LAPORAN TUGAS BESAR PEMBELAJARAN MESIN

TAHAP 2 : CLASSIFICATION (SUPERVISED LEARNING)

Disusun oleh :

Kelompok 2

Daniel Septyadi (1301180009)

Gerald Ergi Bhaskara Natajaya (1301180133)

PEMBELAJARAN MESIN

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

TAHUN AKADEMIK 2021/2022

A. Permasalahan

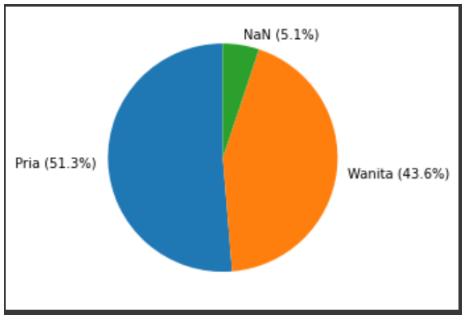
Terdapat data customer pada pembelian kendaraan. Data berisi karakteristik customer yang berkaitan dengan ketertarikan customer untuk membeli kendaraan. Kami diminta untuk melakukan eksplorasi data dan juga membuat model data dengan menggunakan machine learning untuk mengetahui karakteristik customer apakah tertarik atau tidak membeli mobil baru.

B. Eksplorasi dan Persiapan data

#Persentasi data jenis kelamin customer

Kami melakukan presentasi data jenis kelamin costumer dengan codenya sebagai berikut.

```
a = round((len(dfTrain[dfTrain['Jenis_Kelamin']=='Pria'])/len(dfTrain))*100,1)
b = round((len(dfTrain[dfTrain['Jenis_Kelamin']=='Wanita'])/len(dfTrain))*100,1)
c = round((len(dfTrain[(dfTrain['Jenis_Kelamin']!='Pria')&(dfTrain['Jenis_Kelamin']!='Wanita')])/len(dfTrain))*100,1)
y = np.array([a,b,c])
mylabels = ["Pria ({}%)".format(a), "Wanita ({}%)".format(b), "NaN ({}%)".format(c)]

plt.pie(y, labels = mylabels, startangle = 90)

plt.show()
```



Bisa dilihat terdapat 51,3% pria dalam keseluruhan data dan 43,6% Wanita dan juga terdapat 5,1% data blank atau data kosong yang artinya tidak ada keterangan bahwa dia Pria/Wanita.

```
TWanita = len(dfTrain[(dfTrain['Tertarik']==1)&(dfTrain['Jenis_Kelamin']=='Wanita')])/len(dfTrain[dfTrain['Jenis_Kelamin']=='Wanita'])
print('Jumlah Wanita yang tertarik beli mobil baru, yaitu {}% dari jumlah pembeli wanita'.format(round(TWanita*100,1)))

TPria = len(dfTrain[(dfTrain['Tertarik']==1)&(dfTrain['Jenis_Kelamin']=='Pria')])/len(dfTrain[dfTrain['Jenis_Kelamin']=='Pria'])
print('Jumlah Pria yang tertarik beli mobil baru, yaitu {}% dari jumlah pembeli pria'.format(round(TPria*100,1)))
```

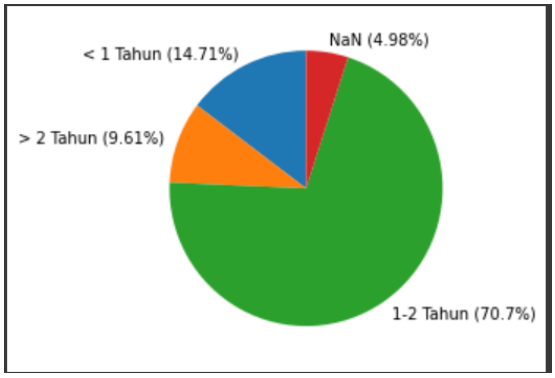
```
Jumlah Wanita yang tertarik beli mobil baru, yaitu 10.3% dari jumlah pembeli wanita
Jumlah Pria yang tertarik beli mobil baru, yaitu 13.8% dari jumlah pembeli pria
```

Lalu disini kami menemukan bahwa jumlah wanita tertarik membeli mobil baru ada

sekitar 10,3% dari seluruh pembeli Wanita, dan juga kami menemukan bahwa jumlah pria yang tertarik membeli mobil baru ada 13,8% dari jumlah seluruh pembeli Pria. Jadi bisa disimpulkan bahwa jumlah pria itu lebih dominan banyak dibandingkan dengan Wanita.

#Persentase ketertarikan pembelian menurut umur kendaraan

```
a = round((len(dfTrain[(dfTrain['Umur_Kendaraan']=='< 1 Tahun')&(dfTrain['Tertarik']==1)])/len(dfTrain[dfTrain['Tertarik']==1]))*100,2)
b = round((len(dfTrain[(dfTrain['Umur_Kendaraan']=='> 2 Tahun')&(dfTrain['Tertarik']==1)])/len(dfTrain[dfTrain['Tertarik']==1]))*100,2)
c = round((len(dfTrain[(dfTrain['Umur_Kendaraan']=='1-2 Tahun')&(dfTrain['Tertarik']==1)])/len(dfTrain[dfTrain['Tertarik']==1]))*100,2)
d = round((len(dfTrain[(dfTrain['Umur_Kendaraan']!='< 1 Tahun')&(dfTrain['Umur_Kendaraan']!='> 2 Tahun')&(dfTrain['Umur_Kendaraan']!='1-2 Tahun')&(dfTrain['Tertarik']==
y = np.array([a,b,c,d])
mylabels = ["< 1 Tahun ({}%)".format(a), "> 2 Tahun ({}%)".format(b), "1-2 Tahun ({}%)".format(c), "NaN ({}%)".format(d)]

plt.pie(y, labels = mylabels, startangle = 90)

plt.show()
```



Bisa kita lihat dari chart tsb bahwa mobil yang berumur 1-2 tahun peminatnya (70.7%) dan yang paling sedikit itu mobil yang berumur sudah tua atau > 2 tahun hanya sekitar (9,61%) dan juga ada sekitar (4,98%) data blank/ tidak ada keterangan.

#Jumlah null atau nan pada setiap column

```
dfTrain.isnull().sum()
```

```
id                      0
Jenis_Kelamin       14440
Umur                14214
SIM                 14404
Kode_Daerah         14306
Sudah_Asuransi      14229
Umur_Kendaraan      14275
Kendaraan_Rusak     14188
Premi               14569
Kanal_Penjualan     14299
Lama_Berlangganan   13992
Tertarik                0
dtype: int64
```

Dari hasil eksplorasi data di atas, dapat dilihat bahwa ada banyak data NaN pada setiap column. Oleh karena itu kita akan melakukan data cleaning.

#Data Cleaning & Preparation

```
# Melakukan standarisasi data, agar nilai antar column tidak jauh berbeda
dfTrain['Premi'] = dfTrain['Premi']/10000
dfTrain['Kanal_Penjualan'] = dfTrain['Kanal_Penjualan']/10
dfTrain['Lama_Berlangganan'] = dfTrain['Lama_Berlangganan']/10

dfTest['Premi'] = dfTest['Premi']/10000
dfTest['Kanal_Penjualan'] = dfTest['Kanal_Penjualan']/10
dfTest['Lama_Berlangganan'] = dfTest['Lama_Berlangganan']/10

# Menghapus row pada column-column selain Premi yang mempunya nilai Nan
a = dfTrain[dfTrain['Jenis_Kelamin'].notnull()]
b = a[a['Umur'].notnull()]
c = b[b['SIM'].notnull()]
d = c[c['Kode_Daerah'].notnull()]
e = d[d['Sudah_Asuransi'].notnull()]
f = e[e['Umur_Kendaraan'].notnull()]
g = f[f['Kendaraan_Rusak'].notnull()]
h = g[g['Kanal_Penjualan'].notnull()]
dfNew = h[h['Lama_Berlangganan'].notnull()]

# Mengisi nilai NaN pada coolumn Premi dengan rata-rata nilai Premi
dfNew = dfNew.fillna(dfNew['Premi'].mean())
```

Jadi disini kita melakukan standarisasi data data train dan test lalu melakukan pembagian data agar data satu sama lain tidak terlalu jauh bedanya, Karena jika nilai terlalu jauh bedannya akan terjadi overfitting (dataset terlalu jauh nilainya). Bisa kita lihat dibagian Premi,Kanal_Penjualan dan Lama Berlangganan data tersebut dibagi atau dikecilkan nilainya. Lalu kita akan memfilter semua data yang dimana kolom kolom tersebut tidak mempunyai nilai null atau data kosong selain Premi. Premi itu angsuran

awal berupa value(nilai) jadi bisa kita cari nilai rata-ratanya lalu isi nilainya dengan rata-rata tersebut.

#Mengubah semua string menjadi numerik agar dapat diolah ke dalam model

```
dfNew.loc[dfNew['Umur_Kendaraan']=='< 1 Tahun', 'Umur_Kendaraan'] = 1
dfNew.loc[dfNew['Umur_Kendaraan']=='> 2 Tahun', 'Umur_Kendaraan'] = 2
dfNew.loc[dfNew['Umur_Kendaraan']=='1-2 Tahun', 'Umur_Kendaraan'] = 3

dfNew.loc[dfNew['Kendaraan_Rusak']=='Pernah', 'Kendaraan_Rusak'] = 1
dfNew.loc[dfNew['Kendaraan_Rusak']=='Tidak', 'Kendaraan_Rusak'] = 0

dfNew.loc[dfNew['Jenis_Kelamin']=='Pria', 'Jenis_Kelamin'] = 1
dfNew.loc[dfNew['Jenis_Kelamin']=='Wanita', 'Jenis_Kelamin'] = 0

del dfNew['id']

dfNew = dfNew.astype(float)
dfNew = dfNew.reset_index(drop=True)
```

```
dfTest.loc[dfTest['Umur_Kendaraan']=='< 1 Tahun', 'Umur_Kendaraan'] = 1
dfTest.loc[dfTest['Umur_Kendaraan']=='> 2 Tahun', 'Umur_Kendaraan'] = 2
dfTest.loc[dfTest['Umur_Kendaraan']=='1-2 Tahun', 'Umur_Kendaraan'] = 3

dfTest.loc[dfTest['Kendaraan_Rusak']=='Pernah', 'Kendaraan_Rusak'] = 1
dfTest.loc[dfTest['Kendaraan_Rusak']=='Tidak', 'Kendaraan_Rusak'] = 0

dfTest.loc[dfTest['Jenis_Kelamin']=='Pria', 'Jenis_Kelamin'] = 1
dfTest.loc[dfTest['Jenis_Kelamin']=='Wanita', 'Jenis_Kelamin'] = 0

dfTest = dfTest.astype(float)
dfTest = dfTest.reset_index(drop=True)
```

Dapat dilihat bahwa data train dan test terdapat jenis kelamin dan umur kendaraan berupa string , disini kita ubah nilainya ke integer /float , agar bisa kita olah ke dalam machine learning

#Hasil akhir data train yang akan digunakan dalam model machine learning

```
[ ] dfNew.head()
```

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	0.0	30.0	1.0	33.0	1.0	1.0	0.0	2.8029	15.2	9.7	0.0
1	1.0	48.0	1.0	39.0	0.0	2.0	1.0	2.5800	2.9	15.8	0.0
2	0.0	58.0	1.0	48.0	0.0	3.0	0.0	0.2630	12.4	6.3	0.0
3	1.0	21.0	1.0	35.0	1.0	1.0	0.0	2.2735	15.2	17.1	0.0
4	0.0	20.0	1.0	8.0	1.0	1.0	0.0	3.0786	16.0	3.1	0.0

Bahwa seluruh nilai dari fitur-fitur diatas ini berupa float artinya desimal jadi bisa kita masukkan dalam modelling

C. **Modelling.**

#Memisahkan data. X merupakan fitur, dan Y merupakan labelnya

```
X_train = dfNew.drop('Tertarik', axis = 1)
Y_train = dfNew[['Tertarik']]

X_test = dfTest.drop('Tertarik', axis = 1)
Y_test = dfTest[['Tertarik']]
```

Disini kita akan melakukan persiapan data yaitu dengan memisahkan data , X merupakan fitur dan Y merupakan labelnya.

X = jenis,kelamin, Umur, SIM, Kode_Daerah, Sudah_asuransi, Umur_Kendaraan, Kendaraan_Rusak, Premi, Kanal_Penjualan, Lama_berlangganan

Y = Tertarik.

#mengubah dataframe/tabel menjadi sebuah array

```
[ ] X_train = X_train.values
    Y_train = Y_train.values

    X_test = X_test.values
    Y_test = Y_test.values


    X_train = X_train.T
    Y_train = Y_train.reshape(1, X_train.shape[1])

    X_test = X_test.T
    Y_test = Y_test.reshape(1, X_test.shape[1])

    Y_train = Y_train.astype(int)
```

X_Train dan Y_Train masih berupa tabel jadi disini kita akan mengekstrak nilai" kedalam bentuk array agar bisa proses kedalam machine learning model.

#Membangun Model

Logistic Regression adalah sebuah algoritma klasifikasi untuk mencari hubungan antara fitur (input) diskrit/kontinu dengan probabilitas hasil output diskrit tertentu menggunakan rumus sigmoid.

$$S(x) = \frac{1}{1 + e^{-x}}$$

$S(x)$ = sigmoid function

e = Euler's number

```
def sigmoid(x):
    return 1/(1 + np.exp(-x))
```

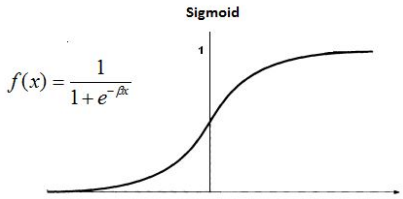
Sigmoid Function adalah suatu fungsi yang dibentuk dengan menyamakan nilai Y pada Linear Function dengan nilai Y pada Sigmoid Function. Tujuan dari Logistic Function adalah merepresentasikan data-data yang kita miliki kedalam bentuk fungsi Sigmoid.

```
def model(X, Y, learning_rate, iterations):

    m = X_train.shape[1]
    n = X_train.shape[0]

    W = np.zeros((n,1))
    B = 0

    cost_list = []

    for i in range(iterations):

        Z = np.dot(W.T, X) + B
        A = sigmoid(Z)

        # cost function
        cost = -(1/m)*np.sum( Y*np.log(A) + (1-Y)*np.log(1-A))

        # Gradient Descent
        dW = (1/m)*np.dot(A-Y, X.T)
        dB = (1/m)*np.sum(A - Y)

        W = W - learning_rate*dW.T
        B = B - learning_rate*dB

        # Keeping track of our cost function value
        cost_list.append(cost)

        if(i%(iterations/10) == 0):
            print("cost after ", i, "iteration is : ", cost)

    return W, B, cost_list
```

Disini kita akan melakukan test terhadap W dengan iterasi yang ditentukan sebagai berikut:

Inisialisasi :

m = jumlah baris
n = jumlah fitur
W = array kosong
Z = garis lurus gradien.
Z akan menghasilkan garis linier , ketika Z sudah mendapatkan garis linear kita akan memasukkan garis Z ini kedalam A = sigmoid(Z) , lalu A ini akan membentuk garis sigmoid pada contoh gambar ini.



Setelah mendapatkan garis sigmoid tersebut , lalu kita hitung errornya dari garis sigmoid dengan hasil label (Y_train) dan akan memperbaharui weight (W & B) untuk menghasilkan hasil yang akurat.

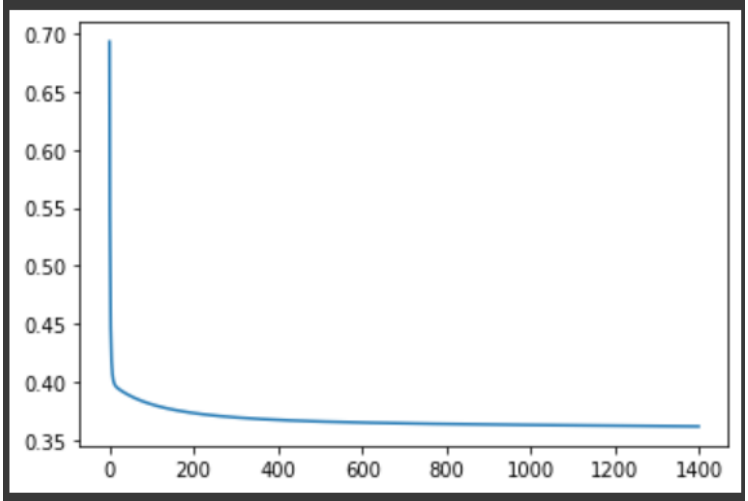
```
Accuracy of the model is :  78.59 %
78.59
```

akurasi model yang dihasilkan 78.59%

D. Evaluasi

```python
def accuracy(X, Y, W, B):
    Z = np.dot(W.T, X) + B
    A = sigmoid(Z)
    #A = A > 0.5
    #A = np.array(A, dtype = 'int64')
    acc = (1 - np.sum(np.absolute(A - Y))/Y.shape[1])*100
    print("Accuracy of the model is : ", round(acc, 2), "%")
    return round(acc, 2)
```

```
plt.plot(np.arange(iterations), cost_list)
plt.show()
```




```
accuracy(X_test, Y_test, W, B)
```

```
Accuracy of the model is :  78.59 %
78.59
```

Dalam metode evaluasi disini bisa kita lihat jika nilai costnya semakin kecil maka akan semakin bagus model yang sudah kita buat, jika dilihat pada gambar grafik diatas interasi yang sudah kita tentukan sudah mendapatkan cost yang paling kecil. Jadi dari cost function tersebut kita bisa tentukan iterasi ke berapa yang paling kecil.

E. Eksperimen

```
akurasi = []
iterasi = []
l_rate = []
div_loop = int(input("Jumlah Looping: ")) #Akan digunakan untuk experimen dengan mengalikan/membagi iterasi dan learning rate
iterations = int(input("Input Jumlah Perulangan awal yang akan dilakukan pada model: ")) #Jumlah iterasi awal yang digunakan oleh model
learning_rate = float(input("Input Besar Learning Rate awal yang akan digunakan pada model: ")) #Besar learning rate yang digunakan oleh model
for i in  range(div_loop):
    print("")
    print("Jumlah Iterasi yang digunakan untuk model adalah : "+str(iterations))
    print("Besar Learning Rate yang digunakan untuk model adalah : "+str(learning_rate))
    W, B, cost_list = model(X_train, Y_train, learning_rate = learning_rate, iterations = iterations)

    iterations = round(iterations*div_loop)
    learning_rate = learning_rate/div_loop
```

Di Bagian ini kita bisa melakukan percobaan eksperimen dengan memasukkan berapa kali yang mau kita ulang modelnya dengan nilai iterasi model dan learning rate yg berbeda. Disini saya mengambil eksperimen inputan :

Jumlah looping : 3

iterations : 1000

Besar Learning Rate awal : 0.01

```
Jumlah Looping: 3
Input Jumlah Perulangan awal yang akan dilakukan pada model: 1000
Input Besar Learning Rate awal yang akan digunakan pada model: 0.01

Jumlah Iterasi yang digunakan untuk model adalah : 1000
Besar Learning Rate yang digunakan untuk model adalah : 0.01
cost after  0 iteration is :  0.6931471805599453
cost after  100 iteration is :  0.3658309895929556
cost after  200 iteration is :  0.6945903180859861
cost after  300 iteration is :  0.4147305626840672
cost after  400 iteration is :  0.423065575699368
cost after  500 iteration is :  0.5295720554763708
cost after  600 iteration is :  0.41319101528011853
cost after  700 iteration is :  0.6555218000750741
cost after  800 iteration is :  0.6523091485474464
cost after  900 iteration is :  0.4093224601622041
Accuracy of the model is :  86.04 %
```

```
Jumlah Iterasi yang digunakan untuk model adalah : 3000
Besar Learning Rate yang digunakan untuk model adalah : 0.0033333333333333335
cost after  0 iteration is :  0.6931471805599453
cost after  300 iteration is :  0.35995011101412877
cost after  600 iteration is :  0.35473912497003796
cost after  900 iteration is :  0.3500342632826715
cost after  1200 iteration is :  0.3457691397947652
cost after  1500 iteration is :  0.34189184536417866
cost after  1800 iteration is :  0.33835897863230946
cost after  2100 iteration is :  0.3351332863442246
cost after  2400 iteration is :  0.3321822889189126
cost after  2700 iteration is :  0.3294774145990739
Accuracy of the model is :  79.59 %
```

```
Jumlah Iterasi yang digunakan untuk model adalah : 9000
Besar Learning Rate yang digunakan untuk model adalah : 0.0011111111111111111
cost after  0 iteration is :  0.6931471805599453
cost after  900 iteration is :  0.3599500918968823
cost after  1800 iteration is :  0.35473953674689307
cost after  2700 iteration is :  0.35003494859047796
cost after  3600 iteration is :  0.34576998411554244
cost after  4500 iteration is :  0.3418927822716722
cost after  5400 iteration is :  0.3383599673139448
cost after  6300 iteration is :  0.3351343002881714
cost after  7200 iteration is :  0.3321833100679217
cost after  8100 iteration is :  0.32947843022513973
Accuracy of the model is :  79.59 %
```

Dengan memasukkan berapa kali mau kita ulang modelnya dengan nilai iterasi model dan learning rate yg berbeda.

F. Kesimpulan

Semakin besar iteration memiliki potensi meningkatkan akurasi karena model lebih banyak belajar, akan tetapi ada batas dimana perulangan tidak akan membuat model bertambah pintar. Sementara learning rate adalah kebalikannya, semakin kecil besar learning rate, maka akan membuat model semakin baik, akan tetapi ada batas dimana besar learning rate tidak akan membuat model bertambah pintar walaupun nilanya diperbesar. Learning rate yang memiliki nilai yang terlalu besar akan membuat model tidak dapat belajar dengan baik

G. Lampiran

Link Code : Tahap2_Classification

Link Video : Tugas Besar Machine Learning Tahap 2 (Classification)

H. Referensi

https://en.wikipedia.org/wiki/Sigmoid_function

https://vincentmichael089.medium.com/machine-learning-2-logistic-regression-96b3d4e7b603