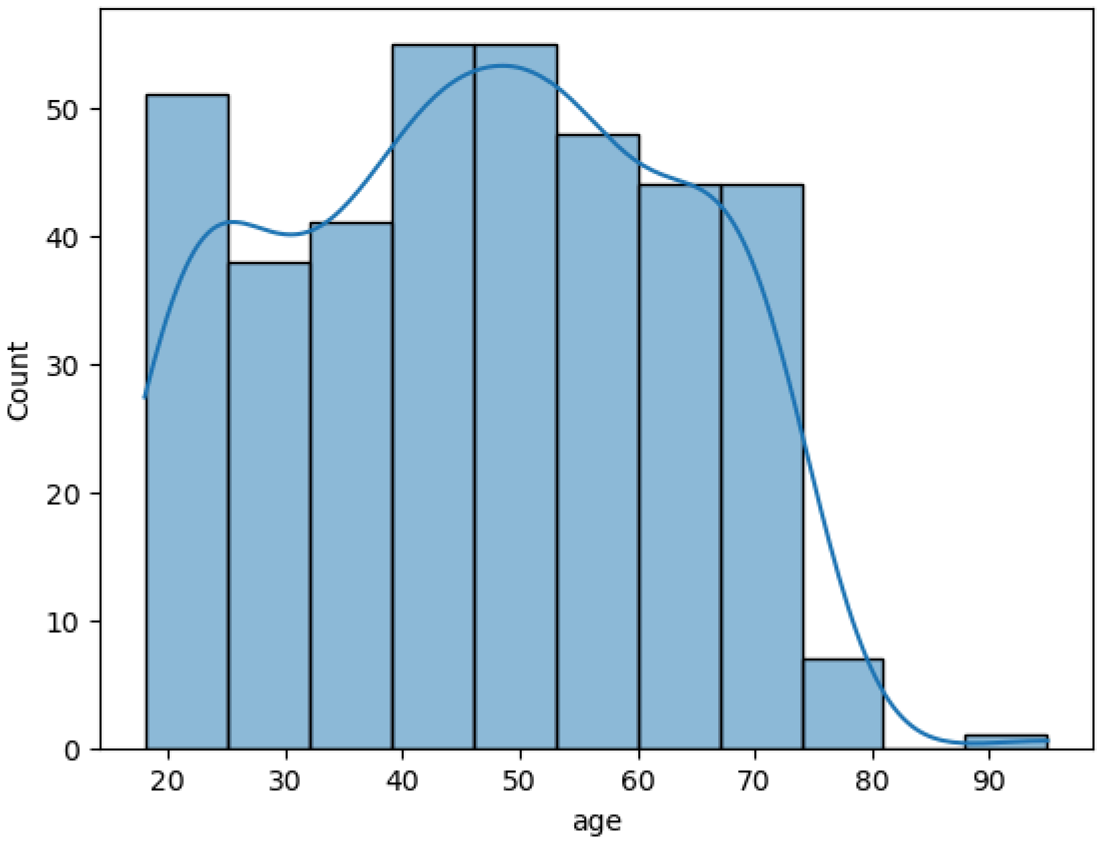
```
# EDA

df['y'].value_counts(normalize=True)
```

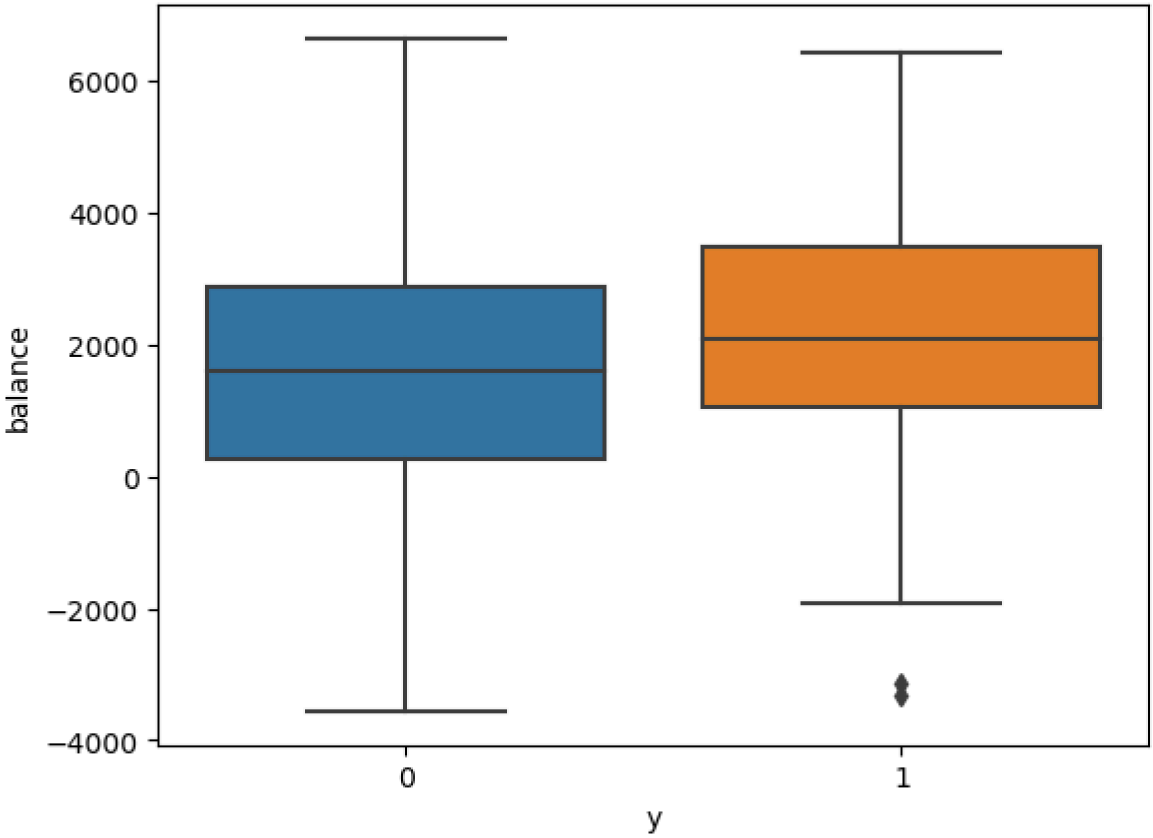
Out[25]:

```
y
0    0.729167
1    0.270833
Name: proportion, dtype: float64
```

In [26]:
```
#plots

import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(x='y', data=df)
plt.show()

sns.histplot(df['age'], kde=True)
plt.show()

sns.boxplot(x='y', y='balance', data=df)
plt.show()
```







In [28]:
```
df = pd.get_dummies(df, drop_first=True)
```

In [29]:

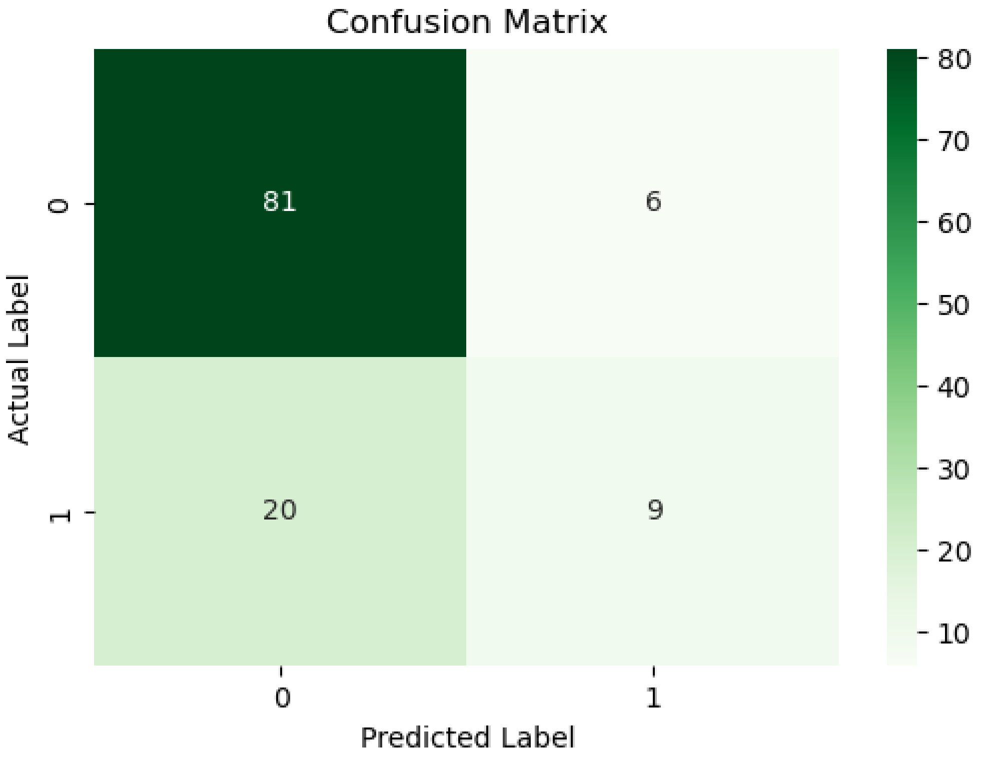
```
# Classification Model

from sklearn.model_selection import train_test_split

X = df.drop('y', axis=1)
y = df['y']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rando
```

In [31]:

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:,1]

print(y_pred[:10])
print(y_prob[:10])
```

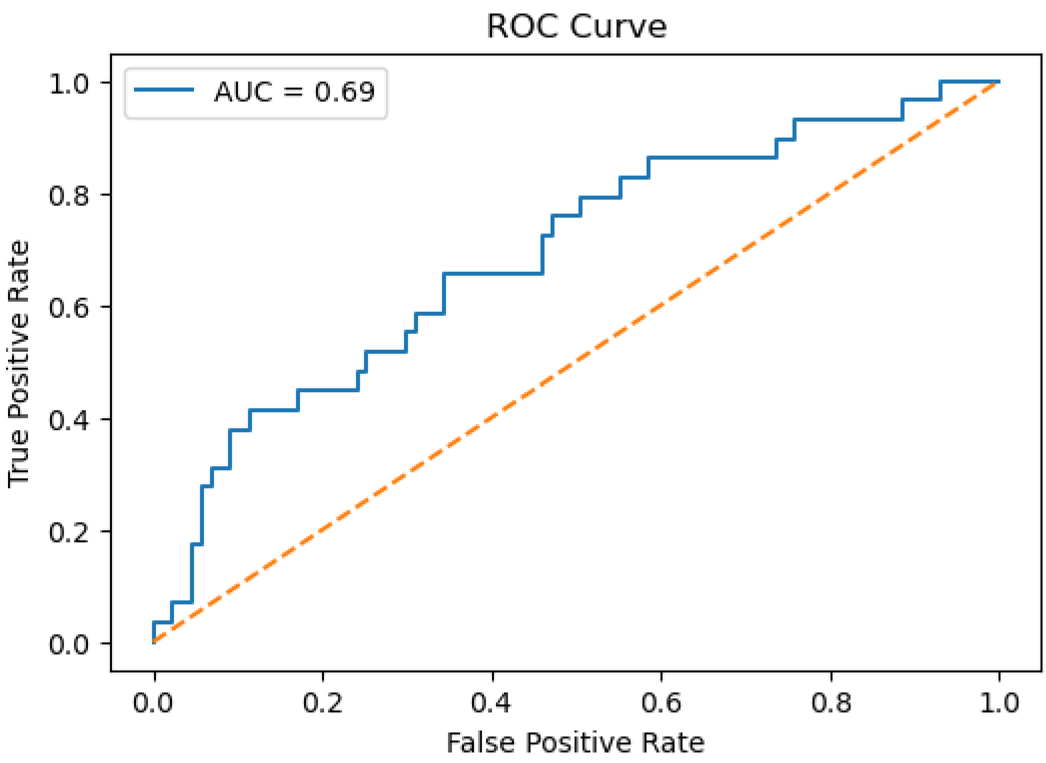
```
[0 1 1 0 0 0 0 1 1 0]
[0.36492744 0.50753374 0.63462157 0.09332663 0.38715414 0.15368983
 0.34137651 0.76177991 0.55286287 0.28726887]
```

```
In [35]: from sklearn.metrics import confusion_matrix
         import seaborn as sns
         import matplotlib.pyplot as plt

         cm = confusion_matrix(y_test, y_pred)

         plt.figure(figsize=(6,4))
         sns.heatmap(cm, annot=True, fmt='d', cmap='Greens')
         plt.xlabel("Predicted Label")
         plt.ylabel("Actual Label")
         plt.title("Confusion Matrix")
         plt.show()
```



In [36]:

```
from sklearn.metrics import roc_curve, roc_auc_score

fpr, tpr, thresholds = roc_curve(y_test, y_prob)
auc_score = roc_auc_score(y_test, y_prob)

plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, label=f"AUC = {auc_score:.2f}")
plt.plot([0,1], [0,1], linestyle='--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.show()

print("ROC-AUC Score:", auc_score)
```



```
ROC-AUC Score: 0.6892588188664288
```

In [37]:

```
# Regression Model
#Now predict revenue
#Assume

df['expected_revenue'] = df['y'] * 200
```

In [38]:

```
 from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

X_reg = df.drop(['y','expected_revenue'], axis=1)
y_reg = df['expected_revenue']

X_train_r, X_test_r, y_train_r, y_test_r = train_test_split(X_reg, y_reg, test

reg = LinearRegression()
reg.fit(X_train_r, y_train_r)

y_pred_r = reg.predict(X_test_r)

print("R2:", r2_score(y_test_r, y_pred_r))
print("MAE:", mean_absolute_error(y_test_r, y_pred_r))
print("RMSE:", np.sqrt(mean_squared_error(y_test_r, y_pred_r)))
```

```
R2: 0.012589405194049164
MAE: 71.60766324227585
RMSE: 86.0556765184298
```

In [39]:

```
# Profit Simulation
#Assume: Campaign cost per customer = $5 , Revenue per responder = $200

threshold = 0.5
predicted_target = (y_prob >= threshold)

revenue = predicted_target.sum() * 200
cost = len(predicted_target) * 5

profit = revenue - cost
print("Estimated Profit:", profit)
```

```
Estimated Profit: 2420
```