In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
import re
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
```

DataSet BMI

In [2]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\C6_bmi.csv")
a
```

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]:

```
a.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]:

```
a.columns
```

Out[4]:

```
Index(['Gender', 'Height', 'Weight', 'Index'], dtype='object')
```

In [7]:

```
b=a[['Height', 'Weight']]
c=a['Index']
```

In [8]:

```
d=StandardScaler().fit_transform(b)
```

In [9]:

```
logr=LogisticRegression()
logr.fit(d,c)
```

Out[9]:

```
LogisticRegression()
```

In [10]:

```
e=[[20,30]]
```

In [11]:

```
prediction=logr.predict(e)
print(prediction)
```

```
[5]
```

In [12]:

```
logr.classes_
```

Out[12]:

```
array([0, 1, 2, 3, 4, 5], dtype=int64)
```

In [13]:

```
logr.predict_proba(e)[0][0]
```

Out[13]:

```
3.746235015459396e-94
```

In [14]:

```
logr.predict_proba(e)[0][1]
```

Out[14]:

```
4.2522455960869615e-88
```

DataSet Used Cars

In [15]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\c7_used_cars.csv")
a
```

Out[15]:

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

99187 rows × 11 columns

In [16]:

```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99187 entries, 0 to 99186
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Unnamed: 0    99187 non-null  int64  
 1   model         99187 non-null  object 
 2   year          99187 non-null  int64  
 3   price         99187 non-null  int64  
 4   transmission  99187 non-null  object 
 5   mileage       99187 non-null  int64  
 6   fuelType      99187 non-null  object 
 7   tax           99187 non-null  int64  
 8   mpg           99187 non-null  float64
 9   engineSize    99187 non-null  float64
 10  Make          99187 non-null  object 
dtypes: float64(2), int64(5), object(4)
memory usage: 8.3+ MB
```

In [17]:

```
a.columns
```

Out[17]:

```
Index(['Unnamed: 0', 'model', 'year', 'price', 'transmission', 'mileage',
       'fuelType', 'tax', 'mpg', 'engineSize', 'Make'],
      dtype='object')
```

In [18]:

```
b=a[['Unnamed: 0', 'year', 'price', 'mileage', 'tax', 'mpg', 'engineSize']]
c=a['Make']
```

In [19]:

```
d=StandardScaler().fit_transform(b)
```

In [20]:

```
logr=LogisticRegression()
logr.fit(d,c)
```

Out[20]:

```
LogisticRegression()
```

In [21]:

```
e=[[12,24,36,48,60,72,84]]
```

In [22]:

```
prediction=logr.predict(e)
print(prediction)
```

```
['BMW']
```

In [23]:

```
logr.classes_
```

Out[23]:

```
array(['Audi', 'BMW', 'VW', 'ford', 'hyundi', 'merc', 'skoda', 'toyota',
       'vauxhall'], dtype=object)
```

In [24]:

```
logr.predict_proba(e)[0][0]
```

Out[24]:

```
3.509334644174678e-56
```

In [25]:

```
logr.predict_proba(e)[0][1]
```

Out[25]:

```
0.9999999999999998
```

DataSet Loan Test

In [26]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\C8_loan-test.csv")
a
```

Out[26]:

Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
Male	Yes	0	Graduate	No	5720	0
Male	Yes	1	Graduate	No	3076	1500
Male	Yes	2	Graduate	No	5000	1800
Male	Yes	2	Graduate	No	2340	2546
Male	No	0	Not Graduate	No	3276	0
...	...	...	...	...	...	...
Male	Yes	3+	Not Graduate	Yes	4009	1777
Male	Yes	0	Graduate	No	4158	709
Male	No	0	Graduate	No	3250	1993
Male	Yes	0	Graduate	No	5000	2393
Male	No	0	Graduate	Yes	9200	0

columns

In [27]:

```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            367 non-null    object 
 1   Gender             356 non-null    object 
 2   Married            367 non-null    object 
 3   Dependents         357 non-null    object 
 4   Education          367 non-null    object 
 5   Self_Employed      344 non-null    object 
 6   ApplicantIncome    367 non-null    int64  
 7   CoapplicantIncome  367 non-null    int64  
 8   LoanAmount         362 non-null    float64
 9   Loan_Amount_Term   361 non-null    float64
 10  Credit_History     338 non-null    float64
 11  Property_Area      367 non-null    object 
dtypes: float64(3), int64(2), object(7)
memory usage: 34.5+ KB
```

In [28]:

```
b=a.fillna(value=12)
b
```

Out[28]:

Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
Male	Yes	0	Graduate	No	5720	0
Male	Yes	1	Graduate	No	3076	1500
Male	Yes	2	Graduate	No	5000	1800
Male	Yes	2	Graduate	No	2340	2546
Male	No	0	Not Graduate	No	3276	0
...	...	...	...	...	...	...
Male	Yes	3+	Not Graduate	Yes	4009	1777
Male	Yes	0	Graduate	No	4158	709
Male	No	0	Graduate	No	3250	1993
Male	Yes	0	Graduate	No	5000	2393
Male	No	0	Graduate	Yes	9200	0

columns

In [29]:

```
b.columns
```

Out[29]:

```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmoun
t',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area'],
      dtype='object')
```

In [30]:

```
c=b[['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History']]
d=b['Property_Area']
```

In [31]:

```
e=StandardScaler().fit_transform(c)
```

In [32]:

```
logr=LogisticRegression()
logr.fit(e,d)
```

Out[32]:

```
LogisticRegression()
```

In [35]:

```
f=[[12,34,56,78,90]]
```

In [36]:

```
prediction=logr.predict(f)
print(prediction)
```

```
['Semiurban']
```

In [37]:

```
logr.classes_
```

Out[37]:

```
array(['Rural', 'Semiurban', 'Urban'], dtype=object)
```

In [38]:

```
logr.predict_proba(f)[0][0]
```

Out[38]:

```
0.3576578823114423
```

In [39]:

```
logr.predict_proba(f)[0][1]
```

Out[39]:

```
0.6416590092728025
```

DataSet Loan Train

In [40]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\C8_loan-train.csv")
a
```

Out[40]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
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 [41]:

```
b=a.fillna(value=20)
b
```

Out[41]:

ried	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount
No	0	Graduate	No	5849	0.0	20.0
Yes	1	Graduate	No	4583	1508.0	128.0
Yes	0	Graduate	Yes	3000	0.0	66.0
Yes	0	Not Graduate	No	2583	2358.0	120.0
No	0	Graduate	No	6000	0.0	141.0
...	...	...	...	...	...	...
No	0	Graduate	No	2900	0.0	71.0
Yes	3+	Graduate	No	4106	0.0	40.0
Yes	1	Graduate	No	8072	240.0	253.0
Yes	2	Graduate	No	7583	0.0	187.0
No	0	Graduate	Yes	4583	0.0	133.0

In [42]:

```
b.columns
```

Out[42]:

```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmoun
t',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Statu
s'],
      dtype='object')
```

In [43]:

```
c=b[['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History']]
d=b['Loan_Status']
```

In [44]:

```
e=StandardScaler().fit_transform(c)
```

In [45]:

```
logr=LogisticRegression()
logr.fit(e,d)
```

Out[45]:

```
LogisticRegression()
```

In [46]:

```
f=[[78,90,32,54,57]]
```

In [47]:

```
prediction=logr.predict(f)
print(prediction)
```

```
['N']
```

In [48]:

```
logr.classes_
```

Out[48]:

```
array(['N', 'Y'], dtype=object)
```

In [49]:

```
logr.predict_proba(f)[0][0]
```

Out[49]:

```
0.9336290376462292
```

In [50]:

```
logr.predict_proba(f)[0][1]
```

Out[50]:

```
0.0663709623537708
```

DataSet Data

In [51]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\C9_Data.csv")
a
```

Out[51]:

	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 [52]:

```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37518 entries, 0 to 37517
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   row_id     37518 non-null  int64 
 1   user_id    37518 non-null  int64 
 2   timestamp  37518 non-null  object
 3   gate_id    37518 non-null  int64 
dtypes: int64(3), object(1)
memory usage: 1.1+ MB
```

In [53]:

```
b=a.head(1000)
b
```

Out[53]:

	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
...	...	...	...	...
995	995	3	2022-08-01 18:33:29	11
996	996	3	2022-08-01 18:33:48	4
997	997	3	2022-08-01 18:33:49	4
998	998	55	2022-08-01 18:35:45	7
999	999	55	2022-08-01 18:36:41	3

1000 rows × 4 columns

In [54]:

```
b.columns
```

Out[54]:

```
Index(['row_id', 'user_id', 'timestamp', 'gate_id'], dtype='object')
```

Linear Regression

In [55]:

```
x=b[['row_id', 'user_id', 'gate_id']]
y=b['gate_id']
```

In [56]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [57]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[57]:

```
LinearRegression()
```

In [58]:

```
print(lr.intercept_)
```

```
-8.881784197001252e-16
```

In [59]:

```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[59]:

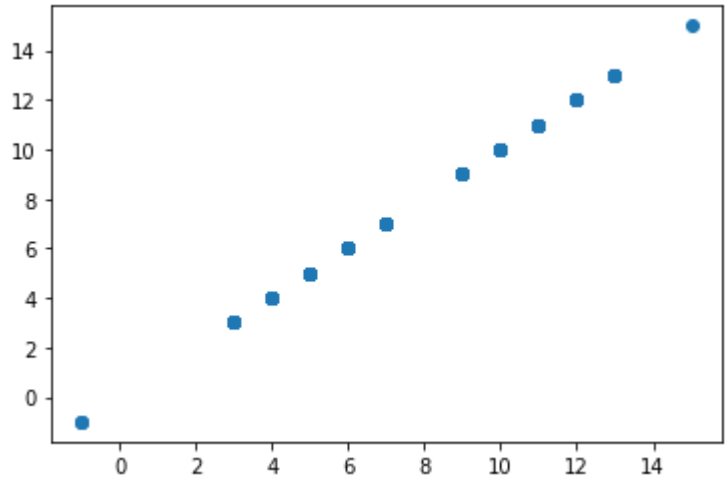
	Co-efficient
row_id	5.782769e-20
user_id	1.634082e-17
gate_id	1.000000e+00

In [60]:

```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[60]:

```
<matplotlib.collections.PathCollection at 0x12c32ead190>
```



In [61]:

```
print(lr.score(x_test,y_test))
```

```
1.0
```

Logistic Regression

In [62]:

```
b
```

Out[62]:

	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
...	...	...	...	...
995	995	3	2022-08-01 18:33:29	11
996	996	3	2022-08-01 18:33:48	4
997	997	3	2022-08-01 18:33:49	4
998	998	55	2022-08-01 18:35:45	7
999	999	55	2022-08-01 18:36:41	3

1000 rows × 4 columns

In [63]:

```
b.columns
```

Out[63]:

```
Index(['row_id', 'user_id', 'timestamp', 'gate_id'], dtype='object')
```

In [64]:

```
c=b[['row_id', 'user_id']]
d=b['gate_id']
```

In [65]:

```
e=StandardScaler().fit_transform(c)
```

In [66]:

```
logr=LogisticRegression()
logr.fit(e,d)
```

Out[66]:

```
LogisticRegression()
```

In [70]:

```
f=[[55,78]]
```

In [71]:

```
prediction=logr.predict(f)
print(prediction)
```

```
[-1]
```

In [72]:

```
logr.classes_
```

Out[72]:

```
array([-1,  3,  4,  5,  6,  7,  9, 10, 11, 12, 13, 15], dtype=int64)
```

In [73]:

```
logr.predict_proba(f)[0][0]
```

Out[73]:

```
0.9999999999997742
```

In [74]:

```
logr.predict_proba(f)[0][1]
```

Out[74]:

```
2.2057310206568094e-13
```

In [ ]:

```

```