Amazon Employee Access Challenge

In [ ]:
```
```

In [1]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
#importing needed modules/packages
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:
```
#importing the data
train=pd.read_csv('train.csv')
test=pd.read_csv('test.csv')
train.shape,test.shape
```

Out[3]:
```
((32769, 10), (58921, 10))
```

FEATURE ENGINEERING

SVD Encoding

Here we are trying to encode categorical data using SVD.

We will construct a matrix of co-occurences for each pair of categorical features. Each row corresponds to a value in feature A, while each column corresponds to a value in feature B. Each element is the count of rows where the value in A appears together with the value in B.

Then apply Truncated SVD on the matrix

In [4]:
```
#https://www.kaggle.com/dmitrylarko/kaggledays-sf-2-amazon-unsupervised-encoding#SVD-Encoding
#https://www.kaggle.com/matleonard/encoding-categorical-features-with-svd
```

In [5]:
```
train_data=train.drop(columns=['ACTION'],axis=1)
train_data.shape
```

Out[5]:
```
(32769, 9)
```

In [6]:
```
train_data.nunique()
```

Out[6]:
```
RESOURCE           7518
MGR_ID             4243
ROLE_ROLLUP_1       128
ROLE_ROLLUP_2       177
ROLE_DEPTNAME       449
ROLE_TITLE          343
ROLE_FAMILY_DESC   2358
ROLE_FAMILY          67
ROLE_CODE           343
dtype: int64
```

In [7]:
```
test_data=test.drop(columns=['id'],axis=1)
test_data.shape
```

Out[7]:
```
(58921, 9)
```

In [8]:
```
test_data.nunique()
```

Out[8]:
```
RESOURCE           4971
MGR_ID             4689
ROLE_ROLLUP_1       126
ROLE_ROLLUP_2       177
ROLE_DEPTNAME       466
ROLE_TITLE          351
ROLE_FAMILY_DESC   2749
ROLE_FAMILY          68
ROLE_CODE           351
dtype: int64
```

In [9]:
```
train_svd = pd.DataFrame()
test_svd = pd.DataFrame()
```

In [ ]:
```
```

In [10]:
```

#train_data.pivot_table(index='RESOURCE',columns='MGR_ID',aggfunc='count')
```

In [ ]:
```
```

In [11]:
```
from itertools import permutations
from sklearn.decomposition import TruncatedSVD
from sklearn.feature_extraction.text import TfidfVectorizer
from tqdm import tqdm

for col1,col2 in tqdm(permutations(train_data.columns,2)):
    res_train=(train_data.groupby([col1,col2])[col2].count())
    res_train=res_train.unstack(fill_value=0)

    svd=TruncatedSVD(n_components=1,random_state=42,).fit(res_train)
    val_train=svd.transform(res_train)


    val_train = pd.DataFrame(val_train)
    val_train = val_train.set_index(res_train.index)
    train_svd[col1+'_'+col2]=train[col1].map(val_train.iloc[:,0])



    test_svd[col1+'_'+col2]=test[col1].map(val_train.iloc[:,0])
```

```
72it [00:18,  3.94it/s]
```

In [12]:
```
train_svd.shape,test_svd.shape
```

Out[12]:
```
((32769, 72), (58921, 72))
```

In [13]:
```
train_svd.fillna(0,inplace=True)
test_svd.fillna(0,inplace=True)
print(train_svd.isna().sum().values)
print(test_svd.isna().sum().values)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

In [14]:
```
train_svd.head()
```

Out[14]:

	RESOURCE_MGR_ID	RESOURCE_ROLE_ROLLUP_1	RESOURCE_ROLE_ROLLUP_2	RESOURCE_ROLE_DEPTNAME	RESOURCE_ROLE_TITLE	RESOURCE_ROLE_FAMILY_DESC	RESOURCE_ROLE_FAMILY	RESOURCE_ROLE_(
0	0.088724	2.995769	1.810303	0.070125	1.593630	2.919560	2.934431	1.5
1	0.559935	25.998514	13.247680	1.084496	4.285689	4.534700	8.583779	4.2
2	0.000108	0.007828	0.022128	0.509533	0.049782	0.007275	0.058700	0.0
3	0.044904	0.998590	0.597128	0.018862	0.669129	0.164492	0.978144	0.6
4	0.059410	2.022416	0.320066	0.804351	0.689633	0.175200	2.033620	0.6

5 rows × 72 columns

In [ ]:
```

```

Normalizing the data

In [16]:
```
from sklearn.preprocessing import Normalizer
columns = (train_svd.columns)
x_vals1=train_svd[columns]
x_vals2=test_svd[columns]
n=Normalizer()
n.fit(x_vals1)
x_vals1 = n.transform(x_vals1)
train_svd = pd.DataFrame(x_vals1,columns=columns)
x_vals2 = n.transform(x_vals2)
test_svd = pd.DataFrame(x_vals2,columns=columns)
train_svd.shape,test_svd.shape
```

Out[16]: ((32769, 72), (58921, 72))

In [17]:
```
train_svd.head()
```

Out[17]:

	RESOURCE_MGR_ID	RESOURCE_ROLE_ROLLUP_1	RESOURCE_ROLE_ROLLUP_2	RESOURCE_ROLE_DEPTNAME	RESOURCE_ROLE_TITLE	RESOURCE_ROLE_FAMILY_DESC	RESOURCE_ROLE_FAMILY	RESOURCE_ROLE_(
0	3.338246e-06	0.000113	0.000068	2.638468e-06	0.000060	0.000110	0.000110	0.0
1	3.290961e-05	0.001528	0.000779	6.374011e-05	0.000252	0.000267	0.000505	0.0
2	1.122108e-07	0.000008	0.000023	5.280569e-04	0.000052	0.000008	0.000061	0.0
3	1.733916e-06	0.000039	0.000023	7.283213e-07	0.000026	0.000006	0.000038	0.0
4	4.072207e-04	0.013863	0.002194	5.513384e-03	0.004727	0.001201	0.013939	0.0

5 rows × 72 columns

In [18]:
```
test_svd.head()
```

Out[18]:

	RESOURCE_MGR_ID	RESOURCE_ROLE_ROLLUP_1	RESOURCE_ROLE_ROLLUP_2	RESOURCE_ROLE_DEPTNAME	RESOURCE_ROLE_TITLE	RESOURCE_ROLE_FAMILY_DESC	RESOURCE_ROLE_FAMILY	RESOURCE_ROLE_(
0	1.748205e-06	0.000014	0.000033	0.006349	0.000224	9.368598e-06	0.000464	0.0
1	4.757212e-07	0.000061	0.000016	0.000002	0.000040	5.865678e-05	0.000061	0.0
2	1.895173e-05	0.000584	0.000352	0.000030	0.000014	8.070228e-07	0.000055	0.0
3	3.237126e-06	0.000120	0.000032	0.000013	0.000080	1.984018e-05	0.000118	0.0
4	3.102218e-04	0.008945	0.004305	0.000712	0.001083	1.242129e-03	0.002395	0.0

5 rows × 72 columns

In [ ]:
```

```

One - hot encoding

A one hot encoding is a representation of categorical variables as binary vectors.

This first requires that the categorical values be mapped to integer values.

Then, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1.

In [19]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html
```

In [20]:
```
# OHE for RESOURCE variable
from sklearn.preprocessing import OneHotEncoder
ohe=OneHotEncoder(handle_unknown='ignore')
ohe.fit(train['RESOURCE'].values.reshape(-1,1))
resource_ohe_train = ohe.transform(train['RESOURCE'].values.reshape(-1,1))
resurce_ohe_test = ohe.transform(test['RESOURCE'].values.reshape(-1,1))
resource_ohe_train.shape,resurce_ohe_test.shape
```

Out[20]: ((32769, 7518), (58921, 7518))

In [21]:
```
# OHE for MGR_ID variable
from sklearn.preprocessing import OneHotEncoder
ohe=OneHotEncoder(handle_unknown='ignore')
ohe.fit(train['MGR_ID'].values.reshape(-1,1))
mgr_id_ohe_train = ohe.transform(train['MGR_ID'].values.reshape(-1,1))
mgr_id_ohe_test = ohe.transform(test['MGR_ID'].values.reshape(-1,1))
mgr_id_ohe_train.shape,mgr_id_ohe_test.shape
```

Out[21]: ((32769, 4243), (58921, 4243))

In [22]:
```
# OHE for ROLE_ROLLUP_1 variable
from sklearn.preprocessing import OneHotEncoder
ohe=OneHotEncoder(handle_unknown='ignore')
ohe.fit(train['ROLE_ROLLUP_1'].values.reshape(-1,1))
rollup1_ohe_train=ohe.transform(train['ROLE_ROLLUP_1'].values.reshape(-1,1))
rollup1_ohe_test=ohe.transform(test['ROLE_ROLLUP_1'].values.reshape(-1,1))
rollup1_ohe_train.shape,rollup1_ohe_test.shape
```

Out[22]: ((32769, 128), (58921, 128))

In [23]:
```
# OHE for ROLE_ROLLUP_2 variable
from sklearn.preprocessing import OneHotEncoder
ohe=OneHotEncoder(handle_unknown='ignore')
ohe.fit(train['ROLE_ROLLUP_2'].values.reshape(-1,1))
rollup2_ohe_train=ohe.transform(train['ROLE_ROLLUP_2'].values.reshape(-1,1))
rollup2_ohe_test=ohe.transform(test['ROLE_ROLLUP_2'].values.reshape(-1,1))
rollup2_ohe_train.shape,rollup2_ohe_test.shape
```

Out[23]: ((32769, 177), (58921, 177))

In [24]:
```
# OHE for ROLE_DEPTNAME variable
from sklearn.preprocessing import OneHotEncoder
ohe=OneHotEncoder(handle_unknown='ignore')
ohe.fit(train['ROLE_DEPTNAME'].values.reshape(-1,1))
deptname_ohe_train = ohe.transform(train['ROLE_DEPTNAME'].values.reshape(-1,1))
deptname_ohe_test = ohe.transform(test['ROLE_DEPTNAME'].values.reshape(-1,1))
deptname_ohe_train.shape,deptname_ohe_test.shape
```

Out[24]: ((32769, 449), (58921, 449))

In [25]:
```
# OHE for ROLE_TITLE variable
from sklearn.preprocessing import OneHotEncoder
ohe=OneHotEncoder(handle_unknown='ignore')
ohe.fit(train['ROLE_TITLE'].values.reshape(-1,1))
title_ohe_train = ohe.transform(train['ROLE_TITLE'].values.reshape(-1,1))
title_ohe_test = ohe.transform(test['ROLE_TITLE'].values.reshape(-1,1))
title_ohe_train.shape,title_ohe_test.shape
```

Out[25]: ((32769, 343), (58921, 343))

In [26]:
```
# OHE for ROLE_FAMILY_DESC variable
from sklearn.preprocessing import OneHotEncoder
ohe=OneHotEncoder(handle_unknown='ignore')
ohe.fit(train['ROLE_FAMILY_DESC'].values.reshape(-1,1))
family_desc_ohe_train = ohe.transform(train['ROLE_FAMILY_DESC'].values.reshape(-1,1))
family_desc_ohe_test = ohe.transform(test['ROLE_FAMILY_DESC'].values.reshape(-1,1))
family_desc_ohe_train.shape,family_desc_ohe_test.shape
```

Out[26]: ((32769, 2358), (58921, 2358))

In [27]:
```
# OHE for ROLE_FAMILY variable
from sklearn.preprocessing import OneHotEncoder
ohe=OneHotEncoder(handle_unknown='ignore')
ohe.fit(train['ROLE_FAMILY'].values.reshape(-1,1))
family_ohe_train = ohe.transform(train['ROLE_FAMILY'].values.reshape(-1,1))
family_ohe_test = ohe.transform(test['ROLE_FAMILY'].values.reshape(-1,1))
family_ohe_train.shape,family_ohe_test.shape
```

Out[27]: ((32769, 67), (58921, 67))

In [28]:
```
# OHE for ROLE_CODE variable
from sklearn.preprocessing import OneHotEncoder
ohe=OneHotEncoder(handle_unknown='ignore')
ohe.fit(train['ROLE_CODE'].values.reshape(-1,1))
code_ohe_train = ohe.transform(train['ROLE_CODE'].values.reshape(-1,1))
code_ohe_test = ohe.transform(test['ROLE_CODE'].values.reshape(-1,1))
code_ohe_train.shape,code_ohe_test.shape
```

Out[28]: ((32769, 343), (58921, 343))

In [29]:
```
from scipy.sparse import hstack
ohe_train = hstack((resource_ohe_train,mgr_id_ohe_train,rollup1_ohe_train,rollup2_ohe_train,deptname_ohe_train,
                    title_ohe_train,family_desc_ohe_train,family_ohe_train,code_ohe_train))
ohe_train_y = train['ACTION'].values
ohe_test = hstack((resource_ohe_test,mgr_id_ohe_test,rollup1_ohe_test,rollup2_ohe_test,deptname_ohe_test,
                    title_ohe_test,family_desc_ohe_test,family_ohe_test,code_ohe_test))

print(ohe_train.shape,ohe_test.shape,ohe_train_y.shape)
```

```
(32769, 15626) (58921, 15626) (32769,)
```

Frequency encoding

It is a way to utilize the frequency of the categories as labels.

In the cases where the frequency is related somewhat with the target variable, it helps the model to understand and assign the weight in direct and inverse proportion, depending on the nature of the data.

In [30]:
```
#https://www.datacamp.com/community/tutorials/encoding-methodologies
```

In [31]:
```
# https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.map.html
```

In [32]:
```
# Frequency coding for MGR_ID
values=train['RESOURCE'].value_counts() /len(train)
resource_fc_train = train['RESOURCE'].map(values)
resource_fc_test = test['RESOURCE'].map(values)
print(resource_fc_train.shape,resource_fc_test.shape,resource_fc_train.isna().sum(),resource_fc_test.isna().sum())
mgr_id_fc_test = resource_fc_test.fillna(0)
print(resource_fc_train.shape,resource_fc_test.shape,resource_fc_train.isna().sum(),resource_fc_test.isna().sum())
```

```
(32769,) (58921,) 0 0
```

In [33]:
```
# Frequency coding for MGR_ID
values=train['MGR_ID'].value_counts() /len(train)
mgr_id_fc_train = train['MGR_ID'].map(values)
mgr_id_fc_test = test['MGR_ID'].map(values)
print(mgr_id_fc_train.shape,mgr_id_fc_test.shape,mgr_id_fc_train.isna().sum(),mgr_id_fc_test.isna().sum())
mgr_id_fc_test = mgr_id_fc_test.fillna(0)
print(mgr_id_fc_train.shape,mgr_id_fc_test.shape,mgr_id_fc_train.isna().sum(),mgr_id_fc_test.isna().sum())
```

```
(32769,) (58921,) 0 1627
(32769,) (58921,) 0 0
```

In [34]:
```
# ROLE_ROLLUP_1 variable
values=train['ROLE_ROLLUP_1'].value_counts() /len(train)
rollup1_fc_train = train['ROLE_ROLLUP_1'].map(values)
rollup1_fc_test = test['ROLE_ROLLUP_1'].map(values)
print(rollup1_fc_train.shape,rollup1_fc_test.shape,rollup1_fc_train.isna().sum(),rollup1_fc_test.isna().sum())
rollup1_fc_test = rollup1_fc_test.fillna(0)
print(rollup1_fc_train.shape,rollup1_fc_test.shape,rollup1_fc_train.isna().sum(),rollup1_fc_test.isna().sum())
```

```
(32769,) (58921,) 0 4
(32769,) (58921,) 0 0
```

In [35]:
```
# ROLE_ROLLUP_2 variable
values=train['ROLE_ROLLUP_2'].value_counts() /len(train)
rollup2_fc_train = train['ROLE_ROLLUP_2'].map(values)
rollup2_fc_test = test['ROLE_ROLLUP_2'].map(values)
print(rollup2_fc_train.shape,rollup2_fc_test.shape,rollup2_fc_train.isna().sum(),rollup2_fc_test.isna().sum())
rollup2_fc_test = rollup2_fc_test.fillna(0)
print(rollup2_fc_train.shape,rollup2_fc_test.shape,rollup2_fc_train.isna().sum(),rollup2_fc_test.isna().sum())
```

```
(32769,) (58921,) 0 12
(32769,) (58921,) 0 0
```

In [36]:
```
# Frequency coding for ROLE_DEPTNAME variable
values=train['ROLE_DEPTNAME'].value_counts()/len(train)
deptname_fc_train = train['ROLE_DEPTNAME'].map(values)
deptname_fc_test = test['ROLE_DEPTNAME'].map(values)
print(deptname_fc_train.shape,deptname_fc_test.shape,deptname_fc_train.isna().sum(),deptname_fc_test.isna().sum())
deptname_fc_test = deptname_fc_test.fillna(0)
print(deptname_fc_train.shape,deptname_fc_test.shape,deptname_fc_train.isna().sum(),deptname_fc_test.isna().sum())
```

```
(32769,) (58921,) 0 62
(32769,) (58921,) 0 0
```

In [37]:
```
# Frequency coding for ROLE_TITLE
values=train['ROLE_TITLE'].value_counts()/len(train)
title_fc_train = train['ROLE_TITLE'].map(values)
title_fc_test = test['ROLE_TITLE'].map(values)
print(title_fc_train.shape,title_fc_test.shape,title_fc_train.isna().sum(),title_fc_test.isna().sum())
title_fc_test = title_fc_test.fillna(0)
print(title_fc_train.shape,title_fc_test.shape,title_fc_train.isna().sum(),title_fc_test.isna().sum())
```

```
(32769,) (58921,) 0 30
(32769,) (58921,) 0 0
```

In [38]:
```
# Freq coding for ROLE_FAMILY_DESC
values=train['ROLE_FAMILY_DESC'].value_counts()/len(train)
family_desc_fc_train = train['ROLE_FAMILY_DESC'].map(values)
family_desc_fc_test = test['ROLE_FAMILY_DESC'].map(values)
print(family_desc_fc_train.shape,family_desc_fc_test.shape,family_desc_fc_train.isna().sum(),family_desc_fc_test.isna().sum())
family_desc_fc_test = family_desc_fc_test.fillna(0)
print(family_desc_fc_train.shape,family_desc_fc_test.shape,family_desc_fc_train.isna().sum(),family_desc_fc_test.isna().sum())
```

```
(32769,) (58921,) 0 1249
(32769,) (58921,) 0 0
```

In [39]:
```
# Freq coding for ROLE_FAMILY
values=train['ROLE_FAMILY'].value_counts()/len(train)
family_fc_train = train['ROLE_FAMILY'].map(values)
family_fc_test = test['ROLE_FAMILY'].map(values)
print(family_fc_train.shape,family_fc_test.shape,family_fc_train.isna().sum(),family_fc_test.isna().sum())
family_fc_test = family_fc_test.fillna(0)
print(family_fc_train.shape,family_fc_test.shape,family_fc_train.isna().sum(),family_fc_test.isna().sum())
```

```
(32769,) (58921,) 0 1
(32769,) (58921,) 0 0
```

In [40]:
```
# Freq coding for ROLE_CODE
values=train['ROLE_CODE'].value_counts()/len(train)
code_fc_train = train['ROLE_CODE'].map(values)
code_fc_test = test['ROLE_CODE'].map(values)
print(code_fc_train.shape,code_fc_test.shape,code_fc_train.isna().sum(),code_fc_test.isna().sum())
code_fc_test = code_fc_test.fillna(0)
print(code_fc_train.shape,code_fc_test.shape,code_fc_train.isna().sum(),code_fc_test.isna().sum())
```

```
(32769,) (58921,) 0 30
(32769,) (58921,) 0 0
```

In [41]:
```
fc_df_train = pd.DataFrame ({'resource_fc_train':resource_fc_train,'mgr_id_fc_train':mgr_id_fc_train,'rollup1_fc_train':rollup1_fc_train,'rollup2_fc_train':rollup2_fc_train,
                         'deptname_fc_train':deptname_fc_train,'title_fc_train':title_fc_train,
                         'family_desc_fc_train':family_desc_fc_train,
                         'family_fc_train':family_fc_train,'code_fc_train':code_fc_train})

fc_df_test = pd.DataFrame ({'resource_fc_test':resource_fc_test,'mgr_id_fc_test':mgr_id_fc_test,'rollup1_fc_test':rollup1_fc_test,'rollup2_fc_test':rollup2_fc_test,
                         'deptname_fc_test':deptname_fc_test,'title_fc_test':title_fc_test,
                         'family_desc_fc_test':family_desc_fc_test,
                         'family_fc_test':family_fc_test,'code_fc_test':code_fc_test})

fc_y_train = train['ACTION'].values
```

In [42]:
```
fc_df_train.shape,fc_y_train.shape,fc_df_test.shape
```

Out[42]:
```
((32769, 9), (32769,), (58921, 9))
```

Response Coding / Target Encoding

It is a technique to represent the categorical data while solving a machine learning classification problem.

As part of this technique, we represent the probability of the data point belonging to a particular class given a category.

In [43]:
```
#RESOURCE
values= train.groupby('RESOURCE')['ACTION'].mean()
rc_resource_train = train['RESOURCE'].map(values)
rc_resource_test = test['RESOURCE'].map(values)
print(rc_resource_train.shape,rc_resource_test.shape,rc_resource_train.isna().sum(),rc_resource_test.isna().sum())
rc_resource_test = rc_resource_test.fillna(0.5)
print(rc_resource_train.shape,rc_resource_test.shape,rc_resource_train.isna().sum(),rc_resource_test.isna().sum())
```

```
(32769,) (58921,) 0 0
(32769,) (58921,) 0 0
```

In [44]:
```
#MGR_ID
values= train.groupby('MGR_ID')['ACTION'].mean()
rc_mgrid_train = train['MGR_ID'].map(values)
rc_mgrid_test = test['MGR_ID'].map(values)
print(rc_mgrid_train.shape,rc_mgrid_test.shape,rc_mgrid_train.isna().sum(),rc_mgrid_test.isna().sum())
rc_mgrid_test = rc_mgrid_test.fillna(0.5)
print(rc_mgrid_train.shape,rc_mgrid_test.shape,rc_mgrid_train.isna().sum(),rc_mgrid_test.isna().sum())
```

```
(32769,) (58921,) 0 1627
(32769,) (58921,) 0 0
```

In [45]:
```
# ROLE_ROLLUP_1
values= train.groupby('ROLE_ROLLUP_1')['ACTION'].mean()
rc_rollup1_train = train['ROLE_ROLLUP_1'].map(values)
rc_rollup1_test = test['ROLE_ROLLUP_1'].map(values)
print(rc_rollup1_train.shape,rc_rollup1_test.shape,rc_rollup1_train.isna().sum(),rc_rollup1_test.isna().sum())
rc_rollup1_test = rc_rollup1_test.fillna(0.5)
print(rc_rollup1_train.shape,rc_rollup1_test.shape,rc_rollup1_train.isna().sum(),rc_rollup1_test.isna().sum())
```

```
(32769,) (58921,) 0 4
(32769,) (58921,) 0 0
```

In [46]:
```
# ROLE_ROLLUP_2
values= train.groupby('ROLE_ROLLUP_2')['ACTION'].mean()
rc_rollup2_train = train['ROLE_ROLLUP_2'].map(values)
rc_rollup2_test = test['ROLE_ROLLUP_2'].map(values)
print(rc_rollup2_train.shape,rc_rollup2_test.shape,rc_rollup2_train.isna().sum(),rc_rollup2_test.isna().sum())
rc_rollup2_test = rc_rollup2_test.fillna(0.5)
print(rc_rollup2_train.shape,rc_rollup2_test.shape,rc_rollup2_train.isna().sum(),rc_rollup2_test.isna().sum())
```

```
(32769,) (58921,) 0 12
(32769,) (58921,) 0 0
```

In [47]:
```
# ROLE_DEPTNAME
values= train.groupby('ROLE_DEPTNAME')['ACTION'].mean()
rc_deptname_train = train['ROLE_DEPTNAME'].map(values)
rc_deptname_test = test['ROLE_DEPTNAME'].map(values)
print(rc_deptname_train.shape,rc_deptname_test.shape,rc_deptname_train.isna().sum(),rc_deptname_test.isna().sum())
rc_deptname_test = rc_deptname_test.fillna(0.5)
print(rc_deptname_train.shape,rc_deptname_test.shape,rc_deptname_train.isna().sum(),rc_deptname_test.isna().sum())
```

```
(32769,) (58921,) 0 62
(32769,) (58921,) 0 0
```

In [48]:
```
# ROLE_TITLE
values= train.groupby('ROLE_TITLE')['ACTION'].mean()
rc_title_train = train['ROLE_TITLE'].map(values)
rc_title_test = test['ROLE_TITLE'].map(values)
print(rc_title_train.shape,rc_title_test.shape,rc_title_train.isna().sum(),rc_title_test.isna().sum())
rc_title_test = rc_title_test.fillna(0.5)
print(rc_title_train.shape,rc_title_test.shape,rc_title_train.isna().sum(),rc_title_test.isna().sum())
```

```
(32769,) (58921,) 0 30
(32769,) (58921,) 0 0
```

In [49]:
```
# ROLE_FAMILY_DESC
values= train.groupby('ROLE_FAMILY_DESC')['ACTION'].mean()
rc_family_desc_train = train['ROLE_FAMILY_DESC'].map(values)
rc_family_desc_test = test['ROLE_FAMILY_DESC'].map(values)
print(rc_family_desc_train.shape,rc_family_desc_test.shape,rc_family_desc_train.isna().sum(),rc_family_desc_test.isna().sum())
rc_family_desc_test = rc_family_desc_test.fillna(0.5)
print(rc_family_desc_train.shape,rc_family_desc_test.shape,rc_family_desc_train.isna().sum(),rc_family_desc_test.isna().sum())
```

```
(32769,) (58921,) 0 1249
(32769,) (58921,) 0 0
```

In [50]:
```
# ROLE_FAMILY
values= train.groupby('ROLE_FAMILY')['ACTION'].mean()
rc_family_train = train['ROLE_FAMILY'].map(values)
rc_family_test = test['ROLE_FAMILY'].map(values)
print(rc_family_train.shape,rc_family_test.shape,rc_family_train.isna().sum(),rc_family_test.isna().sum())
rc_family_test = rc_family_test.fillna(0.5)
print(rc_family_train.shape,rc_family_test.shape,rc_family_train.isna().sum(),rc_family_test.isna().sum())
```

```
(32769,) (58921,) 0 1
(32769,) (58921,) 0 0
```

In [51]:
```
# ROLE_CODE
values= train.groupby('ROLE_CODE')['ACTION'].mean()
rc_code_train = train['ROLE_CODE'].map(values)
rc_code_test = test['ROLE_CODE'].map(values)
print(rc_code_train.shape,rc_code_test.shape,rc_code_train.isna().sum(),rc_code_test.isna().sum())
rc_code_test = rc_code_test.fillna(0.5)
print(rc_code_train.shape,rc_code_test.shape,rc_code_train.isna().sum(),rc_code_test.isna().sum())
```

```
(32769,) (58921,) 0 30
(32769,) (58921,) 0 0
```

In [52]:
```
rc_df_train = pd.DataFrame ({'rc_resource_train':rc_resource_train,'mgr_id_rc_train':rc_mgrid_train,'rollup1_rc_train':rc_rollup1_train,'rollup2_rc_train':rc_rollup1_train,
                         'deptname_rc_train':rc_deptname_train,'title_rc_train':rc_title_train,
                         'family_desc_rc_train':rc_family_desc_train,
                         'family_rc_train':rc_family_train,'code_rc_train':rc_code_train})
rc_df_test = pd.DataFrame ({'rc_resource_test':rc_resource_test,'mgr_id_rc_test':rc_mgrid_test,'rollup1_rc_test':rc_rollup1_test,'rollup2_rc_test':rc_rollup1_test,
                         'deptname_rc_test':rc_deptname_test,'title_rc_test':rc_title_test,
                         'family_desc_rc_test':rc_family_desc_test,
                         'family_rc_test':rc_family_test,'code_rc_test':rc_code_test})

rc_y_train = train['ACTION'].values
```

In [53]:
```
rc_df_train.shape,rc_y_train.shape,rc_df_test.shape
```

Out[53]:
```
((32769, 9), (32769,), (58921, 9))
```

After performing **One Hot Encoding** on the given data,we could see that the transformed data has 15626 features / its a 15626 dimensional data. As the dimennsions are very large, it might take very long time to train the model and it may also lead to the problem of curse of dimensionality. So to solve this , we need to select the important features that might be useful so that we can reduce the dimensionality of data.

In [54]:
```
# https://chrisalbon.com/machine_learning/model_selection/hyperparameter_tuning_using_random_search/
```

Feature Selection

In [55]:
```
from sklearn.feature_selection import SelectKBest,chi2
ktop = SelectKBest(chi2,k=4500).fit(ohe_train,ohe_train_y)
ohe_train=ktop.transform(ohe_train)
ohe_test=ktop.transform(ohe_test)
```

In [56]:
```
ohe_train.shape,ohe_test.shape
```

Out[56]:
```
((32769, 4500), (58921, 4500))
```

Saving all the data into a CSV file

In [57]:
```
# saving all to dataframe
rc_df_train.to_csv('data/rc_df_train.csv',index=False)
rc_df_test.to_csv('data/rc_df_test.csv',index=False)
fc_df_train.to_csv('data/fc_df_train.csv',index=False)
fc_df_test.to_csv('data/fc_df_test.csv',index=False)
train_svd.to_csv('data/train_svd.csv',index=False)
test_svd.to_csv('data/test_svd.csv',index=False)
```

In [58]:
```
#https://stackoverflow.com/questions/8955448/save-load-scipy-sparse-csr-matrix-in-portable-data-format
```

In [59]:
```
from scipy import sparse
sparse.save_npz('data/ohe_train.npz',ohe_train)
sparse.save_npz('data/ohe_test.npz',ohe_test)
```