NATIONAL CHENGCHI UNIVERSITY

Data Mining 2020 - HW

Course Instructor: Man-Kwan Shan

Students: San-Mei Chang 108971020

Date : 2020/07/03

Agenda

- Data Analysis
  - Data (Categorical)
  - Data (Numeric)
  - Feature Selection
  - Attribute Relationship
  - Decision Tree
- Imbalance Data
  - Resampling (SMOTE)
  - Performance Indicator
- Model Evaluation
  - Random Forest Classifier
  - Logistic Regression

NATIONAL CHENGCHI UNIVERSITY

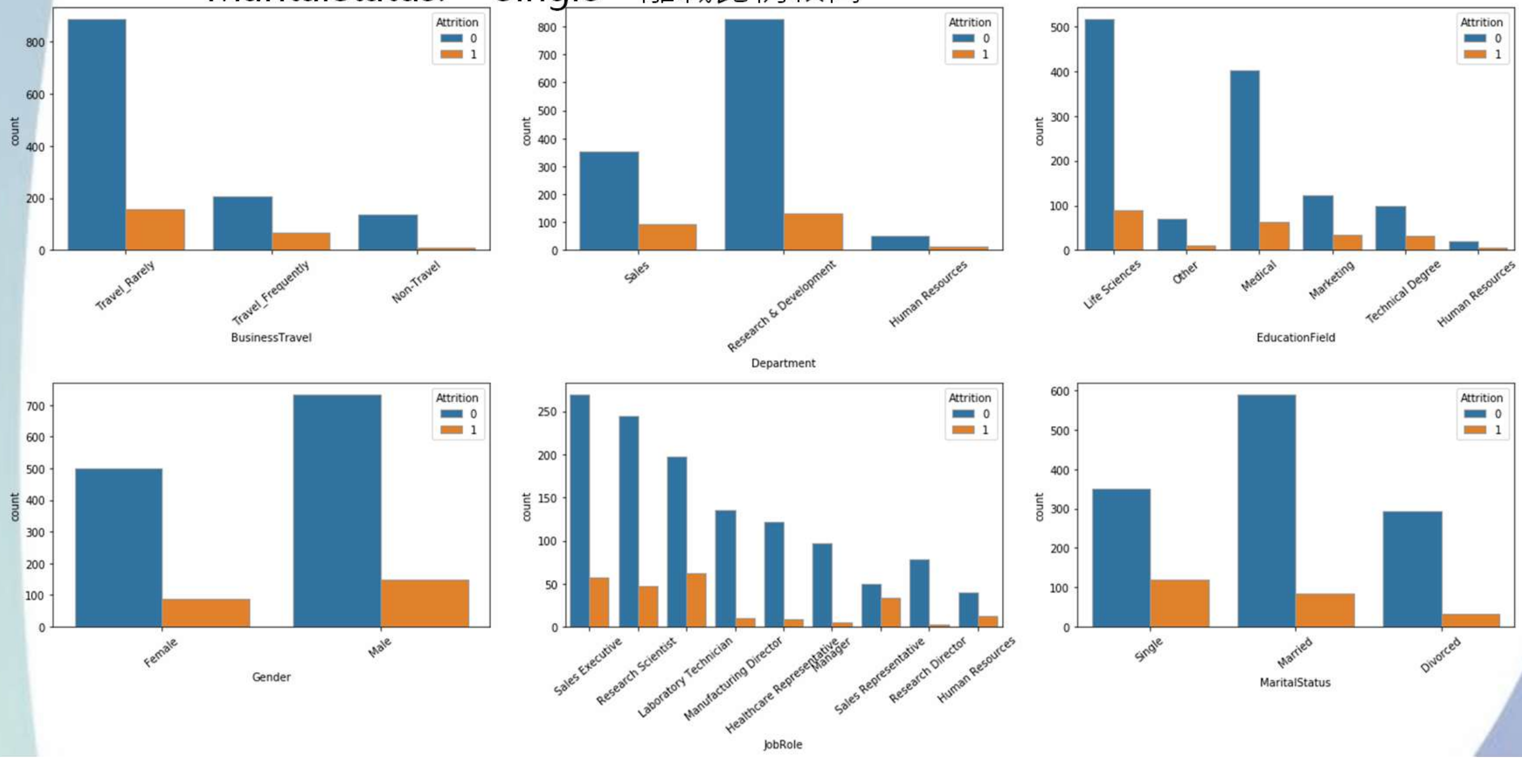
Data (Categorical)

- 共有6個Categorical Feature
- 利用one-hot encoding轉換類別資料
  - pd.get_dummies

```
BusinessTravel ['Travel_Rarely' 'Travel_Frequently' 'Non-Travel’ ]
=========================================================================
Department ['Sales' 'Research & Development' 'Human Resources’ ]
=========================================================================
EducationField ['Life Sciences' 'Other' 'Medical' 'Marketing' 'Technical
Degree' 'Human Resources’ ]
=========================================================================
Gender ['Female' 'Male’ ]
=========================================================================
JobRole ['Sales Executive' 'Research Scientist' 'Laboratory Technician’
'Manufacturing Director' 'Healthcare Representative' 'Manager’
'Sales Representative' 'Research Director' 'Human Resources’ ]
=========================================================================
MaritalStatus ['Single' 'Married' 'Divorced’ ]
=========================================================================
```

Data (Categorical)

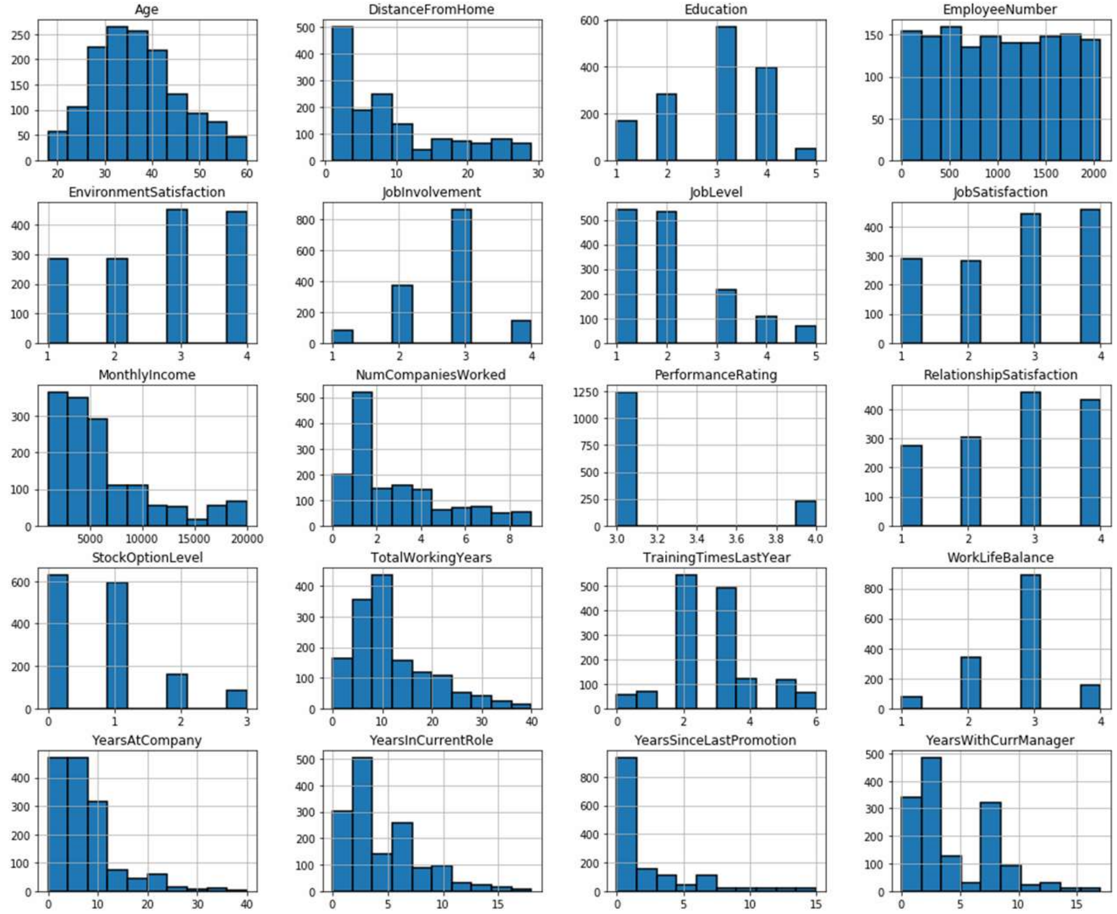
- Check以下欄位和Attrition是否離職的關係
  - BusinessTravel: ‘Non-Travel’ 無離職人數
  - Department: ‘Sales’ 離職比例較高
  - EducationField: ‘Life Sciences’ 離職人數最高
  - Gender: 關聯度不高
  - JobRole: ‘Laboratory Technician ’ 離職人數最高
  - MaritalStatus: ‘Single’ 離職比例較高



NATIONAL CHENGCHI UNIVERSITY

Data (Numeric)

- 共有20個Numeric Feature



NATIONAL CHENGCHI UNIVERSITY

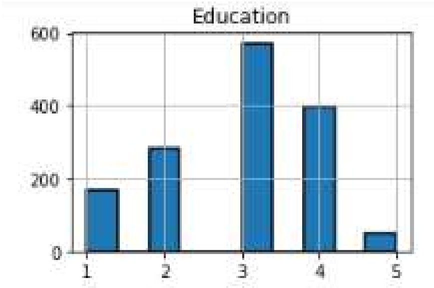
Data (Numeric)

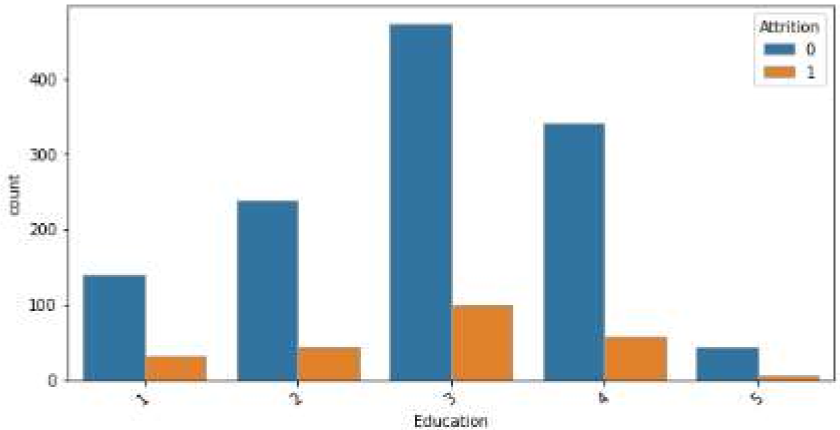
- Check以下欄位和Attrition是否離職的關係

```
Education [1, 2, 3, 4, 5]
============================================================================
DistanceFromHome [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]
============================================================================
EnvironmentSatisfaction [1, 2, 3, 4]
============================================================================
StockOptionLevel [0, 1, 2, 3]
============================================================================
TrainingTimesLastYear [0, 1, 2, 3, 4, 5, 6]
============================================================================
WorkLifeBalance [1, 2, 3, 4]
============================================================================
```

Data (Numeric)

- Education
  - 是否離職和教育程度，3-Doctor離職人數較高

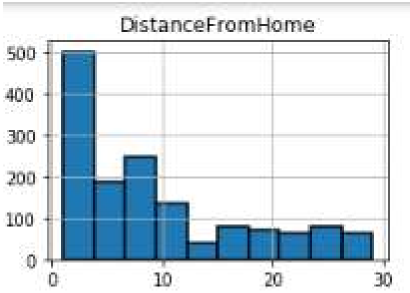


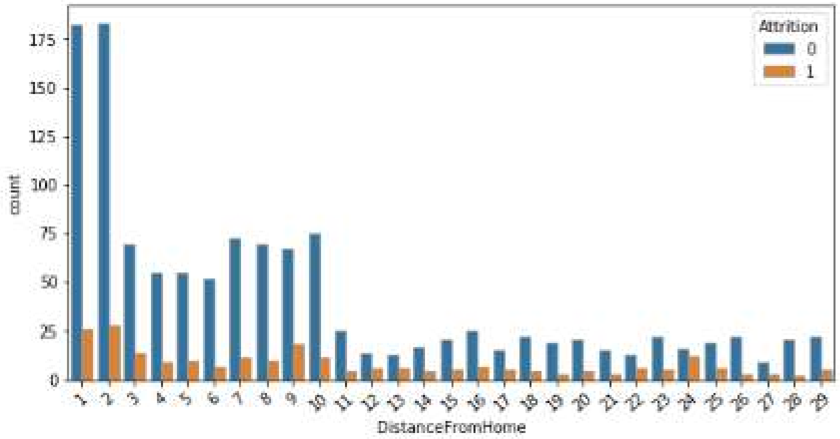


Education:
'Below College’: 0, ‘College’: 1, ’Bachelor’: 2, ’Master’: 3, ‘Doctor’: 4

Data (Numeric)

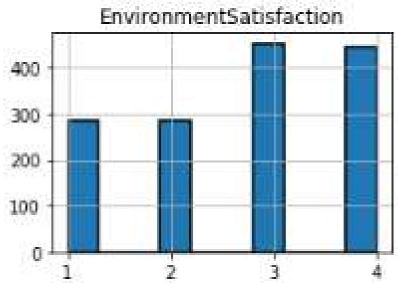
- DistanceFromHome
  - 是否離職和公司離家距離，關連度不高，非常態分佈

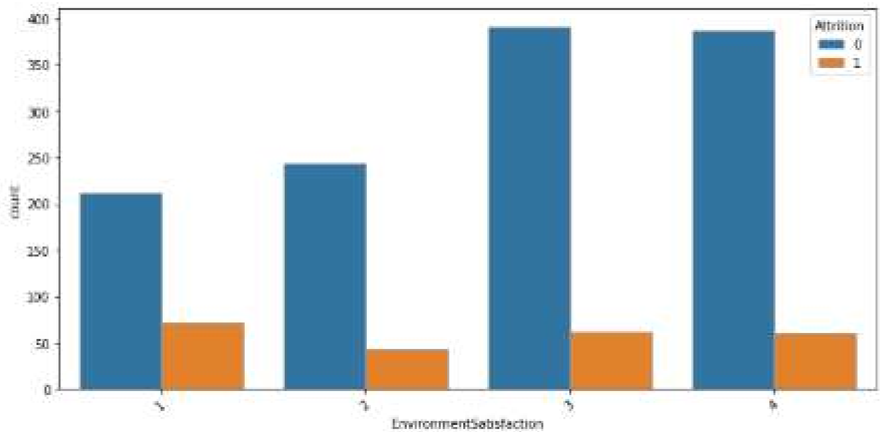




Data (Numeric)

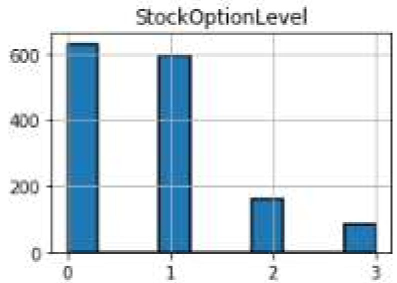
- EnvironmentSatisfaction
  - 是否離職和對工作環境滿意度，關連度不高

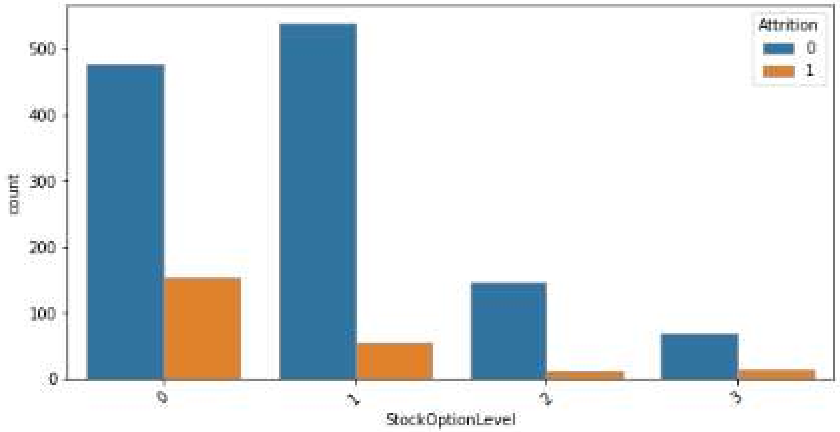




Data (Numeric)

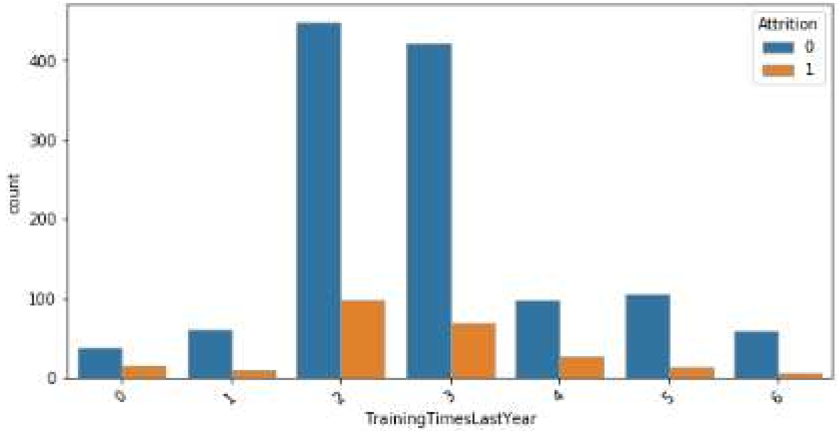
- StockOptionLevel
  - 是否離職和股票選擇權Level有明顯的看到隨Level變高，離職人數降低





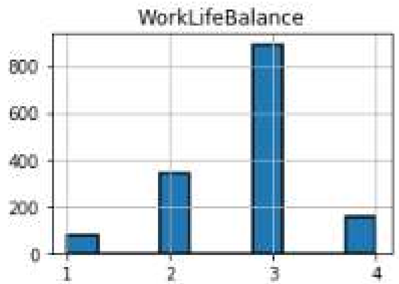
Data (Numeric)

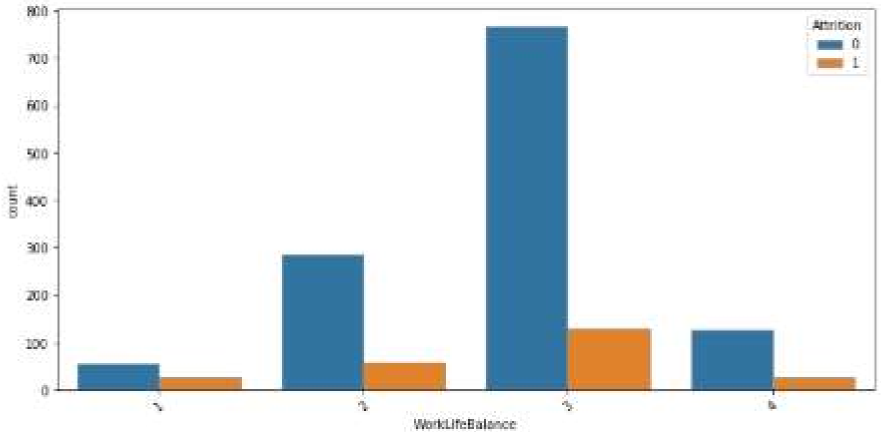
- TrainingTimesLastYear
  - 是否離職和去年訓練次數，在2, 3 時似乎比較高，但也可能是因為分佈的人數比較高



Data (Numeric)

- WorkLifeBalance
  - 是否離職和工作與生活平衡滿意度，關連度不高

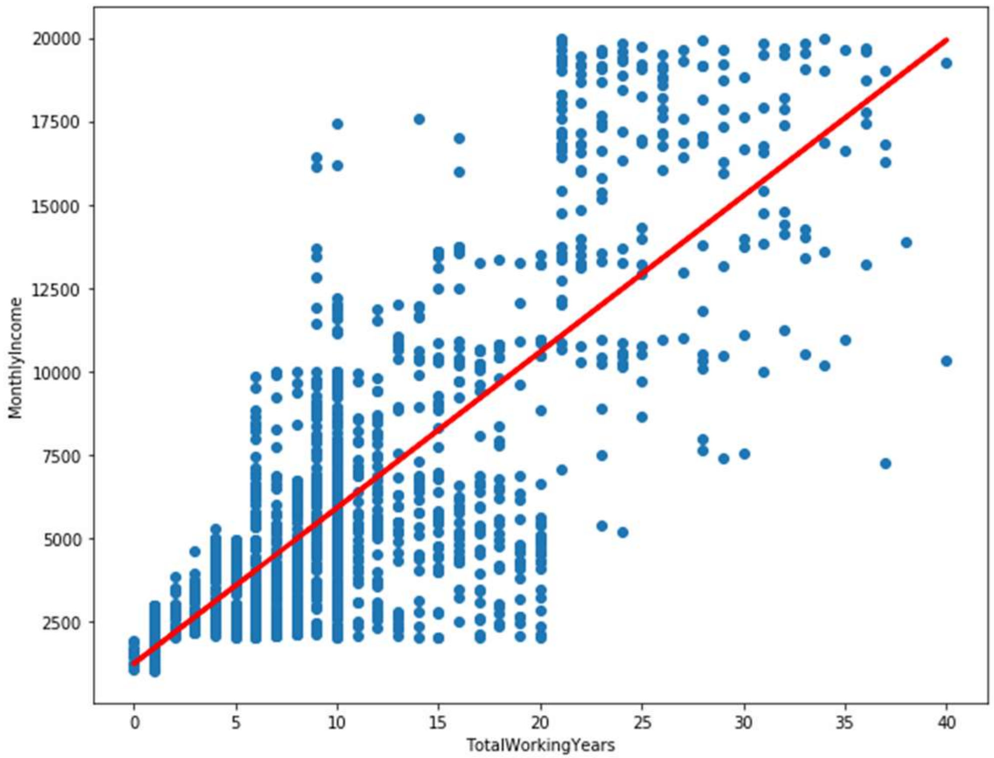




NATIONAL CHENGCHI UNIVERSITY

Attribute Relationship

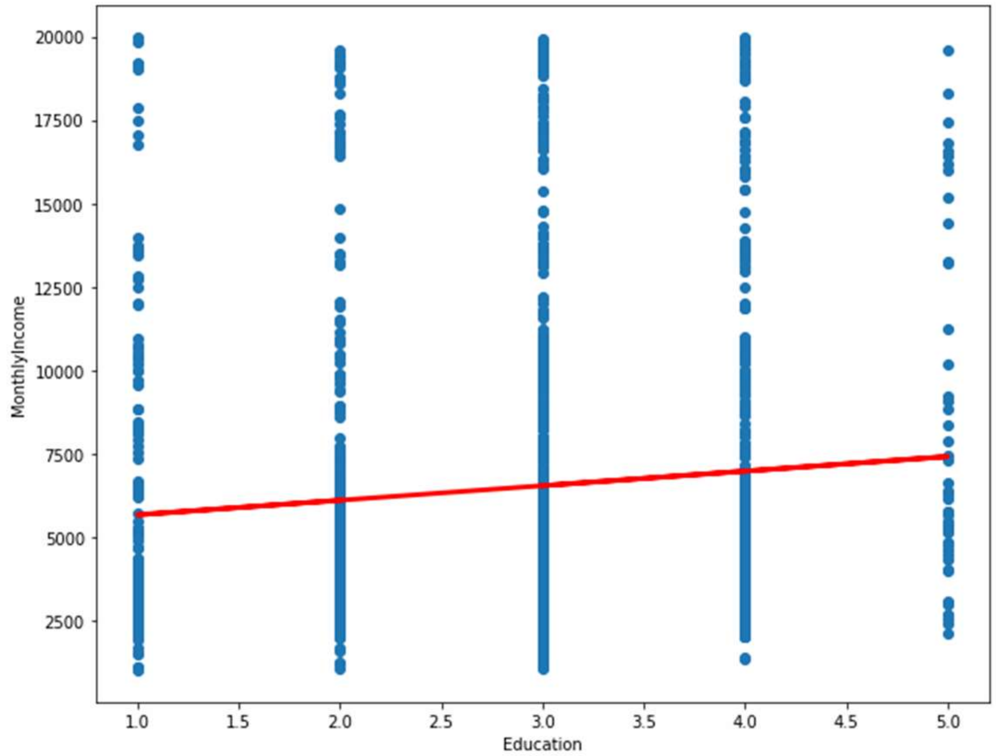
- TotalWorkingYears V.S. MonthlyIncome
  - 總工作年數和月收入，成正比



NATIONAL CHENGCHI UNIVERSITY

Attribute Relationship

- Education V.S. MonthlyIncome
  - 教育程度和月收入，並非成正比，還須其它feature搭配才看得出關係



Feature Selection

- 利用scikit-learn選出可drop之feature，看是否提高準確率

```
## Feature Selection
oh_X = one_hot_encoding_df.iloc[:,2:46]
oh_y = one_hot_encoding_df['Attrition']

Selector = SelectKBest(chi2, k=43)
X_new = Selector.fit_transform(oh_X,oh_y)
train_X_new = Selector.get_support()
train_X_new[0:43]
```

```
array([ True, True, True, True,
True, True, True, True, True, True,
True, True, True, True, True, True,
True, True, True, True, True, True,
True, True, True, True, True, True,
True, True, True, True, True, True,
True, True, True, True, True, False,
True, True, True])
```

```
## Feature Selection - Drop
one_hot_encoding_df = one_hot_encoding_df.drop(['JobRole_Sales Representative'], axis=1)
```

NATIONAL CHENGCHI UNIVERSITY

Feature Selection

- 比較Drop feature後，正確率有稍提升：

```
Training Data: 1176, Testing Data: 294 Fold: 1, Accuracy: 0.850000, Precision: 0.714000,
Recall: 0.106000, F1: 0.185000 Training Data: 1176, Testing Data: 294 Fold: 2, Accuracy:
0.871000, Precision: 0.727000, Recall: 0.186000, F1: 0.296000 Training Data: 1176, Testing
Data: 294 Fold: 3, Accuracy: 0.816000, Precision: 1.000000, Recall: 0.085000, F1: 0.156000
Training Data: 1176, Testing Data: 294 Fold: 4, Accuracy: 0.854000, Precision: 0.600000,
Recall: 0.133000, F1: 0.218000 Training Data: 1176, Testing Data: 294 Fold: 5, Accuracy:
0.867000, Precision: 0.833000, Recall: 0.116000, F1: 0.204000
================================================================================
Avg Accuracy: 0.852000, Avg Precision: 0.775000, Avg Recall: 0.125000, Avg F1: 0.212000
```

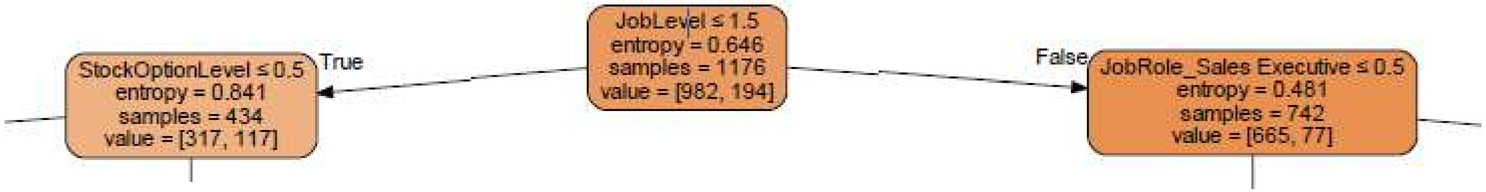
Drop 前

Drop 後

```
Training Data: 1176, Testing Data: 294 Fold: 1, Accuracy: 0.850000, Precision: 0.714000,
Recall: 0.106000, F1: 0.185000 Training Data: 1176, Testing Data: 294 Fold: 2, Accuracy:
0.867000, Precision: 0.750000, Recall: 0.140000, F1: 0.235000 Training Data: 1176,
Testing Data: 294 Fold: 3, Accuracy: 0.816000, Precision: 0.857000, Recall: 0.102000, F1:
0.182000 Training Data: 1176, Testing Data: 294 Fold: 4, Accuracy: 0.871000, Precision:
0.818000, Recall: 0.200000, F1: 0.321000 Training Data: 1176, Testing Data: 294 Fold: 5,
Accuracy: 0.871000, Precision: 0.778000, Recall: 0.163000, F1: 0.269000
================================================================================
Avg Accuracy: 0.855000, Avg Precision: 0.783000, Avg Recall: 0.142000, Avg F1: 0.239000
```

Decision Tree

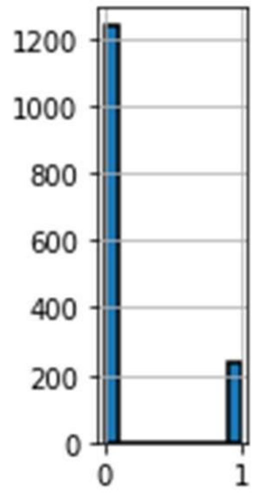
- 計算各個Feature Entropy
  - 呼應前面提到「StockOptionLevel」為重要的feature
  - 詳細內容請參考<<Tree Mode>>



Label-Attrition

- 離職=No/Yes的資料為Imbalance Data
- 為binary classification問題

```
0 1233
1 237
Name: Attrition, dtype: int64
```



Resampling(SMOTE)

- 採用oversampling
  - Replicates the training data of rare class

```
## SMOTE
oversample = SMOTE()
X = one_hot_encoding_df.iloc[:,one_hot_encoding_df.columns != 'Attrition']
y = one_hot_encoding_df['Attrition']
X, y = oversample.fit_resample(X, y)
```

Counter({1: 1233, 0: 1233})

Resampling(SMOTE)

- 比較Resampling前後，跑RFC結果如下：

```
Count: 1470
Accuracy: 0.866000, Precision: 0.800000, Recall: 0.123000, F1: 0.213000
```

前

後

```
Count: 2466
Accuracy: 0.923000, Precision: 1.000000, Recall: 0.923000, F1: 0.960000
```

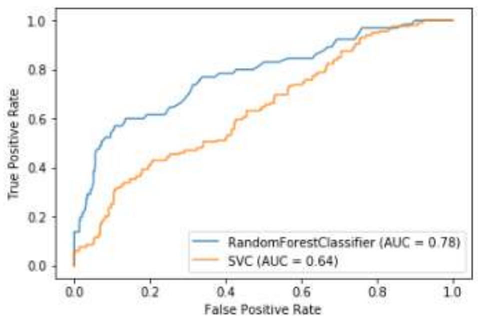
Performance Indicator

- False Positive預測不會離職的，卻離職了
  - cost高
- True Negative預測會離職的，沒有離職
  - Cost不高
- Rank:拿來預測Top 10 會離職的關注

Performance Indicator

- 未Resampling前: Training a Random Forest and Plotting the ROC Curve

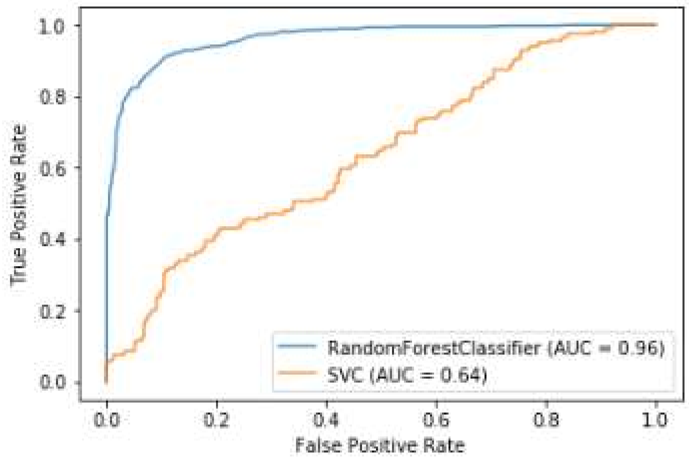
```
In [39]: ## Model Learning (Random Forest)
         total=len(one_hot_encoding_df)
         print(total)

         testPortion = 0.3
         testSize = int(testPortion*total)

         lr_XX = one_hot_encoding_df.loc[:,one_hot_encoding_df.columns != 'Attrition']
         lr_yy = one_hot_encoding_df["Attrition"]

         train_X = lr_XX[0:-testSize]
         train_y = lr_yy[0:-testSize]
         test_X = lr_XX[-testSize:]
         test_y = lr_yy[-testSize:]

         model = RandomForestClassifier(n_estimators=300)
         model = model.fit(train_X, train_y)
         test_predict = model.predict(test_X)


         acc, precision, recall, f1, matrix = evaluation(test_y, test_predict)
         print("Accuracy: %f, Precision: %f, Recall: %f, F1: %f" % (round(acc, 3), round(precision, 3), round(recall, 3), round(f1, 3)))

         1470
         Accuracy: 0.862000, Precision: 0.643000, Recall: 0.138000, F1: 0.228000

In [40]: ax = plt.gca()
         rfc_disp = plot_roc_curve(model, test_X, test_y, ax=ax, alpha=0.8)
         svc_disp.plot(ax=ax, alpha=0.8)
         plt.show()
```



Performance Indicator

- Resampling後: Training a Random Forest and Plotting the ROC Curve

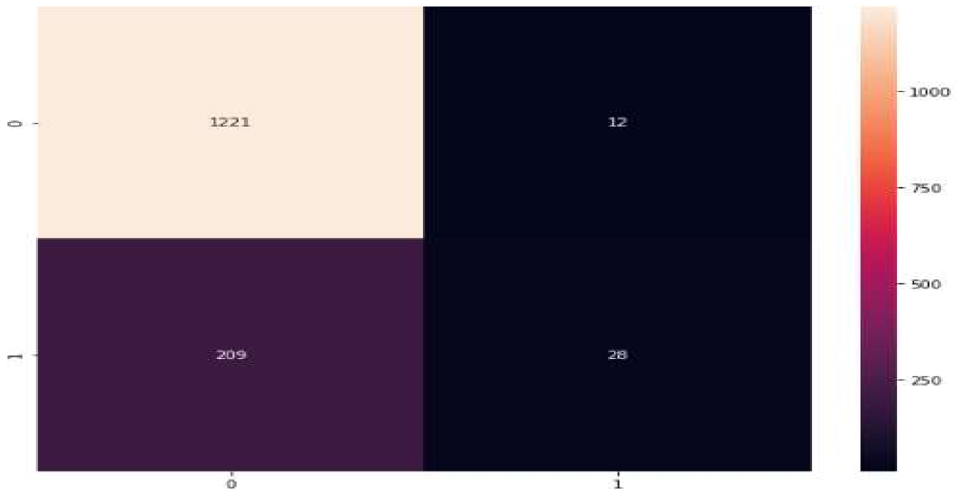
```
In [41]: ax = plt.gca()
         #rfc_disp = plot_roc_curve(model, test_X, test_y, ax=ax, alpha=0.8)
         rfc_disp = plot_roc_curve(model, trainIB_X, trainIB_y, ax=ax, alpha=0.8)
         svc_disp.plot(ax=ax, alpha=0.8)
         plt.show()
```



Random Forest Classifier

```
Training Data: 1176, Testing Data: 294
Fold: 1, Accuracy: 0.847000, Precision: 0.625000, Recall: 0.106000, F1: 0.182000
Training Data: 1176, Testing Data: 294
Fold: 2, Accuracy: 0.867000, Precision: 0.700000, Recall: 0.163000, F1: 0.264000
Training Data: 1176, Testing Data: 294
Fold: 3, Accuracy: 0.810000, Precision: 0.800000, Recall: 0.068000, F1: 0.125000
Training Data: 1176, Testing Data: 294
Fold: 4, Accuracy: 0.857000, Precision: 0.667000, Recall: 0.133000, F1: 0.222000
Training Data: 1176, Testing Data: 294
Fold: 5, Accuracy: 0.867000, Precision: 0.750000, Recall: 0.140000, F1: 0.235000
================================================================================
Avg Accuracy: 0.850000, Avg Precision: 0.708000, Avg Recall: 0.122000, Avg F1: 0.206000
```

```
plt.figure(figsize=(10, 8))
sns.heatmap(np.sum(np.array(avg_confusion_matrix), axis=0), annot=True, fmt="d")
plt.show()
```




```
importance_dict = {}
for col, importance in zip(train_X.columns, np.mean(np.array(avg_feature_importance), axis=0)):
    importance_dict[col] = importance

sorted(importance_dict.items(), key=lambda x: -x[1])[:10]
```

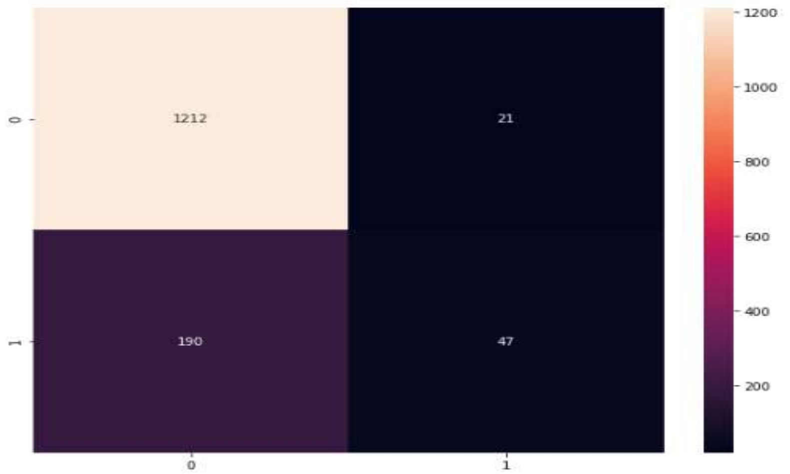
```
[('MonthlyIncome', 0.08095068239878031),
 ('Age', 0.07066656478624714),
 ('EmployeeNumber', 0.06254983310033983),
 ('TotalWorkingYears', 0.05809665479881319),
 ('DistanceFromHome', 0.0564798051189247),
 ('YearsAtCompany', 0.046944051362696734),
 ('NumCompaniesWorked', 0.03998926320401515б),
 ('YearsWithCurrManager', 0.036939229229426726),
 ('EnvironmentSatisfaction', 0.03418396356969071),
 ('JobSatisfaction', 0.03374625687072978)]
```

Logistic Regression

```
    model = LogisticRegression(solver='liblinear')
    model = model.fit(train_X, train_y)
    test_predict = model.predict(test_X)

    acc, precision, recall, f1, matrix = evaluation(test_y, test_predict)
    print("Fold: %d, Accuracy: %f, Precision: %f, Recall: %f, F1: %f" % (fold_count + 1, round(acc, 3), round(precision, 3), rou
    avg_acc += acc
    avg_precision += precision
    avg_recall += recall
    avg_f1 += f1
    avg_confusion_matrix.append(matrix)
    fold_count += 1

print("==========================================================================================")
print("Avg Accuracy: %f, Avg Precision: %f, Avg Recall: %f, Avg F1: %f" % (round(avg_acc / kf.get_n_splits(), 3), \
                                                                            round(avg_precision / kf.get_n_splits(), 3), \
                                                                            round(avg_recall / kf.get_n_splits(), 3), \
                                                                            round(avg_f1 / kf.get_n_splits(), 3)))
```

```
Training Data: 1176, Testing Data: 294
Fold: 1, Accuracy: 0.854000, Precision: 0.833000, Recall: 0.106000, F1: 0.189000
Training Data: 1176, Testing Data: 294
Fold: 2, Accuracy: 0.857000, Precision: 0.529000, Recall: 0.209000, F1: 0.300000
Training Data: 1176, Testing Data: 294
Fold: 3, Accuracy: 0.813000, Precision: 0.625000, Recall: 0.169000, F1: 0.267000
Training Data: 1176, Testing Data: 294
Fold: 4, Accuracy: 0.874000, Precision: 0.786000, Recall: 0.244000, F1: 0.373000
Training Data: 1176, Testing Data: 294
Fold: 5, Accuracy: 0.884000, Precision: 0.800000, Recall: 0.279000, F1: 0.414000
==========================================================================================
Avg Accuracy: 0.856000, Avg Precision: 0.715000, Avg Recall: 0.202000, Avg F1: 0.308000
```

```
plt.figure(figsize=(10, 8))
sns.heatmap(np.sum(np.array(avg_confusion_matrix), axis=0), annot=True, fmt="d")
plt.show()
```



附件

- https://github.com/Sanmei-0807/DM2020/blob/master/HW_DataMining2020_108971020.ipynb
- Decision Tree產出之pdf
  - https://github.com/Sanmei-0807/DM2020/blob/master/tree_model.pdf