kaggle_amazon_access

September 21, 2021

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

[2]:
```
df=pd.read_csv(r"train.csv")
```

[3]:
```
df.head()
```

[3]:

	ACTION	RESOURCE	MGR_ID	ROLE_ROLLUP_1	ROLE_ROLLUP_2	ROLE_DEPTNAME	\
0	1	39353	85475	117961	118300	123472	
1	1	17183	1540	117961	118343	123125	
2	1	36724	14457	118219	118220	117884	
3	1	36135	5396	117961	118343	119993	
4	1	42680	5905	117929	117930	119569	

	ROLE_TITLE	ROLE_FAMILY_DESC	ROLE_FAMILY	ROLE_CODE
0	117905	117906	290919	117908
1	118536	118536	308574	118539
2	117879	267952	19721	117880
3	118321	240983	290919	118322
4	119323	123932	19793	119325

1 Column Name Description

ACTION ACTION is 1 if the resource was approved, 0 if the resource was not RESOURCE An ID for each resource MGR_ID The EMPLOYEE ID of the manager of the current EMPLOYEE ID record; an employee may have only one manager at a time ROLE_ROLLUP_1 Company role grouping category id 1 (e.g. US Engineering) ROLE_ROLLUP_2 Company role grouping category id 2 (e.g. US Retail) ROLE_DEPTNAME Company role department description (e.g. Retail) ROLE_TITLE Company role business title description (e.g. Senior Engineering Retail Manager) ROLE_FAMILY_DESC Company role family extended description (e.g. Retail Manager, Software Engineering) ROLE_FAMILY Company role family description (e.g. Retail Manager) ROLE_CODE Company role code; this code is unique to each role (e.g. Manager)

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32769 entries, 0 to 32768
Data columns (total 10 columns):
ACTION              32769 non-null int64
RESOURCE            32769 non-null int64
MGR_ID              32769 non-null int64
ROLE_ROLLUP_1       32769 non-null int64
ROLE_ROLLUP_2       32769 non-null int64
ROLE_DEPTNAME       32769 non-null int64
ROLE_TITLE          32769 non-null int64
ROLE_FAMILY_DESC    32769 non-null int64
ROLE_FAMILY         32769 non-null int64
ROLE_CODE           32769 non-null int64
dtypes: int64(10)
memory usage: 2.5 MB
```

[5]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

[6]:
```
allcolumns=df.columns
for item in allcolumns:
    print(df[item].nunique())
```

```
2
7518
4243
128
177
449
343
2358
67
343
```

[7]:
```
correl=df.corr()
```

[8]:
```
correl
```

[8]:

	ACTION	RESOURCE	MGR_ID	ROLE_ROLLUP_1	ROLE_ROLLUP_2	\
ACTION	1.000000	0.000185	-0.005167	-0.013702	0.005179	
RESOURCE	0.000185	1.000000	0.011088	-0.005016	0.013438	
MGR_ID	-0.005167	0.011088	1.000000	-0.007132	-0.000364	
ROLE_ROLLUP_1	-0.013702	-0.005016	-0.007132	1.000000	0.033358	
ROLE_ROLLUP_2	0.005179	0.013438	-0.000364	0.033358	1.000000	
ROLE_DEPTNAME	0.001025	0.030004	-0.009551	-0.009548	-0.006056	
ROLE_TITLE	-0.010169	0.002936	0.017864	0.010207	0.008305	

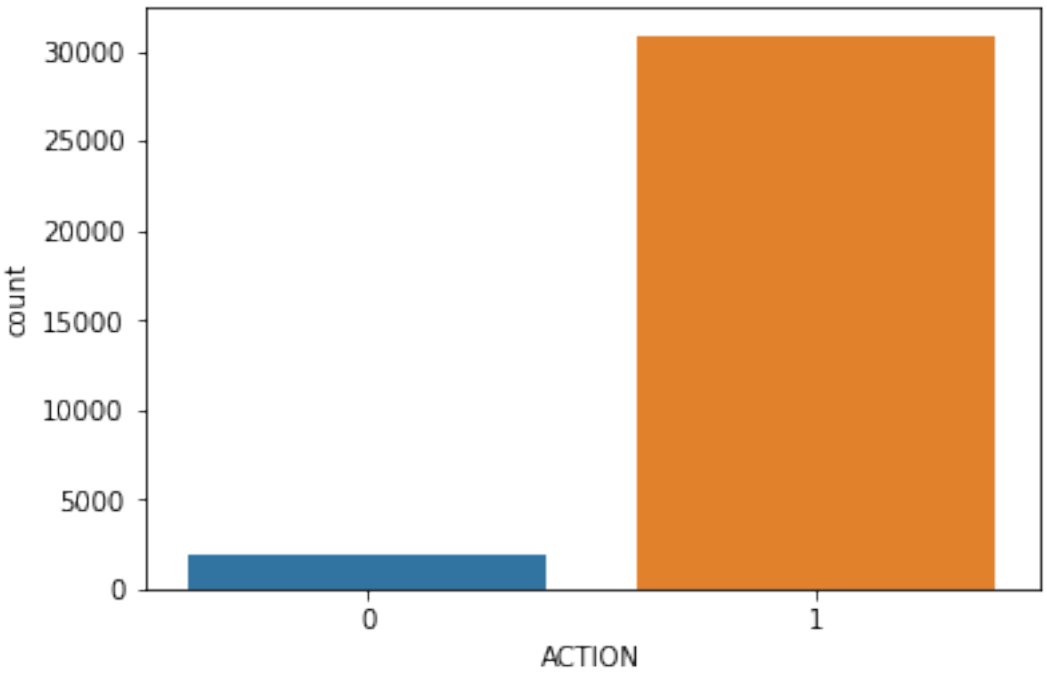
```
ROLE_FAMILY_DESC  0.003565  0.021029 -0.018488      -0.007546       0.018873
ROLE_FAMILY       0.000502  0.031060 -0.118254       0.029468       0.069558
ROLE_CODE         0.017147  0.007733 -0.004067      -0.024927       0.015117

                  ROLE_DEPTNAME  ROLE_TITLE  ROLE_FAMILY_DESC  ROLE_FAMILY  \
ACTION                 0.001025   -0.010169          0.003565     0.000502
RESOURCE               0.030004    0.002936          0.021029     0.031060
MGR_ID                -0.009551    0.017864         -0.018488    -0.118254
ROLE_ROLLUP_1         -0.009548    0.010207         -0.007546     0.029468
ROLE_ROLLUP_2         -0.006056    0.008305          0.018873     0.069558
ROLE_DEPTNAME          1.000000   -0.006932         -0.002877     0.031669
ROLE_TITLE            -0.006932    1.000000          0.170692    -0.012450
ROLE_FAMILY_DESC      -0.002877    0.170692          1.000000    -0.180596
ROLE_FAMILY            0.031669   -0.012450         -0.180596     1.000000
ROLE_CODE              0.010319    0.155920          0.092980    -0.148625

                  ROLE_CODE
ACTION             0.017147
RESOURCE           0.007733
MGR_ID            -0.004067
ROLE_ROLLUP_1     -0.024927
ROLE_ROLLUP_2      0.015117
ROLE_DEPTNAME      0.010319
ROLE_TITLE         0.155920
ROLE_FAMILY_DESC   0.092980
ROLE_FAMILY       -0.148625
ROLE_CODE          1.000000
```

[9]:
```
sns.countplot(df["ACTION"])
```

[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x26b55095e80>
```



[10]:
```
df.ACTION.nunique()
```

[10]: 2

[11]:
```
from sklearn.model_selection import train_test_split
```

[12]:
```
x=df.drop("ACTION",axis=1)
```

[13]:
```
y=df["ACTION"]
```

[14]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.3,␣
↪random_state=101)
```

[15]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

[16]:
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
```

[17]:
```
model = LogisticRegression()
model.fit(x_train, y_train)
predictedvalues=model.predict(x_test)
```

```
print(accuracy_score(y_test,predictedvalues))
print(confusion_matrix(y_test, predictedvalues))
```

```
0.939985759332723
[[   0  590]
 [   0 9241]]
```

2 Logistic Regression give accuracy of 93.9% but if we look into its confusion matrix, then we can reach to the conclusion that its not good model as it predicts everything as class 1 and has not predicted any itema as class 0, so it is affected by the biasness of model

[18]:
```
#Lets try random forest classifier.
model = RandomForestClassifier()
model.fit(x_train, y_train)
predictedvalues=model.predict(x_test)
print(accuracy_score(y_test,predictedvalues))
print(confusion_matrix(y_test, predictedvalues))
```

```
0.9456820262435154
[[ 231  359]
 [ 175 9066]]
```

3 even though improvement in accuracy from logistic regression to random forest is little but here we can see that confusion metrics shows that it has also classfied well better the the class which is having less number of examples in dataset which makes it really good method

[19]:
```
model = AdaBoostClassifier()
model.fit(x_train, y_train)
predictedvalues=model.predict(x_test)
print(accuracy_score(y_test,predictedvalues))
print(confusion_matrix(y_test, predictedvalues))
```

```
0.9400874783847014
[[   1  589]
 [   0 9241]]
```

4 Here also problem is same as in logistic regression that it can't work well with biased class

[20]:
```
model = GradientBoostingClassifier()
model.fit(x_train, y_train)
predictedvalues=model.predict(x_test)
print(accuracy_score(y_test,predictedvalues))
print(confusion_matrix(y_test, predictedvalues))
```

```
0.9405960736445936
[[   7  583]
 [   1 9240]]
```

5 This is also not giving good result compare to random forest as here also it is able to predict corectly only 7 data points of class 0 whose elements are so less.

6 Lets execute random forest on test data as we have choosen random forest as the final model

[21]:
```
model = RandomForestClassifier()
model.fit(x_train, y_train)
predictedvalues=model.predict(x_test)
print(accuracy_score(y_test,predictedvalues))
print(confusion_matrix(y_test, predictedvalues))
```

```
0.9459871833994508
[[ 223  367]
 [ 164 9077]]
```

[24]:
```
test_data = pd.read_csv(r"test.csv")
print (x_train.shape)
print (test_data.shape)
print (test_data.columns)
test_data.drop("id",axis=1, inplace=True)
predictedoutput = model.predict(test_data)
print (predictedoutput)
```

```
(22938, 9)
(912363, 10)
Index(['id', 'RESOURCE', 'MGR_ID', 'ROLE_ROLLUP_1', 'ROLE_ROLLUP_2',
       'ROLE_DEPTNAME', 'ROLE_TITLE', 'ROLE_FAMILY_DESC', 'ROLE_FAMILY',
       'ROLE_CODE'],
```

```
      dtype='object')
[1 1 1 … 1 1 1]
```

[ ]: