LAPORAN TUGAS BESAR 2

Mata Kuliah : Pembelajaran Mesin

Dosen Pengampu : Dr. Suyanto, S.T, M. Sc

Supervised Learning (Classification)

Disusun oleh :

QOMARUDIN SIFAK (1301190396)

MUHAMMAD RIFQI ARRAHIM N (1301190425)

Program Studi S1 Informatika – Fakultas Informatika

Universitas Telkom

Jl. Telekomunikasi Terusan Buah Batu, Bandung Indonesia

A. Formulasi Masalah

Formulasi masalah yang akan diselesaikan adalah membuat model supervised learning pada masalah ini kami menggunakan Random Forest untuk memprediksi apakah pelanggan tertarik untuk membeli kendaraan baru atau tidak berdasarkan data pelanggan di dealer.

B. Eksplorasi dan Persiapan Data

a. Mempersiapkan Dataset

Upload Data set

Data Train

[93] 1 df

	id	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	1	Wanita	30.0	1.0	33.0	1.0	< 1 Tahun	Tidak	28029.0	152.0	97.0	0
1	2	Pria	48.0	1.0	39.0	0.0	> 2 Tahun	Pernah	25800.0	29.0	158.0	0
2	3	NaN	21.0	1.0	46.0	1.0	< 1 Tahun	Tidak	32733.0	160.0	119.0	0
3	4	Wanita	58.0	1.0	48.0	0.0	1-2 Tahun	Tidak	2630.0	124.0	63.0	0
4	5	Pria	50.0	1.0	35.0	0.0	> 2 Tahun	NaN	34857.0	88.0	194.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
285826	285827	Wanita	23.0	1.0	4.0	1.0	< 1 Tahun	Tidak	25988.0	152.0	217.0	0
285827	285828	Wanita	21.0	1.0	46.0	1.0	< 1 Tahun	Tidak	44686.0	152.0	50.0	0
285828	285829	Wanita	23.0	1.0	50.0	1.0	< 1 Tahun	Tidak	49751.0	152.0	226.0	0
285829	285830	Pria	68.0	1.0	7.0	1.0	1-2 Tahun	Tidak	30503.0	124.0	270.0	0
285830	285831	Pria	45.0	1.0	28.0	0.0	1-2 Tahun	Pernah	36480.0	26.0	44.0	0

285831 rows × 12 columns

Data Train

Data Test

[94] 1 test

	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
...	...	...	...	...	...	...	...	...	...	...	...
47634	Pria	61	1	46	0	> 2 Tahun	Pernah	31039	124	67	0
47635	Pria	41	1	15	0	1-2 Tahun	Pernah	2630	157	232	0
47636	Pria	24	1	29	1	< 1 Tahun	Tidak	33101	152	211	0
47637	Pria	59	1	30	0	1-2 Tahun	Pernah	37788	26	239	1
47638	Pria	52	1	31	0	1-2 Tahun	Tidak	2630	124	170	0

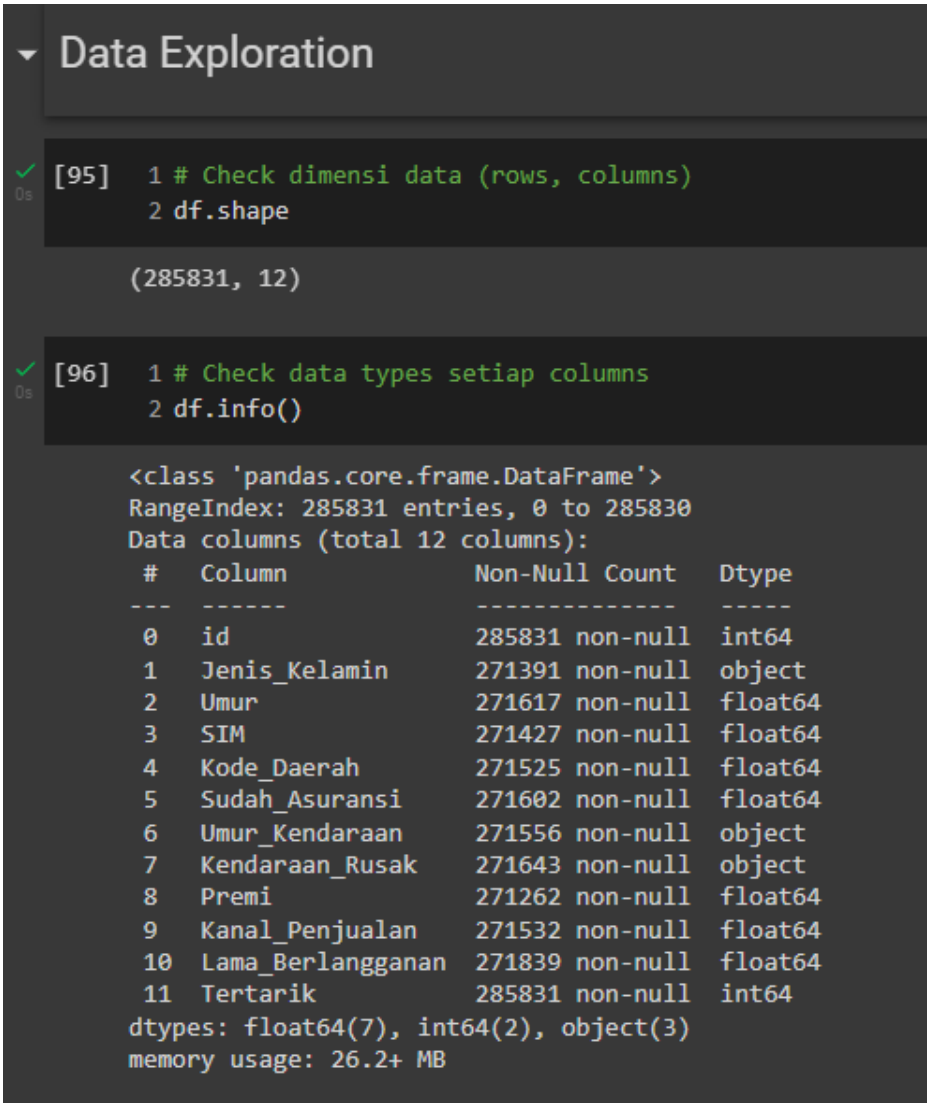
47639 rows × 11 columns

Data Test

Dapat dilihat dari data diatas yaitu data train masih terdapat data yang masih kosong oleh karena itu setelah mempersiapkan data yang kita

lakukan setelah itu adalah melakukan preprocessing agar data tersebut siap digunakan.

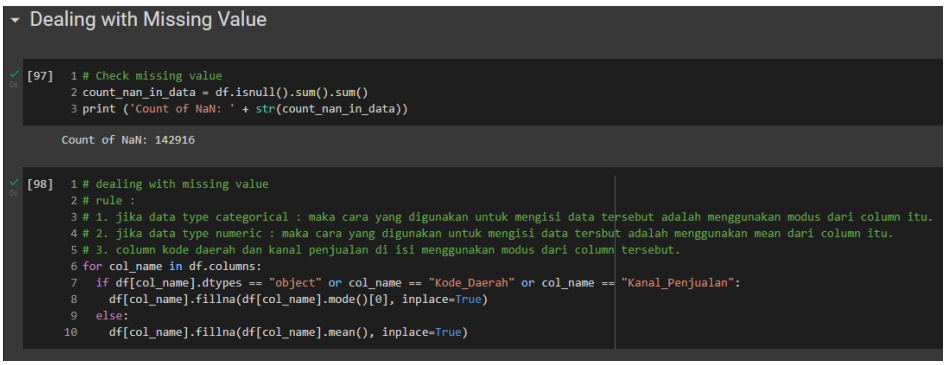
b. Data Exploration



Data Exploration

Disini kita melakukan eksplorasi data agar tahu jenis data apa saja untuk setiap kolomnya.

c. Dealing with Missing Value

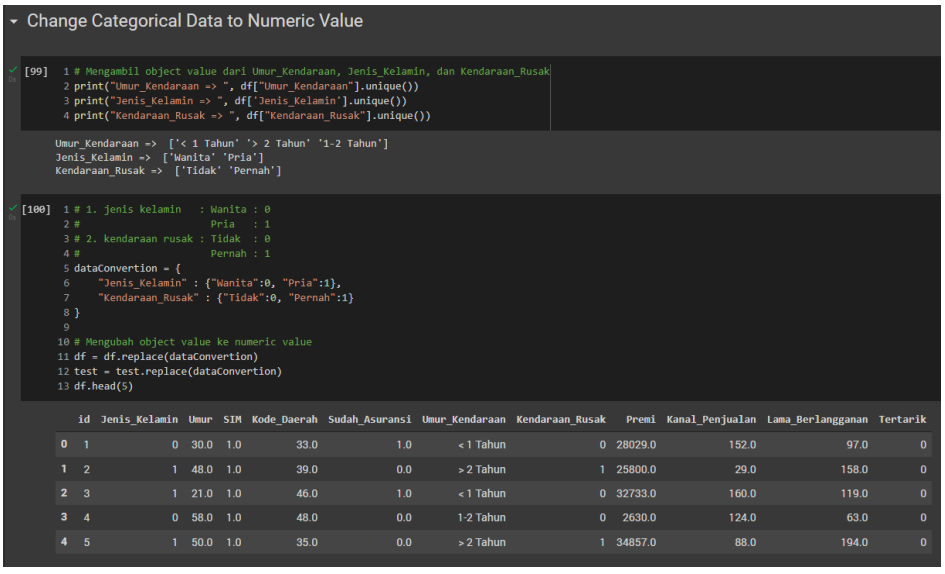


```
Dealing with Missing Value

[97]  1 # Check missing value
      2 count_nan_in_data = df.isnull().sum().sum()
      3 print ('Count of NaN: ' + str(count_nan_in_data))

      Count of NaN: 142916

[98]  1 # dealing with missing value
      2 # rule :
      3 # 1. jika data type categorical : maka cara yang digunakan untuk mengisi data tersebut adalah menggunakan modus dari column itu.
      4 # 2. jika data type numeric : maka cara yang digunakan untuk mengisi data tersbut adalah menggunakan mean dari column itu.
      5 # 3. column kode daerah dan kanal penjualan di isi menggunakan modus dari column tersebut.
      6 for col_name in df.columns:
      7   if df[col_name].dtypes == "object" or col_name == "Kode_Daerah" or col_name == "Kanal_Penjualan":
      8     df[col_name].fillna(df[col_name].mode()[0], inplace=True)
      9   else:
     10     df[col_name].fillna(df[col_name].mean(), inplace=True)
```

Dealing with missing value

Dalam proses ini kita mengisi missing value pada data train setelah mengetahui jumlah dari data yang missing, kami mengisi data tersebut dengan cara menggunakan modus dari kolom tersebut untuk tipe data kategorik dan menggunakan mean atau rata-rata untuk tipe data yang bertipe numerik.

d. Change Categorical Data to Numeric Value



```
Change Categorical Data to Numeric Value

[99]  1 # Mengambil object value dari Umur_Kendaraan, Jenis_Kelamin, dan Kendaraan_Rusak
      2 print("Umur_Kendaraan => ", df["Umur_Kendaraan"].unique())
      3 print("Jenis_Kelamin => ", df['Jenis_Kelamin'].unique())
      4 print("Kendaraan_Rusak => ", df["Kendaraan_Rusak"].unique())

      Umur_Kendaraan =>  ['< 1 Tahun' '> 2 Tahun' '1-2 Tahun']
      Jenis_Kelamin =>  ['Wanita' 'Pria']
      Kendaraan_Rusak =>  ['Tidak' 'Pernah']

[100] 1 # 1. jenis kelamin   : Wanita : 0
      2 #                    Pria   : 1
      3 # 2. kendaraan rusak : Tidak  : 0
      4 #                    Pernah : 1
      5 dataConvertion = {
      6     "Jenis_Kelamin" : {"Wanita":0, "Pria":1},
      7     "Kendaraan_Rusak" : {"Tidak":0, "Pernah":1}
      8 }
      9
     10 # Mengubah object value ke numeric value
     11 df = df.replace(dataConvertion)
     12 test = test.replace(dataConvertion)
     13 df.head(5)
```

	id	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	1	0	30.0	1.0	33.0	1.0	< 1 Tahun	0	28029.0	152.0	97.0	0
1	2	1	48.0	1.0	39.0	0.0	> 2 Tahun	1	25800.0	29.0	158.0	0
2	3	1	21.0	1.0	46.0	1.0	< 1 Tahun	0	32733.0	160.0	119.0	0
3	4	0	58.0	1.0	48.0	0.0	1-2 Tahun	0	2630.0	124.0	63.0	0
4	5	1	50.0	1.0	35.0	0.0	> 2 Tahun	1	34857.0	88.0	194.0	0

Change Categorical Data to Numeric

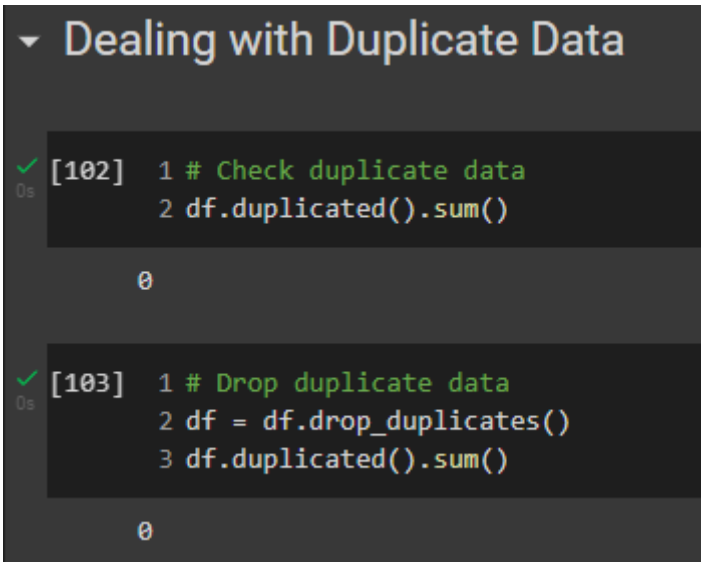
```
[101] 1 # Membagi column Umur_Kendaraan menjadi
      2 #   Umur_Kendaraan_1-2 Tahun
      3 #   Umur_Kendaraan_< 1 Tahun
      4 #   Umur_Kendaraan_> 2 Tahun
      5 df = pd.get_dummies(df)
      6 test = pd.get_dummies(test)
      7 df.head(5)
```

	id	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik	Umur_Kendaraan_1-2 Tahun	Umur_Kendaraan_< 1 Tahun	Umur_Kendaraan_> 2 Tahun
0	1	0	30.0	1.0	33.0	1.0	0	28029.0	152.0	97.0	0	0	1	0
1	2	1	48.0	1.0	39.0	0.0	1	25800.0	29.0	158.0	0	0	0	1
2	3	1	21.0	1.0	46.0	1.0	0	32733.0	160.0	119.0	0	0	1	0
3	4	0	58.0	1.0	48.0	0.0	0	2630.0	124.0	63.0	0	1	0	0
4	5	1	50.0	1.0	35.0	0.0	1	34857.0	88.0	194.0	0	0	0	1

Split Numeric Data to Columns

Pada proses ini program akan mengubah value dari kolom dengan tipe data kategorik lalu diubah menjadi numerik. Kolom yang akan diganti dalam proses ini adalah kolom jenis kelamin, kendaraan rusak, dan umur kendaraan. Untuk kolom umur kendaraan dilakukan split kategorik yang diubah ke kolom.

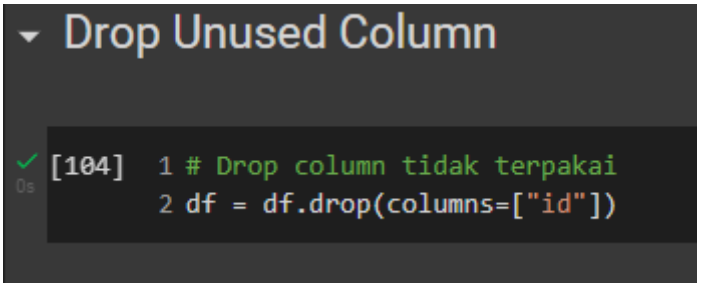
e. Dealing with Duplicate Data



Dealing with Duplicate Data

Dalam proses ini dilakukan cek terlebih dahulu apakah ada ada yang duplikat pada data train ini, jika ada maka hapus duplikasi data tersebut tetapi dalam kasus kami tidak ada data yang duplikat.

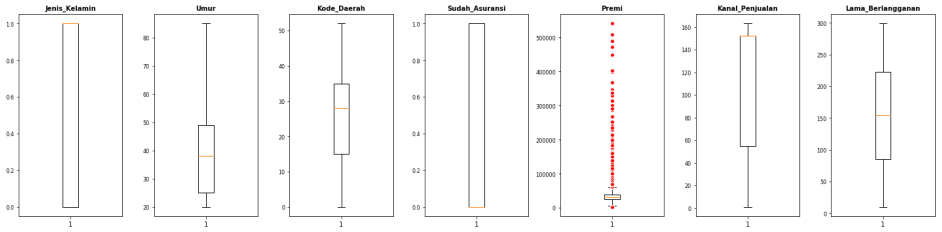
f. Drop Unused Column



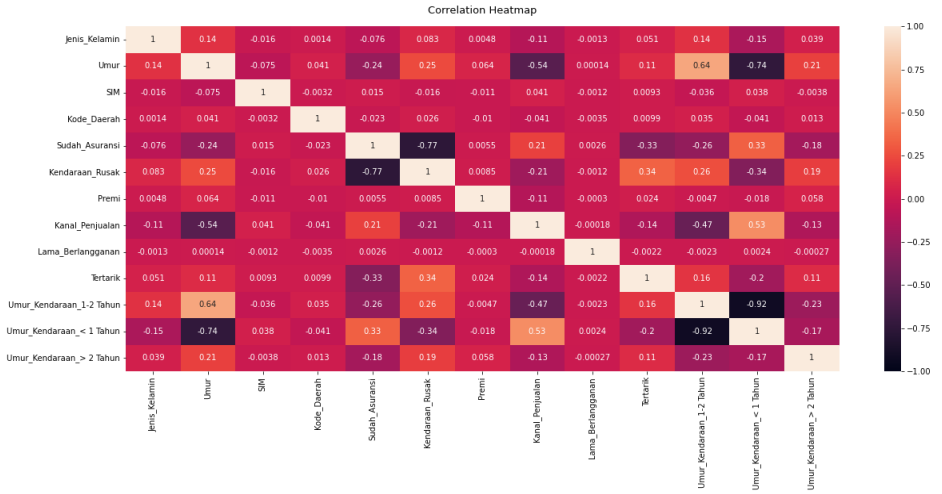
Drop Unused Column

Pada proses ini kami melakukan drop untuk kolom ID karena menurut kami tidak perlu digunakan karena kurang berpengaruh untuk proses klasifikasi.

g. Data Visualization



Show Boxplot



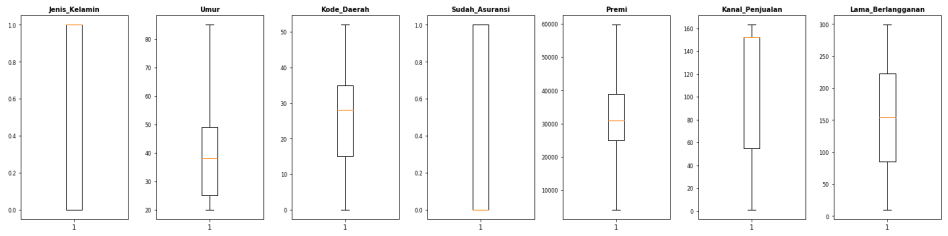
Correlation Heatmap

Dalam proses ini kami melakukan visualisasi data menggunakan boxplot dan correlation heatmap, dapat dilihat di atas boxplot premi terdapat banyak outlier oleh karena itu setelah ini kita harus melakukan proses kepada kolom premi tersebut, pada heatmap dapat dilihat jika korelasinya tinggi maka warna akan terang dan jika warnanya gelap maka semakin tidak ada korelasi.

h. Dealing with Outlier

```
Dealing with Outlier

[106]  1 # Menghapus outlier pada coloumn Premi
       2 Q1 = df["Premi"].quantile(0.25)
       3 Q3 = df["Premi"].quantile(0.75)
       4 IQR=Q3-Q1
       5 lower_whisker = Q1 - (1.5*IQR)
       6 upper_whisker = Q3 + (1.5*IQR)
       7 df["Premi"] = df["Premi"].mask(df["Premi"] >upper_whisker, upper_whisker)
       8 df["Premi"] = df["Premi"].mask(df["Premi"] <lower_whisker, lower_whisker)
```

Drop Outlier Premi Column



Show Boxplot after Drop Outlier

Karena kolom Premi mempunyai outlier yang sangat banyak yang dapat mengakibatkan bias, oleh karena itu kami menghilangkan outlier dari premi agar dataset tersebut menjadi tidak bias. Setelah itu melakukan plot lagi apakah masih ada outlier di kolom lain.

i. Data Split

```
Data Split

[108]  1 # Splitting Data menjadi X, Y untuk data Train dan X_test, Y_test untuk data Test
       2 X = df.drop(["Tertarik"],axis =1)
       3 Y = df['Tertarik']
       4 X_test = test.drop(["Tertarik"],axis =1)
       5 Y_test = test['Tertarik']
```

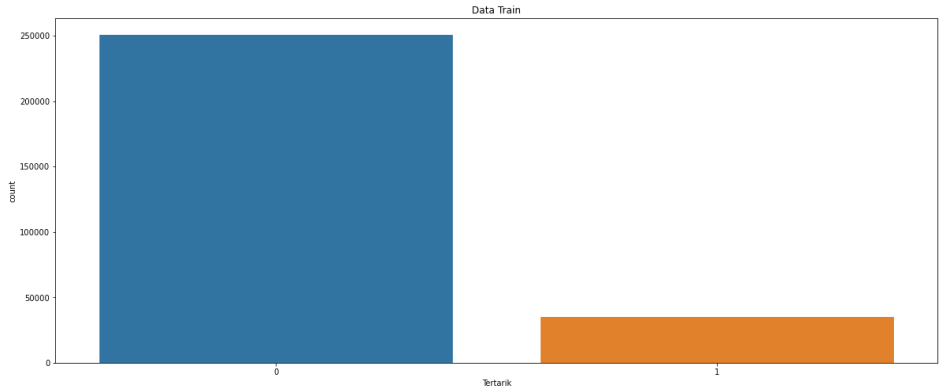
Data Split

Data split dilakukan karena pada Supervised Learning membutuhkan X dan Y dimana X merupakan feature dari dataset tersebut, dan Y merupakan Label dari dataset tersebut.

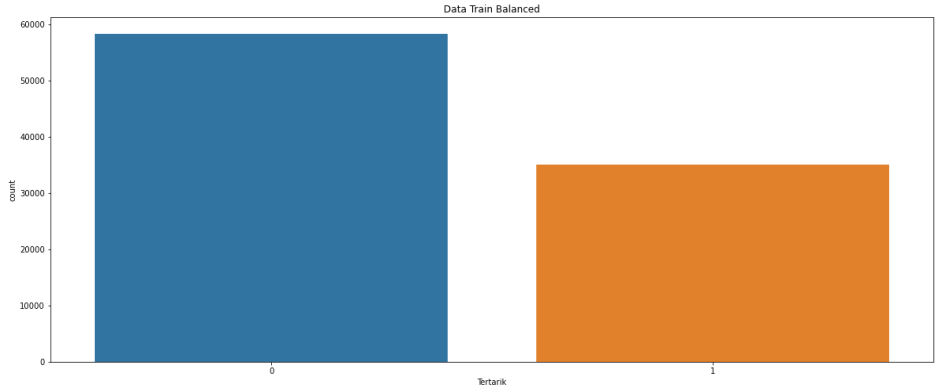
j. Data Balancing

```
Data Balancing

[110]  1 over = RandomOverSampler(sampling_strategy=0.4)
       2 X_balanced, Y_balanced = over.fit_resample(X, Y)
       3 under = RandomUnderSampler(sampling_strategy=0.6)
       4 X_balanced, Y_balanced = under.fit_resample(X, Y)
```

Data Balancing



Sebelum Data Balancing

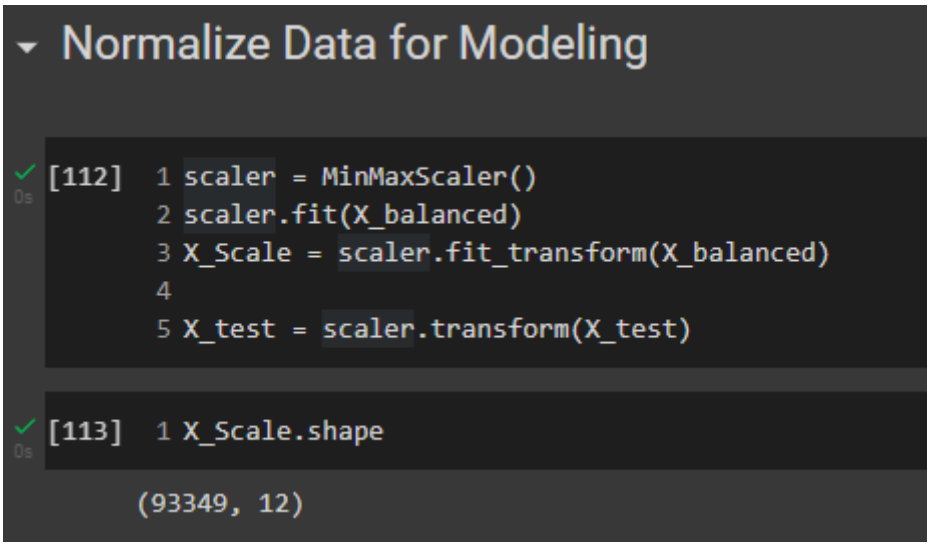


Sesudah Data Balancing

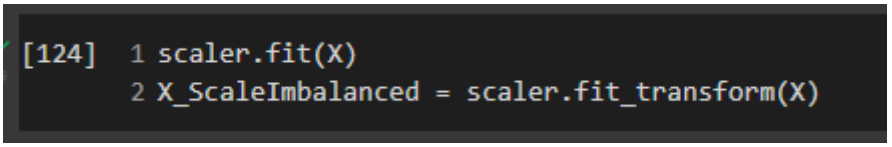
Untuk mengurangi Imbalance pada dataset, kami melakukan data balancing dengan menerapkan metode oversampling dan undersampling dari library randomoversampler. Oversampling untuk menambah minority data lalu setelah itu kami melakukan undersampling untuk mengurangi majority data. Pada proses ini kami melakukan

oversampling terlebih dahulu sebanyak 40% setelah itu dilakukan undersampling sebanyak 60%, dapat dilihat barplot sebelum dan sesudah data balancing, tetapi kami akan melakukan eksperimen apakah data balancing berpengaruh pada klasifikasi.

k. Normalize Data for Modeling



Scaling Data Train Balanced

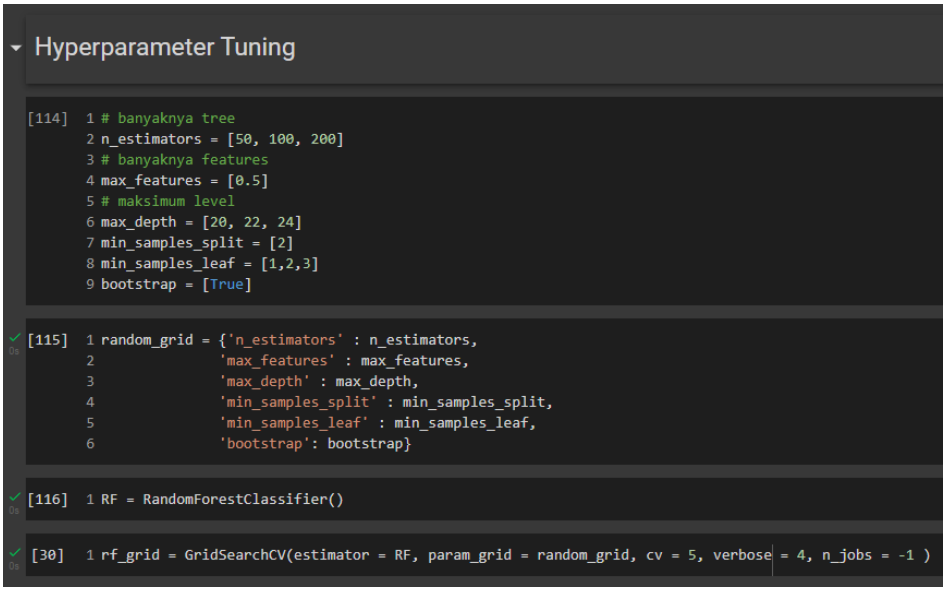


Scaling Data Train Imbalanced

Pada proses ini kami melakukan scaling data agar rangenya menjadi 0-1 yang bertujuan untuk menyamakan range nilai tiap datanya untuk setiap kolom.

C. Pemodelan

a. Random Forest



Hyperparameter Tuning Random Forest

Pada proses pemodelan kami yang pertama menggunakan ensemble Random Forest dalam proses memilih Hyperparameter Tuning untuk mencari parameter terbaik kami menggunakan n_estimators dengan ketentuan 50, 100, 200. Lalu untuk max_feature kami menggunakan 50% dari feature dari seluruh feature yang ada, untuk max_depthnya kami menentukan dari 20, 22, 24. Untuk min_sample_split menggunakan static 2 dan untuk min_sample_leaf menggunakan ketentuan 1, 2, 3.

Setelah itu diperiksa setiap kemungkinan dataset yang keluar pada splitting menggunakan cross validation untuk mendapatkan hasil yang terbaik dari kemungkinan dataset yang ada, disini kami menggunakan 5 kali cross validation untuk model Random Forest ini dan menggunakan grid search untuk melakukan proses pencarian parameter terbaik.

Setelah melakukan proses seleksi parameter terbaik pada model, kemudian memilih parameter tersebut pada model dan dilakukan pelatihan dengan proses model tersebut.

b. Decision Tree

Hyperparameter Tuning Decision Tree

Pada proses pemodelan kami yang kedua kami menggunakan Decision Tree untuk Hyperparameter Tuning nya sendiri kami menggunakan max_feature yang cukup berbeda dari model Random Forest yaitu menggunakan sqrt dan log2, untuk max_depth kami menentukan dari 1, 2, 3, 4, 5, 6, 7, 8, dan 9, untuk min_sample_split kami tentukan dari 3, 4, 5, 6, 7, 8, 9, 10, dan 11, untuk min_sample_leaf kami tentukan 1, 2, 3, 4, 5, 6, 7, 8 , 9, dan 10. Untuk criterion kami menggunakan gini dan entropy.

Setelah itu diperiksa setiap kemungkinan dataset yang keluar pada splitting menggunakan cross validation, disini kami menggunakan 3 kali cross validation untuk model Decision Tree ini dan menggunakan grid search untuk melakukan proses pencarian parameter terbaik.

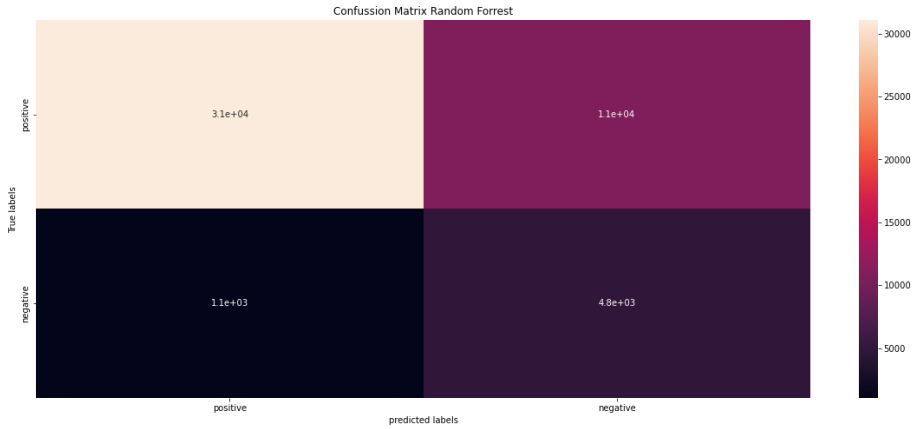
Jika sudah melakukan proses seleksi parameter terbaik pada model, kemudian memilih parameter tersebut pada model dan dilakukan pelatihan dengan proses model tersebut.

D. Evaluasi

Pada tahap ini, kami melakukan pengecekan nilai akurasi pada model yang sudah kami latih sebelumnya. Pengecekan nilai akurasi ini dilakukan dengan cara melakukan prediksi model yang telah dilatih dengan dataset yang sudah di balancing dan data set yang imbalance.

```
[119]  1 y_pred=RF.predict(X_test)

       1 print('Accuracy', metrics.accuracy_score(Y_test, y_pred))

       Accuracy 0.7533743361531519

[144]  1 print("Jumlah Prediksi Tidak Tertarik",pd.DataFrame(y_pred).value_counts()[0])
       2 print("Jumlah Prediksi Tertarik",pd.DataFrame(y_pred).value_counts()[1])

       Jumlah Prediksi Tidak Tertarik 32131
       Jumlah Prediksi Tertarik 15508
```

Gambar diatas merupakan hasil akurasi dari prediksi model Random Forest Tree pada dataset yang sudah di balancing. Dapat dilihat mendapatkan akurasi sebesar 75.3% dengan jumlah prediksi tidak tertarik 32.131 dan jumlah prediksi tertarik 15.508. Setelah itu kami melakukan evaluasi hasil prediksi menggunakan Confusion Matrix dan Classification Report. Confusion Matrix ini bertujuan untuk melihat nilai True Positive (TP), True Negative (TN), False Positive (FP), dan False Negative (FN). Classification Report bertujuan untuk menampilkan precision, recall, dan f1-score dari hasil prediksi. Berikut ini merupakan Confusion Matrix dan Classification Report dari prediksi model Random Forest.



Confusion Matrix Random Forest

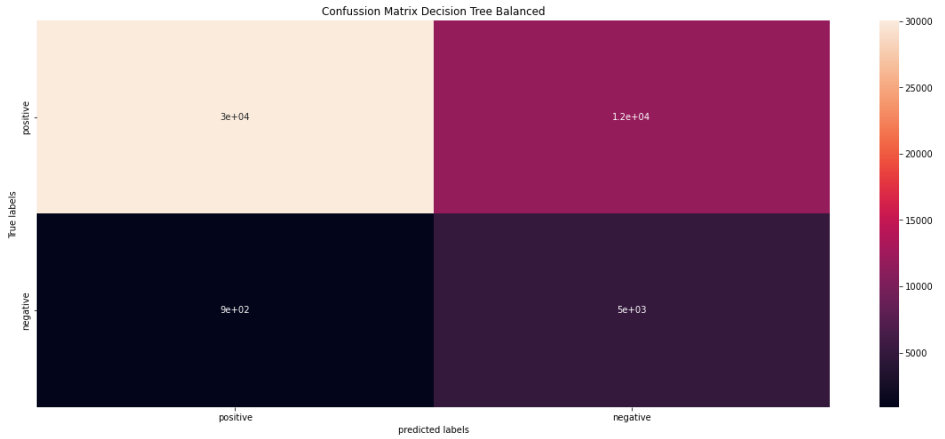
```
[123]  1 print(metrics.classification_report(Y_test, y_pred, labels=[False, True]))

                precision    recall  f1-score   support

         False       0.97      0.74      0.84     41778
          True       0.31      0.82      0.45      5861

      accuracy                           0.75     47639
     macro avg       0.64      0.78      0.65     47639
  weighted avg       0.89      0.75      0.79     47639
```

Classification Report Random Forest

Berdasarkan Classification Report diatas, dapat dilihat bahwa hasil prediksi model Random Forest memiliki f1-score 84% dengan support 41.778 untuk prediksi False atau tidak tertarik dan memiliki f1-score 45% dengan

support 5.861 untuk prediksi True atau tertarik sehingga mendapatkan akurasi 75% dari 47.639 sample test.

Berikut ini merupakan hasil prediksi, Confusion Matrix, dan Classification Report untuk Decision Tree.



Confusion Matrix Decision Tree

```
[179]  1 print(metrics.classification_report(Y_test, y_predict_dct, labels=[False, True]))
0s
                  precision    recall  f1-score   support

           False       0.97      0.72      0.83     41778
            True       0.30      0.85      0.44      5861

        accuracy                           0.73     47639
       macro avg       0.63      0.78      0.63     47639
    weighted avg       0.89      0.73      0.78     47639
```

Classification Report Decision Tree

Berdasarkan Classification Report diatas, dapat dilihat bahwa hasil prediksi model Decision Tree memiliki f1-score 83% dengan support 41.778 untuk prediksi False atau tidak tertarik dan memiliki f1-score 44% dengan support 5.861 untuk prediksi True atau tertarik sehingga mendapatkan akurasi 73% dari 47.639 sample test.

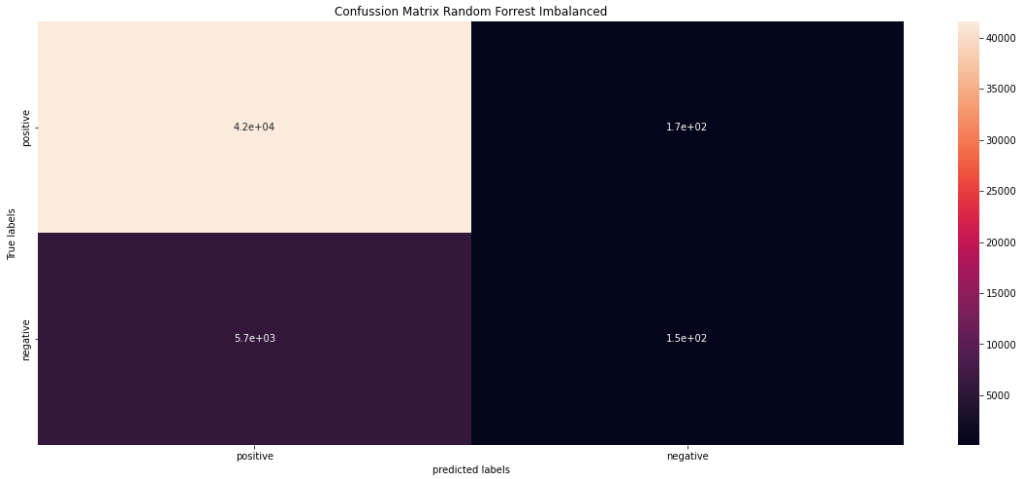
E. Eksperimen

Untuk eksperimen yang coba kami lakukan adalah dengan membandingkan model yang dilatih dengan data yang imbalance dengan data yang telah di balancing. Pertama yang kami lakukan untuk melakukan eksperimen ini adalah memuat kembali data train yang sudah di balancing dan data test imbalance. Setelah itu kami melakukan test pada model Random Forest dan Decision Tree.

Setelah melakukan pembuatan model dan melatihnya, kelompok kami melakukan evaluasi pada model yang baru dibuat dengan cara melakukan prediksi pada data test yang imbalance. Berikut adalah hasil prediksi, Confusion Matrix dan Classification Report untuk Random Forest dengan data imbalance

```
[147]  1 y_pred_imbalanced=RF_imbalanced.predict(X_test)

[148]  1 print('Accuracy', metrics.accuracy_score(Y_test, y_pred_imbalanced))

       Accuracy 0.8764877516320662

[149]  1 print("Jumlah Prediksi Tidak Tertarik",pd.DataFrame(y_pred_imbalanced).value_counts()[0])
       2 print("Jumlah Prediksi Tertarik",pd.DataFrame(y_pred_imbalanced).value_counts()[1])

       Jumlah Prediksi Tidak Tertarik 47314
       Jumlah Prediksi Tertarik 325
```

Prediction Random Forest Imbalance



Confusion Matrix Random Forest Imbalance

```
[150]  1 print(metrics.classification_report(Y_test, y_pred_imbalanced, labels=[False, True]))

              precision    recall  f1-score   support

       False       0.88      1.00      0.93     41778
        True       0.46      0.03      0.05      5861

    accuracy                           0.88     47639
   macro avg       0.67      0.51      0.49     47639
weighted avg       0.83      0.88      0.83     47639
```

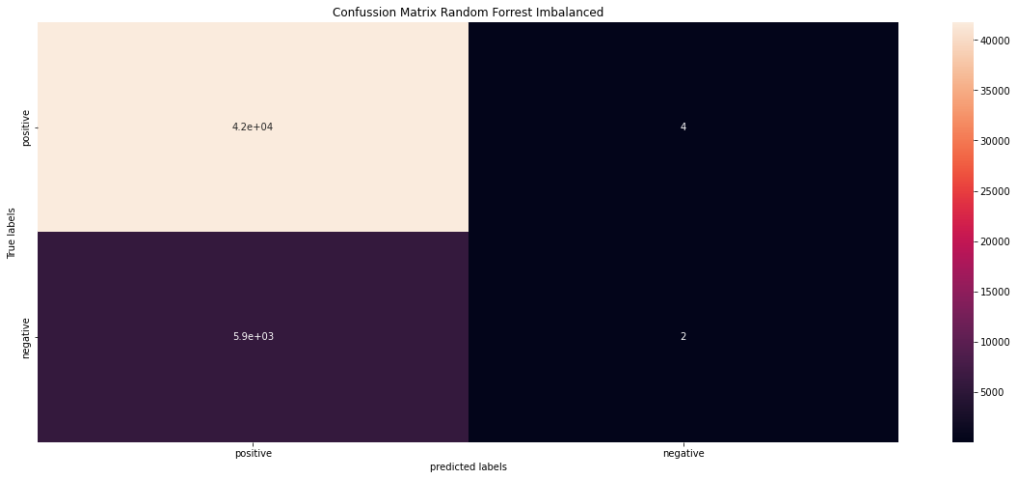
Classification Report Random Forest Imbalance

Dapat dilihat mendapatkan akurasi sebesar 87.6% dengan jumlah prediksi tidak tertarik 47.314 dan jumlah prediksi tertarik 325. Berdasarkan classification report, hasil prediksi model pada dataset imbalance untuk random forest memiliki f1-score 93% dengan support 41.778 untuk prediksi false atau tidak tertarik dan f1-score 5% dengan support 5,861 untuk prediksi true atau tertarik sehingga total akurasi dari semua prediksi menjadi 88% dari 47.639 sample test.

Berikut adalah hasil prediksi, Confusion Matrix dan Classification Report untuk Decision Tree dengan data imbalance

```
[173]  1 y_predict_dct_imbalanced = dct_model_imbalanced.predict(X_test)

[174]  1 print('Accuracy', metrics.accuracy_score(Y_test, y_predict_dct_imbalanced))

       Accuracy 0.8769285669304562

[175]  1 print("Jumlah Prediksi Tidak Tertarik",pd.DataFrame(y_predict_dct_imbalanced).value_counts()[0])
       2 print("Jumlah Prediksi Tertarik",pd.DataFrame(y_predict_dct_imbalanced).value_counts()[1])

       Jumlah Prediksi Tidak Tertarik 47633
       Jumlah Prediksi Tertarik 6
```

Classification Report Decision Tree Imbalance



Classification Report Decision Tree Imbalance

```
[177]  1 print(metrics.classification_report(Y_test, y_predict_dct_imbalanced, labels=[False, True]))

                  precision    recall  f1-score   support

           False       0.88      1.00      0.93     41778
            True       0.33      0.00      0.00      5861

        accuracy                           0.88     47639
       macro avg       0.61      0.50      0.47     47639
    weighted avg       0.81      0.88      0.82     47639
```

Classification Report Decision Tree Imbalance

Dapat dilihat mendapatkan akurasi sebesar 87.6% dengan jumlah prediksi tidak tertarik 47.633 dan jumlah prediksi tertarik 6. Berdasarkan classification report, hasil prediksi model pada dataset imbalance untuk decision tree memiliki f1-score 93% dengan support 41.778 untuk prediksi false atau tidak tertarik dan f1-score kurang dari 1% dengan support 5,861 untuk prediksi true atau tertarik sehingga total akurasi dari semua prediksi menjadi 88% dari 47.639 sample test.

F. Kesimpulan

Berdasarkan hasil dari evaluasi dan eksperimen yang kami lakukan, Model yang dilatih dengan balancing lebih bagus untuk melakukan klasifikasi dibandingkan model yang tidak dilakukan balancing atau imbalance. Balancing sangat mempengaruhi pada hasil akhir prediksi model. Dapat dilihat pada model yang dilatih menggunakan balancing dan tidak dilakukan balancing. Untuk Random Forest dengan data balancing akurasi prediksinya lebih kecil dari pada Random Forest dengan data imbalance dengan skor 75% dengan data balancing dan 87% dengan data Imbalance. Begitu pula dengan model Decision Tree, untuk data balancing lebih kecil daripada data imbalance dengan skor 73% dengan data balancing dan 88% dengan data Imbalance.

Data yang di balancing lebih mudah mengklasifikasikan daripada data yang tidak di balancing contohnya pada Random Forest ketika di balancing dapat dilihat mendapatkan akurasi sebesar 75.3% dengan jumlah prediksi tidak tertarik 32.131 dan jumlah prediksi tertarik 15.508. Sedangkan Random Forest tanpa di balancing mendapat akurasi sebesar f1-score 93% dengan support 41.778 untuk prediksi false atau tidak tertarik dan f1-score 5% dengan support 5,861. Hal ini juga ditemukan pada model Decision Tree, kemungkinan hal ini dikarenakan data imbalance tersebut terlalu banyak label False atau tidak Tertarik sehingga kurangnya data train untuk label True atau Tertarik. Oleh karena itu, model yang dilatih dengan data balancing lebih bagus daripada model yang dilatih dengan data imbalance, dapat juga dibuktikan dengan konsistensi dari model Random Forest dengan data balancing dan model Decision Tree dengan data balancing hasilnya hampir sama.

G. Source Code

https://colab.research.google.com/drive/15yGpQUtBIvZu7U2BDwpwNm1xjUzl5quI?usp=sharing