# Problem Statement

Product Ad Campaign prediction

Performed EDA, feature engineering, and applied ML algorithms

(Logistic Regression, Decision Trees, Random Forest) to predict ad performance and improve targeting strategies.

In [1]:

```
## Step 1: Import Necessary Libraries

import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

import pickle
import json
```

In [2]:

```
## Load the data

df = pd.read_csv('data1.csv')
```

In [4]:

```
df
```

Out[4]:

| | limit_infor | campaign_type | campaign_level | product_level | resource_amount | email_rate | price | discount_rate | hour_resouces | campaig |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 6 | 0 | 1 | 1 | 0.08 | 140.0 | 0.83 | 93 | |
| **1** | 0 | 0 | 0 | 1 | 1 | 0.10 | 144.0 | 0.75 | 150 | |
| **2** | 0 | 1 | 1 | 1 | 1 | 0.12 | 149.0 | 0.84 | 86 | |
| **3** | 0 | 3 | 1 | 2 | 1 | 0.12 | 141.0 | 0.82 | 95 | |
| **4** | 0 | 0 | 0 | 1 | 1 | 0.10 | 146.0 | 0.59 | 73 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **726** | 0 | 5 | 1 | 1 | 8 | 0.79 | 149.0 | 0.83 | 829 | |
| **727** | 0 | 5 | 1 | 1 | 8 | 0.79 | 154.0 | 0.83 | 670 | |
| **728** | 0 | 5 | 1 | 1 | 8 | 0.84 | 158.0 | 0.87 | 562 | |
| **729** | 0 | 6 | 0 | 1 | 8 | 0.80 | 150.0 | 0.87 | 987 | |
| **730** | 0 | 6 | 0 | 1 | 9 | 0.80 | 149.0 | 0.84 | 1448 | |

731 rows × 11 columns

In [5]:

```
## EDA (Exploratary data analysis)
```

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   limit_infor      731 non-null    int64  
 1   campaign_type    731 non-null    int64  
 2   campaign_level   731 non-null    int64  
 3   product_level    731 non-null    int64  
 4   resource_amount  731 non-null    int64  
 5   email_rate       731 non-null    float64
 6   price            729 non-null    float64
 7   discount_rate    731 non-null    float64
 8   hour_resouces    731 non-null    int64  
 9   campaign_fee     731 non-null    int64  
 10  orders           731 non-null    int64  
dtypes: float64(3), int64(8)
memory usage: 62.9 KB
```

In [7]:

```
df

# limit information - Information has some restrcation and limits
# Campaign type - 1) Social media 2) Television 3) Print advertize 4) Direct MAil 5) Internet
# campaign_level - product has to compaign level 1) National 2) Regional 3) local
# product_level -
# resource_amount -
# email_rate - Email delivery rate
# price - Selling price of the product
# discount_rate - Discount and offers with products
# hour_resouces - The number of labour hours and human resources deticated to marketing compaign
# campaign_fee - Fees or costs for add marketing compaign
```

Out[7]:

| | limit_infor | campaign_type | campaign_level | product_level | resource_amount | email_rate | price | discount_rate | hour_resouces | campaig |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 6 | 0 | 1 | 1 | 0.08 | 140.0 | 0.83 | 93 | |
| **1** | 0 | 0 | 0 | 1 | 1 | 0.10 | 144.0 | 0.75 | 150 | |
| **2** | 0 | 1 | 1 | 1 | 1 | 0.12 | 149.0 | 0.84 | 86 | |
| **3** | 0 | 3 | 1 | 2 | 1 | 0.12 | 141.0 | 0.82 | 95 | |
| **4** | 0 | 0 | 0 | 1 | 1 | 0.10 | 146.0 | 0.59 | 73 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **726** | 0 | 5 | 1 | 1 | 8 | 0.79 | 149.0 | 0.83 | 829 | |
| **727** | 0 | 5 | 1 | 1 | 8 | 0.79 | 154.0 | 0.83 | 670 | |
| **728** | 0 | 5 | 1 | 1 | 8 | 0.84 | 158.0 | 0.87 | 562 | |
| **729** | 0 | 6 | 0 | 1 | 8 | 0.80 | 150.0 | 0.87 | 987 | |
| **730** | 0 | 6 | 0 | 1 | 9 | 0.80 | 149.0 | 0.84 | 1448 | |

731 rows × 11 columns

In [8]:

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

Out[8]:

```
limit_infor        0
campaign_type      0
campaign_level     0
product_level      0
resource_amount    0
email_rate         0
price              2
discount_rate      0
hour_resouces      0
campaign_fee       0
orders             0
dtype: int64
```

In [9]:
```
df['price'].mode()[0]
```

Out[9]:
```
np.float64(154.0)
```

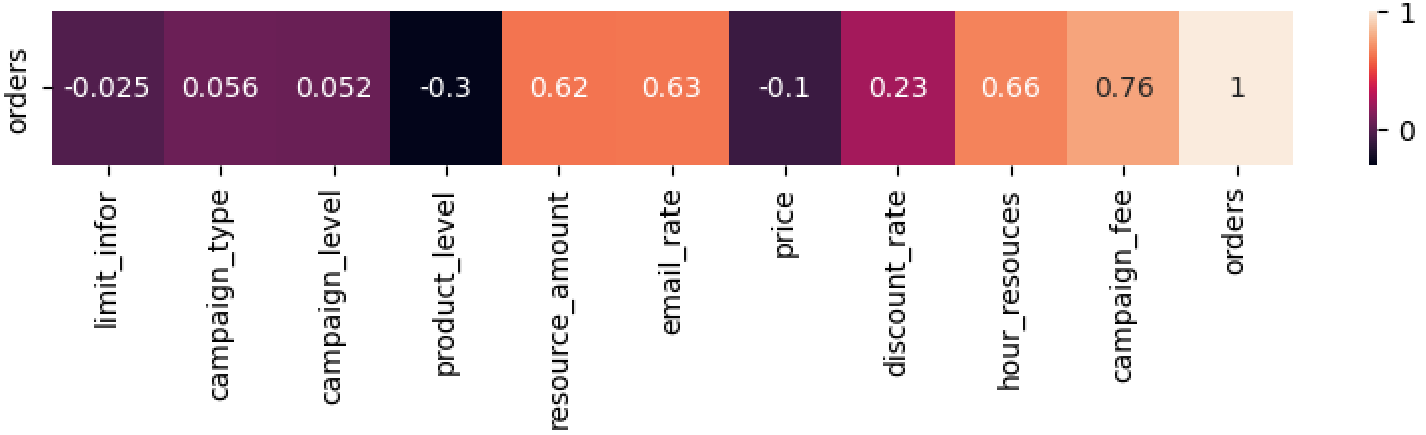
In [10]:
```
df['price'].fillna(df['price'].mode()[0], inplace = True)
```

```
C:\Users\msaad\AppData\Local\Temp\ipykernel_7804\1224124710.py:1: FutureWarning: A value is trying to be set on a copy of a Dat
aFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are set
ting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = d
f[col].method(value) instead, to perform the operation inplace on the original object.


  df['price'].fillna(df['price'].mode()[0], inplace = True)
```

In [11]:
```
df.isna().sum()
```

Out[11]:
```
limit_infor        0
campaign_type      0
campaign_level     0
product_level      0
resource_amount    0
email_rate         0
price              0
discount_rate      0
hour_resouces      0
campaign_fee       0
orders             0
dtype: int64
```

In [12]:
```
plt.figure(figsize=(10,1))
sns.heatmap(df.corr().tail(1), annot = True)
plt.savefig('corr.png')
```

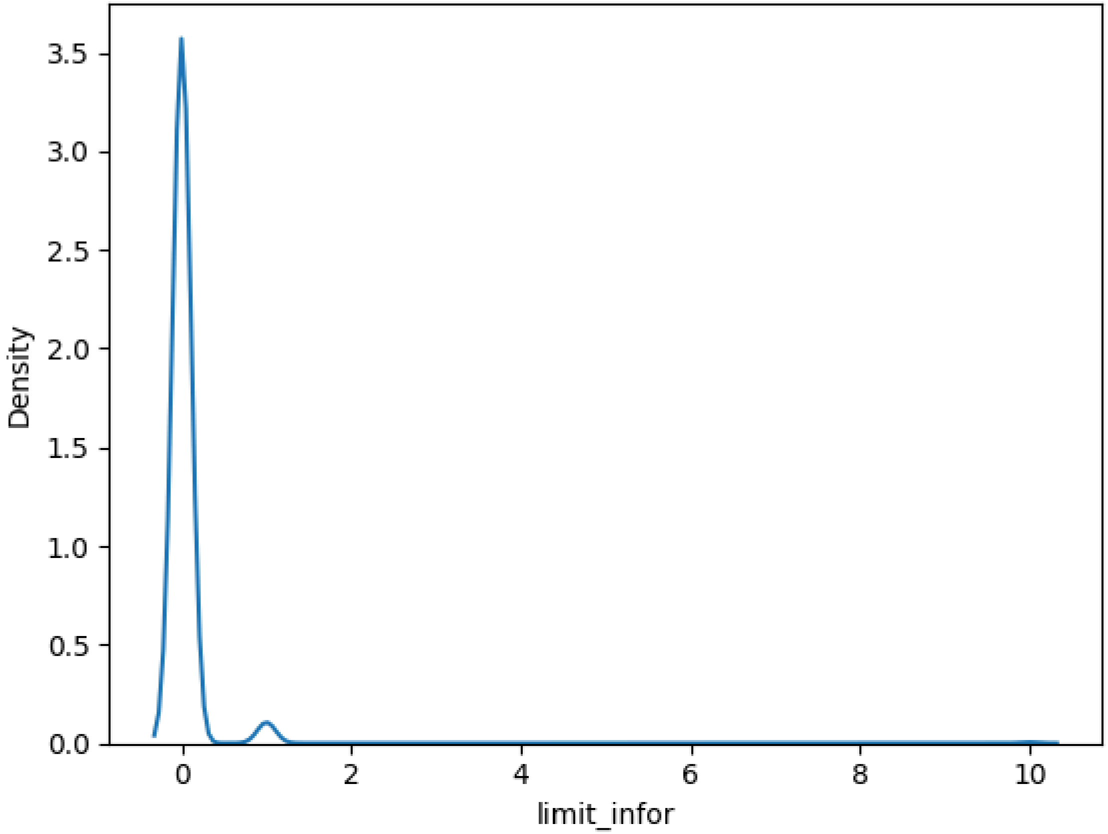


In [13]:
```
df['limit_infor'].value_counts()
```

Out[13]:
```
limit_infor
0     709
1      21
10      1
Name: count, dtype: int64
```

In [14]:
```
sns.kdeplot(df['limit_infor'])
```

Out[14]:
```
<Axes: xlabel='limit_infor', ylabel='Density'>
```



In [15]:
```
df['campaign_type'].value_counts()
```

Out[15]:
```
campaign_type
6    105
0    105
1    105
3    104
4    104
2    104
5    104
Name: count, dtype: int64
```

In [16]:
```
df['campaign_level'].value_counts()
```

Out[16]:

```
campaign_level
1    500
0    231
Name: count, dtype: int64
```

In [17]:

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

Out[17]:

```
product_level
1    463
2    247
3     21
Name: count, dtype: int64
```

In [18]:

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

Out[18]:

```
resource_amount
3    141
7    138
6    115
4    113
5    106
2     55
8     54
1      8
9      1
Name: count, dtype: int64
```

In [19]:

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

Out[19]:

```
email_rate
0.65    25
0.32    23
0.54    20
0.64    20
0.53    20
        ..
0.14     1
0.13     1
0.17     1
0.83     1
0.84     1
Name: count, Length: 71, dtype: int64
```

In [20]:

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

Out[20]:

```
price
154.0    27
149.0    24
169.0    24
165.0    23
159.0    23
         ..
133.0     1
196.0     1
125.0     1
128.0     1
194.0     1
Name: count, Length: 72, dtype: int64
```

In [21]:

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

```
Out[21]: discount_rate
         0.83    46
         0.87    44
         0.81    41
         0.85    39
         0.88    38
         0.86    38
         0.77    37
         0.79    37
         0.84    36
         0.82    35
         0.78    32
         0.76    28
         0.80    23
         0.89    21
         0.75    20
         0.92    19
         0.90    19
         0.73    17
         0.91    15
         0.70    14
         0.72    13
         0.74    13
         0.93    12
         0.71    11
         0.94    11
         0.66    10
         0.65    10
         0.69     9
         0.95     7
         0.64     5
         0.67     5
         0.68     5
         0.58     4
         0.59     3
         0.63     3
         0.62     3
         0.61     3
         0.49     1
         0.60     1
```

```
0.98     1
0.56     1
0.96     1
Name: count, dtype: int64
```

In [22]:

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

Out[22]:

```
hour_resouces
968      4
120      4
163      3
244      3
123      3
        ..
632      1
1421     1
1203     1
1405     1
1052     1
Name: count, Length: 606, dtype: int64
```

In [23]:

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

Out[23]:

```
campaign_fee
4841     3
1707     3
6248     3
4446     2
5265     2
        ..
4634     1
3176     1
2825     1
2298     1
5703     1
Name: count, Length: 679, dtype: int64
```

In [24]:

```
x = df.drop('orders', axis = 1)
x
```

```
y = df['orders']
y
```

Out[24]:

```
0      1981
1       986
2      1416
3      2368
4      1529
       ... 
726    5463
727    3846
728    3387
729    3285
730    4840
Name: orders, Length: 731, dtype: int64
```

## Train test split

In [25]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

In [26]:

```
x_test
```

Out[26]:

| | limit_infor | campaign_type | campaign_level | product_level | resource_amount | email_rate | price | discount_rate | hour_resouces | campaig |
|---|---|---|---|---|---|---|---|---|---|---|
| **703** | 0 | 4 | 1 | 1 | 8 | 0.74 | 160.0 | 0.72 | 1036 | |
| **33** | 0 | 2 | 1 | 1 | 2 | 0.21 | 159.0 | 0.84 | 108 | |
| **300** | 0 | 5 | 1 | 2 | 4 | 0.44 | 184.0 | 0.89 | 548 | |
| **456** | 0 | 0 | 0 | 1 | 6 | 0.56 | 157.0 | 0.91 | 2166 | |
| **633** | 0 | 4 | 1 | 1 | 7 | 0.65 | 142.0 | 0.84 | 812 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **70** | 0 | 3 | 1 | 2 | 3 | 0.27 | 172.0 | 0.87 | 109 | |
| **192** | 0 | 5 | 1 | 1 | 3 | 0.35 | 163.0 | 0.79 | 349 | |
| **328** | 0 | 4 | 1 | 1 | 5 | 0.45 | 141.0 | 0.67 | 745 | |
| **165** | 0 | 3 | 1 | 1 | 3 | 0.31 | 161.0 | 0.73 | 188 | |
| **135** | 0 | 6 | 0 | 1 | 3 | 0.30 | 154.0 | 0.81 | 155 | |

147 rows × 10 columns

In [27]:

```
y_test
```

Out[27]:

```
703    6861
33     1562
300    4378
456    7333
633    4792
       ... 
70     2802
192    4154
328    4189
165    3613
135    1011
Name: orders, Length: 147, dtype: int64
```

In [28]:

```
linear_reg = LinearRegression()

linear_reg.fit(x_train,y_train)
```

Out[28]:

```
▾ LinearRegression

LinearRegression()
```

In [29]:

```
linear_reg.score(x_train,y_train)
```

Out[29]:

```
0.9699087360481338
```

In [30]:

```
linear_reg.predict(x_test)
```

```
Out[30]: array([ 6856.50224661,  1599.40840598,  4333.76438935,  7371.26930154,
                33886.14911538,  5009.16223713,   699.64785951,  6297.1187155 ,
                 5520.98625717,  7934.15253337,  3426.43491171,  5698.33024511,
                 1196.98777201,  4674.60278508,   678.57241716,  2223.07185594,
                 3993.40107979,  4752.36866466,  1448.81348203,  7649.30139101,
                 6926.28369471,  4143.80212804,  4362.91646796,  3078.11055238,
                 1040.06467098,  3330.62783557,  3064.48611817,  7561.60186469,
                 4832.19236623,  4460.92062519,  5886.8571503 ,  4311.58194813,
                 7654.76986909,  4660.66907791,  6980.03568391,  5176.83012683,
                 4673.34561102,  4775.70907393,  3184.60231385,  1446.41394844,
                 3814.33628936,  7048.06136692,  5116.76748324,  3969.0585759 ,
                 5146.22023681,  4755.44881878,  4306.07248659,  5875.53612872,
                 7625.69356651,  6326.21106217,   789.0633425 ,  4579.52296822,
                 4995.49018804,  3015.55939898,  1596.54714004,  2527.20969969,
                 1474.80891848,  1704.45951135,  1736.44093876,  4773.77017691,
                 7389.06153438,  5320.66776111,  3429.34679995,  6204.22707845,
                 5549.79344058,  6996.02946888,  6876.55364768,  4084.67922405,
                 1850.23062294,  7211.20700201,  2879.98853001,  2204.36808247,
                 3313.78635213,  1104.11181526,  3862.9923302 ,  3222.01597432,
                 6906.18689208,  6019.26500018,  1577.82316393,  3210.91566591,
                 5510.91756631,  1895.55475108,  4678.92865201,  3911.40050877,
                 6849.01180995,  3676.5084878 ,  3469.95929112,  4448.7871697 ,
                 2678.69784499,  3920.50404464,  1173.21030067,  7454.01592433,
                 7594.18318408,  7528.57666948,  3915.43698574,  4767.49907455,
                 4062.26234663,  2080.68945389,  2208.91075693,  4116.38267637,
                 6935.92398866,  1467.49606706,  7584.27010412,  1828.77364308,
                 2124.41032794,  4005.17846743,  4513.41046144,  4522.70400402,
                 4629.74529624,  1534.43719133,  5974.43517708,  6031.90614967,
                 5643.51155551,  4320.41311237,  4582.57000576,  1740.44678627,
                 1336.42605254,  3914.69352591,  1846.88404306,  4964.12261045,
                 3975.9725455 ,  5074.17712015,  1909.24592776,  1463.22523852,
                 4203.28874051,  4411.61993704,  2108.36933735,  3160.79953341,
                 1919.94927539,  4820.75347656,  4994.84128753,  5721.50678685,
                 7282.83016943,  7408.5715285 ,  4195.20463393,  1768.704105  ,
                 2064.50852645,  3368.78087014,  1929.36004263,  5556.72965374,
                 2327.14998039,  2880.11333728,  2823.725846  ,  4118.12616726,
                 4200.09642687,  3609.5467299 ,  1117.34639266])
```

```
In [31]: y_test
```

```
Out[31]: 703    6861
         33     1562
         300    4378
         456    7333
         633    4792
                ... 
         70     2802
         192    4154
         328    4189
         165    3613
         135    1011
         Name: orders, Length: 147, dtype: int64
```

```
In [32]: pred_y = linear_reg.predict(x_test)
```

```
In [33]: err = y_test - pred_y
         err
```

```
Out[33]: 703        4.497753
         33       -37.408406
         300       44.235611
         456      -38.269302
         633   -29094.149115
                    ...     
         70       -21.725846
         192       35.873833
         328      -11.096427
         165        3.453270
         135     -106.346393
         Name: orders, Length: 147, dtype: float64
```

```
In [34]: linear_reg.predict(x_train)
```

```
Out[34]:  array([4633.85442654, 2845.96915586, 5324.57493493, 4118.73379271,
                 5005.35285019, 2494.88500434, 4966.13262112,  950.38561051,
                 6190.96373339, 6175.55389314, 4809.91758609, 2954.8079497 ,
                 2284.13124892, 4925.27408172, 3656.53589427, 6344.0883588 ,
                 4074.77202186, 3172.04033052, 3389.60633799, 5397.46015605,
                 3333.20852245, 4412.97866886, 4428.90911116, 5212.23087166,
                 5829.94904796, 1768.86832658, 4875.18775571, 6067.98301107,
                 4681.0144922 , 6181.84084741, 3571.12807171, 1853.65173996,
                 3325.58594175, 1647.73515257, 6526.46947306, 5902.04186067,
                 7325.56024152, 6775.98763692, 6311.47342593, 7391.59073403,
                 3384.28726068, 5753.60168488, 1129.71845156, 4758.91141461,
                 6971.9686996 , 4275.74079627, 4727.9857839 , 3302.41574405,
                 2560.08117209, 5327.11551682, 7316.24646108, 1410.38041662,
                 3098.27781052, 4426.06527187, 6706.45039903, 6804.47206825,
                 1677.42315027, 5754.67129918, 6479.14684819, 4049.73975193,
                 6109.90586825, 5157.88427466, 7387.34445304, 5261.87437828,
                 5805.79859183, 2906.41299698, 2721.05033042, 3720.43375792,
                 5142.68707165, 5079.98775653, 7311.06766614, 4906.36752696,
                 1220.75964945, 2409.10929778, 6690.66861431, 4159.14218358,
                 3417.52840283, 7731.54112114, 5745.32540056, 5130.83478137,
                 6912.45550459, 6515.03168986, 5604.21895853, 6860.30949284,
                 5315.77781446, 3795.77431999, 3650.77635992, 4081.3411537 ,
                 7114.63567194, 2101.45707041, 2185.85041385, 1553.70087781,
                 3688.21724093, 5967.71819775, 7896.11306368, 6081.28065577,
                 4644.35824811, 6266.43838368, 5143.46344784, 7669.35107939,
                 3235.9044812 , 2073.3267427 , 2980.59399895,  458.53948171,
                 1549.02350673, 2426.19407734, 1537.30828576, 2109.08772822,
                 6613.52613713, 6907.40944391, 6420.12249837, 2574.14493261,
                 3987.85085539, 4471.09347814, 5243.28925082, 6969.56157176,
                 6240.59174817, 5384.84974127, 5893.79560656, 1304.24903961,
                 6796.17437825, 1156.02948974, 4174.5687588 , 2947.45729943,
                 3926.61226063, 5909.14976878, 3160.77536828, 3612.85815708,
                 7435.09809975, 1386.84580295, 4549.68710964, 1487.22080324,
                 4367.14675157, 5407.36589718, 4568.95794699, 2430.02661321,
                 4444.62984228, 4842.76573931, 6542.13847511, 3198.23960592,
                 3641.53871649, 4730.30255286, 4563.31664302, 4418.17566766,
                 3682.12479159, 4556.11256529, 4009.88585193, 2162.35461098,
                 1970.52655125, 4063.18765872, 1828.58845101, 5363.53775006,
                 3656.40209414, 2457.70759764, 2544.00416078, 2933.85755266,
                 1666.34339232, 6588.72772977, 2797.11644843, 5018.25014572,
                 7353.65761248, 3422.61371005, 5550.14544255, 1079.10980267,
                 4832.91612741, 4769.59167255, 4652.75765199, 6551.4189512 ,
                 4102.76265878, 3605.16439563, 5636.63864385, 6224.92474419,
                 7374.21975178, 2169.03999482, 6239.22508647, 6073.05648906,
                 3928.86574073, 7841.92531149, 8601.28237164, 4179.99253273,
                 4986.39030952, 3972.36596425, 1882.6921925 ,   67.28824076,
                 2727.64453551, 1542.52122375, 4083.64424154, 1045.03480731,
                 4954.84079763, 6047.90077231, 2048.05518786, 3354.23824581,
                 6258.02654658, 6399.66856913, 2808.89298747, 4503.59467885,
                 7049.14730228, 4712.98510986, 7445.51300733, 4086.47691325,
                 7195.11252209, 7798.93797146, 4221.14606844, 4616.68232407,
                 3014.98111607, 1895.16494499, 4824.19952026, 5365.47639164,
                 2481.9532745 , 5781.35450894, 3178.23802038, 7427.47192364,
                 5671.85616742, 7260.94694271, 1858.61530007, 4467.45848191,
                 4710.9922504 , 5119.86161065, 3734.61061426, 6154.41445203,
                 7415.81283049, 1546.89316793, 2176.5805871 , 7733.59431753,
                 5536.06074912, 7268.39592134, 1047.05420552, 6759.60515872,
                 7995.41185755, 2396.51010891, 5138.55461482, 3528.91171015,
                 1230.09744694, 6047.95312824, 3844.19218979, 6413.63905526,
                 7120.75865743, 5541.89910397, 3413.26047769, 5515.68213709,
                 7702.37860809, 4625.46298659, 3091.94766078, 4162.92258316,
                 6306.84157796, 3690.50119298, 5608.67446707, 4541.54307952,
                 2549.96148182, 8076.58609193, 3843.41058118, 1985.10031205,
                 1344.33433238, 4504.70173186, 3827.60180904, 4775.56295679,
                 6849.10919598, 3786.67342971, 4712.56404845, 6262.43000298,
                 4997.44409192, 7445.87066976, 3416.94925404, 2166.41906065,
                 2512.62704629, 8218.14137454, 7678.55593183, 4704.32366337,
                 5266.25555428, 7110.81064008, 7464.22988055, 1632.21010535,
                 5956.39916229, 1974.90712738, 5123.62980543, 5118.54560656,
                 3938.72805491, 1178.43612071, 2017.20175357, 6719.38332268,
                 6936.79548038, 2398.52507659, 5329.65777455, 4695.39720141,
                 2790.62800399, 4389.56187857, 6826.87719249, 3248.4105152 ,
                 3716.75946528, 7687.72811748, 2951.02944415, 3292.66403888,
                 6830.11528516, 6357.6523222 , 4609.27794885, 2408.83511004,
                 5012.51465748, 2765.68843038, 2750.88728832, 4959.09074761,
                 5506.04178861, 4476.98861678, 4586.37985762, 3254.96426951,
                 7580.43830427, 3567.89927967, 4866.93667792, 5261.47630708,
                 7394.03031157, 2503.81502283, 3841.47047081, 4699.9783341 ,
                 5550.05081066, 4462.75444653, 3865.71005867, 4291.66453326,
                 7330.22515721, 6793.35418605, 2105.96484248, 5766.18593914,
                 5162.59311437, 5855.49429091, 7361.12259751, 5058.77398984,
                 3817.76288709, 5141.95209331, 5374.51259738, 6544.97789672,
                 3432.33041265, 7481.39664701, 5245.91582636, 1119.01277227,
                 3583.49049764, 6691.31744741, 1957.91296245, 2211.8279188 ,
                 4282.28961661, 7697.95316189, 3842.83982044, 4406.68255649,
                 7345.76213318, 3917.97104015, 7555.33048877, 1957.87011797,
                 5478.40456732, 4830.46631202, 5846.31247683, 3947.18243346,
                 1352.28830184, 3763.62719736, 4601.01464376, 4050.97831817,
                 5503.59154989, 4398.22042162, 7262.90985821, 4047.70028785,
                 7046.58098238, 2262.19805179, 7701.46867038, 5219.57165546,
                 6459.8316064 , 1724.01151653, 4726.79948859, 3610.82767644,
                 7048.09579616, 3147.40874059, 7575.2447848 , 4690.70225432,
                 4654.31174039, 4024.18618372, 7525.2478342 , 4207.35049663,
                 2086.99586407, 4679.79478499, 6848.17789631, 2419.50322597,
                 1975.57643871, 4050.56388109, 4821.68519939, 7538.69713295,
                 4751.60308976, 4170.34540813, 2940.64030539, 6324.47454929,
                 4060.0537054 , 1816.13684188, 7474.79046305, 4139.88520233,
                 4370.41032723, 5658.80193529, 2757.38323597, 2664.43991655,
                 4396.1590869 , 4253.99700425, 3545.5198065 , 4186.53692533,
                 6380.68085183, 3812.22771927, 8453.73455714, 4995.75251507,
                 5177.87400842, 6980.28156305, 5413.44153355, 3439.969247  ,
                 2207.9028691 , 4347.24354132, 4188.2634624 , 3890.68584298,
                 7148.10042657, 6629.73931714, 6897.03682992, 3790.89012712,
                 4224.51181249, 5183.66724375, 4842.5823482 , 1773.61444152,
                  716.44101061, 7587.4285742 , 4343.56586421, 4272.01553536,
                 7334.62432481, 3452.9671592 , 5514.81742999, 4802.84443149,
                 7068.62072353, 3773.36766416, 1641.88584209, 5096.54243859,
                 6786.28289655, 5426.85959973, 4399.73966892, 5144.87942967,
                 4379.23899952, 6589.36612324, 4072.76790905, 3838.9202127 ,
                 4978.463359  , 7417.55345963, 6031.33938502, 4960.32986702,
                 4535.29996078, 4866.7440531 , 2557.63507166, 2117.20507362,
                 5422.47838913, 4145.5638772 , 1645.0169385 , 8318.23399461,
                 3596.66426133, 5031.69802498, 4585.6196743 , 4864.5404979 ,
                  897.84588724, 4808.83534017, 5517.26373721, 6675.45852312,
                 8037.99754588, 7318.07495839, 6644.20560345, 5144.75298925,
                 4425.84686536, 4549.56340455, 3971.19963631,  964.2603798 ,
                 1635.71237859, 6660.46659327, 4883.69669512, 6743.64969554,
                 2341.37140604, 1222.61034659, 6249.63702119, 7119.60535752,
                 3474.40605727, 1733.90176386, 4694.91579539, 1574.45824749,
                 1385.82339724, 5393.56956393, 3601.14422768, 8213.04987415,
                 3486.86634907, 6130.8750169 , 6125.78292821, 5348.23183536,
                 5246.03610846, 4059.41800282, 3386.75722902, 2428.07363865,
                 6997.67823247, 5321.17151922, 4929.71149924, 5051.61350494,
                 1427.10802489, 2446.89728519, 5268.09439386,  499.25707219,
                 6147.3897459 , 7698.52480198, 3053.80612062, 3886.37731528,
                  954.87172896, 6770.85097314,  856.33491942, 4590.24358657,
                 7517.00201775, 7771.63291162, 3466.20425491,  718.78064573,
                 3717.23763339, 5133.77280792, 5354.91763122, 4845.49251167,
                 8204.44350426, 5564.67708992, 7370.32081172, 7195.07621867,
                 1122.20066939, 6616.3051801 , 5843.57692934, 3837.22906176,
                 3210.60791291, 1161.7932886 , 4784.77342518, 7582.09251593,
                 3680.78120552, 5236.569033  , 2081.23940607, 7510.4207252 ,
                 7420.48107068, 5296.19716809, 3319.91696907, 1430.18044506,
                 5907.53911779, 4853.97719936, 2943.87024839, 4555.49931512,
                 5607.24315953, 6224.64345923, 8164.27044192, 4093.31728691,
                 5771.05406808, 6811.35996577, 3608.92736998, 4491.86923054,
                 4384.73937437, 6863.73900101, 4304.88107519, 7847.80292704,
                 6153.94179109, 7434.4576103 ,  437.89272682, 8091.10213063,
                 6445.52139712, 4553.38161321, 1553.75316188, 6350.94603328,
                 5724.09192975, 1433.65830268, 2633.2798295 , 4073.21181609,
                 5369.69116278, 2439.23256806, 7769.52377181, 4704.25721656,
                 2569.74842387, 5349.99276297, 5632.70094615, 7095.05730603,
                 3524.41626581, 5205.12803435, 1903.63902452, 4247.67123249,
                 2620.00809852, 5949.27165588, 7341.16671657, 2204.6624659 ,
                 4946.13159025, 2564.53374552, 4755.69452895, 2813.52250225,
                 3555.92857553, 2473.79284466, 4121.87759951, 5672.82615938])
```

In [35]:

```
y_train
```

Out[35]:

```
682    4586
250    2808
336    8298
260    4109
543    4972
       ... 
71     2739
106    3523
270    2429
435    4123
102    5686
Name: orders, Length: 584, dtype: int64
```

In [36]:

```
y_pred_train = linear_reg.predict(x_train)
```

In [37]:

```
err2 = y_train - y_pred_train
err2
```

Out[37]:

```
682     -47.854427
250     -37.969156
336    2973.425065
260      -9.733793
543     -33.352850
           ...
71      -74.522502
106     -32.928576
270     -44.792845
435       1.122400
102      13.173841
Name: orders, Length: 584, dtype: float64
```

## Model Evaulation

TRAINING

In [38]:

```
y_pred_train = linear_reg.predict(x_train)

mse = mean_squared_error(y_train,y_pred_train)
print('MSE :',mse)

rmse = np.sqrt(mse)
print('RMSE :', rmse)

mae = mean_absolute_error(y_train,y_pred_train)
print('MAE :', mae)

r_squared = r2_score(y_train,y_pred_train)
print('r_sqaured :', r_squared)
```

```
MSE : 110805.70593995298
RMSE : 332.8749103491475
MAE : 81.16472117502384
r_sqaured : 0.9699087360481338
```

TESTING

In [39]:

```
pred_y = linear_reg.predict(x_test)

mse = mean_squared_error(y_test,pred_y)
print('MSE :',mse)

rmse = np.sqrt(mse)
print('RMSE :', rmse)

mae = mean_absolute_error(y_test,pred_y)
print('MAE :', mae)

r_squared = r2_score(y_test,pred_y)
print('r_sqaured :', r_squared)
```

```
MSE : 5927458.749918785
RMSE : 2434.637293298282
MAE : 303.6534946947069
r_sqaured : -0.5722921789958788
```

## Save Model

In [40]:

```
with open('linear_reg.pkl', 'wb') as f:
    pickle.dump(linear_reg, f)
```

In [41]:

```
with open('linear_reg.pkl', 'rb') as f:
    linear_model = pickle.load(f)
```

In [42]:

```
linear_model
```

Out[42]:

```
▾ LinearRegression
LinearRegression()
```

In [43]:

```
linear_reg.predict(x_test[4:5])[0]
```

Out[43]:

```
np.float64(33886.149115384316)
```

In [44]:

```
df[34:35]
```

Out[44]:

| | limit_infor | campaign_type | campaign_level | product_level | resource_amount | email_rate | price | discount_rate | hour_resouces | campaigı |
|---|---|---|---|---|---|---|---|---|---|---|
| **34** | 0 | 5 | 1 | 2 | 2 | 0.22 | 179.0 | 0.88 | 38 | |

In [45]:

```
linear_reg.predict(x_test)[55:60]
```

Out[45]:

```
array([2527.20969969, 1474.80891848, 1704.45951135, 1736.44093876,
       4773.77017691])
```

In [46]:

```
y_test[55:60]
```

Out[46]:

```
244    2417
265    3446
120    1650
148    1589
580    4708
Name: orders, dtype: int64
```

In [47]:

```
linear_reg.predict(x_test)[5:10]
```

Out[47]:

```
array([5009.16223713,  699.64785951, 6297.1187155 , 5520.98625717,
       7934.15253337])
```

In [48]:

```
y_test[5:10]
```

```
Out[48]: 557    4978
         39      683
         356    6269
         559    5538
         514    8009
         Name: orders, dtype: int64
```

```
In [49]: pred_y = linear_reg.predict(x_test)
```

```
In [50]: err = y_test - pred_y
         err[5:10]
```

```
Out[50]: 557   -31.162237
         39    -16.647860
         356   -28.118716
         559    17.013743
         514    74.847467
         Name: orders, dtype: float64
```

training data evaluation

```
In [51]: linear_reg.predict(x_train)[100:105]
```

```
Out[51]: array([3235.9044812 , 2073.3267427 , 2980.59399895,  458.53948171,
                1549.02350673])
```

```
In [52]: y_train[100:105]
```

```
Out[52]: 409    3239
         11     1977
         140    2999
         28      431
         43     1530
         Name: orders, dtype: int64
```

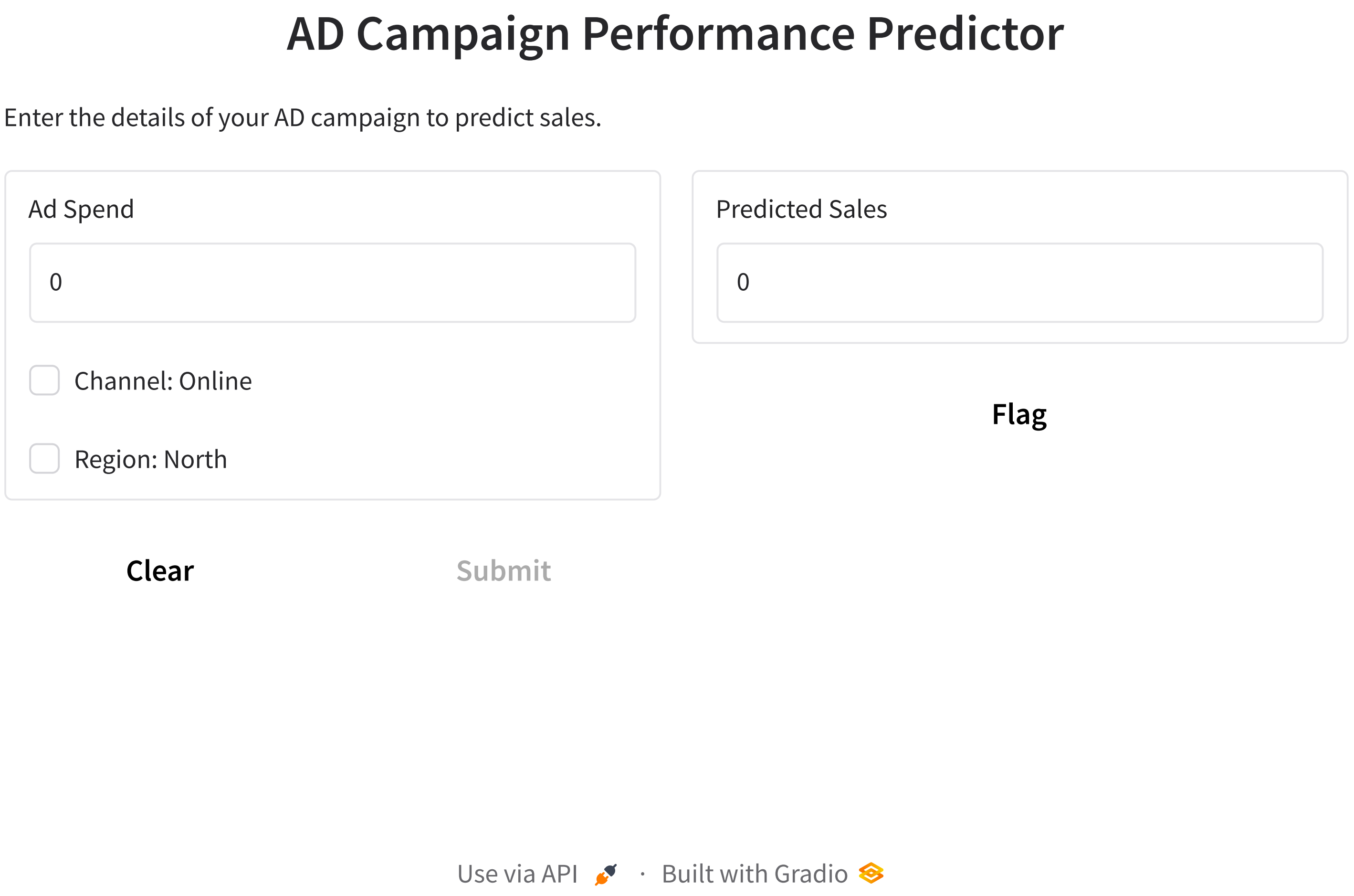
```
In [ ]: 
```

```
In [ ]: 
```

In [53]:

```
import gradio as gr
import joblib

# Load the trained linear regression model
model = joblib.load('linear_reg.pkl')

# Define the prediction function
def predict_sales(discount_rate, resource_amount, hour_resouces):
    # Prepare the input data as a DataFrame
    input_data = pd.DataFrame({
        'discount_rate': [discount_rate],
        'resource_amount': [resource_amount],
        'hour_resouces': [hour_resouces]
    })

    # Ensure the input data matches the model's expected features
    input_data = input_data.reindex(columns=X.columns, fill_value=0)

    # Make a prediction
    prediction = model.predict(input_data)
    return prediction[0]

# Create the Gradio interface
interface = gr.Interface(
    fn=predict_sales,  # Function to call for predictions
    inputs=[
        gr.Number(label="Ad Spend"),  # Input for Ad Spend
        gr.Checkbox(label="Channel: Online"),  # Checkbox for Online channel
        gr.Checkbox(label="Region: North")  # Checkbox for North region
    ],
    outputs=gr.Number(label="Predicted Sales"),  # Output as a number
    title="AD Campaign Performance Predictor",
    description="Enter the details of your AD campaign to predict sales."
)

# Launch the interface
interface.launch(share=True)
```

```
* Running on local URL:  http://127.0.0.1:7860

Could not create share link. Please check your internet connection or our status page: https://status.gradio.app.
```



# AD Campaign Performance Predictor

Enter the details of your AD campaign to predict sales.

Ad Spend

0

□ Channel: Online

□ Region: North

Predicted Sales

0

Out[53]:

In [ ]:

In [ ]:

```
In [ ]:
```