PML Lab #9 Employee Hopping Prediction using RF

Load your data: *attrition = pd.read_csv('Employee_Hopping.csv')*

Check data types: *attrition.dtypes.value_counts()*

Get all categorical columns and perform one hot encoding using get_dummies() method:

You need iterate through the list of column names and collect categorical columns and store it in an empty list. With this categorical column names, create a data frame.

Similarly, gather numerical column names in another list and create another data frame.

Now, concatenate these two new dataframes, using pd.concat() method.

Target variable: One final step that that we have to remember is to generate our target variable. The target in this case is given by the column **Attrition** which contains categorical variables therefore requires numerical encoding. We numerically encode it by creating a dictionary with the mapping given as 1 : Yes and 0 : No

OPTIONAL: SMOTE to oversample due to the skewness in target

Since we have already noted the severe imbalance in the values within the target variable, let us implement the SMOTE method in the dealing with this skewed value via the imblearn Python package. Use the following code:

```
from imblearn.over_sampling import SMOTE
oversampler=SMOTE(random_state=0)
smote_train, smote_target = oversampler.fit_sample(X_train, y_train)
```

Initialising Random Forest parameters

```
seed = 0   # We set our random seed to zero for reproducibility
# Random Forest parameters
rf_params = {
    'n_jobs': -1,
    'n_estimators': 1000,
#     'warm_start': True,
    'max_features': 0.3,
    'max_depth': 4,
    'min_samples_leaf': 2,
    'max_features' : 'sqrt',
    'random_state' : seed,
    'verbose': 0
}
```

Feature Ranking via the Random Forest

The Random Forest classifier in Sklearn also contains a very convenient attribute **feature_importances_** which tells us which features within our dataset has been given most importance through the Random Forest algorithm.

Suppose, *rf = RandomForestClassifier()*
print(rf.feature_importances_)

Now, you can print “Feature Name - Feature Importance Score", as a two column table. The following code snippet will do it.

```
x = attrition_final.columns.values
y = rf.feature_importances_
```

```
lst = []

print("Feature Name - Feature Importance Score")
print("====================================")
for i in range(55):
    lst.append((x[i], y[i]))

sorted(lst,key=itemgetter(1))
```

You can also plot the following chart