logisticRegression

May 11, 2023

[80]:
```
import pandas as pd
df=pd.read_csv(r"C:\Users\Rakesh\Downloads\HR_comma_sep.csv")
df.head()
```

[80]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	Department	salary
0	0.38	0.53	2	157	3	0	1	0	sales	low
1	0.80	0.86	5	262	6	0	1	0	sales	medium
2	0.11	0.88	7	272	4	0	1	0	sales	medium
3	0.72	0.87	5	223	5	0	1	0	sales	low
4	0.37	0.52	2	159	3	0	1	0	sales	low

[81]:
```
left=df[df.left==1]
left.shape
```

[81]: (3571, 10)

[82]:
```
retained=df[df.left==0]
retained.shape
```

[82]: (11428, 10)

[83]:
```
dfleft=df.groupby(df.left)
```

[84]:
```
dfleft.mean()
```

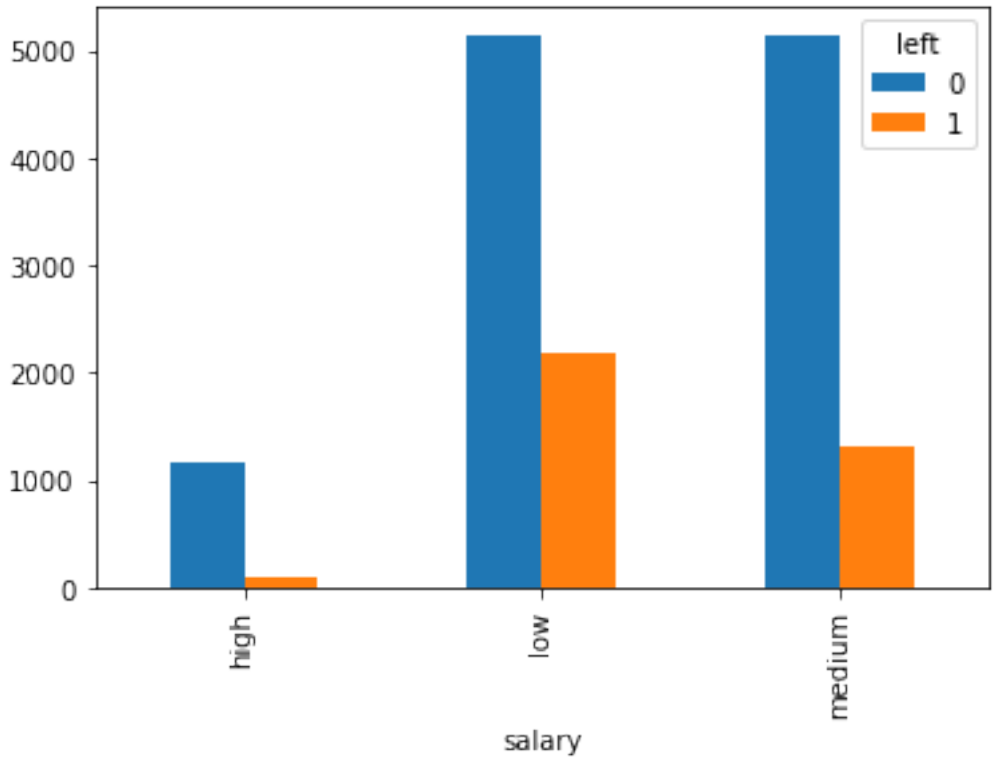
[84]:
```
      satisfaction_level  last_evaluation  number_project  \
left
0               0.666810         0.715473        3.786664
1               0.440098         0.718113        3.855503

      average_montly_hours  time_spend_company  Work_accident  \
left
0               199.060203            3.380032       0.175009
1               207.419210            3.876505       0.047326

      promotion_last_5years
left
0                  0.026251
1                  0.005321
```

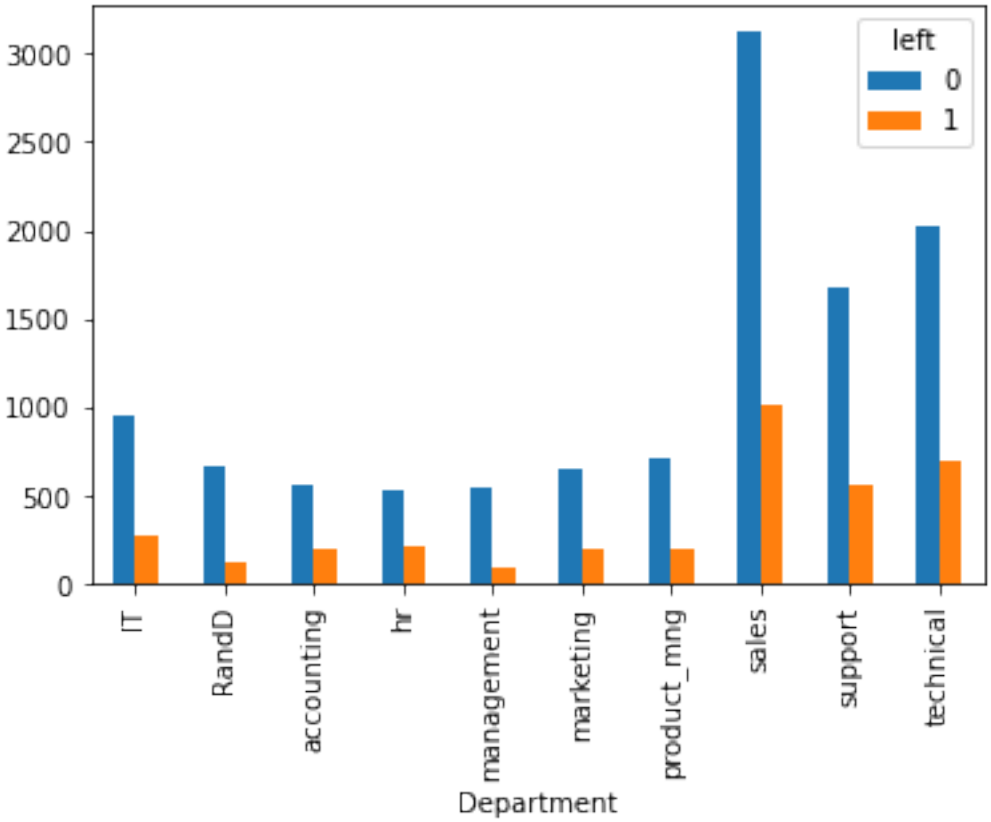
[85]:
```
import matplotlib.pyplot as plt
%matplotlib inline
pd.crosstab(df.salary,df.left).plot(kind='bar')
```

[85]:
```
<AxesSubplot:xlabel='salary'>
```



[86]:
```
pd.crosstab(df.Department,df.left).plot(kind='bar')
```

[86]: <AxesSubplot:xlabel='Department'>



[90]:
```
x=df.drop('Department',axis='columns')
x.head()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [90], in <cell line: 1>()
----> 1 x=df.drop('Department',axis='columns')
      2 x.head()

File ~\anaconda3\lib\site-packages\pandas\util\_decorators.py:311, in␣
 ↪deprecate_nonkeyword_arguments.<locals>.decorate.<locals>.wrapper(*args,␣
 ↪**kwargs)
    305 if len(args) > num_allow_args:
    306     warnings.warn(
    307         msg.format(arguments=arguments),
    308         FutureWarning,
    309         stacklevel=stacklevel,
    310     )
--> 311 return func(*args, **kwargs)
```

```
File ~\anaconda3\lib\site-packages\pandas\core\frame.py:4954, in DataFrame.
 ↪drop(self, labels, axis, index, columns, level, inplace, errors)
   4806 @deprecate_nonkeyword_arguments(version=None, allowed_args=["self",␣
 ↪"labels"])
   4807 def drop(
   4808     self,
   (…)
   4815     errors: str = "raise",
   4816 ):
   4817     """
   4818     Drop specified labels from rows or columns.
   4819
   (…)
   4952             weight  1.0     0.8
   4953     """
-> 4954     return super().drop(
   4955         labels=labels,
   4956         axis=axis,
   4957         index=index,
   4958         columns=columns,
   4959         level=level,
   4960         inplace=inplace,
   4961         errors=errors,
   4962     )

File ~\anaconda3\lib\site-packages\pandas\core\generic.py:4267, in NDFrame.
 ↪drop(self, labels, axis, index, columns, level, inplace, errors)
   4265 for axis, labels in axes.items():
   4266     if labels is not None:
-> 4267         obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4269 if inplace:
   4270     self._update_inplace(obj)

File ~\anaconda3\lib\site-packages\pandas\core\generic.py:4311, in NDFrame.
 ↪_drop_axis(self, labels, axis, level, errors, consolidate, only_slice)
   4309         new_axis = axis.drop(labels, level=level, errors=errors)
   4310     else:
-> 4311         new_axis = axis.drop(labels, errors=errors)
   4312     indexer = axis.get_indexer(new_axis)
   4314 # Case for non-unique axis
   4315 else:

File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:6644, in Index.
 ↪drop(self, labels, errors)
   6642 if mask.any():
   6643     if errors != "ignore":
-> 6644         raise KeyError(f"{list(labels[mask])} not found in axis")
```

```
   6645     indexer = indexer[~mask]
   6646 return self.delete(indexer)

KeyError: "['Department'] not found in axis"
```

[66]:
```
dummies=pd.get_dummies(df.salary)
dummies
```

[66]:
```
        high  low  medium
0          0    1       0
1          0    0       1
2          0    0       1
3          0    1       0
4          0    1       0
…      …   …      …
14994      0    1       0
14995      0    1       0
14996      0    1       0
14997      0    1       0
14998      0    1       0

[14999 rows x 3 columns]
```

[67]:
```
x_f=pd.concat([df,dummies],axis='columns')
x_f.head()
```

[67]:
```
    satisfaction_level  last_evaluation  number_project  average_montly_hours  \
0                 0.38             0.53               2                   157
1                 0.80             0.86               5                   262
2                 0.11             0.88               7                   272
3                 0.72             0.87               5                   223
4                 0.37             0.52               2                   159

   time_spend_company  Work_accident  left  promotion_last_5years  salary  \
0                   3              0     1                      0     low
1                   6              0     1                      0  medium
2                   4              0     1                      0  medium
3                   5              0     1                      0     low
4                   3              0     1                      0     low

   high  low  medium
0     0    1       0
1     0    0       1
2     0    0       1
3     0    1       0
4     0    1       0
```

[68]:
```
x_f1=x_f.drop('salary',axis='columns')
x_f1.head()
```

[68]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	high	low	medium
0	0.38	0.53	2	157	3	0	1	0	0	1	0
1	0.80	0.86	5	262	6	0	1	0	0	0	1
2	0.11	0.88	7	272	4	0	1	0	0	0	1
3	0.72	0.87	5	223	5	0	1	0	0	1	0
4	0.37	0.52	2	159	3	0	1	0	0	1	0

[69]:
```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
```

[70]:
```
y=df.left
```

[71]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_f1,y,test_size=0.2)
```

[72]:
```
x_train
```

[72]:

	satisfaction_level	last_evaluation	number_project
6058	0.58	0.58	5
9565	0.94	0.49	4
11687	0.71	0.78	4
13524	0.50	0.59	4
9139	0.69	0.68	4
…	…	…	…
3425	0.89	0.54	3
10996	0.63	0.57	3
12911	0.52	0.96	4
14212	0.80	0.86	5
4731	0.93	0.51	3

```
       average_montly_hours  time_spend_company  Work_accident  left  \
6058                    171                   2              0     0
9565                    220                   3              0     0
11687                   227                   2              0     0
13524                   214                   3              1     0
9139                    225                   3              0     0
…                       …                     …              …   …
3425                    214                   2              0     0
10996                   242                   3              0     0
12911                   171                   2              0     0
14212                   262                   6              0     1
4731                    196                   2              0     0

       promotion_last_5years  high  low  medium
6058                       0     0    0       1
9565                       0     0    1       0
11687                      0     0    1       0
13524                      0     0    1       0
9139                       0     0    1       0
…                        …   …   …      …
3425                       0     0    0       1
10996                      0     0    1       0
12911                      0     0    0       1
14212                      0     0    0       1
4731                       1     0    0       1

[11999 rows x 11 columns]
```

[73]:
```
model.fit(x_train,y_train)
```

```
C:\Users\Rakesh\anaconda3\lib\site-
packages\sklearn\linear_model\_logistic.py:814: 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(
```

[73]:
```
LogisticRegression()
```

[74]:
```
model.predict(x_test)
```

[74]:
```
array([0, 0, 1, …, 1, 0, 0], dtype=int64)
```

[ ]: