KLASIFIKASI

UNTUK MENGELOMPOKKAN DATA

Oleh :

Deny Ahmad Sofyan(1301194274)
Zahid Athallah Shabir(1301194074)

IF-43-10

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

2021

BAB I
Pendahuluan

A. Latar Belakang

Pada tugas besar kali ini kami sebagai mahasiswa dituntut agar dapat memecahkan masalah pada suatu dealer kendaraan. Masalah yang muncul adalah “klasifikasi pelanggan yang tertarik membeli mobil baru dan tidak tertarik membeli mobil baru”. Masalah tersebut tentu tidak luput dari beberapa faktor lainnya yang ada pada pelanggan tersebut, yaitu ; umur, Jenis kelamin, Sim, kode daerah, sudah asuransi, umur kendaraan, kendaraan rusak, premi, kanal penjualan, lama berlangganan. Pada masalah ‘klasifikasi’ ini mahasiswa harus membuat program yang dapat mempelajari klasifikasi pelanggan yang tertarik dan tidak tertarik untuk membeli kendaraan baru dengan memperhatikan faktor-faktor tersebut.

B. Tujuan

Program *classification* ini bertujuan untuk mengidentifikasi apakah pelanggan tertarik atau tidak dengan mengisi kolom ‘Tertarik’ pada dataset dengan mempertimbangkan beberapa kolom lain sebagai variable.

BAB II
Landasan Teori

C. Klasifikasi

Secara harfiah bisa pula dikatakan bahwa **klasifikasi** adalah pembagian sesuatu menurut kelas-kelas. Menurut Ilmu Pengetahuan,**Klasifikasi** adalah Proses pengelompokkan benda berdasarkan ciri-ciri persamaan dan perbedaan. Pada Machine Learning Klasifikasi diartikan sebagai pengelompokkan yang dilakukan oleh program komputer dengan cara membuat mesin mempelajari suatu sifat atau karakter agar dapat mengenali suatu kelompok

D. Regresi Linear

Regresi **linear** merupakan pendekatan untuk memodelkan hubungan antara suatu variabel bebas dengan variabel terikat. Variabel bebas dilambangkan dengan X sedangkan variable terikat dilambangkan dengan Y. Regresi umumnya digunakan untuk prediksi (prediction) dan peramalan (forecasting). Prediksi merupakan proses memperkirakan nilai-nilai data bertipe apa saja.

BAB III
Implementasi

E. Eksplorasi dan Persiapan Data

a. Import Data
Mengimport data csv menggunakan liblary pandas melalui raw github yang telah di upload sebelumnya

```
[3]  url1 = 'https://raw.githubusercontent.com/abil223/Malin2-classification/main/kendaraan_train.csv'
     url2 = 'https://raw.githubusercontent.com/abil223/Malin2-classification/main/kendaraan_test.csv'

     df_test = pd.read_csv(url2)
     df_train = pd.read_csv(url1)
```

b. Menghilangkan file yang tidak penting
Menghilangkan kolom yang tidak penting seperti kolom id dimana isinya hanya angka urut dari baris dan akan membuat data tidak relefan

```
#remove unused value
df_train=df_train.drop(['id'],axis=1)
df_train
```

c. Menangani file yang kosong
Menangani isi dari baris kolom yang kosong dengan menggunakan mean dan median sebagai inputannya

```
#handling missing value
df_train['Jenis_Kelamin']=df_train['Jenis_Kelamin'].fillna(df_train['Jenis_Kelamin'].mode()[0])
df_train['Kode_Daerah'].fillna(df_train['Kode_Daerah'].mean(),inplace=True)
df_train['Umur'].fillna(df_train['Umur'].mean(),inplace=True)
df_train['Sudah_Asuransi'].fillna(df_train['Sudah_Asuransi'].mean(),inplace=True)
df_train['SIM'].fillna(df_train['SIM'].median(),inplace=True)
df_train['Umur_Kendaraan']=df_train['Umur_Kendaraan'].fillna(df_train['Umur_Kendaraan'].mode()[0])
df_train['Kendaraan_Rusak']=df_train['Kendaraan_Rusak'].fillna(df_train['Kendaraan_Rusak'].mode()[0])
df_train['Premi'].fillna(df_train['Premi'].median(),inplace=True)
df_train['Kanal_Penjualan'].fillna(df_train['Kanal_Penjualan'].mean(),inplace=True)
df_train['Lama_Berlangganan'].fillna(df_train['Lama_Berlangganan'].mean(),inplace=True)
```

d. Menangani pencilan data (outliers)
Menangani ouliers dari data karena dapat menjadikan data banyak sekali noise dan mengganggu tingkat keakurasian data

```
def interquartile(data,x):
  q1 = (df_train[x]).quantile(0.25)
  q3 = (df_train[x]).quantile(0.75)
  iqr = q3 - q1 #range q3 - q1
  maximum = q3 + (1.5 *iqr)
  minimum = q1 - (1.5 *iqr)
  return maximum,minimum

max,min = interquartile(df_train,'Premi')
print('max: ',max,' | min:',min)
```

```
max:  59719.0  | min: 4055.0
```

```
def outliersFix(data,x,max,min):
  above = (df_train[x] > max)
  below = (df_train[x] < min)
  print('above: ',above,' | below: ',below)
  data[x] = df_train[x].mask(above, max,axis=0)
  data[x] = df_train[x].mask(below, min,axis=0)
  return df_train

max,min = interquartile(df_train,'Premi')
df_train = outliersFix(df_train,'Premi',max,min)
boxplot = df_train.boxplot(column=['Premi'])
```

Menangani pencilan data pada data test menggunakan metode interquartile

```
#fixing outlier for data_test
max,min = interquartile(df_test,'Premi')
print('max: ',max,' | min:',min)
```

```
max:  59719.0  | min: 4055.0
```

```
max,min = interquartile(df_test,'Premi')
df_test = outliersFix(df_test,'Premi',max,min)
boxplot = df_test.boxplot(column=['Premi'])
```

e. labelling(mengubah data menjadi numerik)
Disini kami mengubah data kategorik menjadi numerik pada data train dan data test. Data Jenis_Kelamin (Pria = 1, Wanita = 0), data Umur_Kendaraan (< 1 Tahun = 0, 1-2 Tahun = 1, > 2 Tahun = 2), dan data Kendaraan_Rusak (Pernah = 1, Tidak = 0)

```
replace_values = {'Wanita' : 0, 'Pria' : 1}
df_train = df_train.replace({"Jenis_Kelamin": replace_values})
df_test = df_test.replace({"Jenis_Kelamin": replace_values})
df_train['Jenis_Kelamin'] = df_train['Jenis_Kelamin'].astype('float')
df_test['Jenis_Kelamin'] = df_test['Jenis_Kelamin'].astype('float')

replace_values = {'< 1 Tahun' : 0, '1-2 Tahun' : 1, '> 2 Tahun' : 2 }
df_train = df_train.replace({"Umur_Kendaraan": replace_values})
df_test = df_test.replace({"Umur_Kendaraan": replace_values})
df_train['Umur_Kendaraan'] = df_train['Umur_Kendaraan'].astype('float')
df_test['Umur_Kendaraan'] = df_test['Umur_Kendaraan'].astype('float')

replace_values = {'Pernah' : 1, 'Tidak' : 0}
df_train = df_train.replace({"Kendaraan_Rusak": replace_values})
df_test = df_test.replace({"Kendaraan_Rusak": replace_values})
df_train['Kendaraan_Rusak'] = df_train['Kendaraan_Rusak'].astype('float')
df_test['Kendaraan_Rusak'] = df_test['Kendaraan_Rusak'].astype('float')
```

f. Scaling
Scaling data dengan menggunakan metode min max scaller

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df_train[['Umur','Kode_Daerah', 'Premi', 'Kanal_Penjualan', 'Lama_Berlangganan', 'Umur_Kendaraan']]=scaler.fit_transform(df_train[['Umur','K
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df_test[['Jenis_Kelamin','Umur','SIM','Kode_Daerah','Sudah_Asuransi','Umur_Kendaraan','Kendaraan_Rusak', 'Premi',
```

g. Export
Mengexport data yang sudah di preparation sebelumnya untuk di olah di tahap selanjutnya

```
df_train.to_csv('cleaned_kendaraan_train.csv', index=False)
df_test.to_csv('cleaned_kendaraan_test.csv', index=False)
```

F. Modeling

a. Split data

Disini kami memisahkan label dan fitur menjadi

Fitur = x_train dan x_test

Label = y_train dan y_test

```
#split data
X_train = df_train.loc[:, df_train.columns != 'Tertarik']
X_test = df_test.loc[:, df_test.columns  != 'Tertarik']

y_train = df_train['Tertarik']
y_test = df_test['Tertarik']
```

b. Classification

i. Linear Regression

```
class Linear_Regression():

  # initiating the parameters (learning rate & no. of iterations)
  def __init__(self, learning_rate, no_of_iterations):

    self.learning_rate = learning_rate
    self.no_of_iterations = no_of_iterations


  def fit(self, X, Y ):

    # number of training examples & number of features

    self.m, self.n = X.shape  # number of rows & columns

    # initiating the weight and bias

    self.w = np.zeros(self.n)
    self.b = 0
    self.X = X
    self.Y = Y

    # implementing Gradient Descent

    for i in range(self.no_of_iterations):
```

```
  for i in range(self.no_of_iterations):
    self.update_weights()


def update_weights(self):

  Y_prediction = self.predict(self.X)

  # calculate gradients

  dw = - (2 * (self.X.T).dot(self.Y - Y_prediction)) / self.m

  db = - 2 * np.sum(self.Y - Y_prediction)/self.m

  # upadating the weights

  self.w = self.w - self.learning_rate*dw
  self.b = self.b - self.learning_rate*db


def predict(self, X):

  pred = X.dot(self.w) + self.b
  pred_hasil = [1 if i>0.5 else 0 for i in pred]
```

```
def predict(self, X):

  pred = X.dot(self.w) + self.b
  pred_hasil = [1 if i>0.5 else 0 for i in pred]
  return np.array(pred_hasil)
```

```
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import numpy as np

def mean_squared_error(y_true, y_pred):
        return np.mean((y_true - y_pred) ** 2)


regressor = Linear_Regression(learning_rate=0.01, no_of_iterations=1000
regressor.fit(X_train, y_train)
predictions = regressor.predict(X_test)

mse = mean_squared_error(y_test, predictions)
print("MSE:", mse)

def accuracy(y_true,y_pred):
  accuracy = np.sum(y_true == y_pred)/len(y_true)
  return accuracy


print("Accuracy: ",accuracy(y_test, predictions))


print("pred", predictions)
```

ii. Naive bayes (perbandingan)

```
#Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB

#Create a Gaussian Classifier
gnb = GaussianNB()

#Train the model using the training sets
gnb.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = gnb.predict(X_test)
```

```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
y_pred
```

G. Eksperimen

a. Eksperimen 1 membanding kan nilai confusion matrix

```
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import f1_score
import seaborn as sns

restLROVR = predictions

conf_matrix = confusion_matrix(y_test, restLROVR)
print(conf_matrix)
```

```
[[247683   3142]
 [ 33004   2002]]
```

```
prediksi = metrics.accuracy_score(y_test, y_pred)

resultNB = prediksi

print("Confusion Matrix")
print(confusion_matrix(y_test, y_pred))
```

```
Confusion Matrix
[[164466  86359]
 [  3686  31320]]
```

b. Eksperimen 2 membandingkan classification linear regresi sebelum dan sesudah data di oversampling

```
from collections import Counter
from imblearn.over_sampling import SMOTE

print("sebelum oversampling: ",Counter(y_train))
SMOTE = SMOTE()

# fit dan transform data training
X_train_SMOTE, y_train_SMOTE = SMOTE.fit_resample(X_train, y_train)
print("Setelah oversampling: ",Counter(y_train_SMOTE))
```

```
sebelum oversampling:  Counter({0: 250825, 1: 35006})
Setelah oversampling:  Counter({0: 250825, 1: 250825})
```

```
regres_eksperimen = Linear_Regression(learning_rate=0.0001,no_of_iterations=1000)
regres_eksperimen.fit(X_train_SMOTE, y_train_SMOTE)

pred_eks = regres_eksperimen.predict(X_train_SMOTE)

pred_eks
```

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

```
print("Accuracy: ",accuracy(y_train_SMOTE, pred_eks))
print()
print("Confusion Matrix")
print(confusion_matrix(y_train_SMOTE, pred_eks))
print("Classification Report")
print(classification_report(y_train_SMOTE,pred_eks))
```

```
Accuracy:  0.5731506030100668

Confusion Matrix
[[222610  28215]
 [185914  64911]]
Classification Report
              precision    recall  f1-score   support

           0       0.54      0.89      0.68    250825
           1       0.70      0.26      0.38    250825

    accuracy                           0.57    501650
   macro avg       0.62      0.57      0.53    501650
weighted avg       0.62      0.57      0.53    501650
```

H. Evaluasi

Akurasi Linear Regresi

```
Accuracy:  0.8735406586409452
```

Akurasi Naive bayes

```
Accuracy: 0.6849711892691835
```

BAB IV

kesimpulan

Berdasarkan eksperimen data yang telah kami lakukan dan diolah dalam klasifikasi dengan menggunakan metode naive bayes dan linear regression, metode yang terbaik dapat dilihat berdasarkan nilai akurasi positif tertinggi yaitu metode linear regression dengan nilai akurasi 0,87 atau 87,7% dan apabila dibulatkan menjadi 88%.

Link Video :
https://youtu.be/1sk-ypOTlpY

Link Google Colab : :
https://colab.research.google.com/drive/1glAt_RPQi6snGkGcDd0f_NJHSl2Jthf9?usp=sharing