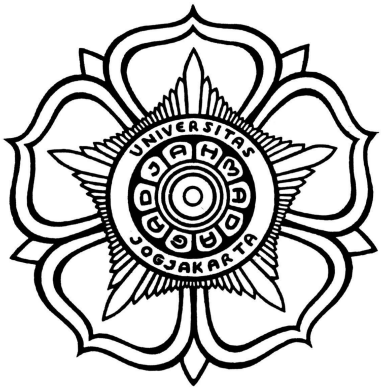
LAPORAN TUGAS BIOINFORMATIKA

IMPLEMENTASI SUPPORT VECTOR MACHINE UNTUK PREDIKSI STRUKTUR SEKUNDER PROTEIN



HERY CIAPUTRA

15/385649/PA/17034

PROGRAM STUDI S1 ILMU KOMPUTER

DEPARTEMEN ILMU KOMPUTER DAN ELEKTRONIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS GADJAH MADA

YOGYAKARTA

2019

Analisis Data

Data adalah kumpulan 126 sekuens primer dan sekunder protein yang berkorespondensi secara berturut-turut tiap baris oleh Rost dan Sander. Dataset kemudian dinamakan RS126. Contoh tampilan dari data:

```
TQSHYGQCGGIGYSGPTVCASGTTCQVLNPYYSQCL
CCCCCCEEECCCCCCCCCCCCCCCEEEEECCEEEEC
LKCNKLIPIAYKTCPEGKNLCYKMMLASKKMVPVKRGCINVCPKNSALVKYVCCSTDRCN
CEEECCCCCCCEEECCCCCCEEEEEEECCCCCEEEEEEEECCCCCCCCCCEEEEEECCCCCC
TTCCPSIVARSNFNVCRLPGTPEAICATYTGCIIIPGATCPGDYAN
CEECCCHHHHHHHHHHHHCCCCCHHHHHHHHHCCEECCCCCCCHHHCC
KSFPEVVGKTVDQAREYFTLHYPQYNVYFLPEGSPVTLDLRYNRVRVFYNPGTNVVNHVPHVG
CECHHHCCCEHHHHHHHHHHHHCCCCEEEEEECCCCEECCCCCCEEEEEEECCCCEECCCCEEC
AAPCFCSGKPGRGDLWILRGTCPGGYGYTSNCYKWPNICCYPH
CCCCCCCCCCCCCEEECCCCCCCCCCCCCCCEEEECCEEEECCC
```

Membaca data dan membuat list untuk data

```
lineList = [line.rstrip('\n') for line in open('RS126.data')]
```

Membagi data ke list primer dan sekunder

```
prm,skn = lineList[::2], lineList[1::2]
```

Mencari dan menghapus sekuens yang tidak memiliki panjang yang sama antara primer dan sekundernya

```
for i in range(len(prm)-1):
        if (len(prm[i]) != len(skn[i])):
        prm.pop(i)
        skn.pop(i)
```

Memisahkan tiap asam amino menjadi elemen sendiri

```
def split(word):
        return [char for char in word]

prm2 = []
skn2 = []
for i in range(len(prm)-1):
        prm2.append(split(prm[i]))
        skn2.append(split(skn[i]))
```

Menggabungkan tiap baris

```
import numpy as np
prm2 = np.concatenate(prm2)
```

```
skn2 = np.concatenate(skn2)
```

Feature Embedding

Interpretasi data kemudian dilakukan dengan mengubah tiap asam amino primer menjadi One Hot Encoding dan sekunder menjadi Integer Encoding

```
from numpy import array
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

val1 = array(prm2)
val2 = array(skn2)
le = LabelEncoder()
ie = le.fit_transform(val1)
skn3 = le.fit_transform(val2)
ohe = OneHotEncoder(sparse=False)
ie = ie.reshape(len(ie), 1)
prm3 = ohe.fit_transform(ie)
```

Contoh tampilan list primer menjadi

```
array([[1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.]])
```

Contoh tampilan list sekunder menjadi

```
array([0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0])
```

Penerapan SVM

Penerapan SVM menggunakan library scikit-learn

```
from sklearn.model_selection import train_test_split
from sklearn import svm

X_train, X_test, y_train, y_test = train_test_split(prm3, skn3, test_size = 0.20)
svc = SVC()
svc.fit(X_train, y_train)
```

Hasil

```
                precision    recall  f1-score   support

             0       0.52      0.69      0.59      1971
             1       0.39      0.22      0.28      1036
             2       0.45      0.40      0.42      1478

      accuracy                           0.48      4485
     macro avg       0.45      0.43      0.43      4485
  weighted avg       0.47      0.48      0.46      4485
```

Digunakan GridSearch untuk melakukan parameter tuning

```
from sklearn.model_selection import GridSearchCV

param_grid = {'C': [0.1, 1, 10, 100, 1000],
          'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
          'kernel': ['rbf']}

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)

grid.fit(X_train, y_train)
print(grid.best_params_)
print(grid.best_estimator_)
```

Dan didapatkan hasil

```
{'C': 0.1, 'gamma': 1, 'kernel': 'rbf'}
SVC(C=0.1, cache_size=200, class_weight=None, coef0=0.0, decision_function_shape='ovr',
degree=3, gamma=1, kernel='rbf', max_iter=-1, probability=False, random_state=None,
shrinking=True, tol=0.001, verbose=False)
```

Hasil yang didapatkan tidak berbeda dengan tanpa GridSearch yaitu

```
                precision    recall  f1-score   support

             0       0.52      0.69      0.59      1971
             1       0.39      0.22      0.28      1036
             2       0.45      0.40      0.42      1478

      accuracy                           0.48      4485
     macro avg       0.45      0.43      0.43      4485
  weighted avg       0.47      0.48      0.46      4485
```