# nalukanga winnie logistic regression

March 19, 2024

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
ds=pd.read_csv("C:\\Users\\HP\\Desktop\\Social_Network_Ads.csv")
ds
```

[2]:

| | User ID | Gender | Age | EstimatedSalary | Purchased |
|---|---|---|---|---|---|
| 0 | 15624510 | Male | 19 | 19000 | 0 |
| 1 | 15810944 | Male | 35 | 20000 | 0 |
| 2 | 15668575 | Female | 26 | 43000 | 0 |
| 3 | 15603246 | Female | 27 | 57000 | 0 |
| 4 | 15804002 | Male | 19 | 76000 | 0 |
| .. | … | … | … | … | … |
| 395 | 15691863 | Female | 46 | 41000 | 1 |
| 396 | 15706071 | Male | 51 | 23000 | 1 |
| 397 | 15654296 | Female | 50 | 20000 | 1 |
| 398 | 15755018 | Male | 36 | 33000 | 0 |
| 399 | 15594041 | Female | 49 | 36000 | 1 |

[400 rows x 5 columns]

[3]:
```
x=ds.drop(['Purchased','Gender'], axis=1)
x
```

[3]:

| | User ID | Age | EstimatedSalary |
|---|---|---|---|
| 0 | 15624510 | 19 | 19000 |
| 1 | 15810944 | 35 | 20000 |
| 2 | 15668575 | 26 | 43000 |
| 3 | 15603246 | 27 | 57000 |
| 4 | 15804002 | 19 | 76000 |
| .. | … | … | … |
| 395 | 15691863 | 46 | 41000 |
| 396 | 15706071 | 51 | 23000 |
| 397 | 15654296 | 50 | 20000 |
| 398 | 15755018 | 36 | 33000 |
| 399 | 15594041 | 49 | 36000 |

[400 rows x 3 columns]

[4]:
```
y=ds['Purchased']
y
```

[4]:
```
0      0
1      0
2      0
3      0
4      0
      ..
395    1
396    1
397    1
398    0
399    1
Name: Purchased, Length: 400, dtype: int64
```

[5]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y, test_size=0.
 ↪25,random_state=43)
x_test.shape
```

[5]:
```
(100, 3)
```

[6]:
```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model
```

[6]:
```
LogisticRegression()
```

[7]:
```
#checking for missing values
ds.isnull().sum()
```

[7]:
```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

[8]:
```
model.fit(x_train,y_train)
```

[8]:
```
LogisticRegression()
```

[9]:
```
y_pred=model.predict(x_test)
y_pred
```

[9]:
```
array([0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,
```

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], dtype=int64)
```

[10]:
```
#evaluation
from sklearn.metrics import precision_score,recall_score,accuracy_score,f1_score
```

[11]:
```
accuracy_score(y_test,y_pred)
```

[11]: 0.71

[12]:
```
precision_score(y_test,y_pred)
```

[12]: 0.7222222222222222

[13]:
```
recall_score(y_test,y_pred)
```

[13]: 0.35135135135135137

[14]:
```
f1_score(y_test,y_pred)
```

[14]: 0.4727272727272727

optimization

[15]:
```
winnie=LogisticRegression()
winnie
```

[15]: LogisticRegression()

[16]:
```
from sklearn.model_selection import GridSearchCV
param_grid={
    'penalty':['l1', 'l2', 'elasticnet', None],
    'solver':['lbfgs', 'liblinear', 'newton-cg', 'newton-cholesky', 'sag',␣
 ↪'saga'],
    'C':[1],
    'dual':[True,False]
}
param_grid
```

[16]:
```
{'penalty': ['l1', 'l2', 'elasticnet', None],
 'solver': ['lbfgs',
  'liblinear',
  'newton-cg',
  'newton-cholesky',
  'sag',
  'saga'],
 'C': [1],
```

```
 'dual': [True, False]}
```

[17]:
```
grid_search=GridSearchCV(winnie,param_grid,cv=5)
grid_search
```

[17]:
```
GridSearchCV(cv=5, estimator=LogisticRegression(),
             param_grid={'C': [1], 'dual': [True, False],
                         'penalty': ['l1', 'l2', 'elasticnet', None],
                         'solver': ['lbfgs', 'liblinear', 'newton-cg',
                                    'newton-cholesky', 'sag', 'saga']})
```

[18]:
```
import warnings
warnings.filterwarnings('ignore')
grid_search.fit(x_train,y_train)
```

[18]:
```
GridSearchCV(cv=5, estimator=LogisticRegression(),
             param_grid={'C': [1], 'dual': [True, False],
                         'penalty': ['l1', 'l2', 'elasticnet', None],
                         'solver': ['lbfgs', 'liblinear', 'newton-cg',
                                    'newton-cholesky', 'sag', 'saga']})
```

[19]:
```
best_params= grid_search.best_params_
best_params
```

[19]:
```
{'C': 1, 'dual': False, 'penalty': 'l2', 'solver': 'newton-cholesky'}
```

[20]:
```
best_winnie=LogisticRegression(**best_params)
best_winnie.fit(x_train,y_train)
```

[20]:
```
LogisticRegression(C=1, solver='newton-cholesky')
```

[21]:
```
y_pred=best_winnie.predict(x_test)
y_pred
```

[21]:
```
array([0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], dtype=int64)
```

[22]:
```
accuracy_score(y_test,y_pred)
```

[22]:
```
0.78
```

[23]:
```
precision_score(y_test,y_pred)
```

[23]:
```
0.7777777777777778
```

[24]:
```
recall_score(y_test,y_pred)
```

[24]: 0.5675675675675675

[25]:
```
f1_score(y_test,y_pred)
```

[25]: 0.65625

[ ]:
```

```