Assignment

You have been provided with a dataset containing information about customers of an e-commerce company. The task is to build a binary classification model using logistic regression to predict whether a customer will make a purchase or not based on their demographic and browsing behavior data. The dataset consists of the following features:

```
email
address
avatar
time on app
 time on website
length of membership
yearly amount spent
```

The target variable is: Purchase (binary: 1 if the customer made a purchase over $450, 0 otherwise)

Instructions: Load the dataset and perform any necessary data preprocessing steps. Split the data into training and testing sets (e.g., 80% training, 20% testing). Train a logistic regression model using the training data. Evaluate the model's performance on the testing data using appropriate evaluation metrics (e.g., accuracy, precision, recall, F1-score).

Provide a brief summary of the model's performance and any insights you gather from the results. Note: You can use any programming language or machine learning libraries of your choice. The aim of this problem is to assess your ability to quickly understand the problem, preprocess the data, build a logistic regression model, evaluate its performance, and derive meaningful insights from the results within a limited timeframe.

In [340]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import RandomOverSampler

%matplotlib inline
```

In [341]:

```
df = pd.read_csv("scpl_folder/data.csv")
```

In [342]:

```
df.head(10)
```

Out[342]:

	\tEmail	Address	Avatar	Time on App	Time on Website	Length of Membership	Yearly Amount Spent	C
0	aaron04@yahoo.com	16338 Scott Corner Suite 727West Alexandra, AR...	SeaGreen	10.16	37.76	4.78	521.24	
1	aaron11@luna.com	672 Jesus Roads Apt. 443Thompsonland, WY 69228	LightSkyBlue	13.46	37.24	2.94	503.98	
2	aaron22@gmail.com	38678 Sean Drive Suite 293Karentown, IA 78306-...	DarkGray	12.01	36.53	4.71	576.48	
3	aaron89@gmail.com	0128 Sampson Loop Suite 943Hoffmanton, MO 02122	SaddleBrown	10.10	38.04	4.24	418.60	
4	acampbell@sanchez-velasquez.info	5791 Jessica CoveMckinneyborough, OK 64460-7536	Wheat	11.45	37.58	2.59	420.74	
5	acontreras@hotmail.com	88995 Edwards Row Suite 456North Jo, DE 02062-...	Sienna	10.74	37.46	3.86	476.19	
6	adam75@gmail.com	9991 Macdonald SquaresVasquezborough, WY 73586...	Purple	10.97	36.61	2.87	404.82	
7	adamperkins@terrell.com	2595 James Creek Apt. 571Millerberg, HI 82236	PaleVioletRed	11.76	37.92	3.53	482.14	
8	afry@ford.biz	399 Jeremy Skyway Suite 377North Keithville, I...	PaleTurquoise	12.19	36.15	3.78	494.55	
9	agolden@yahoo.com	PSC 2490, Box 2120APO AE 15445-2876	Black	12.88	37.44	1.56	419.94	

In [343]:

```
df.columns = ['Email', 'Address', 'Avatar', 'Time on App', 'Time on Website',
       'Length of Membership', 'Yearly Amount Spent', 'Clean_Address_Loc','Clean_Address_Count
```

In [344]:

```
df.loc[df['Yearly Amount Spent']>450,'purchase']=1
df.loc[df['Yearly Amount Spent']<=450,'purchase']=0
```

In [345]:

```
df.describe()
```

Out[345]:

	Time on App	Time on Website	Length of Membership	Yearly Amount Spent	purchase
count	500.000000	500.000000	500.00000	500.000000	500.000000
mean	12.052620	37.060480	3.53336	499.314240	0.730000
std	0.994418	1.010555	0.99926	79.314764	0.444404
min	8.510000	33.910000	0.27000	256.670000	0.000000
25%	11.390000	36.347500	2.93000	445.037500	0.000000
50%	11.980000	37.070000	3.53500	498.890000	1.000000
75%	12.752500	37.720000	4.13000	549.312500	1.000000
max	15.130000	40.010000	6.92000	765.520000	1.000000

In [346]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 10 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Email                 500 non-null    object 
 1   Address               500 non-null    object 
 2   Avatar                500 non-null    object 
 3   Time on App           500 non-null    float64
 4   Time on Website       500 non-null    float64
 5   Length of Membership  500 non-null    float64
 6   Yearly Amount Spent   500 non-null    float64
 7   Clean_Address_Loc     500 non-null    object 
 8   Clean_Address_County  500 non-null    object 
 9   purchase              500 non-null    float64
dtypes: float64(5), object(5)
memory usage: 39.2+ KB
```

In [347]:

```
df.corr()
```

Out[347]:

	Time on App	Time on Website	Length of Membership	Yearly Amount Spent	purchase
Time on App	1.000000	0.082285	0.029240	0.499315	0.353636
Time on Website	0.082285	1.000000	-0.047443	-0.002601	0.003681
Length of Membership	0.029240	-0.047443	1.000000	0.809184	0.601839
Yearly Amount Spent	0.499315	-0.002601	0.809184	1.000000	0.737246
purchase	0.353636	0.003681	0.601839	0.737246	1.000000

Vectorize the words

In [289]:

```
df = df.drop(['Email','Address','Clean_Address_Loc'], axis=1)
df
```

Out[289]:

	Avatar	Time on App	Time on Website	Length of Membership	Yearly Amount Spent	Clean_Address_County	purchase
0	SeaGreen	10.16	37.76	4.78	521.24	AR	1.0
1	LightSkyBlue	13.46	37.24	2.94	503.98	WY	1.0
2	DarkGray	12.01	36.53	4.71	576.48	IA	1.0
3	SaddleBrown	10.10	38.04	4.24	418.60	MO	0.0
4	Wheat	11.45	37.58	2.59	420.74	OK	0.0
...	...	...	...	...	...	...	...
495	DodgerBlue	12.94	36.73	4.56	544.41	UT	1.0
496	OldLace	11.83	36.84	3.61	502.09	MI	1.0
497	Purple	11.68	38.72	3.59	463.59	MT	1.0
498	Moccasin	12.75	36.71	3.28	548.28	Bo	1.0
499	PeachPuff	12.13	38.19	4.02	597.74	SC	1.0

500 rows × 7 columns

In [290]:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(analyzer = "word",
                             tokenizer = None,
                             preprocessor = None,
                             stop_words = None,
                             max_features = 2)
```

In [291]:

```
def vect(df, col):
    dfw = vectorizer.fit_transform(df[col])
    dfw = dfw.toarray()
    df = np.hstack((df.drop(col, axis =1),np.reshape(dfw,(-1,2))))
    df= pd.DataFrame(df)
    print(df)
    return df
```

In [292]:

```
df = vect(df,'Avatar')
df.columns =['Time on App', 'Time on Website','Length of Membership', 'Yearly Amount Spent','C
df = vect(df,'Clean_Address_County')
df.columns =['Time on App', 'Time on Website','Length of Membership', 'Yearly Amount Spent','p
df.head()
```

```
         0      1     2       3   4    5  6  7
0    10.16  37.76  4.78  521.24  AR  1.0  0  0
1    13.46  37.24  2.94  503.98  WY  1.0  0  0
2    12.01  36.53  4.71  576.48  IA  1.0  0  0
3     10.1  38.04  4.24   418.6  MO  0.0  0  0
4    11.45  37.58  2.59  420.74  OK  0.0  0  0
..     ...    ...   ...     ...  ..  ... .. ..
495  12.94  36.73  4.56  544.41  UT  1.0  0  0
496  11.83  36.84  3.61  502.09  MI  1.0  0  0
497  11.68  38.72  3.59  463.59  MT  1.0  0  0
498  12.75  36.71  3.28  548.28  Bo  1.0  0  0
499  12.13  38.19  4.02  597.74  SC  1.0  0  0

[500 rows x 8 columns]
         0      1     2       3    4  5  6  7  8
0    10.16  37.76  4.78  521.24  1.0  0  0  0  0
1    13.46  37.24  2.94  503.98  1.0  0  0  0  0
2    12.01  36.53  4.71  576.48  1.0  0  0  0  0
3     10.1  38.04  4.24   418.6  0.0  0  0  0  0
4    11.45  37.58  2.59  420.74  0.0  0  0  0  0
..     ...    ...   ...     ...  ... .. .. .. ..
495  12.94  36.73  4.56  544.41  1.0  0  0  0  0
496  11.83  36.84  3.61  502.09  1.0  0  0  0  0
497  11.68  38.72  3.59  463.59  1.0  0  0  0  0
498  12.75  36.71  3.28  548.28  1.0  0  0  1  0
499  12.13  38.19  4.02  597.74  1.0  0  0  0  1

[500 rows x 9 columns]
```

Out[292]:

	Time on App	Time on Website	Length of Membership	Yearly Amount Spent	purchase	avatar1	avatar2	clc1	clc2
0	10.16	37.76	4.78	521.24	1.0	0	0	0	0
1	13.46	37.24	2.94	503.98	1.0	0	0	0	0
2	12.01	36.53	4.71	576.48	1.0	0	0	0	0
3	10.1	38.04	4.24	418.6	0.0	0	0	0	0
4	11.45	37.58	2.59	420.74	0.0	0	0	0	0

In [294]:

```
train,test = np.split(df.sample(frac=1),[int(0.8*len(df))])
```

In [295]:

```
train.shape, test.shape
```

Out[295]:

```
((400, 9), (100, 9))
```

In [296]:

```
train = pd.DataFrame(train)
train.columns =df.columns
print(train.head())

test = pd.DataFrame(test)
test.columns =df.columns
print(test.head())
```

```
     Time on App Time on Website Length of Membership Yearly Amount Spent  \
235        11.08           37.96                 4.72              517.17
11          12.6           37.37                 3.47              501.93
55         12.36           38.04                 3.31              468.91
473        11.47           35.68                 1.81              374.27
201        12.52           37.15                 2.67              487.38

     purchase avatar1 avatar2 clc1 clc2
235       1.0       1       0    0    0
11        1.0       0       0    0    0
55        1.0       0       0    1    0
473       0.0       0       0    0    0
201       1.0       0       0    0    0
     Time on App Time on Website Length of Membership Yearly Amount Spent  \
97         12.91           36.05                 3.49              547.71
496        11.83           36.84                 3.61              502.09
57         11.17           35.63                 5.46              587.57
334        13.29           38.63                 3.87              543.34
95         11.33           35.46                 4.54              568.72

     purchase avatar1 avatar2 clc1 clc2
97        1.0       0       0    0    0
496       1.0       0       0    0    0
57        1.0       0       0    0    0
334       1.0       0       0    0    0
95        1.0       0       0    0    0
```

In [297]:

```
def column_to_move(df):
    column_to_move = df.pop("purchase")
    df.insert(8, "purchase", column_to_move)
    return df

train = column_to_move(train)
test = column_to_move(test)

train,test
```

Out[297]:

```
(     Time on App Time on Website Length of Membership Yearly Amount Spent  \
 235        11.08           37.96                 4.72              517.17   
 11          12.6           37.37                 3.47              501.93   
 55         12.36           38.04                 3.31              468.91   
 473        11.47           35.68                 1.81              374.27   
 201        12.52           37.15                 2.67              487.38   
 ..           ...             ...                  ...                 ...   
 439        13.15           36.62                 2.49              470.45   
 339        11.56           35.98                 1.48              282.47   
 432         12.7           35.36                  4.0               553.6   
 160        11.75           36.94                  0.8              298.76   
 375        12.43           37.63                 4.33              532.72   
 
     avatar1 avatar2 clc1 clc2 purchase  
 235       1       0    0    0      1.0  
 11        0       0    0    0      1.0  
 55        0       0    1    0      1.0  
 473       0       0    0    0      0.0  
 201       0       0    0    0      1.0  
 ..      ...     ...  ...  ...      ...  
 439       0       0    0    0      1.0  
 339       0       0    0    0      0.0  
 432       0       0    0    0      1.0  
 160       0       0    0    0      0.0  
 375       0       0    0    0      1.0  
 
 [400 rows x 9 columns],
     Time on App Time on Website Length of Membership Yearly Amount Spent  \
 97         12.91           36.05                 3.49              547.71   
 496        11.83           36.84                 3.61              502.09   
 57         11.17           35.63                 5.46              587.57   
 334        13.29           38.63                 3.87              543.34   
 95         11.33           35.46                 4.54              568.72   
 ..           ...             ...                  ...                 ...   
 393        11.54           37.53                 2.92              431.62   
 210        12.05           38.51                 2.85              409.09   
 215        11.67           37.34                 4.26              567.48   
 461         12.5           38.05                 4.64              616.85   
 368        11.41           36.38                 4.04              541.05   
 
     avatar1 avatar2 clc1 clc2 purchase  
 97        0       0    0    0      1.0  
 496       0       0    0    0      1.0  
 57        0       0    0    0      1.0  
 334       0       0    0    0      1.0  
 95        0       0    0    0      1.0  
 ..      ...     ...  ...  ...      ...  
 393       0       0    0    0      0.0  
 210       0       0    0    0      0.0  
 215       0       0    0    0      1.0  
 461       0       0    0    0      1.0  
 368       0       0    0    0      1.0  
 
 [100 rows x 9 columns])
```

In [305]:

```
def resample(dataframe, oversample=False):
    x = dataframe[dataframe.columns[:-1]].values
    y = dataframe[dataframe.columns[-1]].astype('int').values

    if oversample:
        ros = RandomOverSampler()
        x,y = ros.fit_resample(x,y)

    data = np.hstack((x, np.reshape(y,(-1,1))))
    return data, x, y
```

In [306]:

```
train, X_train, y_train = resample(train, oversample = True)
test, X_test, y_test = resample(test, oversample = False)
```

In [332]:

```
from sklearn.linear_model import LogisticRegression

lg_model = LogisticRegression(solver='lbfgs', max_iter=100)
lg_model.fit(X_train, y_train)
```

```
C:\Users\creat\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:458: Co
nvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-lear
n.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (h
ttps://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[332]:

```
▾ LogisticRegression
LogisticRegression()
```

In [333]:

```
y_pred = lg_model.predict(X_test)
```

In [334]:

```
from sklearn.metrics import classification_report
```

In [335]:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.93      1.00      0.96        26
           1       1.00      0.97      0.99        74

    accuracy                           0.98       100
   macro avg       0.96      0.99      0.97       100
weighted avg       0.98      0.98      0.98       100
```

Summary

1. Created and cleaned address to get relaitable data point for the users to see similarity in behaviour-- built 2 more colunms -- cleaned_address_loc and cleaned_address_county
2. Load the data and cleaned the columns name
3. Built the purchase colunms based on the contition mentioned
4. Looked on the basic stats before cleaning and resampling the data (describe and info)
5. Vectorize clean_address_county and avatar to use them in the regression
6. Split into train and test as mentioned (80:20)
7. Resample -- oversample to have a decent data to make generic model
8. build logistic regression model and predict using the same on the test data
9. Showcase the stats [F1 : 98% accuracy]

Insights

1. Higher the time spend on App and lenght of membership -- higher the probaboility they will make a purchase
2. App is more efective then web for purchase conversion
3. Lenght of membership has highest impact on the purchase