Adult Data Set Analysis and Classification

Kelompok 4

- Nuzha Musyafira	(05111640000014)
- Ferdinand Jason Gondowijoyo	(05111640000033)
- Nurlita Dhuha Fatmawati	(05111640000092)
- Jonathan Rehuel Lewerissa	(05111640000105)

Supervisor
Dr. Chastine Fatichah, M.Kom.

DATA MINING

DEPARTEMEN INFORMATIKA

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

2019

1 Adult Data Set Analysis and Classification

We continue the analysis on assignment 1, with classification, so that the source code below continuation in assignment 1.

1.1 Dimension Reduction and Feature Selection

Data reduction using Dimensionality Reduction is needed to avoid the curse of dimensionality, help eliminate irrelevant features, reduce noise, time and space required in data mining, and also allow an easier visualization. In this analysis, we're going to use Principal Component Analysis (PCA) as our dimensionality reduction technique. Before using Principal Component Analysis on the dataset, dataset need standardized first using `StandardScaler`.

On this dataset, we decided to reduce feature from 14 to 9.

```
In [12]: from sklearn.preprocessing import StandardScaler
         from sklearn.decomposition import PCA

         # Field Education and Education-Num represent the same feature, so we can drop one
         encoded_data = encoded_data.drop('Education-Num',1)

         feature_dataframe = encoded_data.drop('Target',1)

         sc = StandardScaler()
         standard_feature_dataframe = sc.fit_transform(feature_dataframe.astype('float64'))

         pca = PCA(n_components=9)
         pca_feature_dataframe = pca.fit_transform(standard_feature_dataframe)

         print(pca_feature_dataframe)
```

```
[[ 0.67920063 -0.61895733 -0.63335428 ...  0.19893869 -0.7327437
   1.37479831]
 [ 0.65654677 -0.62899563 -0.14880203 ... -0.73591687 -0.61334228
  -0.40134703]
 [ 1.03391718 -0.24882968 -0.00978604 ... -0.84665026  0.21546684
  -0.97173434]
 ...
 [-1.91657756 -1.3383945   0.25273515 ... -0.56554022 -0.36181964
   0.92643081]
 [-1.57334105  0.0877552  -1.10546941 ... -0.93597509 -0.53386515
   0.42197529]
 [-0.58480885 -1.88611787  1.94542111 ... -0.57958575  2.81574889
   2.40946941]]
```

1.2 Handle Imbalanced Dataset

There is way to handle imbalanced dataset, called Sampling-based Approaches. Sampling the data modifies the distribution of training data so that minority class would be well-represented in training set. Sampling can be done by either undersampling the majority class or oversampling the minority class. Below we're going to try `SMOTE`,`ADASYN`, `BorderLineSMOTE`, `SMOTE + TomekLink`, `ADASYN + TomekLinks` and `BorderLineSMOTE + TomekLinks` methods for data sampling.

Before start, let's `import` some library we are going to need for handling imbalanced dataset and classification

```
In [13]: from imblearn.under_sampling import TomekLinks
         from imblearn.over_sampling import SMOTE
         from imblearn.over_sampling import ADASYN
         from imblearn.over_sampling import BorderlineSMOTE

         from sklearn.metrics import precision_recall_fscore_support
         from sklearn.metrics import accuracy_score

         from sklearn.ensemble import RandomForestClassifier

         from sklearn.model_selection import StratifiedKFold
```

```
In [105]: dataframe_sampling = {}

          def make_sampling_dataset(feature, target):
              global dataframe_sampling

              sm = SMOTE(ratio='auto', random_state=1)
              new_data_smote, new_target_smote = sm.fit_sample(feature, target)
              dataframe_sampling['SMOTE'] = {}
              dataframe_sampling['SMOTE']['feature'] = new_data_smote
              dataframe_sampling['SMOTE']['target'] = new_target_smote

              ada = ADASYN(ratio='auto', random_state=1)
              new_data_adasyn, new_target_adasyn = ada.fit_sample(feature, target)
              dataframe_sampling['ADASYN']
              {}
              dataframe_sampling['ADASYN']['feature'] = new_data_adasyn
              dataframe_sampling['ADASYN']['target'] = new_target_adasyn

              blsm = BorderlineSMOTE(random_state=1)
              new_data_blsmote, new_target_blsmote = blsm.fit_sample(feature, target)
              dataframe_sampling['BorderLineSMOTE'] = {}
              dataframe_sampling['BorderLineSMOTE']['feature'] = new_data_blsmote
              dataframe_sampling['BorderLineSMOTE']['target'] = new_target_blsmote

              t1 = TomekLinks(random_state=1, ratio='not minority')
              new_data_smote_tomek, new_target_smote_tomek =
              t1.fit_sample(new_data_smote, new_target_smote)
```

```
    dataframe_sampling['SMOTE_Tomek'] = {}
    dataframe_sampling['SMOTE_Tomek']['feature'] = new_data_smote_tomek
    dataframe_sampling['SMOTE_Tomek']['target'] = new_target_smote_tomek

    t2 = TomekLinks(random_state=1, ratio='not minority')
    new_data_adasyn_tomek, new_target_adasyn_tomek =
    t2.fit_sample(new_data_adasyn, new_target_adasyn)
    dataframe_sampling['ADASYN_Tomek'] = {}
    dataframe_sampling['ADASYN_Tomek']['feature'] = new_data_adasyn_tomek
    dataframe_sampling['ADASYN_Tomek']['target'] = new_target_adasyn_tomek

    t3 = TomekLinks(random_state=1, ratio='not minority')
    new_data_blsmote_tomek, new_target_blsmote_tomek =
    t3.fit_sample(new_data_blsmote, new_target_blsmote)
    dataframe_sampling['BorderLineSMOTE_Tomek'] = {}
    dataframe_sampling['BorderLineSMOTE_Tomek']['feature'] =
    new_data_blsmote_tomek
    dataframe_sampling['BorderLineSMOTE_Tomek']['target'] =
    new_target_blsmote_tomek

make_sampling_dataset(pca_feature_dataframe, encoded_data['Target'])
```

1.3 Data Sampling and Classification

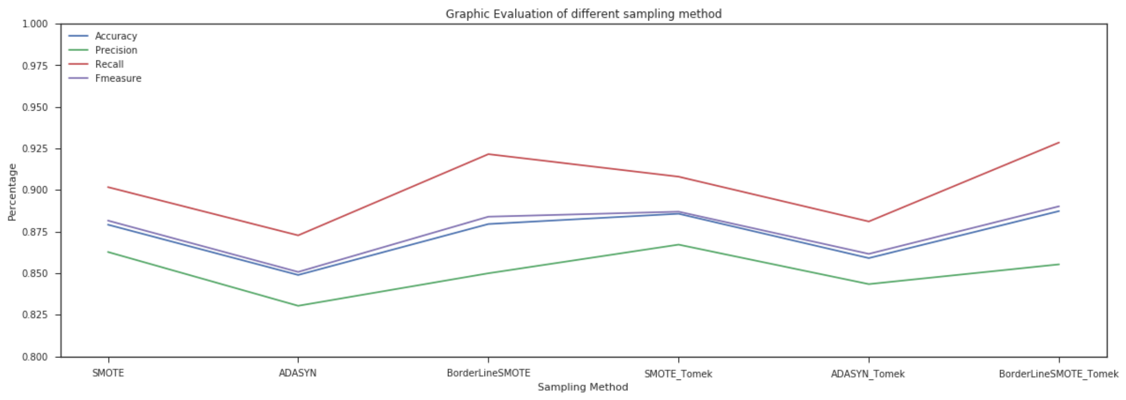
To reduce size of data, we use sampling method called `StratifiedKFold`, which is sampling the data and make cross validation through it. For classification we use `RandomForestClassifier` from `sklean`.

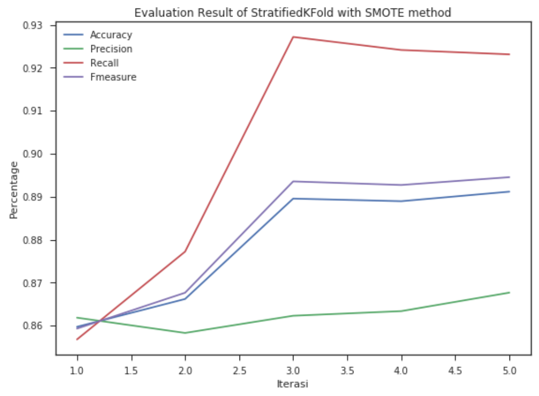
For evaluation we use `Accuracy`, `Precision`, `Recall`, and `Fmeasure`.

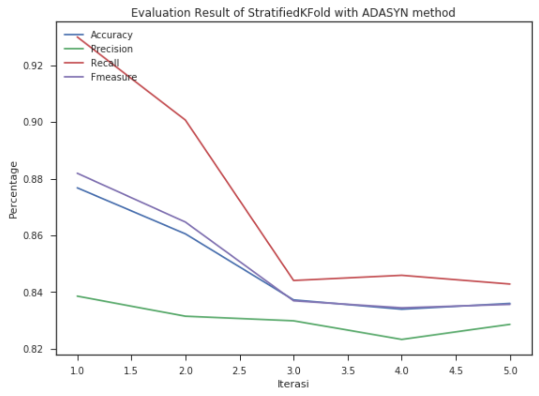
```
In [106]: from collections import Counter

          dataframe_result = {}

          def sampling_and_classify(key, feature, target):
              global dataframe_result
              dataframe_result[key] = {}
              skf = StratifiedKFold(n_splits=5)
              rfc = RandomForestClassifier(n_estimators=20,random_state=1)

              dataframe_result[key]['accuracy'] = []
              dataframe_result[key]['precision'] = []
              dataframe_result[key]['recall'] = []
              dataframe_result[key]['fmeasure'] = []

              print('Original shape = '+str(feature.shape))
              print(Counter(target))

              first = True
              for train,test in skf.split(feature, target):
                  if first :
                      print(Counter(target[train]))
                      first = False
                  rfc.fit(feature[train], target[train])
                  predict = rfc.predict(feature[test])
                  score = precision_recall_fscore_support
                  (target[test], predict, average='binary')
                  dataframe_result[key]['accuracy'].append
                  (accuracy_score(target[test], predict))
                  dataframe_result[key]['precision'].append(score[0])
                  dataframe_result[key]['recall'].append(score[1])
                  dataframe_result[key]['fmeasure'].append(score[2])

              print('Accuracy = ', np.mean(dataframe_result[key]['accuracy']))
              print('Precision = ', np.mean(dataframe_result[key]['precision']))
              print('Recall = ', np.mean(dataframe_result[key]['recall']))
              print('Fmeasure = ', np.mean(dataframe_result[key]['fmeasure']))
              print()

          for key in dataframe_sampling:
              print('Classify and sampling with method '+key+' :')
```

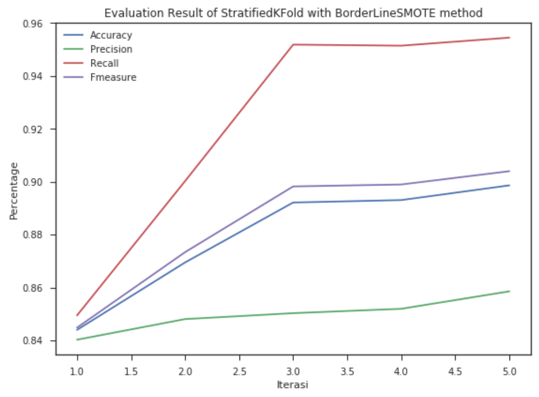
```
            print()
            sampling_and_classify(key, dataframe_sampling[key]['feature'],
            dataframe_sampling[key]['target'])
```

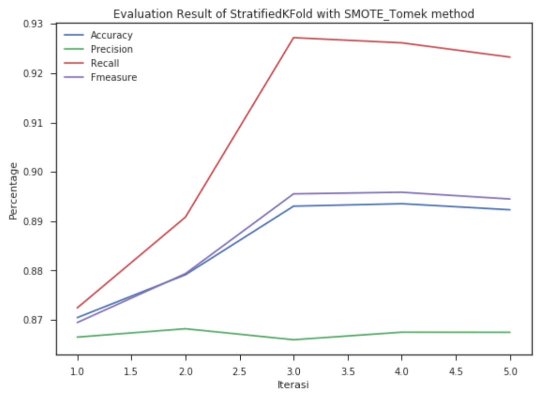
```
Classify and sampling with method SMOTE :

Original shape = (49440, 9)
Counter({0: 24720, 1: 24720})
Counter({0: 19776, 1: 19776})
Accuracy =  0.8791262135922329
Precision =  0.8627030500262272
Recall =  0.9016990291262136
Fmeasure =  0.881565320480498

Classify and sampling with method ADASYN :

Original shape = (48928, 9)
Counter({0: 24720, 1: 24208})
Counter({0: 19776, 1: 19366})
Accuracy =  0.8488794175947808
Precision =  0.8303458867459842
Recall =  0.8726843742536834
Fmeasure =  0.8507110563206499

Classify and sampling with method BorderLineSMOTE :

Original shape = (49440, 9)
Counter({0: 24720, 1: 24720})
Counter({0: 19776, 1: 19776})
Accuracy =  0.8795105177993527
Precision =  0.8498984489159623
Recall =  0.9215210355987054
Fmeasure =  0.8839226470113845

Classify and sampling with method SMOTE_Tomek :

Original shape = (48898, 9)
Counter({0: 24720, 1: 24178})
Counter({0: 19776, 1: 19342})
Accuracy =  0.8857011424882542
Precision =  0.8671270371503207
Recall =  0.9079755761468409
Fmeasure =  0.8869452836279926

Classify and sampling with method ADASYN_Tomek :

Original shape = (48498, 9)
Counter({0: 24290, 1: 24208})
Counter({0: 19432, 1: 19366})
Accuracy =  0.8590246884841992
Precision =  0.8433856712874427
Recall =  0.8810293052228995
Fmeasure =  0.861556045193416

Classify and sampling with method BorderLineSMOTE_Tomek :

Original shape = (48778, 9)
Counter({0: 24720, 1: 24058})
Counter({0: 19776, 1: 19246})
Accuracy =  0.8872856930004461
Precision =  0.8552505607917109
Recall =  0.9285078977882669
Fmeasure =  0.8901544032911133
```

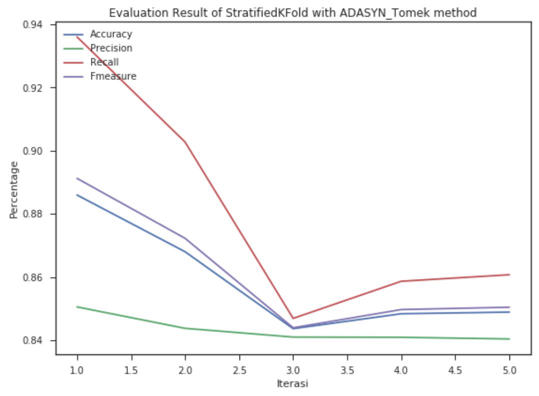

Graphic Evaluation of different sampling method
Accuracy
Precision
Recall
Fmeasure
Percentage
Sampling Method
SMOTE
ADASYN
BorderLineSMOTE
SMOTE_Tomek
ADASYN_Tomek
BorderLineSMOTE_Tomek

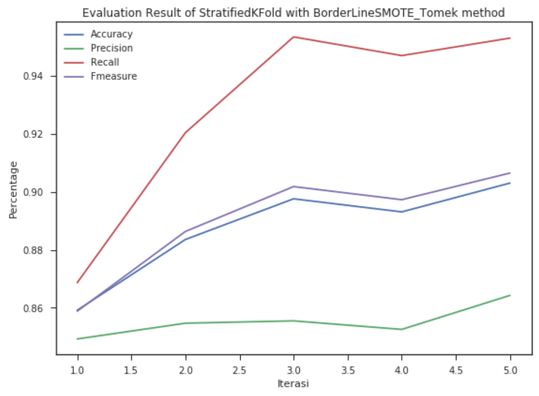

Evaluation Result of StratifiedKFold with SMOTE method
Accuracy
Precision
Recall
Fmeasure
Percentage
Iterasi


Evaluation Result of StratifiedKFold with ADASYN method
Accuracy
Precision
Recall
Fmeasure
Percentage
Iterasi


Evaluation Result of StratifiedKFold with BorderLineSMOTE method
Accuracy
Precision
Recall
Fmeasure
Percentage
Iterasi


Evaluation Result of StratifiedKFold with SMOTE_Tomek method
Accuracy
Precision
Recall
Fmeasure
Percentage
Iterasi


Evaluation Result of StratifiedKFold with ADASYN_Tomek method
Accuracy
Precision
Recall
Fmeasure
Percentage
Iterasi


Evaluation Result of StratifiedKFold with BorderLineSMOTE_Tomek method
Accuracy
Precision
Recall
Fmeasure
Percentage
Iterasi