Amazon Employee Access Challenge (Part-3)

##Final Pipeline

In [22]:

```
#Importing libraries
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import pickle
from tqdm import tqdm
from itertools import combinations
from collections import Counter
from scipy import sparse
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import StratifiedKFold
import sklearn.metrics as metrics

import warnings
warnings.filterwarnings("ignore")
```

In [7]:

```
!pip install category_encoders
```

```
Collecting category_encoders
  Downloading https://files.pythonhosted.org/packages/44/57/fcef41c248701ee6
2e8325026b90c432adea35555cbc870aff9cfba23727/category_encoders-2.2.2-py2.py3
-none-any.whl (https://files.pythonhosted.org/packages/44/57/fcef41c248701ee
62e8325026b90c432adea35555cbc870aff9cfba23727/category_encoders-2.2.2-py2.py
3-none-any.whl) (80kB)
     |████████████████████████████████| 81kB 3.6MB/s
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.6/dist
-packages (from category_encoders) (0.5.1)
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.6/dis
t-packages (from category_encoders) (1.18.5)
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.
6/dist-packages (from category_encoders) (0.10.2)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python
3.6/dist-packages (from category_encoders) (0.22.2.post1)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.6/dist
-packages (from category_encoders) (1.4.1)
Requirement already satisfied: pandas>=0.21.1 in /usr/local/lib/python3.6/di
st-packages (from category_encoders) (1.0.5)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages
(from patsy>=0.5.1->category_encoders) (1.12.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist
-packages (from scikit-learn>=0.20.0->category_encoders) (0.15.1)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist
-packages (from pandas>=0.21.1->category_encoders) (2018.9)
Requirement already satisfied: python-dateutil>=2.6.1 in /usr/local/lib/pyth
on3.6/dist-packages (from pandas>=0.21.1->category_encoders) (2.8.1)
Installing collected packages: category-encoders
Successfully installed category-encoders-2.2.2
```

In [8]:

```
#Reading data
data = pd.read_csv('train.csv')
data_test = pd.read_csv('test.csv')
```

In [9]:

```
Y = data['ACTION']
X = data.drop('ACTION', axis = 1)

#Dropping ROLE_CODE feature.
X = X.drop('ROLE_CODE', axis = 1)

X_test = data_test.drop('ROLE_CODE', axis = 1)
X_test = X_test.drop('id', axis = 1)
```

In [10]:

```
def concat_features_duplet(df_train, cols):
    dup_features = []
    for indicies in combinations(range(len(cols)), 2):
        dup_features.append([hash(tuple(v)) for v in df_train[:,list(indicies)]])
    return np.array(dup_features).T
```

In [11]:

```
def concat_features_triplet(df_train, cols):
    tri_features = []
    for indicies in combinations(range(len(cols)), 3):
        tri_features.append([hash(tuple(v)) for v in df_train[:,list(indicies)]])
    return np.array(tri_features).T
```

In [12]:

```
def category_freq(X):
    X_new = X.copy()
    for f in X_new.columns:
        col_count = dict(Counter(X_new[f].values))

        for r in X_new.itertuples():
            X_new.at[r[0], f'{f}_freq'] = col_count[X_new.loc[r[0], f]]
    return X_new
```

###Function-1 - Predictor

In [18]:

```
#Loading models from disk

with open('one_hot.pickle', 'rb') as f:
    one_enc = pickle.load(f)

with open('lab_dup.pickle', 'rb') as g:
    lab_dup_enc = pickle.load(g)

with open('lab_tri.pickle', 'rb') as h:
    lab_tri_enc = pickle.load(h)

with open('scaler.pickle', 'rb') as i:
    scaler = pickle.load(i)

filename = 'logreg1_updated.sav'
loaded_model = pickle.load(open(filename, 'rb'))

def final_fun_1(X):

    X_dup_test = concat_features_duplet(np.array(X), ['RESOURCE', 'MGR_ID', 'ROLE_ROLLUP_1'
          'ROLE_TITLE', 'ROLE_FAMILY_DESC', 'ROLE_FAMILY'])

    X_tri_test = concat_features_triplet(np.array(X), ['RESOURCE', 'MGR_ID', 'ROLE_ROLLUP_1
          'ROLE_TITLE', 'ROLE_FAMILY_DESC', 'ROLE_FAMILY'])

    X_dup_test= lab_dup_enc.transform(X_dup_test)
    X_tri_test= lab_tri_enc.transform(X_tri_test)

    X_freq_test = np.array(category_freq(X).iloc[:,8:])

    X_all_categorical = np.hstack((X, X_dup_test, X_tri_test))

    X_freq = scaler.transform(X_freq_test)

    X_all_categorical_selected= X_all_categorical[:, [64, 42, 69, 11, 85, 0, 65, 67, 29, 9,
    X_freq_selected = X_freq[:, [1, 5, 7]]

    X_selected = sparse.hstack((one_enc.transform(X_all_categorical_selected), X_freq_selec

    preds = loaded_model.predict_proba(X_selected)[:, 1]

    access_or_not = loaded_model.predict(X_selected)

    return preds, access_or_not
```

In [31]:

```
_, access_or_not = final_fun_1(X_test.iloc[[1,2,3]])

print(access_or_not)
```

```
[1 1 1]
```

1, 1, 1 means that access can be granted for these 3 users.

###Function-2 - Evaluator

In [29]:

```
def final_fun_2(X, Y):

    mean_auc = []

    kf = StratifiedKFold(n_splits = 5,shuffle=True,random_state=37)

    for idx, (train_index, test_index) in enumerate(kf.split(X, Y)):

        X_train = X.iloc[train_index]
        y_train = Y.iloc[train_index]
        X_cv = X.iloc[test_index]
        y_cv = Y.iloc[test_index]

        preds, _ = final_fun_1(X_cv)

        fpr, tpr, thresholds = metrics.roc_curve(y_cv, preds)
        roc_auc = metrics.auc(fpr, tpr)
        mean_auc.append(roc_auc)
        print(f"For KFold: {idx+1}/5, AUC = {roc_auc}")

    print(f"\nAverage AUC Score: {np.mean(mean_auc)}")
```

In [30]:

```
final_fun_2(X.iloc[1:1000],Y.iloc[1:1000])
```

```
For KFold: 1/5, AUC = 0.9800531914893618
For KFold: 2/5, AUC = 0.9238996297819827
For KFold: 3/5, AUC = 0.9819004524886877
For KFold: 4/5, AUC = 0.9802550390785686
For KFold: 5/5, AUC = 0.9901960784313725

Average AUC Score: 0.9712608782539947
```

###NOTE: The Model is giving overfitted results here as we are predicting the same training data again which we used to train the model.