LAPORAN TUGAS BESAR
MATA KULIAH PEMBELAJARAN
MODEL KLASIFIKASI

Oleh :

- Artisa Bunga Syahputri (1301194007)
- Salsabila Martono (1301194469)

Program Studi S1 Informatika
Fakultas Informatika
Universitas Telkom

2021

Daftar Isi

A. Formulasi Masalah

Pada tugas besar kedua Pembelajaran masin ini, akan dilakukan klasifikasi (supervised learning) terhadap data 'kendaraan_train.csv' untuk melakukan prediksi apakah pelanggan untuk membeli kendaraan atau tidak berdasarkan dataset yang disediakan. Pada percobaan klasifikasi dataset tersebut, kami menggunakan beberapa model untuk mengklasifikasikan dataset berdasarkan ketertarikan pelanggan untuk membeli kendaraan atau tidak.

B. Eksplorasi dan Persiapan Data

1. Eksplorasi

Pada tahapan Eksplorasi data, kami melihat beberapa informasi yang dibutuhkan pada data latih. Sebelum melakukan eksplorasi, kami mendrop fitur Id, karena tidak akan digunakan saat proses pre-processing dan pemodelan nanti.

```
[ ]  #Drop feature ID pada dataset
     df.drop(['id'],inplace=True, axis=1)
```

Berikut adalah hasil eksplorasi data train yang kami lakukan,

a. Informasi dataset

Berikut adalah informasi pada setiap fitur yang ada pada dataset yang id nya sudah di drop hal ini bertujuan untuk mengetahui tipe data dari setiap fitur yang ada pada dataset

```
[ ]  1 #melihat informasi data
     2 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285831 entries, 0 to 285830
Data columns (total 11 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   Jenis_Kelamin      271391 non-null  object
 1   Umur               271617 non-null  float64
 2   SIM                271427 non-null  float64
 3   Kode_Daerah        271525 non-null  float64
 4   Sudah_Asuransi     271602 non-null  float64
 5   Umur_Kendaraan     271556 non-null  object
 6   Kendaraan_Rusak    271643 non-null  object
 7   Premi              271262 non-null  float64
 8   Kanal_Penjualan    271532 non-null  float64
 9   Lama_Berlangganan  271839 non-null  float64
 10  Tertarik           285831 non-null  int64
dtypes: float64(7), int64(1), object(3)
memory usage: 24.0+ MB
```

b. Deskripsi statistik dataset

Berikut adalah hasil deskripsi statistik dataset. Ini dilakukan bertujuan untuk mengetahui informasi statistik dari setiap fitur numerik pada dataset seperti nilai minimum, maksimum, rata-rata, quartil,dll.

```
[ ]  1 #melihat deskripsi statistik dari data
     2 df.describe()
```

	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
count	271617.000000	271427.000000	271525.000000	271602.000000	271262.000000	271532.000000	271839.000000	285831.000000
mean	38.844336	0.997848	26.405410	0.458778	30536.683472	112.021567	154.286302	0.122471
std	15.522487	0.046335	13.252714	0.498299	17155.000770	54.202457	83.694910	0.327830
min	20.000000	0.000000	0.000000	0.000000	2630.000000	1.000000	10.000000	0.000000
25%	25.000000	1.000000	15.000000	0.000000	24398.000000	29.000000	82.000000	0.000000
50%	36.000000	1.000000	28.000000	0.000000	31646.000000	132.000000	154.000000	0.000000
75%	49.000000	1.000000	35.000000	1.000000	39377.750000	152.000000	227.000000	0.000000
max	85.000000	1.000000	52.000000	1.000000	540165.000000	163.000000	299.000000	1.000000

c. Dimensi dataset

Dapat diketahui bahwa dataset yang kita miliki memiliki jumlah baris sebanyak 285831 baris dengan jumlah kolom sebanyak 11 kolom.

```
[ ]  1 #melihat dimensi data
     2 df.shape

(285831, 11)
```

d. Missing Value

Missing value terjadi ketika data dari sebuah record tidak lengkap. Kami melihat missing value yang ada pada dataset bertujuan untuk melihat jumlah missing value pada setiap fitur pada dataset, hal ini yang nantinya akan di handling pada preprocessing, karena missing value akan menyebabkan masalah dan mempengaruhi performa saat pemodelan jika masih terdapat missing value pada data.

```
[ ]  1 # cek Missing value
     2
     3 print(df.isnull().sum())

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
```

e. Duplicate Data

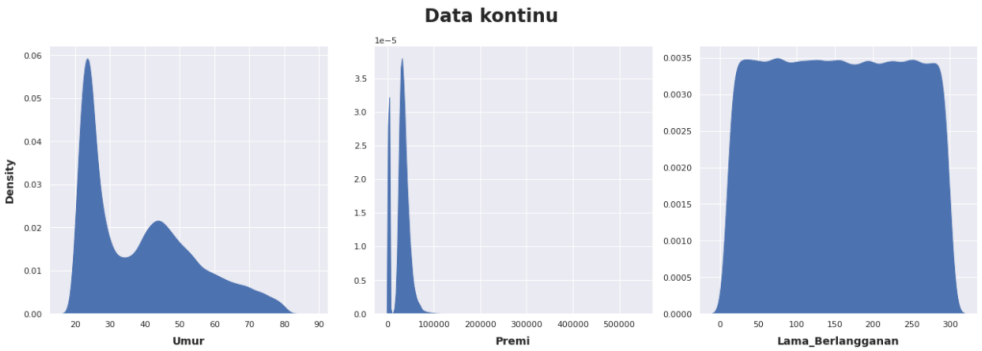
Data yang memiliki duplikat nantinya akan mempengaruhi model machine learning yang kita buat, apalagi jika ternyata data yang duplikat berjumlah sangat besar, oleh karena itu disini kami melakukan pengecekan terlebih dahulu apakah terdapat data yang duplikat pada dataset yang kita miliki.

```
[ ]  1 #Cek Data Duplikat
     2 duplicate = list(df_train.duplicated())
     3 print("Data Duplikasi :", duplicate.count(True))

     Data Duplikasi : 82
```

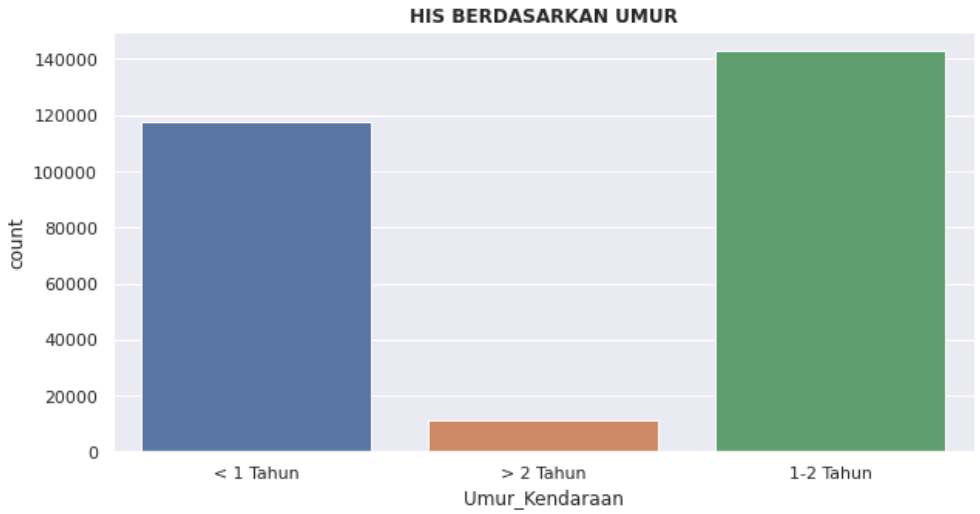
f. Visualisasi persebaran data

Visualisasi data dilakukan untuk melihat persebaran jumlah data dari setiap fitur pada dataset. Disini kami melakukan eksplorasi dengan melihat persebaran data berdasarkan jenis data kontinu (Umur, Premi, Lama_Berlangganan) dan juga data kategori_ordinal(Umur_Kendaraan). Dengan melihat sebaran data dari fitur-fitur tersebut bertujuan agar kita dapat melihat kecenderungan jumlah data dari setiap fitur. Lalu kami juga melihat korelasi antar setiap fitur menggunakan heatmap dimana korelasi tertinggi yaitu yang nilainya mendekati 1 atau -1 yaitu korelasi antara Umur dengan Kanal_Penjualan dengan besar korelasi sebesar -0.58, sedangkan korelasi terkecil yaitu korelasi dengan nilai yang mendekati 0 yaitu dimiliki oleh korelasi antara Umur dengan Lama_Berlangganan yaitu sebesar 0.00015

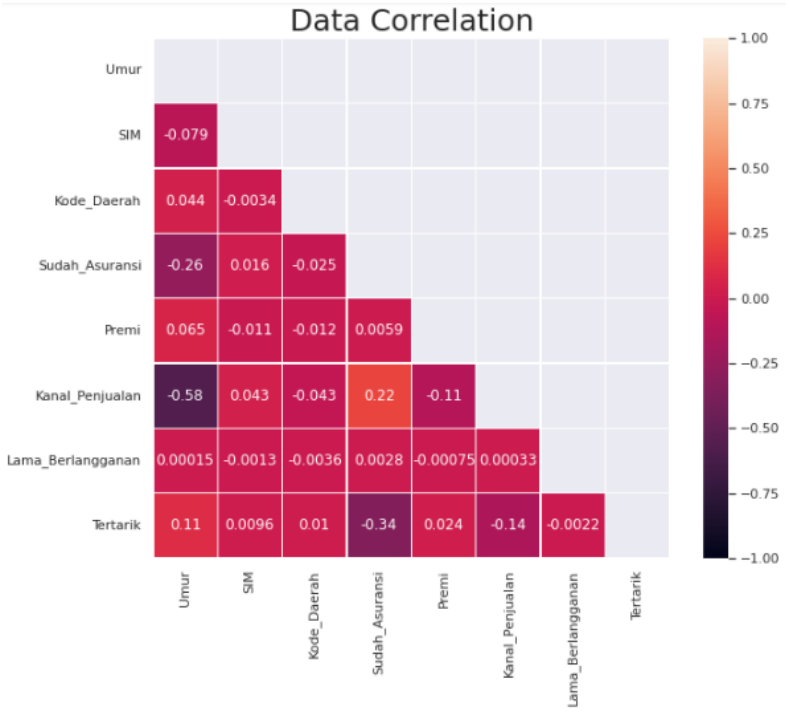
1) Data Kontinu



2) Data Histogram berdasarkan “Umur_Kendaraan”



3) Korelasi Data



2. Persiapan Data

Proses persiapan data atau data cleaning adalah tahap yang sangat penting yang bertujuan untuk meningkatkan kualitas data dengan membuat data atau informasi yang tidak dibutuhkan sehingga kita mendapatkan data yang berkualitas sehingga akan memberikan pengaruh positif dalam proses pengambilan keputusan dan meningkatkan produktivitas kerja model secara keseluruhan. Proses persiapan data atau cleaning data yang kami lakukan antara lain adalah sebagai berikut:

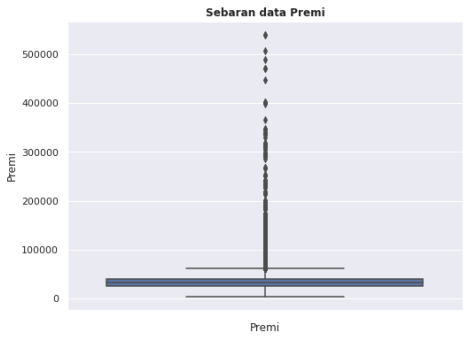
a. Handle Outlier

Outlier atau pencilan adalah kumpulan nilai yang berbeda jauh dengan nilai lainnya yang akan mengacaukan hasil dari analisis statistik dari data, biasanya disebabkan karena kesalahan dalam pengumpulan data atau karena data nya memang unik dari data lain,sehingga harus di handling.

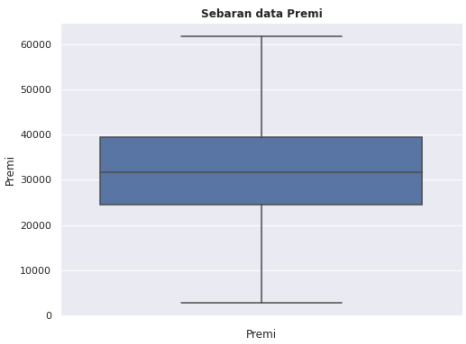
Metode yang kami gunakan untuk menghandle outlier adalah metode interquartile range(IQR). Yaitu menghapus outlier dengan menghitung batas atas dan batas bawah dari datanya. Pada dataset, fitur Premi memiliki outlier, sehingga harus di handling. Berikut adalah kode yang kami buat untuk handling outlier,

```
# Handle Outlier dengan metode IQR

Q1 = df['Premi'].quantile(0.25)
Q3 = df['Premi'].quantile(0.75)
IQR = Q3 - Q1

Max = Q3 + (1.5 * IQR)
Min = Q1 - (1.5 * IQR)

more_than = df['Premi'] > Max
lower_than = df['Premi'] < Min

df['Premi'] = df['Premi'].mask(more_than, Max)
df['Premi'] = df['Premi'].mask(lower_than, Min)

```

Sebelum



Sesudah



b. Label Encoding

Pada dataset yang dimiliki, terdapat jenis data kategorik yang tidak dapat diukur atau didefinisikan dengan bilangan. Model machine learning tidak dapat mengolah data kategorik sehingga kita harus mengkonversi data kategorik menjadi data numerik. Salah satu cara untuk mengkonversi data kategorik menjadi data numerik adalah dengan menggunakan one hot encoding atau label encoding. Di tugas ini kami menggunakan metode label encoding untuk mengkonversi data kategorik menjadi data numerik agar dapat diproses oleh model. Fitur yang kami label encoding:

1) Jenis Kelamin

2) Umur Kendaraan
3) Kendaraan Rusak

Label Encoding

```
[ ] #Label Encoding

    df['Jenis_Kelamin'] = df['Jenis_Kelamin'].replace(['Wanita', 'Pria'], [0, 1])
    df['Umur_Kendaraan'] = df['Umur_Kendaraan'].replace(['< 1 Tahun', '1-2 Tahun', '> 2 Tahun'], [0, 1, 2])
    df['Kendaraan_Rusak'] = df['Kendaraan_Rusak'].replace(['Tidak', 'Pernah'], [0, 1])
```

```
df.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	0.0	0.0	28029.0	152.0	97.0	0
1	1.0	48.0	1.0	39.0	0.0	2.0	1.0	25800.0	29.0	158.0	0
2	NaN	21.0	1.0	46.0	1.0	0.0	0.0	32733.0	160.0	119.0	0
3	0.0	58.0	1.0	48.0	0.0	1.0	0.0	2630.0	124.0	63.0	0
4	1.0	50.0	1.0	35.0	0.0	2.0	NaN	34857.0	88.0	194.0	0

c. Handle Missing Value

Pada saat eksplorasi data kita mendapati bahwa pada dataset terdapat missing value yang akan berpengaruh buruk saat kita melatih model machine learning sehingga kita harus menghandling nya. Ada dua cara untuk melakukan handling terhadap missing value, yaitu dengan mengisi data yang kosong atau dengan melakukan drop pada data yang kosong tersebut.

Pada tugas besar ini,Untuk handle missing value, kami mendrop semua data yang memiliki value Null/NaN. Kami mendrop data missing value, karena jumlah missing value data < 5% dari dataset yang dimiliki. Hasil setelah kami melakukan drop terhadap missing value pada data latih:

```
[ ]  1 #Handle Missing Value dengan mengisi missing value dengan mean
     2 #df_train = df.fillna(df_train.mean())
     3 #Handle Missing Value dengan mendrop missing value
     4 df_train = df.dropna()
     5 print(df_train.isnull().sum())
```

```
Jenis_Kelamin        0
Umur                 0
SIM                  0
Kode_Daerah          0
Sudah_Asuransi       0
Umur_Kendaraan       0
Kendaraan_Rusak      0
Premi                0
Kanal_Penjualan      0
Lama_Berlangganan    0
Tertarik             0
dtype: int64
```

d. Handle Duplicate Data

Duplicate data berarti terdapat data yang sama pada data latih yang kita miliki, ini akan memberi pengaruh buruk pada saat melati model apalagi jika jumlah data yang terduplikat berjumlah banyak. Pada data latih terdapat 82 duplikat data, sehingga untuk mengatasinya kami melakukan drop pada data yang duplikat di data set yang kami miliki.

Handle Data Duplikat

```
#Cek Data Duplikat
duplicate = list(df_train.duplicated())
print("Data Duplikasi :", duplicate.count(True))
```

```
Data Duplikasi : 82
```

```
[ ] #Handle Data Duplikasi
df_train.drop_duplicates(inplace=True)
duplicate = list(df_train.duplicated())
print("Data Duplikasi :", duplicate.count(True))
```

```
Data Duplikasi : 0
```

e. Normalization

Normalisasi bertujuan untuk mengubah nilai dari setiap fitur sehingga menjadi skala yang sama sehingga memungkinkan kenaikan performa dan stabilitas dari model machine learning. Ada banyak metode yang dapat digunakan untuk seperti menggunakan standar scaller atau menggunakan minmaxscaller.

Pada tugas besar ini kami menggunakan minmaxscallar untuk melakukan normalisasi terhadap dataset yang kami miliki dimana Min-Max Scaling sendiri bekerja dengan menyesuaikan data dalam rentang/range tertentu. Dengan rumus matematisnya:

$$X_{sc} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Berikut hasil mormalisasi yang kami lakukan dengan terlebih dahulu mengimport library minmaxscaler pada sklearn.

Normalisasi

```
#Normalisasi dengan menggunakan MinMaxScaller
from sklearn.preprocessing import MinMaxScaler
df_train = pd.DataFrame(MinMaxScaler().fit_transform(df_train))
df_train.columns = ['Jenis_Kelamin', 'Umur', 'SIM', 'Kode_Daerah', 'Sudah_Asuransi', 'Umur_Kendaraan', 'Kendaraan_Rusak', 'Premi', 'Kanal_Penjualan', 'Lama_Berlangganan', 'Tr
df_train
```

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	0.0	0.153846	1.0	0.634615	1.0	0.0	0.0	0.428911	0.932099	0.301038	0.0
1	1.0	0.430769	1.0	0.750000	0.0	1.0	1.0	0.391270	0.172840	0.512111	0.0
2	0.0	0.584615	1.0	0.923077	0.0	0.5	0.0	0.000000	0.759259	0.183391	0.0
3	1.0	0.015385	1.0	0.673077	1.0	0.0	0.0	0.339512	0.932099	0.557093	0.0
4	0.0	0.000000	1.0	0.153846	1.0	0.0	0.0	0.475469	0.981481	0.072664	0.0
...	...	...	...	...	...	...	...	...	...	...	...
170981	0.0	0.046154	1.0	0.076923	1.0	0.0	0.0	0.394445	0.932099	0.716263	0.0
170982	0.0	0.015385	1.0	0.884615	1.0	0.0	0.0	0.710197	0.932099	0.138408	0.0
170983	0.0	0.046154	1.0	0.961538	1.0	0.0	0.0	0.795729	0.932099	0.747405	0.0
170984	1.0	0.738462	1.0	0.134615	1.0	0.5	0.0	0.470690	0.759259	0.899654	0.0
170985	1.0	0.384615	1.0	0.538462	0.0	0.5	1.0	0.571623	0.154321	0.117647	0.0

170986 rows × 11 columns

f. Split Dataset

Kami membagi data latih menjadi dua yaitu data train dan data test untuk melakukan pelatihan model pada data latih. Perbandingan yang kami gunakan untuk melatih model ii adalah 90:10 dimana kami menggunakan 90% dari data latih sebagai data train dan 10% sebagai data tesnya, hal ini karena dimensi data yang kita miliki disini kurang lebih sebanyak 171068 data sehingga yang kami gunakan sebagai data tes nya sebanyak 10% dari data latih nya yaitu kurang lebih 17 ribu data.

```
#pisahkan atribut dengan label

x = df.drop("Tertarik", axis=1)
y = df["Tertarik"]
```

DATA SPLIT

```
[ ] # Membagi dataset menjadi data latih & data uji
    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.1, random_state=123)
```

Selain membersihkan dataset train, kami juga melakukan persiapan data pada data test yang disediakan pada folder dataset untuk tugas ini yang nanti akan digunakan untuk evaluasi model machine learning yang dibuat. Persiapan data test yang kami lakukan diantaranya:

a. Download data test

Kami mendownload data test yang nanti akan digunakan pada evaluasi model machine learning

```
#download data test
!gdown --id 1QrTecS39gNEyndR3DEZ7TIEgXJWg6UnB

Downloading...
From: https://drive.google.com/uc?id=1QrTecS39gNEyndR3DEZ7TIEgXJWg6UnB
To: /content/kendaraan_test.csv
100% 2.31M/2.31M [00:00<00:00, 73.6MB/s]

#read data test
df_test = pd.read_csv('kendaraan_test.csv')

df_test.head()
```

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	Wanita	49	1	8	0	1-2 Tahun	Pernah	46963	26	145	0
1	Pria	22	1	47	1	< 1 Tahun	Tidak	39624	152	241	0
2	Pria	24	1	28	1	< 1 Tahun	Tidak	110479	152	62	0
3	Pria	46	1	8	1	1-2 Tahun	Tidak	36266	124	34	0
4	Pria	35	1	23	0	1-2 Tahun	Pernah	26963	152	229	0

b. Label Encoding dan Normalisasi

Selanjutnya kami melakukan label encoding terhadap data kategorikal pada data test yaitu data Jenis Kelamin, Umur Kendaraan, dan Kendaraan Rusak.

Lalu kami melakukan normalisasi pada data test untuk menyamakan skala pada setiap fitur yang terdapat pada datatest dengan menggunakan metode minmaxscaler. Sehingga didapatkan hasil sebagai berikut:

```
df_test['Jenis_Kelamin'] = df_test['Jenis_Kelamin'].replace(['Wanita', 'Pria'], [0, 1])
df_test['Umur_Kendaraan'] = df_test['Umur_Kendaraan'].replace(['< 1 Tahun', '1-2 Tahun', '> 2 Tahun'], [0, 1, 2])
df_test['Kendaraan_Rusak'] = df_test['Kendaraan_Rusak'].replace(['Tidak', 'Pernah'], [0, 1])
```

```
from sklearn.preprocessing import MinMaxScaler
df_test = pd.DataFrame(MinMaxScaler().fit_transform(df_test))
df_test.columns = ['Jenis_Kelamin', 'Umur', 'SIM', 'Kode_Daerah', 'Sudah_Asuransi', 'Umur_Kendaraan', 'Kendaraan_Rusak', 'Premi', 'Kanal_Penjualan', 'Lama_Berlangganan', 'Te
df_test
```

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	0.0	0.446154	1.0	0.153846	0.0	0.5	1.0	0.082475	0.154321	0.467128	0.0
1	1.0	0.030769	1.0	0.903846	1.0	0.0	0.0	0.068822	0.932099	0.799308	0.0
2	1.0	0.061538	1.0	0.538462	1.0	0.0	0.0	0.200636	0.932099	0.179931	0.0
3	1.0	0.400000	1.0	0.153846	1.0	0.5	0.0	0.062575	0.759259	0.083045	0.0
4	1.0	0.230769	1.0	0.442308	0.0	0.5	1.0	0.045268	0.932099	0.757785	0.0
...	...	...	...	...	...	...	...	...	...	...	...
47634	1.0	0.630769	1.0	0.884615	0.0	1.0	1.0	0.052851	0.759259	0.197232	0.0
47635	1.0	0.323077	1.0	0.288462	0.0	0.5	1.0	0.000000	0.962963	0.768166	0.0
47636	1.0	0.061538	1.0	0.557692	1.0	0.0	0.0	0.056687	0.932099	0.695502	0.0
47637	1.0	0.600000	1.0	0.576923	0.0	0.5	1.0	0.065406	0.154321	0.792388	1.0
47638	1.0	0.492308	1.0	0.596154	0.0	0.5	0.0	0.000000	0.759259	0.553633	0.0

47639 rows × 11 columns

c. Missing value dan duplicate data

Pada data test yang disediakan sudah tidak terdapat missing value, sehingga tidak perlu dilakukan handling terhadap mingging value dan duplicate data hanya ada 3 data duplikat.

```
print(df_test.isnull().sum())
```

```
Jenis_Kelamin        0
Umur                 0
SIM                  0
Kode_Daerah          0
Sudah_Asuransi       0
Umur_Kendaraan       0
Kendaraan_Rusak      0
Premi                0
Kanal_Penjualan      0
Lama_Berlangganan    0
Tertarik             0
dtype: int64
```

```
[8] #Cek Data Duplikat
    duplicate = list(df_test.duplicated())
    print("Data Duplikasi :", duplicate.count(True))

    Data Duplikasi : 3

    #Handle Data Duplikasi
    df_test.drop_duplicates(inplace=True)
    duplicate = list(df_test.duplicated())
    print("Data Duplikasi :", duplicate.count(True))

    Data Duplikasi : 0
```

C. Pemodelan

Pemodelan yang kami gunakan adalah Random Forest. Dimana Random Forest mengkombinasikan masing-masing tree yang baik menjadi satu model. Model Random Forest akan dilatih menggunakan data train yang sudah kami pre-processing dan split data menjadi data train dan data test dengan perbandingan 90:10. Kemudian kami menguji data train nya untuk diuji akurasi Random Forest. Dari hasil pemodelan yang kami buat, skor akurasi Random Forest adalah 0.876.

1. Model Random Forest

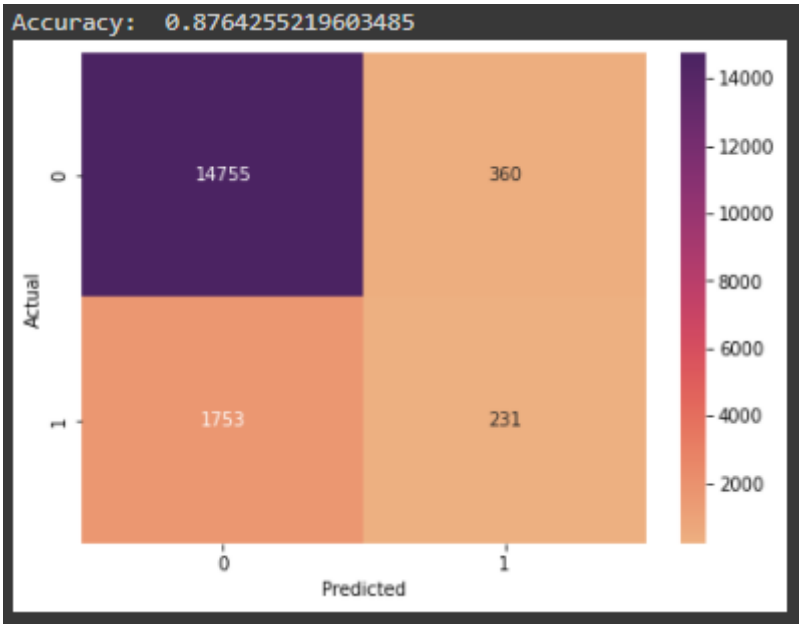
```
[60] #Pisahakan atribut dan label dari data test
     x_testing = df_test.drop("Tertarik", axis=1)
     y_testing = df_test["Tertarik"]
```

```
[81] #Import library random forest
     from sklearn.ensemble import RandomForestClassifier

     rf_model = RandomForestClassifier(n_estimators = 500, random_state = 42)

     rf_model = rf_model.fit(X_train, y_train)
```

```
[82] from sklearn.metrics import accuracy_score
     pred = rf_model.predict(X_test)

     secore = round(accuracy_score(pred, y_test), 3)

     print('Random Forest Accuracy: ',secore)
```

```
     Random Forest Accuracy:  0.876
```

Berikut adalah hasil Confusion Matrix dari data train yang sudah diuji coba,



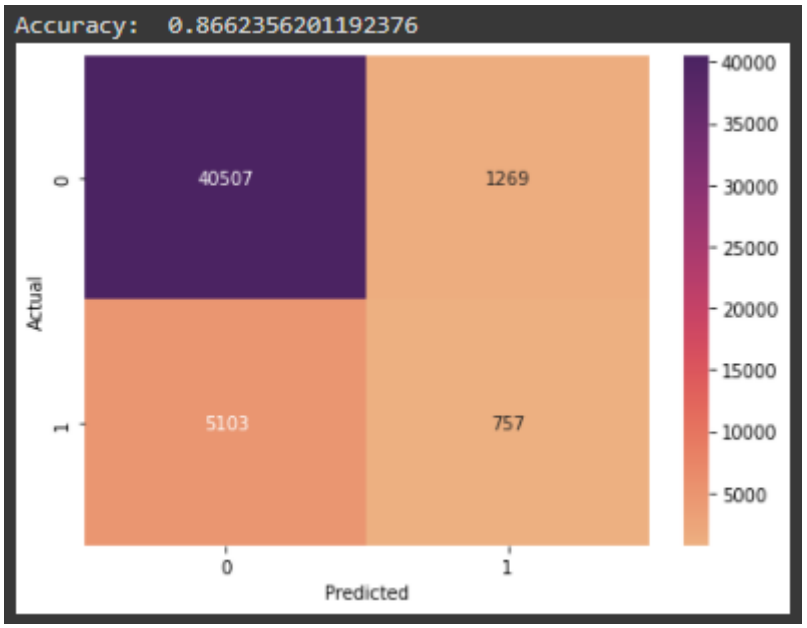
D. Evaluasi

Untuk evaluasi kami menggunakan dataset test yang diberikan dari tugas ke dalam pemodelan yang sudah kami buat yaitu Random Forest. Dari hasil evaluasi, skor akurasi Random Forest yang didapatkan menggunakan data test yang diberikan dari tugas adalah 0.866.

```
[60] #Pisahakan atribut dan label dari data test
     x_testing = df_test.drop("Tertarik", axis=1)
     y_testing = df_test["Tertarik"]
```

```
Evaluasi Model Random Forest

[ ] prediksi = rf_model.predict(x_testing)

    secor = round(accuracy_score(prediksi, y_testing), 3)

    print('Random Forest Accuracy: ',secor)

    Random Forest Accuracy:  0.866
```

Berikut adalah hasil Confusion Matrix dari dataset test yang diberikan,



E. Eksperimen

Kami melakukan beberapa kali eksperimen dengan menggunakan beberapa model yang berbeda dan menggunakan random forest dengan jumlah pohon yang berbeda juga.

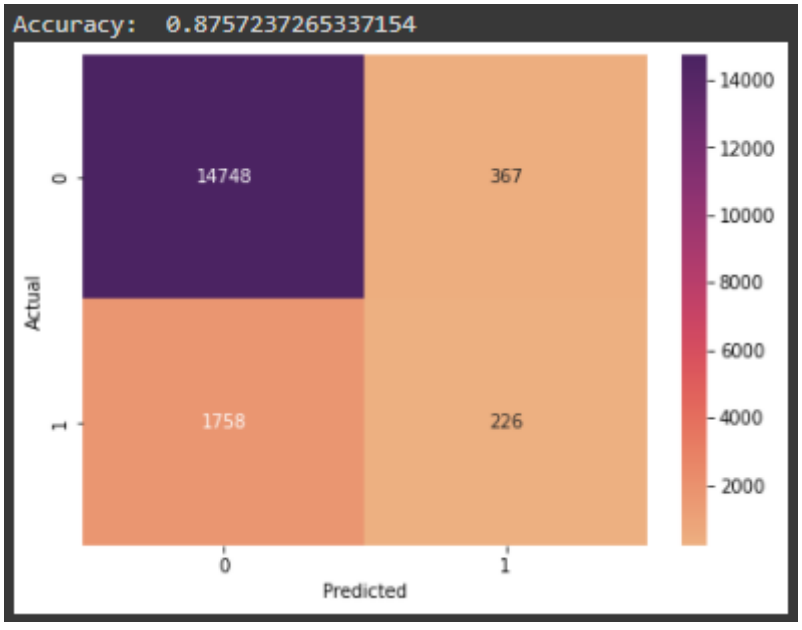
Berikut adalah eksperimen yang telah kami lakukan:

1. **Model random forest dengan jumlah pohon 100**

Kami menggunakan model random forest yang sama dengan model sebelumnya, yang berbeda hanya jumlah n_estimator yang digunakan pada eksperimen ini sebanyak 100 n_estimator, dari pengujian model didapatkan akurasi sebesar 0.876

```
100 pohon

[86] rf_model2 = RandomForestClassifier(n_estimators = 100, random_state = 42)

     rf_model2 = rf_model2.fit(X_train, y_train)

[87] pred = rf_model2.predict(X_test)

     secore2 = round(accuracy_score(pred, y_test), 3)

     print('Random Forest Accuracy: ',secore2)

     Random Forest Accuracy:  0.876
```

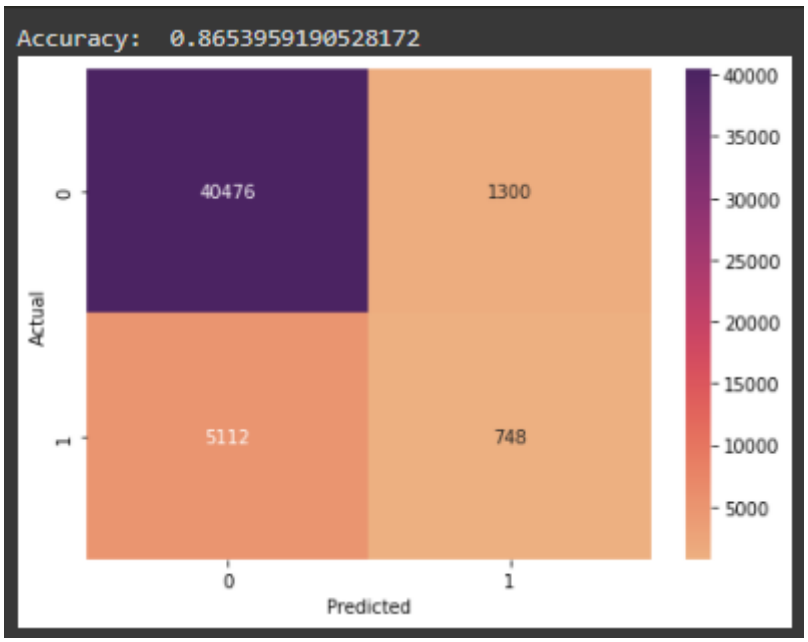
Berikut adalah Confusion Matrix dari model random forest dengan jumlah n_estimator =100



Setelah dilakukan evaluasi dengan menggunakan data test yang disediakan oleh soal didapatkan akurasi dengan menggunakan model random forest ini sebesar 0.865.

```
Evaluasi dengan data test

+ Code  + Text

prediksi2 = rf_model2.predict(x_testing)

secor2 = round(accuracy_score(prediksi2, y_testing), 3)

print('Random Forest Accuracy: ',secor2)

Random Forest Accuracy:  0.865

[99] confusion_matrix = pd.crosstab(y_testing, prediksi2, rownames=['Actual'], colnames=['Predicted'])
     plt.figure(figsize=(7, 5))
     sb.heatmap(confusion_matrix, annot=True,xticklabels=[0, 1], yticklabels=[0, 1], cmap='flare', fmt='d')
     print('Accuracy: ',metrics.accuracy_score(y_testing, prediksi2))
     plt.show()
```

Berikut Confusion matrix dari hasil evaluasi terhadap model random forest dengan jumlah n_estimator nya 100,



2. Decision tree

Decision tree adalah salah satu model machine learning yang dapat dipakai pada clasification yang mampu dipakai pada persoalan yang kompleks. Decision tree akan memprediksi sebuah kelas dan nilai berdasarkan aturan aturan yang telah dipelajari dari data-data yang ada sebelumnya.

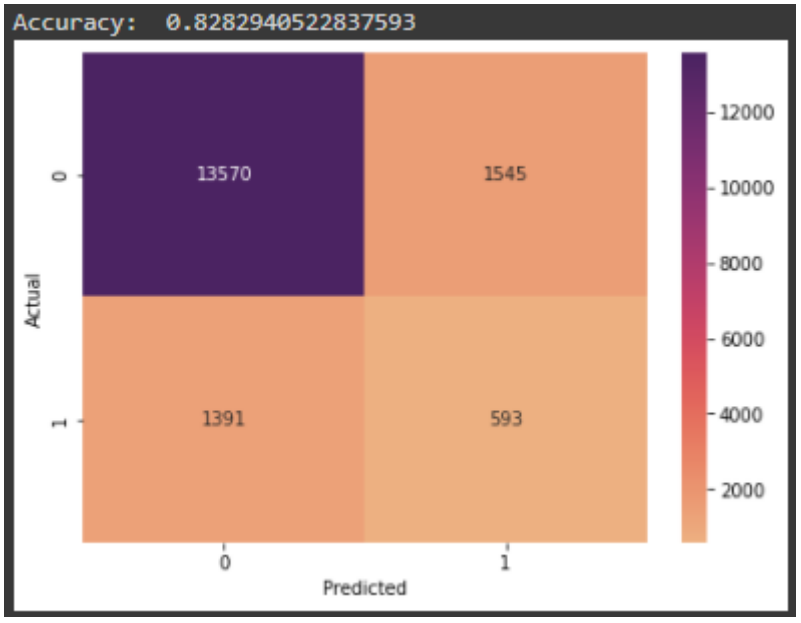
Pada Eksperimen ke dua, kami mencoba mengukur akurasi dari model untuk memprediksi ketertarikan pelanggan untuk membeli mobil dengan menggunakan model decision tree dengan mengimport library DecisionTreeClassifier pada sklearn. Dari hasil pemodelan didapatkan nilai akurasi sebesar 0.828. Nilai ini lebih kecil dari pada ketika menggunakan model random forest

2. DecisionTreeModel

```
[89] #membuat model descision tree classifier
     #import moder decisoion tree
     from sklearn.tree import DecisionTreeClassifier

     # membuat model Decision Tree
     tree_model = DecisionTreeClassifier()

     # Melatih model dengan menggunakan data latih
     tree_model = tree_model.fit(X_train, y_train)
```

```
[90] # Evaluasi Model
     y_pred = tree_model.predict(X_test)

     acc_secore = round(accuracy_score(y_pred, y_test), 3)

     print('Decision tree Accuracy: ', acc_secore)

     Decision tree Accuracy:  0.828
```

Berikut adalah Confusion matrix dari model Decision tree yang kami buat:



Selanjutnya kami melakukan evaluasi terhadap model yang dibangun dengan menggunakan data test yang sudah disediakan pada tugas. Didapat akurasi dari model terhadap data test kita sebesar 0.826

Validasi dengan data test

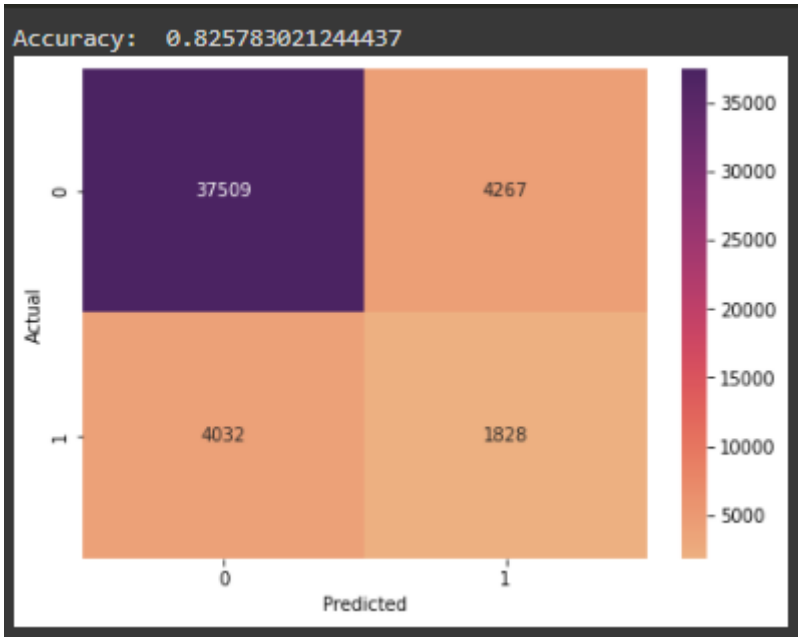
```
[91] #Pisahakan atribut dan label dari data test
     x_testing = df_test.drop("Tertarik", axis=1)
     y_testing = df_test["Tertarik"]
```

```
[93] #from sklearn.metrics import classification_report
     result = tree_model.predict(x_testing)
     df_result = pd.DataFrame(result)
     df_result.columns =["Tertarik"]
```

```
     # Evaluasi Model
     from sklearn.metrics import accuracy_score

     y_predic = tree_model.predict(x_testing)

     acc_secoring = round(accuracy_score(y_predic, y_testing), 3)

     print('Decision tree Accuracy: ', acc_secoring)
```

```
Decision tree Accuracy:  0.826
```

Berikut adalah Confusion Matrix dari hasil evaluasi dari model decision tree terhadap data test:



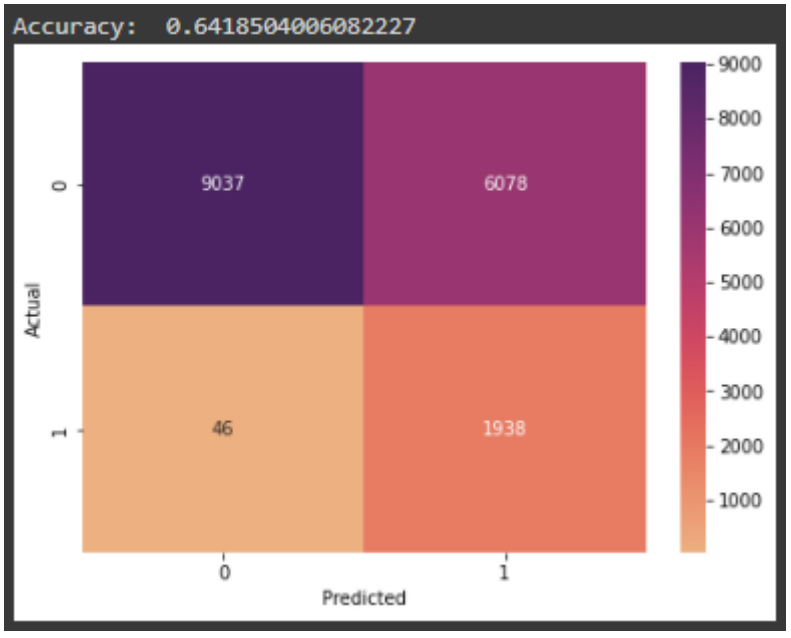
3. Model Naive Bayes

Model Naive Bayes adalah salah satu metode klasifikasi dimana Naive Bayes menggunakan metode probabilitas dan statistik dengan memprediksi peluang masa depan berdasarkan pengalaman sebelumnya.

Pada eksperimen ketiga kami menggunakan Naive Bayes untuk menghitung akurasi dari model dan memprediksi ketertarikan pelanggan untuk membeli mobil dengan menggunakan model Naive Bayes dengan menggunakan library GaussianNB pada sklearn.naive_bayes. Dari hasil eksperimen yang kami coba menggunakan dataset train ke dalam model Naive Bayes, skor akurasi yang didapatkan adalah 0.642.

```
3. Naive Bayes

[102] #import Library Baive bayes
      from sklearn.naive_bayes import GaussianNB
      #buat model naive bayes
      naive_bayes = GaussianNB()
      # latih model dengan fungsi fit
      model = naive_bayes.fit(X_train , y_train)

[103] # Evaluasi Model
      y_predicted = naive_bayes.predict(X_test)
      # uji akurasi model
      acc = round(accuracy_score(y_predicted, y_test), 3)
      print('Naive bayes Accuracy: ', acc)

      Naive bayes Accuracy:  0.642
```

Berikut adalah hasil Confusion Matrix dataset Train yang kami uji coba menggunakan model Naive Bayes,

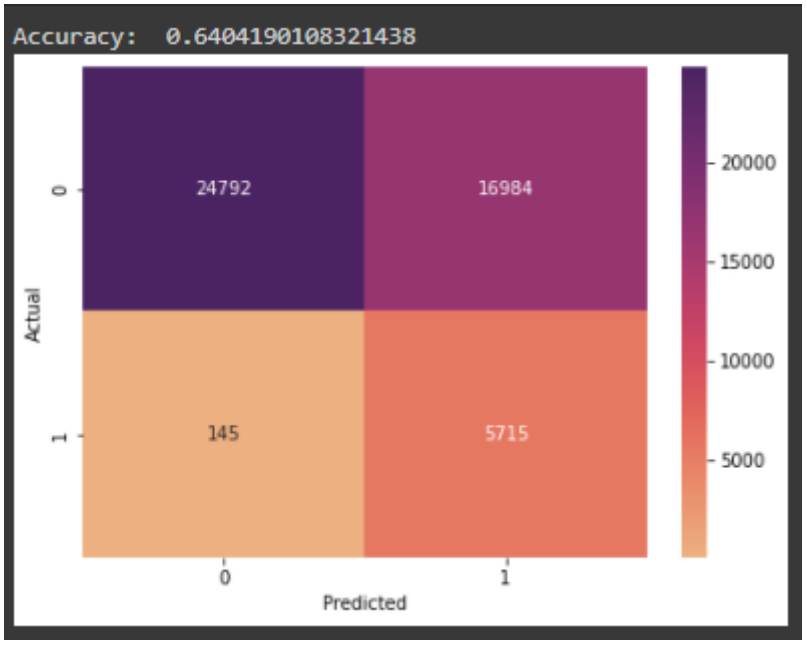


Selanjutnya kami melakukan evaluasi terhadap model yang dibangun dengan menggunakan dataset test yang sudah disediakan pada tugas. Didapat akurasi dari model terhadap data test kita sebesar 0.64

validasi dengan data test

```
[105] #Pisahakan atribut dan label dari data test
      x_testing = df_test.drop("Tertarik", axis=1)
      y_testing = df_test["Tertarik"]

[108] predik = model.predict(x_testing)

      skor = round(accuracy_score(predik, y_testing), 3)

      print('Random Forest Accuracy: ',skor)

      Random Forest Accuracy:  0.64
```

Berikut adalah hasil Confusion Matrix dataset Train yang kami uji coba menggunakan model Naive Bayes,



F. Kesimpulan

Model machine learning ada banyak, penggunaannya pun disesuaikan dengan jenis dataset yang kita miliki. Pada tugas besar Classification ini kami menggunakan 3 jenis model machine learning yaitu, Random Forest, Decision tree, dan Naive Bayes, dari ketiga model tersebut didapatkan akurasi terbesar ketika menggunakan model random forest dengan jumlah n_esimator(pohon) 500 yaitu 0.866 pada saat

evaluasi, dan pada model random forest sendiri jumlah pohon (n_estimator) nya juga berpengaruh terhadap tingkat akurasi data walaupun tidak terlalu signifikan.

Selain itu pada eksperimen yang dilakukan juga didapatkan akurasi terendah adalah ketika menggunakan model naive bayes dengan tingkat akurasi hanya 0.66. Hasil akurasi pada model Naive Bayes menjadi rendah dikarenakan, dataset yang digunakan kurang cocok dengan model Naive Bayes.

Akurasi dari satu model machine learning dapat berbeda-beda pada jenis data yang berbeda, pada data set yang kita miliki model terbaik adalah dengan menggunakan model random forest.

Link GoogleColab :

PreProcessing
https://colab.research.google.com/drive/1JQAPfvcgafUyrf3g4f3b4gp-9IW6qxKR?usp=sharing

Classification:
https://colab.research.google.com/drive/1mm55_WSwcUcrmnT8HIGEg-DbFkeBSDGP?usp=sharing