Nama : Talitha Farah K

Kelas : IF-40-02

Nim : 1301164262

Laporan Tugas 1 Machine Learning

1. **Deskripsi Masalah**
   Diberikan sebuah Trainset berupa himpunan data berisi 160 objek data yang memiliki 7 atribut input (age, workclass, education, marital-status, occupation, relationship, hours-per-week) dan 1 output (label kelas income) yang memiliki 4 kelas/label (0, 1, 2, dan 3). Bangunlah sebuah sistem klasifikasi menggunakan metode Naïve Bayes untuk menentukan kelas/label data testing dalam Testset. Sistem membaca masukan file TrainsetTugas1ML.csv dan TestsetTugas1ML.csv dan mengeluarkan output berupa file TebakanTugas1ML.csv berupa satu kolom berisi 40 baris yang menyatakan kelas/label baris yang bersesuaian pada file TestsetTugas1ML.csv.
2. Analisis dan Penyelesaian Masalah
   Naive Bayes merupakan sebuah metoda klasifikasi menggunakan metode probabilitas dan statistik dengan cara memprediksi peluang di **Data Test** berdasarkan pengalaman di **Data Train**. Tahapan dalam Algoritma ini yaitu :
   a. Load Data Train
   b. Menghitung Probabilitas Keseluruhan

```
lebihdari = trainset[trainset[8]=='>50K']
kurangdari = trainset[trainset[8]=='<=50K']

prob_lebih = len(lebihdari) / len(trainset)
prob_kurang = len(kurangdari) / len(trainset)
# print(prob_lebih)
# print(prob_kurang)
```

   Data Train dibagi menjadi 2 sesuai dengan income nya yaitu “>50K” dan “<=50K”. Variabel prob_lebih menampung probabilitas keseluruhan dengan income >50K, didapat dari jumlah data dengan income >50K dibagi dengan total data Train. Variabel prob_kurang menampung probabilitas keseluruhan dengan income <=50K, didapat dari jumlah data dengan income >50K dibagi dengan total data Train.
   c. Menghitung Probabilitas Kasus Per Kelas

```
def tabel(data):
    hasil = []
    for i in range(1,8):
        state = data.groupby(i)
        num = state.size()
        prob = np.asarray(num.apply(lambda jumlahdata : jumlahdata/len(data)))
        hasil.append(prob)
    return hasil

a = pd.DataFrame(np.asarray(tabel(lebihdari)))
b = pd.DataFrame(np.asarray(tabel(kurangdari)))
```

   fungsi tabel berisikan cara menghitung probabilitas data perkelas. data dihitung berdasakan header nya. Kemudian dimasukkan rumus jumlah data/ panjang data menurut kelas nya. Hasil kemudian dimasukkan ke array. Fungsi ini akan menghasilkan probabilitas dari data kurang dari maupun lebih dari.
   d. Load Data Test
   e. Dictionary Probabilitas

```
dict_prob = {
'>50K':{ 'adult':a[0][0], 'old':a[1][0], 'young':a[2][0],
'Local-gov':a[0][1],'Private':a[1][1], 'Self-emp-not-inc':a[2][1],
'Bachelors':a[0][2],'HS-grad':a[1][2], 'Some-college':a[2][2],
'Divorced':a[0][3],'Married-civ-spouse':a[1][3], 'Never-married':a[2][3],
'Craft-repair':a[0][4], 'Exec-managerial':a[1][4], 'Prof-specialty':a[2][4],
'Husband':a[0][5],'Not-in-family':a[1][5],'Own-child':a[2][5],
'low':a[0][6],'many':a[1][6],'normal':a[2][6]
},
'<=50K':{ 'adult':b[0][0], 'old':b[1][0], 'young':b[2][0],
'Local-gov':b[0][1], 'Private':b[1][1],'Self-emp-not-inc':b[2][1],
'Bachelors':b[0][2],'HS-grad':b[1][2], 'Some-college':b[2][2],
'Divorced':b[0][3],'Married-civ-spouse':b[1][3], 'Never-married':b[2][3],
'Craft-repair':b[0][4], 'Exec-managerial':b[1][4], 'Prof-specialty':b[2][4],
'Husband':b[0][5],'Not-in-family':b[1][5],'Own-child':b[2][5],
'low':b[0][6],'many':b[1][6],'normal':b[2][6]
}}
```

Dictionary ini berisikan probabilitas tiap tiap atribut yang telah dihitung dari fungsi sebelumnya. Dictionary ini berfungsi untuk membandingkan data dari data tes untuk dicari propabilitasnya.

f. Mencari Probabilitas dari Data Tes

```
result = []
for i in range(1,len(testSet)):
    if (testSet[1][0] == 'age'):
        if (testSet[1][i] == 'young'):
            age_a = dict_prob['>50K']['young']
            age_b = dict_prob['<=50K']['young']
        elif(testSet[1][i] == 'adult'):
            age_a = dict_prob['>50K']['adult']
            age_b = dict_prob['<=50K']['adult']
        else:
            age_a = dict_prob['>50K']['old']
            age_b = dict_prob['<=50K']['old']
    if (testSet[2][0] == 'workclass'):
        if (testSet[2][i] == 'Local-gov'):
            work_a = dict_prob['>50K']['Local-gov']
            work_b = dict_prob['<=50K']['Local-gov']
        elif(testSet[2][i] == 'Private'):
            work_a = dict_prob['>50K']['Private']
            work_b = dict_prob['<=50K']['Private']
        else:
            work_a = dict_prob['>50K']['Self-emp-not-inc']
            work_b = dict_prob['<=50K']['Self-emp-not-inc']
    if (testSet[3][0] == 'education'):
        if (testSet[3][i] == 'Bachelors'):
            edu_a = dict_prob['>50K']['Bachelors']
            edu_b = dict_prob['<=50K']['Bachelors']
        elif(testSet[3][i] == 'HS-grad'):
            edu_a = dict_prob['>50K']['HS-grad']
            edu_b = dict_prob['<=50K']['HS-grad']
```

Result berfungsi untuk menampung hasi akhir dari hasil perhitungan. Data Test di load dan ditentukan probabiliasnya. Di cek terus dari atribut ke 1 hingga atribut ke 6. Jika sudah, semua atribut dikalikan dan dikalikan juga dengan probabilitas keseluruhan nya. Hasil kemudian ditampung di variabel hasil_lebihdari dan hasil_kurangdari

```
            hours_b = dict_prob['<=50K']['normal']
    hasil_lebihdari = age_a * work_a * edu_a * marital_a * ocu_a * rela_a * hours_a * prob_lebih
    hasil_kurangdari = age_b * work_b * edu_b * marital_b * ocu_b * rela_b * hours_b * prob_kurang
    # print(i, hasil_lebihdari)
```

g. Bandingkan Hasil

```
if (hasil_lebihdari > hasil_kurangdari):
    result.append(">50K")
else:
    result.append("<=50K")
```

Hasil dari keduanya dibandingkan siapa yang lebih besar. Jika hasil_lebihdari lebih besar dari hasil_kurangdari maka akan dimasukan “>50K” jika tidak “<=50K”

h. Export Hasil

Hasil kemudian di Export ke TebakanTugas1ML.csv

```
array_result = np.asarray(result)
with open('TebakanTugas1ML.csv', 'w') as fp:
    a = csv.writer(fp,quoting=csv.QUOTE_ALL)
    for word in array_result:
        a.writerow([word])
```