Hasil Observasi

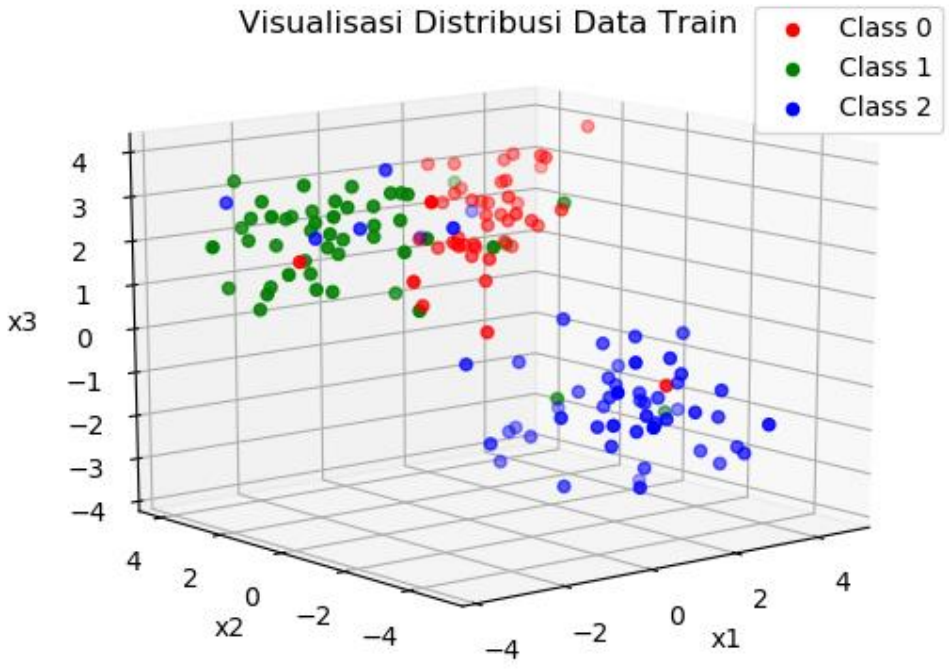
Nama : Naufal Ihsan Kusumayadhi

Kelas : IF-39-01

NIM : 1301154409

Bagian A

1.



Grafik diatas merupakan visualisasi distribusi data train.

2.

```
def patternLayerKFold(testrow,trainsheet,start,end) : # mencari hasil fungsi g(x) dari sebuah row data terhadap seluruh row dari sebuah sheet dimulai dari baris ke start berakhir di baris ke end
    smooth = pow(1.5,2) # nilai smoothing setelah dikuadratkan
    i = start
    gx = []
    while (i <= end) :
        pangkat = -1*(pow(testrow[0] - trainsheet.cell(i,0).value,2) + pow(testrow[1] - trainsheet.cell(i,1).value,2) + pow(testrow[2] - trainsheet.cell(i,2).value,2))/(2*smooth) # isi pangkat dari exponen rumus gaussian
        gx.append(np.exp(pangkat)) # memasukkan hasil perhitungan terhadap tiap baris data train menjadi g(x)
        i += 1
    return gx

def patternLayerKFoldsbg(testrow,trainsheet,start1,end1,start2,end2) : # mencari hasil fungsi g(x) dari sebuah row data terhadap seluruh row dari sebuah sheet tetapi loncat, dimulai dari baris ke start1 berakhir di baris ke end1, kemudian
    smooth = pow(1.5,2) # nilai smoothing setelah dikuadratkan
    i = start1
    gx = []
    while (i <= end1) :
        pangkat = -1*(pow(testrow[0] - trainsheet.cell(i,0).value,2) + pow(testrow[1] - trainsheet.cell(i,1).value,2) + pow(testrow[2] - trainsheet.cell(i,2).value,2))/(2*smooth) # isi pangkat dari exponen rumus gaussian
        gx.append(np.exp(pangkat)) # memasukkan hasil perhitungan terhadap tiap baris data train menjadi g(x)
        i += 1
    i = start2
    while (i <= end2) :
        pangkat = -1*(pow(testrow[0] - trainsheet.cell(i,0).value,2) + pow(testrow[1] - trainsheet.cell(i,1).value,2) + pow(testrow[2] - trainsheet.cell(i,2).value,2))/(2*smooth) # isi pangkat dari exponen rumus gaussian
        gx.append(np.exp(pangkat)) # memasukkan hasil perhitungan terhadap tiap baris data train menjadi g(x)
        i += 1
    return gx
```

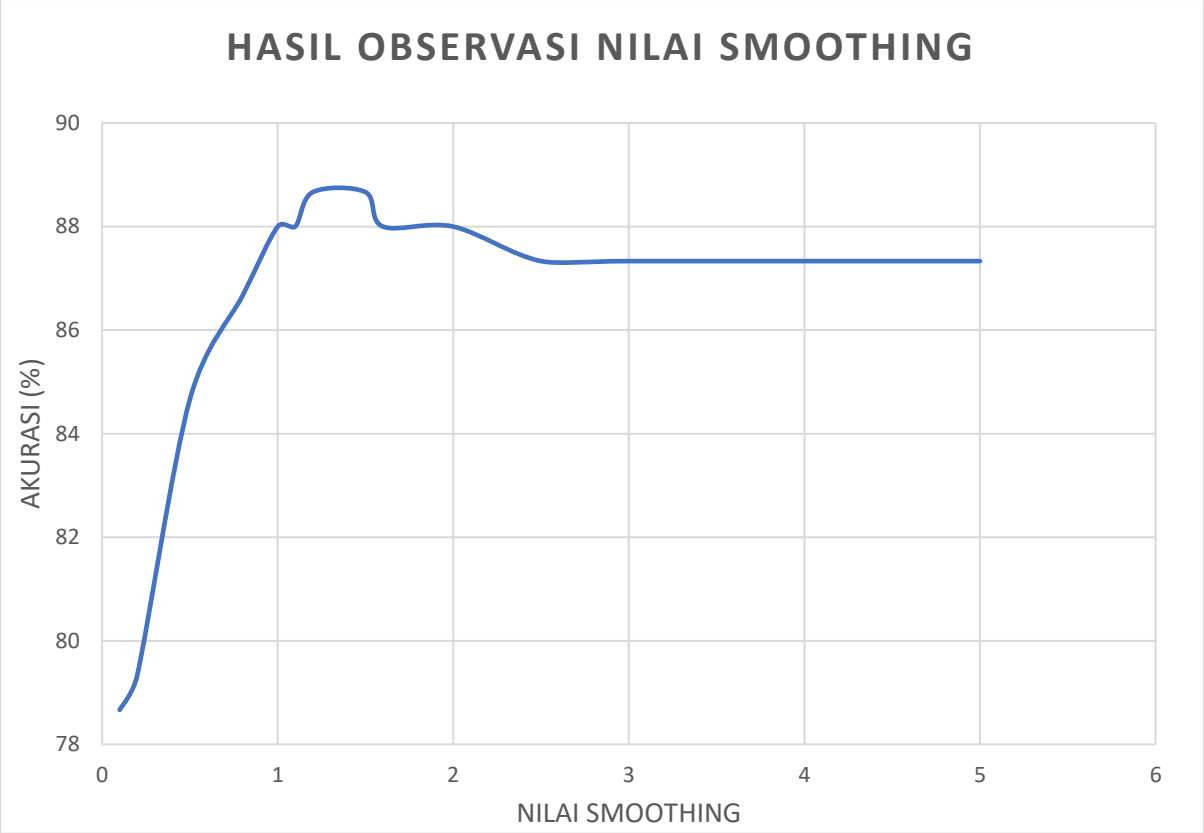
```
def sumLayerKFold(gx, sheet, start, end) : # menjumlahkan dan merata-ratakan nilai g(x) di setiap classnya
    sum1 = 0
    sum2 = 0
    sum3 = 0
    jml1 = 0
    jml2 = 0
    jml3 = 0
    i = start
    j = 0
    while (i <= end) : # memilah penjumlahan tergantung dari class y data trainnya
        if (sheet.row_values(i)[3] == 0) :
            sum1 += gx[j]
            jml1 += 1
        elif (sheet.row_values(i)[3] == 1.0) :
            sum2 += gx[j]
            jml2 += 1
        elif (sheet.row_values(i)[3] == 2.0) :
            sum3 += gx[j]
            jml3 += 1
        i += 1
        j += 1
    i = 0
    sum1 = sum1/jml1
    sum2 = sum2/jml2
    sum3 = sum3/jml3
    return sum1,sum2,sum3
```

```python
def sumLayerKFoldsbg(gx, sheet, start1, end1, start2, end2) : # menjumlahkan dan merata-ratakan nilai g(x) di setiap classnya, tetapi loncat
    sum1 = 0
    sum2 = 0
    sum3 = 0
    jml1 = 0
    jml2 = 0
    jml3 = 0
    i = start1
    j = 0
    while (i <= end1) : # memilah penjumlahan tergantung dari class y data trainnya
        if (sheet.row_values(i)[3] == 0) :
            sum1 += gx[j]
            jml1 += 1
        elif (sheet.row_values(i)[3] == 1.0) :
            sum2 += gx[j]
            jml2 += 1
        elif (sheet.row_values(i)[3] == 2.0) :
            sum3 += gx[j]
            jml3 += 1
        i += 1
        j += 1
    i = start2
    while (i <= end2) : # memilah penjumlahan tergantung dari class y data trainnya
        if (sheet.row_values(i)[3] == 0) :
            sum1 += gx[j]
            jml1 += 1
        elif (sheet.row_values(i)[3] == 1.0) :
            sum2 += gx[j]
            jml2 += 1
        elif (sheet.row_values(i)[3] == 2.0) :
            sum3 += gx[j]
            jml3 += 1
        i += 1
        j += 1
    sum1 = sum1/jml1
    sum2 = sum2/jml2
    sum3 = sum3/jml3
    return sum1,sum2,sum3
```

```python
def outputLayer(sum1, sum2, sum3) : # mencari nilai sum mana yang lebih besar dari setiap classnya untuk menentukan prediksi class di row tersebut
    if ((sum1 > sum2) and (sum1 > sum3)) :
        return(0.0)
    elif ((sum2 > sum1) and (sum2 > sum3)) :
        return(1.0)
    elif ((sum3 > sum1) and (sum3 > sum2)) :
        return(2.0)
```

3. Observasi yang telah dilakukan adalah mencari nilai smoothing yang tepat untuk data train. Metode yang digunakan untuk mancari perkiraan akurasi adalah dengan menggunakan K-Fold Cross Validation pada data train. Data train dibagi menjadi 3 bagian yang akan secara bergantian menjadi data train dan data validation. Setelah dibangun fungsi-fungsi PNN, maka dilakukan observasi terhadap nilai smoothing. Didapatkan data sebagai berikut :

Nilai smoothing	Perkiraan akurasi dari data train
0.1	78.66666667
0.2	79.33333333
0.5	84.66666667
0.8	86.66666667
1	88
1.1	88
1.2	88.66666667
1.5	88.66666667
1.6	88
2	88
2.5	87.33333333
3	87.33333333
5	87.33333333



Akurasi maksimal didapatkan ketika nilai smoothingnya 1,2-1,5. Akurasi yang didapat sebesar 88,666667%.