test3

February 3, 2026

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#
plt.rcParams['font.sans-serif'] = ['SimHei']  #
plt.rcParams['axes.unicode_minus'] = False  #

#
try:
    df=pd.read_csv('./data/train.csv',encoding='utf-8')
except:
    df=pd.read_csv('./data/train.csv',encoding='gbk')

#    yes/no  1/0
df['subscribe'] = df['subscribe'].map({'yes': 1, 'no': 0})

#        0
df = df[df['duration'] > 0].reset_index(drop=True)

print("     ",df.shape)
print("   \n", df.isnull().sum())
```

```
      (22498, 22)

 id                 0
age                 0
job                 0
marital             0
education           0
default             0
housing             0
loan                0
contact             0
month               0
day_of_week         0
duration            0
campaign            0
```

```
pdays                0
previous             0
poutcome             0
emp_var_rate         0
cons_price_index     0
cons_conf_index      0
lending_rate3m       0
nr_employed          0
subscribe            0
dtype: int64
```

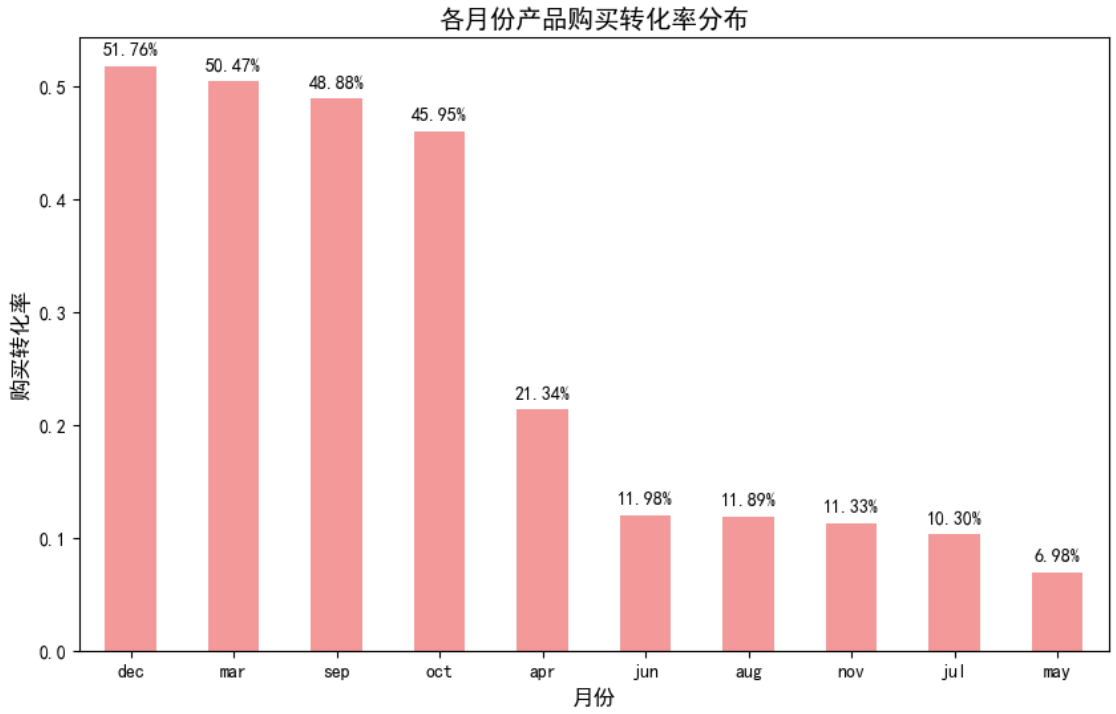
[3]:
```
# 1.      (  /  )
#
df['duration_group'] = pd.cut(df['duration'], bins=[0,␣
 ↪100,300,500,1000],labels=['  ', '   ', '   ', '  '])
print("      \n", df.groupby('duration_group')['subscribe'].mean())

#
month_conversion = df.groupby('month')['subscribe'].mean().
 ↪sort_values(ascending=False)
plt.figure(figsize=(10, 6))
month_conversion.plot(kind='bar', color='lightcoral', alpha=0.8)
plt.title('       ', fontsize=14)
plt.xlabel(' ', fontsize=12)
plt.ylabel('   ', fontsize=12)
plt.xticks(rotation=0)

#
for x, y in enumerate(month_conversion):
    plt.text(x, y+0.01, f'{y:.2%}', ha='center')
plt.show()
```

```
 duration_group
       0.034580
      0.095364
      0.178270
       0.290118
Name: subscribe, dtype: float64
```

```
C:\Users\Origin01\AppData\Local\Temp\ipykernel_36456\1851726172.py:4:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  print("      \n", df.groupby('duration_group')['subscribe'].mean())
```



[4]:
```
 #

# 0.      EDA
base_features = ['age', 'duration','job', 'month','poutcome', 'campaign',␣
 ↪'previous', 'pdays']
df_model = df[base_features + ['subscribe']].copy()

# 1.  1       /
df_model['duration_per_campaign'] = df_model['duration'] /␣
 ↪(df_model['campaign'] + 1)

# 2.  2       *
df_model['previous_x_duration'] = df_model['previous'] * df_model['duration']

# 3.  3
df_model['age_group'] = pd.cut(df_model['age'],␣
 ↪bins=[0,25,35,45,55,100],labels=[' ',' ',' ',' ',' '])

# 4.  4
categorical_cols = ['job', 'month', 'poutcome', 'age_group']
df_model = pd.get_dummies(df_model, columns=categorical_cols, drop_first=True)

print("       ", df_model.shape[1]-1)
```

33

[5]:
```
#
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# 1.   ( )
duration_99 = df_model['duration'].quantile(0.99)
df_model['duration'] = np.where(df_model['duration'] > duration_99,␣
 ↪duration_99, df_model['duration'])

# 2.
numeric_cols = ['age', 'duration', 'campaign', 'previous', 'pdays',␣
 ↪'duration_per_campaign', 'previous_x_duration']
scaler = StandardScaler()
df_model[numeric_cols] = scaler.fit_transform(df_model[numeric_cols])

# 3.
X = df_model.drop('subscribe', axis=1)
y = df_model['subscribe']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

print("    ", X_train.shape[0])
print("    ", X_test.shape[0])
```

```
    17998
    4500
```

[6]:
```
#
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, roc_auc_score

# 1.
model_lr = LogisticRegression(
    max_iter=1000,
    random_state=42,
    class_weight='balanced',    #
    C=0.8,                      #
    solver='liblinear'          #
)

# 2.
model_lr.fit(X_train, y_train)

# 3.
```

```
y_pred = model_lr.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print(f"   : {acc:.4f} --> {acc*100:.2f}%")
```

```
   : 0.7420 --> 74.20%
```