Design an algorithm to accurately predict the access status to certain resources of employees.

Problem Statement Scenario: When employees start working in an organization, they first need to obtain the computer access necessary to fulfill their role. This access may allow employees to read/manipulate resources through various applications or web portals. It is assumed that employees fulfilling the functions of a given role will access the same or similar resources. Often, employees figure out the access they need as they encounter roadblocks during their daily work (such as, not being able to log into a reporting portal). A knowledgeable supervisor then takes time to manually grant the access needed to overcome these obstacles. As employees change roles within a company, this access discovery/recovery cycle wastes a huge amount of time and money. There is a considerable amount of data regarding employees' roles within an organization and the resources to which they have access. Given the data related to current employees and their provisioned access, models can be built that automatically determine access privileges as employees enter and leave roles within a company. These auto-access models seek to minimize human involvement required to grant or revoke employee access.

Following actions should be performed:

Understand the type of data.

Identify the output variable.

Identify the factors which affect the output variable.

Check if there are any biases in your dataset.

Perform train test split.

Predict the accuracy using classification models.

Check and compare the accuracy of the different models.

In [4]:
```
import numpy as np
import pandas as pd
```

In [5]:
```
df=pd.read_csv(r"train.csv")
```

In [6]:
```
df.head()
```

Out[6]:

	ACTION	RESOURCE	MGR_ID	ROLE_ROLLUP_1	ROLE_ROLLUP_2	ROLE_DEPTNAME	ROLE_TITLE	ROLE_FAMILY_DESC	ROLE_FAMILY	ROLE_CO

	ACTION	RESOURCE	MGR_ID	ROLE_ROLLUP_1	ROLE_ROLLUP_2	ROLE_DEPTNAME	ROLE_TITLE	ROLE_FAMILY_DESC	ROLE_FAMILY	ROLE_CO
0	1	39353	85475	117961	118300	123472	117905	117906	290919	1179
1	1	17183	1540	117961	118343	123125	118536	118536	308574	1185
2	1	36724	14457	118219	118220	117884	117879	267952	19721	1178
3	1	36135	5396	117961	118343	119993	118321	240983	290919	1183
4	1	42680	5905	117929	117930	119569	119323	123932	19793	1193

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)

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32769 entries, 0 to 32768
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ACTION            32769 non-null  int64
 1   RESOURCE          32769 non-null  int64
```

```
 2   MGR_ID            32769 non-null  int64
 3   ROLE_ROLLUP_1     32769 non-null  int64
 4   ROLE_ROLLUP_2     32769 non-null  int64
 5   ROLE_DEPTNAME     32769 non-null  int64
 6   ROLE_TITLE        32769 non-null  int64
 7   ROLE_FAMILY_DESC  32769 non-null  int64
 8   ROLE_FAMILY       32769 non-null  int64
 9   ROLE_CODE         32769 non-null  int64
dtypes: int64(10)
memory usage: 2.5 MB
```

In [8]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [10]:
```
allcolumns=df.columns
for item in allcolumns:
    print(df[item].nunique())
```

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

In [11]:
```
correl=df.corr()
```

In [12]:
```
correl
```

Out[12]:

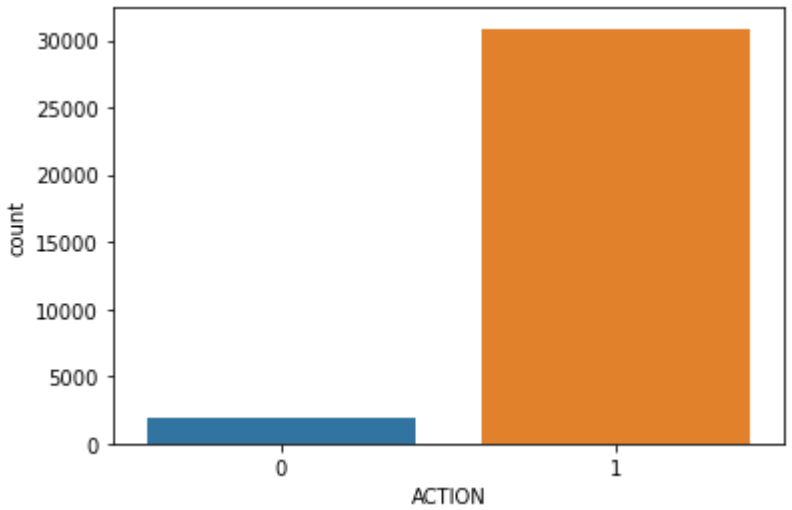
	ACTION	RESOURCE	MGR_ID	ROLE_ROLLUP_1	ROLE_ROLLUP_2	ROLE_DEPTNAME	ROLE_TITLE	ROLE_FAMILY_DESC	ROL
ACTION	1.000000	0.000185	-0.005167	-0.013702	0.005179	0.001025	-0.010169	0.003565	
RESOURCE	0.000185	1.000000	0.011088	-0.005016	0.013438	0.030004	0.002936	0.021029	
MGR_ID	-0.005167	0.011088	1.000000	-0.007132	-0.000364	-0.009551	0.017864	-0.018488	
ROLE_ROLLUP_1	-0.013702	-0.005016	-0.007132	1.000000	0.033358	-0.009548	0.010207	-0.007546	
ROLE_ROLLUP_2	0.005179	0.013438	-0.000364	0.033358	1.000000	-0.006056	0.008305	0.018873	
ROLE_DEPTNAME	0.001025	0.030004	-0.009551	-0.009548	-0.006056	1.000000	-0.006932	-0.002877	

	ACTION	RESOURCE	MGR_ID	ROLE_ROLLUP_1	ROLE_ROLLUP_2	ROLE_DEPTNAME	ROLE_TITLE	ROLE_FAMILY_DESC	ROL
ROLE_TITLE	-0.010169	0.002936	0.017864	0.010207	0.008305	-0.006932	1.000000	0.170692	
ROLE_FAMILY_DESC	0.003565	0.021029	-0.018488	-0.007546	0.018873	-0.002877	0.170692	1.000000	
ROLE_FAMILY	0.000502	0.031060	-0.118254	0.029468	0.069558	0.031669	-0.012450	-0.180596	
ROLE_CODE	0.017147	0.007733	-0.004067	-0.024927	0.015117	0.010319	0.155920	0.092980	

In [13]:
```
sns.countplot(df["ACTION"])
```

```
C:\Users\ctoqu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a key word arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[13]: <AxesSubplot:xlabel='ACTION', ylabel='count'>



In [14]:
```
df.ACTION.nunique()
```

Out[14]: 2

In [15]:
```
from sklearn.model_selection import train_test_split
```

In [16]:
```
x=df.drop("ACTION",axis=1)
```

In [17]:
```
y=df["ACTION"]
```

In [18]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.3, random_state=101)
```

In [19]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

In [20]:
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
```

In [23]:
```
#Logistic regression

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]]
```

The LR model is giving accuracy of 93% but the confussion matrix shows it predicts everything as class 1 so there is a bias in the model

In [28]:
```
# Random Forest
model = RandomForestClassifier()
model.fit(x_train, y_train)
predictedmodelvalues=model.predict(x_test)
print(accuracy_score(y_test,predictedvalues))
print(confusion_matrix(y_test, predictedmodelvalues))
```

```
0.939985759332723
[[ 221  369]
 [ 126 9115]]
```

The improvement in accuracy is little with respect to LR model but the confusion metricts shows that it has also classified better the class

In [32]:
```
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]]
```

Here there is the same problem as in LR that can't work with a biased class

In [33]:
```
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]]
```

This model didn't give a good result compare with random forest as it is able to predict correctly only 7 data points os class 0.

Now will execute random forest on test data as we have choosen random forest as final model

In [34]:
```
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))
```

```
0.9486318787508901
[[ 216  374]
 [ 131 9110]]
```

In [36]:
```
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]
```

In [ ]: