qqhnq2jje

August 1, 2023

[6]:
```
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
```

[7]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[8]:
```
df=pd.read_csv("/content/drive/MyDrive/mydatasets/C9_Data.csv")
df
```

[8]:

	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 x 4 columns]

[9]:
```
df.head()
```

[9]:

	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
```

1 Data Cleaning and Data Preprocessing

[10]:
```
df.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
```

[11]:
```
df.describe()
```

[11]:

	row_id	user_id	gate_id
count	37518.000000	37518.000000	37518.000000
mean	18758.500000	28.219015	6.819607
std	10830.658036	17.854464	3.197746
min	0.000000	0.000000	-1.000000
25%	9379.250000	12.000000	4.000000
50%	18758.500000	29.000000	6.000000
75%	28137.750000	47.000000	10.000000
max	37517.000000	57.000000	16.000000

[12]:
```
df.columns
```

[12]:
```
Index(['row_id', 'user_id', 'timestamp', 'gate_id'], dtype='object')
```

[13]:
```
feature_matrix = df[['row_id', 'user_id']]
target_vector = df[['gate_id']]
```

[14]:
```
fs = StandardScaler().fit_transform(feature_matrix)
logr = LogisticRegression()
logr.fit(fs,target_vector)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
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
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[14]:
```
LogisticRegression()
```

[15]:
```
observation=[[1,2]]
prediction = logr.predict(observation)
print(prediction)
```

```
[3]
```

[16]:
```
logr.classes_
```

[16]:
```
array([-1,  0,  1,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16])
```

[17]:
```
logr.predict_proba(observation)
```

[17]:
```
array([[5.36517679e-03, 2.43221075e-05, 9.36568351e-05, 2.22025633e-01,
        2.19695882e-01, 7.52352405e-02, 5.84513730e-02, 7.17956781e-02,
        2.68284044e-03, 7.98655513e-02, 1.24425419e-01, 1.07054385e-01,
        2.51118120e-03, 7.57336969e-03, 2.68214159e-05, 2.29125763e-02,
        2.60893089e-04]])
```

[23]:
```
x = df[['row_id', 'user_id']]
y = df['gate_id']
```

[24]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

[25]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

[25]:
```
LinearRegression()
```

[26]:
```
lr.intercept_
```

[26]:
```
7.301593327105421
```

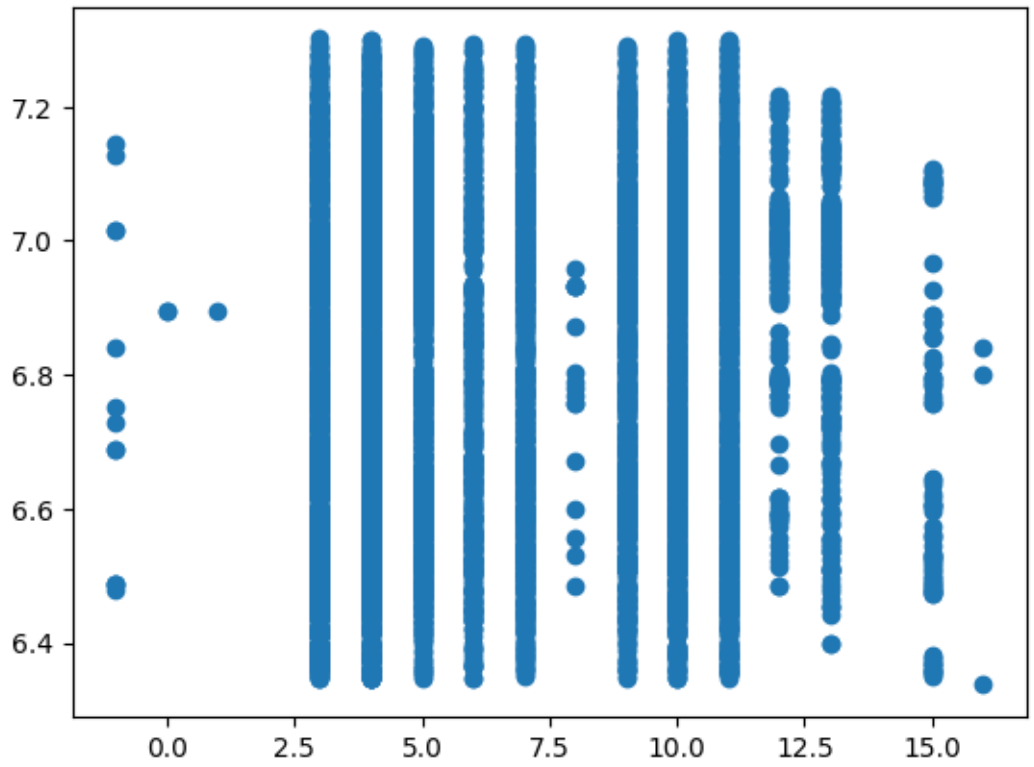
[27]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

[27]:

	Co-efficient
row_id	-0.000006
user_id	-0.013273

[29]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

[29]: <matplotlib.collections.PathCollection at 0x7c58e8ac1840>



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

[31]: 0.004821018408127764

[32]:
```
lr.score(x_train,y_train)
```

[32]: 0.005812611039229809