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
```

C6

In [2]:
```
df=pd.read_csv("C6_bmi.csv")
df
```

Out[2]:

	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 [3]:
```
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 [4]:
```
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 [5]:
```
lr=LogisticRegression()
lr.fit(x_train,y_train)
```

Out[5]:
```
LogisticRegression()
```

In [6]:
```
lr.predict(x_test)
```

Out[6]:
```
array(['Male', 'Female', 'Male', 'Male', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Male', 'Female', 'Female', 'Male', 'Female',
       'Male', 'Female', 'Female', 'Female', 'Male', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Male', 'Female', 'Male',
       'Female', 'Male', 'Female', 'Female', 'Female', 'Male', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Male', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Male', 'Female',
       'Female', 'Female', 'Male', 'Male', 'Female', 'Female', 'Male',
       'Female', 'Female', 'Female', 'Male', 'Male', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Male', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Male', 'Female', 'Male', 'Female',
       'Female', 'Male', 'Male', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Male', 'Male', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female', 'Male',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Male', 'Female', 'Female', 'Male',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Male', 'Female'], dtype=object)
```

In [7]:
```
lr.score(x_test,y_test)
```

Out[7]:
```
0.4666666666666667
```

C7

In [8]:
```
df1=pd.read_csv("c7_used_cars.csv")
df1
```

Out[8]:

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

99187 rows × 11 columns

In [9]:
```
df2=df1.drop(["transmission","Make","model","Unnamed: 0"],axis=1)
df2
```

Out[9]:

	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 [10]:
```
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 [11]:
```
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 [12]:
```
lr=LogisticRegression()
lr.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
763: ConvergenceWarning: 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/modules/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[12]:
```
LogisticRegression()
```

In [13]:
```
val=[[2019,25000,16545,145,44.6,1],[2018,68748,1235,108,38,2]]
lr.predict(val)
```

Out[13]:
```
array(['Diesel', 'Diesel'], dtype=object)
```

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

Out[14]:
```
0.7067580737305508
```

C8

In [15]:
```
df3=pd.read_csv("C8_loan-train.csv")
df4=pd.read_csv("C8_loan-test.csv")
df3
```

Out[15]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coap
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	
4	LP001008	Male	No	0	Graduate	No	6000	
...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	
610	LP002979	Male	Yes	3+	Graduate	No	4106	
611	LP002983	Male	Yes	1	Graduate	No	8072	
612	LP002984	Male	Yes	2	Graduate	No	7583	
613	LP002990	Female	No	0	Graduate	Yes	4583	

614 rows × 13 columns

In [16]:
```
df3["Loan_Status"]=df3["Loan_Status"].replace("Y",1,regex=True)
df3["Loan_Status"]=df3["Loan_Status"].replace("N",0,regex=True)
df3
```

Out[16]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coap
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	
4	LP001008	Male	No	0	Graduate	No	6000	
...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	
610	LP002979	Male	Yes	3+	Graduate	No	4106	
611	LP002983	Male	Yes	1	Graduate	No	8072	
612	LP002984	Male	Yes	2	Graduate	No	7583	
613	LP002990	Female	No	0	Graduate	Yes	4583	

614 rows × 13 columns

In [17]:
```
df3_tr=df3.drop(["Dependents","Married","Loan_ID","Education","Gender","Proper
df3_tr
```

Out[17]:

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

614 rows × 6 columns

In [18]:
```
df_tr=df3_tr.dropna()
```

In [19]:
```
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 [20]:
```
y=df_tr["Self_Employed"]
x=df_tr.drop(["Self_Employed"],axis=1)
f=StandardScaler().fit_transform(x)
lr.fit(f,y)
```

Out[20]:
```
LogisticRegression()
```

In [21]:
```
df4_te=df4.drop(["Education","Loan_ID","Gender","Married","Dependents","Proper
```

In [22]:
```
df4_te=df4_te.dropna()
```

In [23]:
```
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 [24]:
```
lr.predict(df4_te)
```

Out[24]:
```
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 [25]:
```
lr.predict_proba(df4_te)
```

Out[25]:
```
array([[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],
```

C9

In [26]:
```
df5=pd.read_csv("C9_Data.csv")
df5
```

Out[26]:

	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 [27]:
```
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 [28]:
```
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 [29]:
```
lr=LogisticRegression()
lr.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
763: ConvergenceWarning: 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/modules/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[29]:
```
LogisticRegression()
```

In [30]:
```
lr.predict(x_test)
```

Out[30]:
```
array([55, 55, 55, ..., 55, 55, 55], dtype=int64)
```

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

Out[31]:
```
0.0570362473347548
```