In [1]:

```python
# import libraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import OneHotEncoder
from scipy.sparse import hstack
import category_encoders as ce
from scipy import sparse

from itertools import permutations
from sklearn.decomposition import TruncatedSVD
from sklearn.feature_extraction.text import TfidfVectorizer
from tqdm import tqdm

# import warnings
# warnings.filterwarnings('ignore')
```

Amazon Employee Access Challenge

In [2]:

```python
train = pd.read_csv('data/train.csv')
test = pd.read_csv('data/test.csv')
```

In [3]:

```python
train.shape
```

Out[3]: (32769, 10)

In [4]:

```python
test.shape
```

Out[4]: (58921, 10)

One Hot Encoding

In [5]:
```
# One hot encoding of RESOURCE Feature
ohe = OneHotEncoder(handle_unknown='ignore')

ohe.fit(train['RESOURCE'].values.reshape(-1, 1))# Fit has to happen onl
y on train data

train_resource_ohe = ohe.transform(train['RESOURCE'].values.reshape(-1,
 1))
test_resource_ohe = ohe.transform(test['RESOURCE'].values.reshape(-1, 1
))

print(train_resource_ohe.shape, test_resource_ohe.shape)
```

```
(32769, 7518) (58921, 7518)
```

In [6]:
```
# One hot encoding of MGR_ID Feature
ohe = OneHotEncoder(handle_unknown='ignore')

ohe.fit(train['MGR_ID'].values.reshape(-1, 1))# Fit has to happen only
 on train data

train_mgr_id_ohe = ohe.transform(train['MGR_ID'].values.reshape(-1, 1))
test_mgr_id_ohe = ohe.transform(test['MGR_ID'].values.reshape(-1, 1))

print(train_mgr_id_ohe.shape, test_mgr_id_ohe.shape)
```

```
(32769, 4243) (58921, 4243)
```

In [7]:
```
# One hot encoding of ROLE_ROLLUP_1 Feature
ohe = OneHotEncoder(handle_unknown='ignore')

ohe.fit(train['ROLE_ROLLUP_1'].values.reshape(-1, 1))# Fit has to happe
n only on train data

train_role_rollup_1_ohe = ohe.transform(train['ROLE_ROLLUP_1'].values.r
eshape(-1, 1))
test_role_rollup_1_ohe = ohe.transform(test['ROLE_ROLLUP_1'].values.res
hape(-1, 1))
```

```
print(train_role_rollup_1_ohe.shape, test_role_rollup_1_ohe.shape)
```

```
(32769, 128) (58921, 128)
```

In [8]:
```
# One hot encoding of ROLE_ROLLUP_2 Feature
ohe = OneHotEncoder(handle_unknown='ignore')

ohe.fit(train['ROLE_ROLLUP_2'].values.reshape(-1, 1))# Fit has to happe
n only on train data

train_role_rollup_2_ohe = ohe.transform(train['ROLE_ROLLUP_2'].values.r
eshape(-1, 1))
test_role_rollup_2_ohe = ohe.transform(test['ROLE_ROLLUP_2'].values.res
hape(-1, 1))

print(train_role_rollup_2_ohe.shape, test_role_rollup_2_ohe.shape)
```

```
(32769, 177) (58921, 177)
```

In [9]:
```
# One hot encoding of ROLE_DEPTNAME Feature
ohe = OneHotEncoder(handle_unknown='ignore')

ohe.fit(train['ROLE_DEPTNAME'].values.reshape(-1, 1))# Fit has to happe
n only on train data

train_role_deptname_ohe = ohe.transform(train['ROLE_DEPTNAME'].values.r
eshape(-1, 1))
test_role_deptname_ohe = ohe.transform(test['ROLE_DEPTNAME'].values.res
hape(-1, 1))

print(train_role_deptname_ohe.shape, test_role_deptname_ohe.shape)
```

```
(32769, 449) (58921, 449)
```

In [10]:
```
# One hot encoding of ROLE_TITLE Feature
ohe = OneHotEncoder(handle_unknown='ignore')

ohe.fit(train['ROLE_TITLE'].values.reshape(-1, 1))# Fit has to happen o
```

```
nly on train data

train_role_title_ohe = ohe.transform(train['ROLE_TITLE'].values.reshape
(-1, 1))
test_role_title_ohe = ohe.transform(test['ROLE_TITLE'].values.reshape(-
1, 1))

print(train_role_title_ohe.shape, test_role_title_ohe.shape)
```

(32769, 343) (58921, 343)

In [11]:

```
# One hot encoding of ROLE_FAMILY_DESC Feature
ohe = OneHotEncoder(handle_unknown='ignore')

ohe.fit(train['ROLE_FAMILY_DESC'].values.reshape(-1, 1))# Fit has to ha
ppen only on train data

train_role_family_desc_ohe = ohe.transform(train['ROLE_FAMILY_DESC'].va
lues.reshape(-1, 1))
test_role_family_desc_ohe = ohe.transform(test['ROLE_FAMILY_DESC'].valu
es.reshape(-1, 1))

print(train_role_family_desc_ohe.shape, test_role_family_desc_ohe.shape
)
```

(32769, 2358) (58921, 2358)

In [12]:

```
# One hot encoding of ROLE_FAMILY Feature
ohe = OneHotEncoder(handle_unknown='ignore')

ohe.fit(train['ROLE_FAMILY'].values.reshape(-1, 1))# Fit has to happen
 only on train data

train_role_family_ohe = ohe.transform(train['ROLE_FAMILY'].values.resha
pe(-1, 1))
test_role_family_ohe = ohe.transform(test['ROLE_FAMILY'].values.reshape
(-1, 1))

print(train_role_family_ohe.shape, test_role_family_ohe.shape)
```

```
(32769, 67) (58921, 67)
```

In [13]:
```
# One hot encoding of ROLE_CODE Feature
ohe = OneHotEncoder(handle_unknown='ignore')

ohe.fit(train['ROLE_CODE'].values.reshape(-1, 1))# Fit has to happen on
ly on train data

train_role_code_ohe = ohe.transform(train['ROLE_CODE'].values.reshape(-
1, 1))
test_role_code_ohe = ohe.transform(test['ROLE_CODE'].values.reshape(-1,
 1))

print(train_role_code_ohe.shape, test_role_code_ohe.shape)
```

```
(32769, 343) (58921, 343)
```

In [14]:
```
train_ohe = hstack((train_resource_ohe, train_mgr_id_ohe,  train_role_r
ollup_1_ohe, train_role_rollup_2_ohe, train_role_deptname_ohe, train_ro
le_title_ohe, train_role_family_desc_ohe, train_role_family_ohe, train_
role_code_ohe))
```

In [15]:
```
test_ohe = hstack((test_resource_ohe, test_mgr_id_ohe, test_role_rollup
_1_ohe, test_role_rollup_2_ohe, test_role_deptname_ohe, test_role_title
_ohe, test_role_family_desc_ohe, test_role_family_ohe, test_role_code_o
he))
```

In [16]:
```
y_train_ohe = train['ACTION']
```

In [17]:
```
train_ohe.shape, test_ohe.shape, y_train_ohe.shape
```

Out[17]:
```
((32769, 15626), (58921, 15626), (32769,))
```

Frequency Encoding

https://python-data-science.readthedocs.io/en/latest/preprocess.html

In [18]:
```
### FREQUENCY ENCODING

# size of each category
# encoding = titanic.groupby('Embarked').size()
# get frequency of each category
# encoding = encoding/len(titanic)
# titanic['enc'] = titanic.Embarked.map(encoding)
```

In [19]:
```
### FREQUENCY ENCODING RESOURCE

# size of each category
encoding = train.groupby('RESOURCE').size()

# get frequency of each category
encoding = encoding/len(train)
train_resource_fc = train.RESOURCE.map(encoding)
test_resource_fc = test.RESOURCE.map(encoding)

print(train_resource_fc.shape, test_resource_fc.shape, train_resource_f
c.isna().sum(), test_resource_fc.isna().sum())
# fill missing values
test_resource_fc = test_resource_fc.fillna(0)
print(train_resource_fc.shape, test_resource_fc.shape, train_resource_f
c.isna().sum(), test_resource_fc.isna().sum())
```

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

In [20]:
```
### FREQUENCY ENCODING MGR_ID

# size of each category
encoding = train.groupby('MGR_ID').size()

# get frequency of each category
encoding = encoding/len(train)
train_mgr_id_fc = train.MGR_ID.map(encoding)
```

```
test_mgr_id_fc = test.MGR_ID.map(encoding)

print(train_mgr_id_fc.shape, test_mgr_id_fc.shape, train_mgr_id_fc.isna
().sum(), test_mgr_id_fc.isna().sum())
# fill missing values
test_mgr_id_fc = test_mgr_id_fc.fillna(0)
print(train_mgr_id_fc.shape, test_mgr_id_fc.shape, train_mgr_id_fc.isna
().sum(), test_mgr_id_fc.isna().sum())
```

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

In [21]:

```
### FREQUENCY ENCODING ROLE_ROLLUP_1

# size of each category
encoding = train.groupby('ROLE_ROLLUP_1').size()

# get frequency of each category
encoding = encoding/len(train)
train_rollup_1_fc = train.ROLE_ROLLUP_1.map(encoding)
test_rollup_1_fc = test.ROLE_ROLLUP_1.map(encoding)

print(train_rollup_1_fc.shape, test_rollup_1_fc.shape, train_rollup_1_f
c.isna().sum(), test_rollup_1_fc.isna().sum())
# fill missing values
test_rollup_1_fc = test_rollup_1_fc.fillna(0)
print(train_rollup_1_fc.shape, test_rollup_1_fc.shape, train_rollup_1_f
c.isna().sum(), test_rollup_1_fc.isna().sum())
```

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

In [22]:

```
### FREQUENCY ENCODING ROLE_ROLLUP_2

# size of each category
encoding = train.groupby('ROLE_ROLLUP_2').size()

# get frequency of each category
encoding = encoding/len(train)
```

```
train_rollup_2_fc = train.ROLE_ROLLUP_2.map(encoding)
test_rollup_2_fc = test.ROLE_ROLLUP_2.map(encoding)

print(train_rollup_2_fc.shape, test_rollup_2_fc.shape, train_rollup_2_f
c.isna().sum(), test_rollup_2_fc.isna().sum())
# fill missing values
test_rollup_2_fc = test_rollup_2_fc.fillna(0)
print(train_rollup_2_fc.shape, test_rollup_2_fc.shape, train_rollup_2_f
c.isna().sum(), test_rollup_2_fc.isna().sum())
```

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

In [23]:

```
### FREQUENCY ENCODING ROLE_DEPTNAME

# size of each category
encoding = train.groupby('ROLE_DEPTNAME').size()

# get frequency of each category
encoding = encoding/len(train)
train_role_deptname_fc = train.ROLE_DEPTNAME.map(encoding)
test_role_deptname_fc = test.ROLE_DEPTNAME.map(encoding)

print(train_role_deptname_fc.shape, test_role_deptname_fc.shape, train_
role_deptname_fc.isna().sum(), test_role_deptname_fc.isna().sum())
# fill missing values
test_role_deptname_fc = test_role_deptname_fc.fillna(0)
print(train_role_deptname_fc.shape, test_role_deptname_fc.shape, train_
role_deptname_fc.isna().sum(), test_role_deptname_fc.isna().sum())
```

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

In [24]:

```
### FREQUENCY ENCODING ROLE_TITLE

# size of each category
encoding = train.groupby('ROLE_TITLE').size()

# get frequency of each category
```

```
encoding = encoding/len(train)
train_role_title_fc = train.ROLE_TITLE.map(encoding)
test_role_title_fc = test.ROLE_TITLE.map(encoding)

print(train_role_title_fc.shape, test_role_title_fc.shape, train_role_title_fc.isna().sum(), test_role_title_fc.isna().sum())
# fill missing values
test_role_title_fc = test_role_title_fc.fillna(0)
print(train_role_title_fc.shape, test_role_title_fc.shape, train_role_title_fc.isna().sum(), test_role_title_fc.isna().sum())
```

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

In [25]:

```
### FREQUENCY ENCODING ROLE_FAMILY_DESC

# size of each category
encoding = train.groupby('ROLE_FAMILY_DESC').size()

# get frequency of each category
encoding = encoding/len(train)
train_role_family_desc_fc = train.ROLE_FAMILY_DESC.map(encoding)
test_role_family_desc_fc = test.ROLE_FAMILY_DESC.map(encoding)

print(train_role_family_desc_fc.shape, test_role_family_desc_fc.shape, train_role_family_desc_fc.isna().sum(), test_role_family_desc_fc.isna().sum())
# fill missing values
test_role_family_desc_fc = test_role_family_desc_fc.fillna(0)
print(train_role_family_desc_fc.shape, test_role_family_desc_fc.shape, train_role_family_desc_fc.isna().sum(), test_role_family_desc_fc.isna().sum())
```

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

In [26]:

```
### FREQUENCY ENCODING ROLE_FAMILY

# size of each category
```

```
encoding = train.groupby('ROLE_FAMILY').size()

# get frequency of each category
encoding = encoding/len(train)
train_role_family_fc = train.ROLE_FAMILY.map(encoding)
test_role_family_fc = test.ROLE_FAMILY.map(encoding)

print(train_role_family_fc.shape, test_role_family_fc.shape, train_role
_family_fc.isna().sum(), test_role_family_fc.isna().sum())
# fill missing values
test_role_family_fc = test_role_family_fc.fillna(0)
print(train_role_family_fc.shape, test_role_family_fc.shape, train_role
_family_fc.isna().sum(), test_role_family_fc.isna().sum())
```

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

In [27]:

```
### FREQUENCY ENCODING ROLE_CODE

# size of each category
encoding = train.groupby('ROLE_CODE').size()

# get frequency of each category
encoding = encoding/len(train)
train_role_code_fc = train.ROLE_CODE.map(encoding)
test_role_code_fc = test.ROLE_CODE.map(encoding)

print(train_role_code_fc.shape, test_role_code_fc.shape, train_role_cod
e_fc.isna().sum(), test_role_code_fc.isna().sum())
# fill missing values
test_role_code_fc = test_role_code_fc.fillna(0)
print(train_role_code_fc.shape, test_role_code_fc.shape, train_role_cod
e_fc.isna().sum(), test_role_code_fc.isna().sum())
```

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

In [28]:

```
type(test_role_code_fc[0:10])
```

Out[28]: pandas.core.series.Series

In [29]:
```
train_df_fc = pd.DataFrame({'resource_fc':train_resource_fc, 'mgr_id_fc':train_mgr_id_fc,'rollup_1_fc':train_rollup_1_fc, 'rollup_2_fc':train_rollup_2_fc, 'role_deptname_fc':train_role_deptname_fc, 'role_title_fc':train_role_title_fc, 'role_family_desc_fc':train_role_family_desc_fc, 'role_family_fc':train_role_family_fc, 'role_code_fc':train_role_code_fc})
```

In [30]:
```
test_df_fc = pd.DataFrame({'resource_fc':test_resource_fc, 'mgr_id_fc':test_mgr_id_fc, 'rollup_1_fc':test_rollup_1_fc, 'rollup_2_fc':test_rollup_2_fc, 'role_deptname_fc':test_role_deptname_fc, 'role_title_fc':test_role_title_fc, 'role_family_desc_fc':test_role_family_desc_fc, 'role_family_fc':test_role_family_fc, 'role_code_fc':test_role_code_fc})
```

In [31]:
```
train_df_fc.shape
```

Out[31]: (32769, 9)

In [32]:
```
test_df_fc.shape
```

Out[32]: (58921, 9)

In [33]:
```
train_y_fc = train['ACTION'].values
```

In [34]:
```
train_y_fc.shape
```

Out[34]: (32769,)

Response Encoding

https://medium.com/analytics-vidhya/types-of-categorical-data-encoding-schemes-a5bbeb4ba02b

In [35]:
```
# sample
data = pd.DataFrame({
    'color' : ['Blue', 'Black', 'Black','Blue', 'Blue'],
    'outcome' : [1,      2,        1,     1,      2,]
})
# column to perform encoding
X = data['color']
Y = data['outcome']
# create an object of the TargetEncoder
ce_TE = ce.TargetEncoder(cols=['color'])
# fit and transform and you will get the encoded data
ce_TE.fit(X,Y)
ce_TE.transform(X)
```

Out[35]:

	color
0	1.341280
1	1.473106
2	1.473106
3	1.341280
4	1.341280

In [36]:
```
### RESPONSE ENCODING RESOURCE

# column to perform encoding
X = train['RESOURCE']
Y = train['ACTION']
# create an object of the TargetEncoder
ce_TE = ce.TargetEncoder(cols=['RESOURCE'])
# fit and transform and you will get the encoded data
ce_TE.fit(X,Y)
train_resource_rc = ce_TE.transform(X)
test_resource_rc = ce_TE.transform(test['RESOURCE'])

print(train_resource_rc.shape, test_resource_rc.shape)
```

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

In [37]:
```
train_resource_rc[:10]
```

Out[37]:

	RESOURCE
0	0.993099
1	0.966667
2	0.984431
3	0.942110
4	0.999947
5	0.802556
6	0.953545
7	1.000000
8	0.997255
9	1.000000

In [38]:
```
### RESPONSE ENCODING MGR_ID

# column to perform encoding
X = train['MGR_ID']
Y = train['ACTION']
# create an object of the TargetEncoder
ce_TE = ce.TargetEncoder(cols=['MGR_ID'])
# fit and transform and you will get the encoded data
ce_TE.fit(X,Y)
train_mgr_id_rc = ce_TE.transform(X)
test_mgr_id_rc = ce_TE.transform(test['MGR_ID'])

print(train_mgr_id_rc.shape, test_mgr_id_rc.shape)
```

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

In [39]:
```
### RESPONSE ENCODING ROLE_ROLLUP_1
```

```
# column to perform encoding
X = train['ROLE_ROLLUP_1']
Y = train['ACTION']
# create an object of the TargetEncoder
ce_TE = ce.TargetEncoder(cols=['ROLE_ROLLUP_1'])
# fit and transform and you will get the encoded data
ce_TE.fit(X,Y)
train_rollup_1_rc = ce_TE.transform(X)
test_rollup_1_rc = ce_TE.transform(test['ROLE_ROLLUP_1'])

print(train_rollup_1_rc.shape, test_rollup_1_rc.shape)
```

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

In [40]:

```
### RESPONSE ENCODING ROLE_ROLLUP_2

# column to perform encoding
X = train['ROLE_ROLLUP_2']
Y = train['ACTION']
# create an object of the TargetEncoder
ce_TE = ce.TargetEncoder(cols=['ROLE_ROLLUP_2'])
# fit and transform and you will get the encoded data
ce_TE.fit(X,Y)
train_rollup_2_rc = ce_TE.transform(X)
test_rollup_2_rc = ce_TE.transform(test['ROLE_ROLLUP_2'])

print(train_rollup_2_rc.shape, test_rollup_2_rc.shape)
```

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

In [41]:

```
### RESPONSE ENCODING ROLE_DEPTNAME

# column to perform encoding
X = train['ROLE_DEPTNAME']
Y = train['ACTION']
# create an object of the TargetEncoder
ce_TE = ce.TargetEncoder(cols=['ROLE_DEPTNAME'])
# fit and transform and you will get the encoded data
```

```
ce_TE.fit(X,Y)
train_role_deptname_rc = ce_TE.transform(X)
test_role_deptname_rc = ce_TE.transform(test['ROLE_DEPTNAME'])

print(train_role_deptname_rc.shape, test_role_deptname_rc.shape)
```

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

In [42]:

```
### RESPONSE ENCODING ROLE_TITLE

# column to perform encoding
X = train['ROLE_TITLE']
Y = train['ACTION']
# create an object of the TargetEncoder
ce_TE = ce.TargetEncoder(cols=['ROLE_TITLE'])
# fit and transform and you will get the encoded data
ce_TE.fit(X,Y)
train_role_title_rc = ce_TE.transform(X)
test_role_title_rc = ce_TE.transform(test['ROLE_TITLE'])

print(train_role_title_rc.shape, test_role_title_rc.shape)
```

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

In [43]:

```
### RESPONSE ENCODING ROLE_FAMILY_DESC

# column to perform encoding
X = train['ROLE_FAMILY_DESC']
Y = train['ACTION']
# create an object of the TargetEncoder
ce_TE = ce.TargetEncoder(cols=['ROLE_FAMILY_DESC'])
# fit and transform and you will get the encoded data
ce_TE.fit(X,Y)
train_role_family_desc_rc = ce_TE.transform(X)
test_role_family_desc_rc = ce_TE.transform(test['ROLE_FAMILY_DESC'])

print(train_role_family_desc_rc.shape, test_role_family_desc_rc.shape)
```

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

In [44]:

```
### RESPONSE ENCODING ROLE_FAMILY

# column to perform encoding
X = train['ROLE_FAMILY']
Y = train['ACTION']
# create an object of the TargetEncoder
ce_TE = ce.TargetEncoder(cols=['ROLE_FAMILY'])
# fit and transform and you will get the encoded data
ce_TE.fit(X,Y)
train_role_family_rc = ce_TE.transform(X)
test_role_family_rc = ce_TE.transform(test['ROLE_FAMILY'])

print(train_role_family_rc.shape, test_role_family_rc.shape)
```

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

In [45]:

```
### RESPONSE ENCODING ROLE_CODE

# column to perform encoding
X = train['ROLE_CODE']
Y = train['ACTION']
# create an object of the TargetEncoder
ce_TE = ce.TargetEncoder(cols=['ROLE_CODE'])
# fit and transform and you will get the encoded data
ce_TE.fit(X,Y)
train_role_code_rc = ce_TE.transform(X)
test_role_code_rc = ce_TE.transform(test['ROLE_CODE'])

print(train_role_code_rc.shape, test_role_code_rc.shape)
```

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

In [46]:

```
train_df_rc = pd.DataFrame ({'resource_rc':train_resource_rc['RESOURCE'
],'mgr_id_rc':train_mgr_id_rc['MGR_ID'], 'rollup_1_rc':train_rollup_1_r
c['ROLE_ROLLUP_1'],  'rollup_2_rc':train_rollup_2_rc['ROLE_ROLLUP_2'],
'role_deptname_rc':train_role_deptname_rc['ROLE_DEPTNAME'], 'role_title
_rc':train_role_title_rc['ROLE_TITLE'], 'role_family_desc_rc':train_rol
```

```
e_family_desc_rc['ROLE_FAMILY_DESC'], 'role_family_rc':train_role_famil
y_rc['ROLE_FAMILY'], 'role_code_rc':train_role_code_rc['ROLE_CODE']})
```

In [47]:
```
test_df_rc = pd.DataFrame ({'resource_rc':test_resource_rc['RESOURCE'],
'mgr_id_rc':test_mgr_id_rc['MGR_ID'], 'rollup_1_rc':test_rollup_1_rc['R
OLE_ROLLUP_1'],  'rollup_2_rc':test_rollup_2_rc['ROLE_ROLLUP_2'], 'role
_deptname_rc':test_role_deptname_rc['ROLE_DEPTNAME'], 'role_title_rc':t
est_role_title_rc['ROLE_TITLE'], 'role_family_desc_rc':test_role_family
_desc_rc['ROLE_FAMILY_DESC'], 'role_family_rc':test_role_family_rc['ROL
E_FAMILY'], 'role_code_rc':test_role_code_rc['ROLE_CODE']})
```

In [48]:
```
train_df_rc
```

Out[48]:

	resource_rc	mgr_id_rc	rollup_1_rc	rollup_2_rc	role_deptname_rc	role_title_rc	role_fami
0	0.993099	1.000000	0.949222	0.956148	0.958333	0.967625	
1	0.966667	0.999993	0.949222	0.969075	0.893082	0.962963	
2	0.984431	0.993099	0.918478	0.918478	0.923077	0.889331	
3	0.942110	1.000000	0.949222	0.969075	0.989474	0.920413	
4	0.999947	0.999981	0.931159	0.876812	0.755556	0.866667	
...	...	...	...	...	...	...	
32764	0.901961	0.965517	0.949222	0.956148	0.989474	0.920413	
32765	0.984431	0.999981	0.963939	0.963939	1.000000	1.000000	
32766	0.962733	0.998959	0.949222	0.954563	1.000000	0.993099	
32767	0.999857	0.687500	0.734545	0.719844	0.864947	0.913706	
32768	0.905512	0.999947	0.925424	0.935484	0.906198	0.925216	

32769 rows × 9 columns

In [49]:
```
test_df_rc
```

Out[49]:

	resource_rc	mgr_id_rc	rollup_1_rc	rollup_2_rc	role_deptname_rc	role_title_rc	role_fami
0	1.000000	0.802556	0.809955	0.809955	0.937445	0.889331	
1	0.618902	1.000000	0.949222	0.954563	0.943820	0.991736	
2	0.999993	1.000000	0.949222	0.956148	1.000000	0.937500	
3	0.984431	0.866667	0.949222	0.957205	0.979323	0.913495	
4	0.941176	1.000000	0.949222	0.969075	0.977901	0.992021	
...	...	...	...	...	...	...	
58916	0.990220	0.956522	0.949222	0.912584	0.992188	0.929458	
58917	0.946488	0.814815	0.949222	0.957205	0.891304	0.970284	
58918	0.961240	0.999613	0.949222	0.969075	0.884615	0.979592	
58919	0.882353	0.998959	0.949222	0.954563	0.884058	0.920413	
58920	0.923077	1.000000	0.949222	0.969075	0.978571	0.989950	

58921 rows × 9 columns

In [50]:
```
train_y_rc = train['ACTION'].values
```

In [51]:
```
train_y_rc.shape
```

Out[51]:
```
(32769,)
```

Feature Engineering

Encoding with Singular Value Decomposition

Here I'll use singular value decomposition (SVD) to learn encodings from pairs of categorical features. SVD is one of the more complex encodings, but it can also be very effective. We'll 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.

You then use singular value decomposition to find two smaller matrices that equal the count matrix when multiplied.

In [52]:
```
#https://www.kaggle.com/dmitrylarko/kaggledays-sf-2-amazon-unsupervised
-encoding#SVD-Encoding
#https://www.kaggle.com/matleonard/encoding-categorical-features-with-s
vd
```

In [53]:
```
train_data=train.drop(columns=['ACTION'],axis=1)
```

In [54]:
```
train_data.shape
```

Out[54]:
```
(32769, 9)
```

In [55]:
```
train_data.nunique()
```

Out[55]:
```
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 [56]:
```
test_data=test.drop(columns=['id'],axis=1)
```

In [57]:
```
test_data.shape
```

Out[57]:
```
(58921, 9)
```

In [58]:
```
test_data.nunique()
```

Out[58]:
```
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 [59]:
```
train_svd = pd.DataFrame()
test_svd = pd.DataFrame()
```

In [60]:
```
temp = train_data.groupby(['ROLE_ROLLUP_1','ROLE_ROLLUP_2'])['ROLE_ROLLUP_1'].count()
temp=temp.unstack(fill_value=0)
```

In [61]:
```
temp
```

Out[61]:

ROLE_ROLLUP_2	23779	31010	32137	117877	117883	117891	117894	117903	117911	117917
ROLE_ROLLUP_1										
4292	0	0	0	0	0	0	0	0	0	0
5110	0	0	0	0	0	0	0	0	0	0
11146	0	0	0	0	0	0	0	0	0	0
91261	0	0	0	0	0	0	0	0	0	0
117876	0	0	0	171	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...
203209	0	0	0	0	0	0	0	0	0	0
209434	0	0	0	0	0	0	0	0	0	0

ROLE_ROLLUP_2	23779	31010	32137	117877	117883	117891	117894	117903	117911	117917
ROLE_ROLLUP_1										
216705	0	0	0	0	0	0	0	0	0	0
247952	0	0	0	0	0	0	0	0	0	0
311178	0	0	0	0	0	0	0	0	0	0

128 rows × 177 columns

In [62]:
```
temp = train_data.groupby(['RESOURCE','MGR_ID'])['MGR_ID'].count()
temp=temp.unstack(fill_value=0)
```

In [63]:
```
temp
```

Out[63]:

MGR_ID	25	27	30	32	33	36	43	46	47	55	...	311251	311338	311355	311433	31143
RESOURCE																
0	0	0	0	0	1	1	0	0	0	0	...	0	0	0	0	
38	0	0	0	0	0	0	0	0	1	0	...	0	0	0	0	
136	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
138	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
153	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
312136	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
312139	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
312140	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
312152	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
312153	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	

7518 rows × 4243 columns

In [64]:
```
train_data.columns
```

Out[64]:
```
Index(['RESOURCE', 'MGR_ID', 'ROLE_ROLLUP_1', 'ROLE_ROLLUP_2', 'ROLE_DE
PTNAME',
       'ROLE_TITLE', 'ROLE_FAMILY_DESC', 'ROLE_FAMILY', 'ROLE_CODE'],
      dtype='object')
```

In [65]:
```
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 [00:23,  3.06it/s]
```

In [66]:
```
train_svd.shape,test_svd.shape
```

Out[66]:
```
((32769, 72), (58921, 72))
```

In [67]:
```
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 [68]:
```
train_svd.head()
```

Out[68]:

	RESOURCE_MGR_ID	RESOURCE_ROLE_ROLLUP_1	RESOURCE_ROLE_ROLLUP_2	RESOURCE_
0	0.088724	2.995769	1.810303	
1	0.559935	25.998514	13.247680	
2	0.000108	0.007828	0.022128	
3	0.044904	0.998590	0.597128	
4	0.059410	2.022416	0.320066	

5 rows × 72 columns

Normalizing the data

In [69]:
```
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)
```

In [70]:
```
train_svd.shape,test_svd.shape
```

Out[70]: ((32769, 72), (58921, 72))

In [71]:
```
train_svd.head()
```

Out[71]:

	RESOURCE_MGR_ID	RESOURCE_ROLE_ROLLUP_1	RESOURCE_ROLE_ROLLUP_2	RESOURCE_

	RESOURCE_MGR_ID	RESOURCE_ROLE_ROLLUP_1	RESOURCE_ROLE_ROLLUP_2	RESOURCE_
0	3.338246e-06	0.000113	0.000068	
1	3.290961e-05	0.001528	0.000779	
2	1.122108e-07	0.000008	0.000023	
3	1.733916e-06	0.000039	0.000023	
4	4.072207e-04	0.013863	0.002194	

5 rows × 72 columns

In [72]:
```
test_svd.head()
```

Out[72]:

	RESOURCE_MGR_ID	RESOURCE_ROLE_ROLLUP_1	RESOURCE_ROLE_ROLLUP_2	RESOURCE_
0	1.748205e-06	0.000014	0.000033	
1	4.757212e-07	0.000061	0.000016	
2	1.895173e-05	0.000584	0.000352	
3	3.237126e-06	0.000120	0.000032	
4	3.102218e-04	0.008945	0.004305	

5 rows × 72 columns

In [73]:
```
# Save data into csv files
```

In [74]:
```
train_df_fc.to_csv('data/train_df_fc.csv', index=False)
test_df_fc.to_csv('data/test_df_fc.csv', index=False)

train_df_rc.to_csv('data/train_df_rc.csv', index=False)
test_df_rc.to_csv('data/test_df_rc.csv', index=False)
```

```
train_svd.to_csv('data/train_svd.csv', index=False)
test_svd.to_csv('data/test_svd.csv', index=False)
```

In [75]:
```
# feature selection for one hot encoding
train_ohe.shape, test_ohe.shape, y_train_ohe.shape
```

Out[75]: ((32769, 15626), (58921, 15626), (32769,))

In [76]:
```
from sklearn.feature_selection import SelectKBest,chi2
ktop = SelectKBest(chi2,k=4500).fit(train_ohe,y_train_ohe)
train_ohe=ktop.transform(train_ohe)
test_ohe=ktop.transform(test_ohe)
```

In [77]:
```
train_ohe.shape, test_ohe.shape, y_train_ohe.shape
```

Out[77]: ((32769, 4500), (58921, 4500), (32769,))

In [78]:
```
sparse.save_npz('data/train_ohe.npz', train_ohe)
sparse.save_npz('data/test_ohe.npz', test_ohe)
```

In [ ]:
```

```