LAPORAN TUGAS 2

Implementasi Oversampling dan Undersampling Methods pada Imbalance Datasets

Kelompok 6

Muhammad Adib Arinanda – 5115100111

Wahyu Ivan Satyagraha – 5115100155

Alfindio Muhammad Abdallah – 5115100164

Library yang digunakan :

1. Pandas
2. Numpy
3. Scikit-learn
4. Matplotlib

Datasets yang digunakan :

PubChem Bioassay Data Dataset

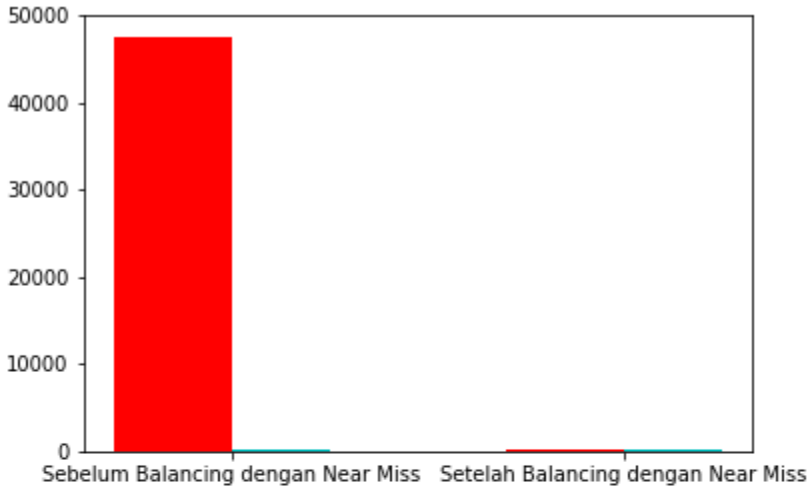
STEP :

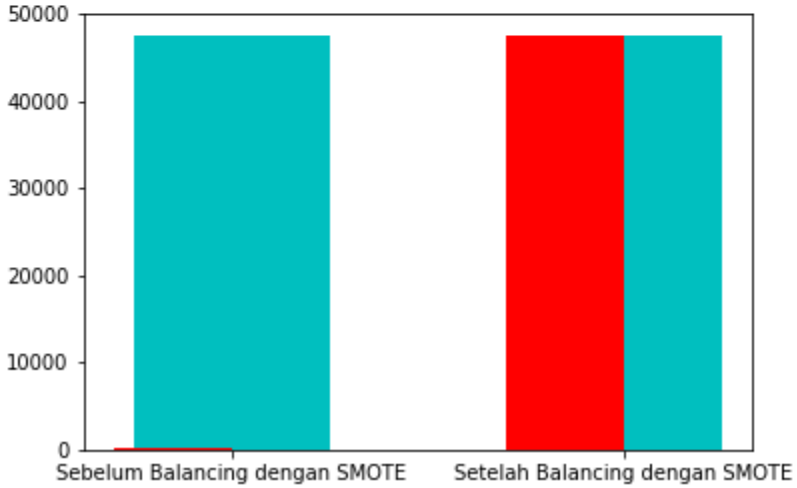
1. Load dataset.
2. Memisahkan atribut kelas dengan non kelas.
3. Melakukan balancing data dengan metode oversampling (SMOTE) dan undersampling (Near Miss).
4. Menghitung akurasi.
5. Melakukan Feature Reduction dengan PCA.

HASIL BALANCING DATA DAN AKURASI :

```
Under-sampling using Near Miss
==============================
Original dataset shape Counter({'Inactive': 47538, 'Active': 293})
Accuracy value knn dengan data Imbalance: 0.99 (+/- 0.00)
Resampled dataset shape Counter({'Active': 293, 'Inactive': 293})
================================================================
Accuracy value knn untuk Near Miss: 0.81 (+/- 0.12)
Over-sampling using SMOTE
=========================
Original dataset shape Counter({'Inactive': 47538, 'Active': 293})
Resampled dataset shape Counter({'Active': 47538, 'Inactive': 47538})
================================================================
Accuracy value knn untuk SMOTE: 0.91 (+/- 0.02)
```

PLOT :





CODE :

```
import matplotlib.pyplot as plt
from collections import Counter
from imblearn.under_sampling import NearMiss
from imblearn.over_sampling import SMOTE
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.model_selection import cross_val_score
from sklearn import neighbors

data = pd.read_csv('dataset.csv')

knn = neighbors.KNeighborsClassifier(n_neighbors=5, algorithm='auto')

X = data.drop('class', axis=1)
y = data['class']

besar = data.groupby(['class']).size()
besar = list(besar)

koor_x = ['Setelah Balancing dengan Near Miss','Sebelum Balancing dengan Near Miss']
koor_y = besar

besar_smote = data.groupby(['class']).size()
besar_smote = list(besar_smote)

koor_x_smote = ['Sebelum Balancing dengan SMOTE','Setelah Balancing dengan SMOTE']
koor_y_smote = besar_smote

pca = PCA(n_components=2)
X_vis = pca.fit_transform(X)

print('Under-sampling using Near Miss')
print('==============================')
print('Original dataset shape {}'.format(Counter(y)))
#Original dataset shape Counter({1: 900, 0: 100})

nm = NearMiss(random_state=42)
X_under_resampled, y_under_resampled = nm.fit_sample(X, y)
X_und_res_vis = pca.transform(X_under_resampled)

y_res = list(y_under_resampled)

valp = y_res.count('Inactive')
valn = y_res.count('Active')

new_y = []
new_y.append(valp)
new_y.append(valn)

print('Resampled dataset shape {}'.format(Counter(y_under_resampled)))
print('============================================================')
```

```
xnya = pd.DataFrame(X_under_resampled)
ynya = pd.DataFrame(y_under_resampled)
ynya.columns = ['class']

scores3 = cross_val_score(knn,xnya, ynya['class'], cv=10)
print("Accuracy value knn untuk Near Miss: %0.2f (+/- %0.2f)" % (scores3.mean(), scores3.std() * 2))

print('Over-sampling using SMOTE')
print('=========================')
print('Original dataset shape {}'.format(Counter(y)))

sm = SMOTE(random_state=42)
X_over_resampled, y_over_resampled = sm.fit_sample(X, y)
X_ov_res_vis = pca.transform(X_over_resampled)

y_res_smote = list(y_over_resampled)

valp_smote = y_res_smote.count('Inactive')
valn_smote = y_res_smote.count('Active')

new_y_smote = []
new_y_smote.append(valp_smote)
new_y_smote.append(valn_smote)

print('Resampled dataset shape {}'.format(Counter(y_over_resampled)))
print('================================================================')

xnya1 = pd.DataFrame(X_over_resampled)
ynya1 = pd.DataFrame(y_over_resampled)
ynya1.columns = ['class']

scores3 = cross_val_score(knn,xnya1, ynya1['class'], cv=10)
print("Accuracy value knn untuk SMOTE: %0.2f (+/- %0.2f)" % (scores3.mean(), scores3.std() * 2))

plt.figure(1)
plt.bar(koor_x, new_y, label = 'after Near Miss', color='c', width= 0.5, align = 'center')
plt.bar(koor_x, koor_y, label = 'before Near Miss', color='r', width= -0.3, align = 'edge')

plt.figure(2)
plt.bar(koor_x_smote, new_y_smote, label = 'after SMOTE', color='c', width= 0.5, align = 'center')
plt.bar(koor_x_smote, koor_y_smote, label = 'before SMOTE', color='r', width= -0.3, align = 'edge')

```

KESIMPULAN

Dalam menentukan sebuah akurasi data mining, metode balancing data sangat diperlukan untuk ke akuratan hasil yang di peroleh, pada data set ini, oversampling memiliki akurasi yang lebih tinggi namun dengan compile time yang lebih lama disbanding undersampling karena data yang di olah menjadi lebih banyak. Pada data imbalance akurasi yang didapat lebih tinggi namun dalam keadaan imbalance, hasil dari akurasi tersebut masih harus dipertanyakan.