In [1]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [3]:
```
df=pd.read_csv("C3_bot_detection_data.csv")
df
```

Out[3]:

	User ID	Username	Tweet	Retweet Count	Mention Count	Follower Count	Verified	Bot Label	Location	Creat
0	132131	flong	Station activity person against natural majori...	85	1	2353	False	1	Adkinston	202 05-15:29:
1	289683	hinesstephanie	Authority research natural life material staff...	55	5	9617	True	0	Sanderston	202 11-05:18:
2	779715	roberttran	Manage whose quickly especially foot none to g...	6	2	4363	True	0	Harrisonfurt	202 08-03:16:
3	696168	pmason	Just cover eight opportunity strong policy which.	54	5	2242	True	1	Martinezberg	202 08-22:27:
4	704441	noah87	Animal sign six data good or.	26	3	8438	False	1	Camachoville	202 04-21:24:
...	...	...	...	...	...	...	...	...	...	
49995	491196	uberg	Want but put card direction know miss former h...	64	0	9911	True	1	Lake Kimberlyburgh	202 04-11:06:
49996	739297	jessicamunoz	Provide whole maybe agree church respond most ...	18	5	9900	False	1	Greenbury	202 10-03:57:
49997	674475	lynncunningham	Bring different everyone international capital...	43	3	6313	True	1	Deborahfort	202 07-03:54:
49998	167081	richardthompson	Than about single generation itself seek sell ...	45	1	6343	False	0	Stephenside	202 03-12:13:
49999	311204	daniel29	Here morning class various room human true bec...	91	4	4006	False	0	Novakberg	202 12-06:11:

50000 rows × 11 columns

In [5]:
```
df1=df.iloc[:,3:8]
df1
```

Out[5]:

	Retweet Count	Mention Count	Follower Count	Verified	Bot Label
0	85	1	2353	False	1
1	55	5	9617	True	0
2	6	2	4363	True	0
3	54	5	2242	True	1
4	26	3	8438	False	1
...	...	...	...	...	...
49995	64	0	9911	True	1
49996	18	5	9900	False	1
49997	43	3	6313	True	1
49998	45	1	6343	False	0
49999	91	4	4006	False	0

50000 rows × 5 columns

In [6]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Retweet Count   50000 non-null  int64
 1   Mention Count   50000 non-null  int64
 2   Follower Count  50000 non-null  int64
 3   Verified        50000 non-null  bool
 4   Bot Label       50000 non-null  int64
dtypes: bool(1), int64(4)
memory usage: 1.6 MB
```

In [8]:
```
y=df1["Verified"]
x=df1.drop(["Verified"],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [9]:
```
lr=LogisticRegression()
lr.fit(x_train,y_train)
```

Out[9]:
```
LogisticRegression()
```

In [13]:
```
lr.predict(x_test)
```

Out[13]:
```
array([ True,  True,  True, ..., False,  True,  True])
```

In [14]:
```
lr.score(x_test,y_test)
```

Out[14]: 0.494

In [15]:
```
df2=pd.read_csv("C4_framingham.csv")
df2
```

Out[15]:

	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBl
	39	4.0	0	0.0	0.0	0	0	0	195.0	106.
	46	2.0	0	0.0	0.0	0	0	0	250.0	121.
	48	1.0	1	20.0	0.0	0	0	0	245.0	127.
	61	3.0	1	30.0	0.0	0	1	0	225.0	150.
	46	3.0	1	23.0	0.0	0	0	0	285.0	130.
	...	...	...	...	...	...	...	...	...	.
	50	1.0	1	1.0	0.0	0	1	0	313.0	179.
	51	3.0	1	43.0	0.0	0	0	0	207.0	126.
	48	2.0	1	20.0	NaN	0	0	0	248.0	131.
	44	1.0	1	15.0	0.0	0	0	0	210.0	126.
	52	2.0	0	0.0	0.0	0	0	0	269.0	133.

× 16 columns

In [16]:
```
df2=df2.dropna()
```

In [17]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3656 entries, 0 to 4237
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   male             3656 non-null   int64  
 1   age              3656 non-null   int64  
 2   education        3656 non-null   float64
 3   currentSmoker    3656 non-null   int64  
 4   cigsPerDay       3656 non-null   float64
 5   BPMeds           3656 non-null   float64
 6   prevalentStroke  3656 non-null   int64  
 7   prevalentHyp     3656 non-null   int64  
 8   diabetes         3656 non-null   int64  
 9   totChol          3656 non-null   float64
 10  sysBP            3656 non-null   float64
 11  diaBP            3656 non-null   float64
 12  BMI              3656 non-null   float64
 13  heartRate        3656 non-null   float64
 14  glucose          3656 non-null   float64
 15  TenYearCHD       3656 non-null   int64  
dtypes: float64(9), int64(7)
memory usage: 485.6 KB
```

In [26]:
```
y=df2["diabetes"]
x=df2.drop(["diabetes"],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [27]:
```
lr=LogisticRegression()
lr.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: Conve
rgenceWarning: 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-learn.or
g/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 (http
s://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[27]:
```
LogisticRegression()
```

In [29]:
```
val=[[1,34,5,1,4,1,0,1,123,108,89,29,84,70,1]]
lr.predict(val)
```

Out[29]:
```
array([0], dtype=int64)
```

In [31]:
```
lr.score(x_test,y_test)
```

Out[31]:
```
0.9817684594348223
```

In [30]:
```
df3=pd.read_csv("C5_health care diabetes.csv")
df3
```

Out[30]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Out
0	6	148	72	35	0	33.6	0.627	50	
1	1	85	66	29	0	26.6	0.351	31	
2	8	183	64	0	0	23.3	0.672	32	
3	1	89	66	23	94	28.1	0.167	21	
4	0	137	40	35	168	43.1	2.288	33	
...	...	...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.171	63	
764	2	122	70	27	0	36.8	0.340	27	
765	5	121	72	23	112	26.2	0.245	30	
766	1	126	60	0	0	30.1	0.349	47	
767	1	93	70	31	0	30.4	0.315	23	

768 rows × 9 columns

In [32]:
```
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [34]:
```
y=df3["Outcome"]
x=df3.drop(["Outcome"],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [35]:
```
lr=LogisticRegression()
lr.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: Conve
rgenceWarning: 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-learn.or
g/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 (http
s://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[35]:
```
LogisticRegression()
```

In [36]:
```
lr.predict(x_test)
```

Out[36]:
```
array([0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0,
       0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
       0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0,
       1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0], dtype=int64)
```

In [37]:
```
val1=[[1,34,5,1,4,1,123,10]]
lr.predict(val1)
```

Out[37]:
```
array([1], dtype=int64)
```

In [ ]:
```

```

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [3]:
```
df=pd.read_csv("C6_bmi.csv")
df
```

Out[3]:

	Gender	Height	Weight	Index
0	Male	174	96	4
1	Male	189	87	2
2	Female	185	110	4
3	Female	195	104	3
4	Male	149	61	3
...	...	...	...	...
495	Female	150	153	5
496	Female	184	121	4
497	Female	141	136	5
498	Male	150	95	5
499	Male	173	131	5

500 rows × 4 columns

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Gender  500 non-null    object
 1   Height  500 non-null    int64 
 2   Weight  500 non-null    int64 
 3   Index   500 non-null    int64 
dtypes: int64(3), object(1)
memory usage: 15.8+ KB
```

In [5]:
```
y=df["Gender"]
x=df.drop(["Gender"],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [7]:
```
lr=LogisticRegression()
lr.fit(x_train,y_train)
```

Out[7]:
```
LogisticRegression()
```

In [8]:
```
lr.predict(x_test)
```

Out[8]:
```
array(['Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female'],
      dtype=object)
```

In [9]:
```
lr.score(x_test,y_test)
```

Out[9]:
```
0.48
```

In [10]:
```
df1=pd.read_csv("c7_used_cars.csv")
df1
```

Out[10]:

	Unnamed: 0	model	year	price	transmission	mileage	fuelType	tax	mpg	engineSize	Make
0	0	T-Roc	2019	25000	Automatic	13904	Diesel	145	49.6	2.0	VW
1	1	T-Roc	2019	26883	Automatic	4562	Diesel	145	49.6	2.0	VW
2	2	T-Roc	2019	20000	Manual	7414	Diesel	145	50.4	2.0	VW
3	3	T-Roc	2019	33492	Automatic	4825	Petrol	145	32.5	2.0	VW
4	4	T-Roc	2019	22900	Semi-Auto	6500	Petrol	150	39.8	1.5	VW
...	...	...	...	...	...	...	...	...	...	...	...
99182	10663	A3	2020	16999	Manual	4018	Petrol	145	49.6	1.0	Audi
99183	10664	A3	2020	16999	Manual	1978	Petrol	150	49.6	1.0	Audi
99184	10665	A3	2020	17199	Manual	609	Petrol	150	49.6	1.0	Audi
99185	10666	Q3	2017	19499	Automatic	8646	Petrol	150	47.9	1.4	Audi
99186	10667	Q3	2016	15999	Manual	11855	Petrol	150	47.9	1.4	Audi

99187 rows × 11 columns

In [12]:
```
df2=df1.drop(["transmission","Make","model","Unnamed: 0"],axis=1)
df2
```

Out[12]:

	year	price	mileage	fuelType	tax	mpg	engineSize
0	2019	25000	13904	Diesel	145	49.6	2.0
1	2019	26883	4562	Diesel	145	49.6	2.0
2	2019	20000	7414	Diesel	145	50.4	2.0
3	2019	33492	4825	Petrol	145	32.5	2.0
4	2019	22900	6500	Petrol	150	39.8	1.5
...	...	...	...	...	...	...	...
99182	2020	16999	4018	Petrol	145	49.6	1.0
99183	2020	16999	1978	Petrol	150	49.6	1.0
99184	2020	17199	609	Petrol	150	49.6	1.0
99185	2017	19499	8646	Petrol	150	47.9	1.4
99186	2016	15999	11855	Petrol	150	47.9	1.4

99187 rows × 7 columns

In [13]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99187 entries, 0 to 99186
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   year        99187 non-null  int64  
 1   price       99187 non-null  int64  
 2   mileage     99187 non-null  int64  
 3   fuelType    99187 non-null  object 
 4   tax         99187 non-null  int64  
 5   mpg         99187 non-null  float64
 6   engineSize  99187 non-null  float64
dtypes: float64(2), int64(4), object(1)
memory usage: 5.3+ MB
```

In [14]:
```
y=df2["fuelType"]
x=df2.drop(["fuelType"],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [15]:
```
lr=LogisticRegression()
lr.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarn
ing: 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-learn.org/stable/m
odules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit
-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[15]:
```
LogisticRegression()
```

In [17]:
```
val=[[2019,25000,16545,145,44.6,1],[2018,68748,1235,108,38,2]]
lr.predict(val)
```

Out[17]:
```
array(['Diesel', 'Diesel'], dtype=object)
```

In [20]:
```
lr.score(x_test,y_test)
```

Out[20]:
```
0.7083375340256074
```

In [19]:
```
df3=pd.read_csv("C8_loan-train.csv")
df4=pd.read_csv("C8_loan-test.csv")
df3
```

Out[19]:

ried	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit
No	0	Graduate	No	5849	0.0	NaN	360.0	
Yes	1	Graduate	No	4583	1508.0	128.0	360.0	
Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	
Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	
No	0	Graduate	No	6000	0.0	141.0	360.0	
...	...	...	...	...	...	...	...	
No	0	Graduate	No	2900	0.0	71.0	360.0	
Yes	3+	Graduate	No	4106	0.0	40.0	180.0	
Yes	1	Graduate	No	8072	240.0	253.0	360.0	
Yes	2	Graduate	No	7583	0.0	187.0	360.0	
No	0	Graduate	Yes	4583	0.0	133.0	360.0	

In [22]:
```
df3["Loan_Status"]=df3["Loan_Status"].replace("Y",1,regex=True)
df3["Loan_Status"]=df3["Loan_Status"].replace("N",0,regex=True)
df3
```

Out[22]:

ried	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit
No	0	Graduate	No	5849	0.0	NaN	360.0	
Yes	1	Graduate	No	4583	1508.0	128.0	360.0	
Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	
Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	
No	0	Graduate	No	6000	0.0	141.0	360.0	
...	...	...	...	...	...	...	...	
No	0	Graduate	No	2900	0.0	71.0	360.0	
Yes	3+	Graduate	No	4106	0.0	40.0	180.0	
Yes	1	Graduate	No	8072	240.0	253.0	360.0	
Yes	2	Graduate	No	7583	0.0	187.0	360.0	
No	0	Graduate	Yes	4583	0.0	133.0	360.0	

In [45]:
```
df3_tr=df3.drop(["Dependents","Married","Loan_ID","Education","Gender","Property_Area","Loan_Statu
df3_tr
```

Out[45]:

	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	No	5849	0.0	NaN	360.0	1.0
1	No	4583	1508.0	128.0	360.0	1.0
2	Yes	3000	0.0	66.0	360.0	1.0
3	No	2583	2358.0	120.0	360.0	1.0
4	No	6000	0.0	141.0	360.0	1.0
...	...	...	...	...	...	...
609	No	2900	0.0	71.0	360.0	1.0
610	No	4106	0.0	40.0	180.0	1.0
611	No	8072	240.0	253.0	360.0	1.0
612	No	7583	0.0	187.0	360.0	1.0
613	Yes	4583	0.0	133.0	360.0	0.0

614 rows × 6 columns

In [46]:
```
df_tr=df3_tr.dropna()
```

In [53]:
```
df_tr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 504 entries, 1 to 613
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Self_Employed      504 non-null    object 
 1   ApplicantIncome    504 non-null    int64  
 2   CoapplicantIncome  504 non-null    float64
 3   LoanAmount         504 non-null    float64
 4   Loan_Amount_Term   504 non-null    float64
 5   Credit_History     504 non-null    float64
dtypes: float64(4), int64(1), object(1)
memory usage: 27.6+ KB
```

In [54]:
```
y=df_tr["Self_Employed"]
x=df_tr.drop(["Self_Employed"],axis=1)
f=StandardScaler().fit_transform(x)
lr.fit(f,y)
```

Out[54]:
```
LogisticRegression()
```

In [55]:
```
df4_te=df4.drop(["Education","Loan_ID","Gender","Married","Dependents","Property_Area","Self_Emplo
```

In [56]:
```
df4_te=df4_te.dropna()
```

In [57]:
```
df4_te.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 328 entries, 0 to 366
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   ApplicantIncome    328 non-null    int64  
 1   CoapplicantIncome  328 non-null    int64  
 2   LoanAmount         328 non-null    float64
 3   Loan_Amount_Term   328 non-null    float64
 4   Credit_History     328 non-null    float64
dtypes: float64(3), int64(2)
memory usage: 15.4 KB
```

In [58]:
```python
lr.predict(df4_te)
```

Out[58]:
```
array(['Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes'], dtype=object)
```

In [60]:
```python
lr.predict_proba(df4_te)
```

```
       [0.00000000e+00, 1.00000000e+00],
       [0.00000000e+00, 1.00000000e+00],
       [0.00000000e+00, 1.00000000e+00],
       [0.00000000e+00, 1.00000000e+00],
       [0.00000000e+00, 1.00000000e+00],
       [0.00000000e+00, 1.00000000e+00],
       [0.00000000e+00, 1.00000000e+00],
       [0.00000000e+00, 1.00000000e+00],
       [0.00000000e+00, 1.00000000e+00],
       [0.00000000e+00, 1.00000000e+00],
       [0.00000000e+00, 1.00000000e+00],
       [0.00000000e+00, 1.00000000e+00],
       [0.00000000e+00, 1.00000000e+00],
       [0.00000000e+00, 1.00000000e+00],
       [0.00000000e+00, 1.00000000e+00],
       [0.00000000e+00, 1.00000000e+00],
       [0.00000000e+00, 1.00000000e+00],
       [0.00000000e+00, 1.00000000e+00],
       [0.00000000e+00, 1.00000000e+00],
       [0.00000000e+00, 1.00000000e+00],
```

In [67]:
```
df5=pd.read_csv("C9_Data.csv")
df5
```

Out[67]:

	row_id	user_id	timestamp	gate_id
0	0	18	2022-07-29 09:08:54	7
1	1	18	2022-07-29 09:09:54	9
2	2	18	2022-07-29 09:09:54	9
3	3	18	2022-07-29 09:10:06	5
4	4	18	2022-07-29 09:10:08	5
...	...	...	...	...
37513	37513	6	2022-12-31 20:38:56	11
37514	37514	6	2022-12-31 20:39:22	6
37515	37515	6	2022-12-31 20:39:23	6
37516	37516	6	2022-12-31 20:39:31	9
37517	37517	6	2022-12-31 20:39:31	9

37518 rows × 4 columns

In [68]:
```
df5=df5.drop(["timestamp"],axis=1)
df5.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37518 entries, 0 to 37517
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   row_id   37518 non-null  int64
 1   user_id  37518 non-null  int64
 2   gate_id  37518 non-null  int64
dtypes: int64(3)
memory usage: 879.5 KB
```

In [69]:
```
y=df5["user_id"]
x=df5.drop(["user_id"],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [70]:
```
lr=LogisticRegression()
lr.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarn
ing: 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-learn.org/stable/m
odules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit
-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[70]:
```
LogisticRegression()
```

In [71]:
```
lr.predict(x_test)
```

Out[71]:
```
array([55, 55, 55, ..., 55, 55, 55], dtype=int64)
```

In [72]:
```
lr.score(x_test,y_test)
```

Out[72]:
```
0.05863539445628998
```

In [ ]:
```

```