

Challenge 2 - Studi Kasus Machine Learning

Oleh Itsna Hikmatul Maula

Alur Pengerjaan Project

Studi Kasus Machine Learning ini adalah untuk membuat model yang memprediksi customer yang churn (pindah ke provider lain). Adapun alur pengerjaan secara garis besar adalah sebagai berikut:

1. Data preprocessing
2. Melihat seluruh informasi data baik secara statistik deskriptif atau membuat diagram untuk mendapatkan insight
3. Membuat model machine learning algoritma klasifikasi (Decision Tree, Logistic Regression, dan KNN)
4. Melihat nilai akurasi, confusion matrix, dan classification report pada masing-masing model yang sudah dibuat
5. Implementasi model machine learning dengan data test yang sudah disediakan

Alur Pengerjaan Project

1. Lakukan import libraries yang diperlukan (termasuk import library untuk upload file dari komputer)

```
✓ [1] import matplotlib
1 d      import pandas as pd
      import seaborn as sns
      import matplotlib.pyplot as plt
```


```
✓ [2] from google.colab import files
17 d      uploaded = files.upload()

Choose Files train.csv
• train.csv(text/csv) - 387621 bytes, last modified: 10/9/2022 - 100% done
Saving train.csv to train.csv
```

2. Melakukan deklarasi variabel pada file csv yang akan digunakan dengan df. Lalu mengecek data dengan df.head() untuk mengetahui gambaran bentuk data , mengecek tipe data dengan df.info(), mengecek banyaknya data null dengan df.isna().null(), mengecek data duplikat dengan df.duplicated().sum()

✓ [4] # 2. Cek data (apakah bersih atau belum) untuk melakukan data preprocessing sesuai kebutuhan
0 d df.head()

	state	account_length	area_code	international_plan	voice_mail_plan	number_vmail_messages	total_day_minutes	total_day_calls	total_charges
0	OH	107	area_code_415	no	yes	26	161.6	123	11.44
1	NJ	137	area_code_415	no	no	0	243.4	114	18.89
2	OH	84	area_code_408	yes	no	0	299.4	71	15.37
3	OK	75	area_code_415	yes	no	0	166.7	113	10.90
4	MA	121	area_code_510	no	yes	24	218.2	88	14.51



2. Melakukan deklarasi variabel 'df' pada file csv yang akan digunakan. Lalu mengecek data dengan `df.head()` untuk mengetahui gambaran bentuk data dengan `df.head()`, mengecek tipe data dengan `df.info()`, mengecek banyaknya data null dengan `df.isna().null()`, mengecek data duplikat dengan `df.duplicated().sum()`

```
✓ [6] df.isna().sum()
0d
state 0
account_length 0
area_code 0
international_plan 0
voice_mail_plan 0
number_vmail_messages 0
total_day_minutes 0
total_day_calls 0
total_day_charge 0
total_eve_minutes 0
total_eve_calls 0
total_eve_charge 0
total_night_minutes 0
total_night_calls 0
total_night_charge 0
total_intl_minutes 0
total_intl_calls 0
total_intl_charge 0
number_customer_service_calls 0
churn 0
dtype: int64
```

```
✓ [7] df.duplicated().sum()
0d
0
```

```
▶ df.info()
[+] <class 'pandas.core.frame.DataFrame'>
RangeIndex: 4250 entries, 0 to 4249
Data columns (total 20 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   state                                4250 non-null   object
1   account_length                      4250 non-null   int64
2   area_code                           4250 non-null   object
3   international_plan                  4250 non-null   object
4   voice_mail_plan                     4250 non-null   object
5   number_vmail_messages               4250 non-null   int64
6   total_day_minutes                   4250 non-null   float64
7   total_day_calls                     4250 non-null   int64
8   total_day_charge                    4250 non-null   float64
9   total_eve_minutes                   4250 non-null   float64
10  total_eve_calls                     4250 non-null   int64
11  total_eve_charge                    4250 non-null   float64
12  total_night_minutes                 4250 non-null   float64
13  total_night_calls                   4250 non-null   int64
14  total_night_charge                  4250 non-null   float64
15  total_intl_minutes                  4250 non-null   float64
16  total_intl_calls                    4250 non-null   int64
17  total_intl_charge                   4250 non-null   float64
18  number_customer_service_calls       4250 non-null   int64
19  churn                              4250 non-null   object
dtypes: float64(8), int64(7), object(5)
memory usage: 664.2+ KB
```

3. Setelah dirasai tidak menemukan missing value dan data duplikat, selanjutnya adalah melihat data dengan statistika deskriptif dan diagram plot untuk mendapatkan insight dari dataset ini.



```
df.describe()
```

	account_length	number_vmail_messages	total_day_minutes	total_day_calls	total_day_charge	total_eve_minutes	total_eve_calls	total_eve_charge
count	4250.000000	4250.000000	4250.000000	4250.000000	4250.000000	4250.000000	4250.000000	4250.000000
mean	100.236235	7.631765	180.259600	99.907294	30.644682	200.173906	100.176471	100.176471
std	39.698401	13.439882	54.012373	19.850817	9.182096	50.249518	19.908591	19.908591
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	73.000000	0.000000	143.325000	87.000000	24.365000	165.925000	87.000000	87.000000
50%	100.000000	0.000000	180.450000	100.000000	30.680000	200.700000	100.000000	100.000000
75%	127.000000	16.000000	216.200000	113.000000	36.750000	233.775000	114.000000	114.000000
max	243.000000	52.000000	351.500000	165.000000	59.760000	359.300000	170.000000	170.000000

4. Langkah selanjutnya adalah mengetahui kolom apa saja yang memiliki data numeric. Lalu, kolom-kolom tersebut diberi standar/disesuaikan menggunakan scaler sehingga akan diperoleh value tiap kolom dengan minimal 0 dan maksimal 1.

```
✓ [17] numeric_column = ['account_length', 'number_vmail_messages', 'total_day_minutes',  
0 d      'total_day_calls', 'total_day_charge', 'total_eve_minutes',  
      'total_eve_calls', 'total_eve_charge', 'total_night_minutes',  
      'total_night_calls', 'total_night_charge', 'total_intl_minutes',  
      'total_intl_calls', 'total_intl_charge',  
      'number_customer_service_calls']
```

```
✓ [18] from sklearn.preprocessing import MinMaxScaler  
0 d      scaler = MinMaxScaler()
```

```
✓ [19] df[numeric_column] = scaler.fit_transform(df[numeric_column])  
0 d
```

```
✓ [20] print(df[numeric_column].describe().T[['min', 'max']])  
0 d
```

	min	max
account_length	0.0	1.0
number_vmail_messages	0.0	1.0
total_day_minutes	0.0	1.0
total_day_calls	0.0	1.0
total_day_charge	0.0	1.0
total_eve_minutes	0.0	1.0
total_eve_calls	0.0	1.0
total_eve_charge	0.0	1.0
total_night_minutes	0.0	1.0
total_night_calls	0.0	1.0
total_night_charge	0.0	1.0
total_intl_minutes	0.0	1.0
total_intl_calls	0.0	1.0
total_intl_charge	0.0	1.0
number_customer_service_calls	0.0	1.0

5. Langkah selanjutnya adalah mengubah kolom-kolom categorical menjadi kolom-kolom yang memiliki value numerik. Langkah ini dilakukan karena machine learning dapat bekerja jika seluruh data bernilai numerik. Untuk melakukannya, digunakan library Encoder untuk konversi numerik secara instan.

```
✓ [23] LE = LabelEncoder()
0d df['state'] = LE.fit_transform(df['state'])
print(LE.classes_)
print(np.sort(df['state'].unique()))
print('')

LE = LabelEncoder()
df['area_code'] = LE.fit_transform(df['area_code'])
print(LE.classes_)
print(np.sort(df['area_code'].unique()))
print('')

LE = LabelEncoder()
df['voice_mail_plan'] = LE.fit_transform(df['voice_mail_plan'])
print(LE.classes_)
print(np.sort(df['voice_mail_plan'].unique()))
print('')

LE = LabelEncoder()
df['international_plan'] = LE.fit_transform(df['international_plan'])
print(LE.classes_)
print(np.sort(df['international_plan'].unique()))
print('')
```

5. Hasil dari encoder categorical value pada kolom 'state', 'area_code', 'international_plan', 'voice_mail_plan', 'churn' adalah sebagai berikut.

```
✓ [23] df['churn'] = LE.fit_transform(df['churn'])
0d      print(LE.classes_)
      print(np.sort(df['churn'].unique()))
      print('')

['AK' 'AL' 'AR' 'AZ' 'CA' 'CO' 'CT' 'DC' 'DE' 'FL' 'GA' 'HI' 'IA' 'ID'
 'IL' 'IN' 'KS' 'KY' 'LA' 'MA' 'MD' 'ME' 'MI' 'MN' 'MO' 'MS' 'MT' 'NC'
 'ND' 'NE' 'NH' 'NJ' 'NM' 'NV' 'NY' 'OH' 'OK' 'OR' 'PA' 'RI' 'SC' 'SD'
 'TN' 'TX' 'UT' 'VA' 'VT' 'WA' 'WI' 'WV' 'WY']
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50]

['area_code_408' 'area_code_415' 'area_code_510']
[0 1 2]

['no' 'yes']
[0 1]

['no' 'yes']
[0 1]

['no' 'yes']
[0 1]
```

6. Dataset sudah siap digunakan! Selanjutnya, dibuat variabel X dan y untuk membuat model. Variabel X adalah features yang ada pada dataset sedangkan variabel y adalah variabel dependent yang hasilnya dipengaruhi oleh features (hasil akhir, yes or no, true or false). Untuk membuat variabel X, kita cukup drop/hapus kolom yang memuat hasil akhir pada dataset dalam hal ini adalah kolom 'churn'

```
✓ [24] X = df.drop(['churn'], axis = 1)
    0 d y = df['churn']
        print("Shape of X", X.shape)
        print("Shape of y:", y.shape)
```

```
Shape of X (4250, 19)
```

```
Shape of y: (4250,)
```

7. Selanjutnya, dataset train yang sudah diolah perlu dipisahkan untuk membuat model. Dalam hal ini, dataset akan dipecah menjadi 80% data train dan 20% data test.

```
✓ [25] from sklearn.model_selection import train_test_split
```

0 d

```
✓ 0 d ▶ X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

```
print("Shape of X_train :", X_train.shape)
print("Shape of y_train :", y_train.shape)
print("Shape of X_test :", X_test.shape)
print("Shape of y_test :", y_test.shape)
```

```
Shape of X_train : (3400, 19)
Shape of y_train : (3400,)
Shape of X_test : (850, 19)
Shape of y_test : (850,)
```

Membuat Machine Learning

8. Setelah deklarasi variabel `X_train`, `X_test`, `y_train`, dan `y_test`, dibuat model machine learning dengan algoritma supervised learning yaitu Decision Tree, Logistic Regression, dan KNN. Algoritma supervised learning ini dapat digunakan untuk prediksi data yang memiliki hasil klasifikasi true or false. Dalam dataset train yang digunakan dalam pengerjaan project ini, model digunakan untuk memprediksi customer yang melakukan churn (pindah ke provider lain) atau tidak. Dalam tahap ini, akan diketahui skor Accuracy, Confussion Metrix, dan Classification Report

Adapun libraries yang dibutuhkan dalam tahapan ini adalah:

1. `from sklearn.tree import DecisionTreeClassifier`
2. `from sklearn.linear_model import LogisticRegression`
3. `from sklearn.neighbors import KNeighborsClassifier`
4. `from sklearn.metrics import confusion_matrix, classification_report`

9. Membuat model Decision Tree

```
✓ [28] model = DecisionTreeClassifier()  
0d      model = model.fit(X_train,y_train)  
      y_pred = model.predict(X_test)
```

```
✓ # Hasil evaluasi akurasi model  
0d print('Training Accuracy :', model.score(X_train, y_train))  
    print('Testing Accuracy :', model.score(X_test, y_test))
```

```
Training Accuracy : 1.0  
Testing Accuracy : 0.928235294117647
```

```
✓ [31] print('\nConfusion matrix:')  
0d      cm = confusion_matrix(y_test, y_pred)  
      print(cm)
```

```
Confusion matrix:  
[[699  36]  
 [ 25  90]]
```

9. Membuat model Decision Tree

```
✓ [32] print('\nClassification report:')  
0d cr = classification_report(y_test, y_pred)  
    print(cr)
```

```
Classification report:  
              precision    recall  f1-score   support  
  
      0           0.97       0.95      0.96         735  
      1           0.71       0.78      0.75         115  
  
   accuracy              0.93         850  
  macro avg           0.84       0.87      0.85         850  
weighted avg           0.93       0.93      0.93         850
```

10. Membuat model Linear Regression

✓
1 d



```
logreg = LogisticRegression()  
logreg = logreg.fit(X_train,y_train)  
y_pred = logreg.predict(X_test)
```

/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,

10. Membuat model Linear Regression

```
✓ [35] print('Training Accuracy :', logreg.score(X_train, y_train))  
0d print('Testing Accuracy :', logreg.score(X_test, y_test))
```

```
Training Accuracy : 0.8661764705882353  
Testing Accuracy : 0.8788235294117647
```

```
✓ [36] print('\nConfusion matrix')  
0d cm = confusion_matrix(y_test, y_pred)  
print(cm)
```

```
Confusion matrix  
[[727   8]  
 [ 95 20]]
```

```
✓ [37] print('\nClassification report')  
0d cr = classification_report(y_test, y_pred)  
print(cr)
```

```
Classification report
```

	precision	recall	f1-score	support
0	0.88	0.99	0.93	735
1	0.71	0.17	0.28	115
accuracy			0.88	850
macro avg	0.80	0.58	0.61	850
weighted avg	0.86	0.88	0.85	850

11. Membuat model KNN

```
✓ [38] from sklearn.neighbors import KNeighborsClassifier  
0d
```

```
knn = KNeighborsClassifier(n_neighbors=5)  
knn = knn.fit(X_train,y_train)  
y_pred = knn.predict(X_test)
```

```
✓ [39] print('Training Accuracy :', knn.score(X_train, y_train))  
0d      print('Testing Accuracy :', knn.score(X_test, y_test))
```

```
Training Accuracy : 0.8720588235294118  
Testing Accuracy : 0.8623529411764705
```

11. Membuat model KNN

```
✓ [40] print('\nConfusion matrix')  
0 d   cm = confusion_matrix(y_test, y_pred)  
      print(cm)
```

```
Confusion matrix  
[[726   9]  
 [108   7]]
```

```
✓ [41] print('\nClassification report')  
0 d   cr = classification_report(y_test, y_pred)  
      print(cr)
```

```
Classification report
```

	precision	recall	f1-score	support
0	0.87	0.99	0.93	735
1	0.44	0.06	0.11	115
accuracy			0.86	850
macro avg	0.65	0.52	0.52	850
weighted avg	0.81	0.86	0.81	850

12. Langkah terakhir dari project ini adalah mengimplementasikan model yang sudah dibuat dengan menggunakan dataset test untuk memprediksi customer yang churn.

13. Adapun tahapannya adalah mengunggah file dataset test dari komputer dan melakukan data preprocessing seperti yang telah dilakukan pada dataset train. Setelah itu, copy dataset test pada variabel baru yaitu 'testing' dan drop kolom 'id' yang ada pada dataset testing. Lalu lakukan prediksi dengan menggunakan tiga model yang sudah dibuat tadi.

14. Hasil akhir disimpan dalam bentuk dataframe yang memuat customer_id dan hasil churn pada tiga model. Lalu simpan hasilnya dalam format csv.

```
testing = testing.drop(['id'], axis=1)

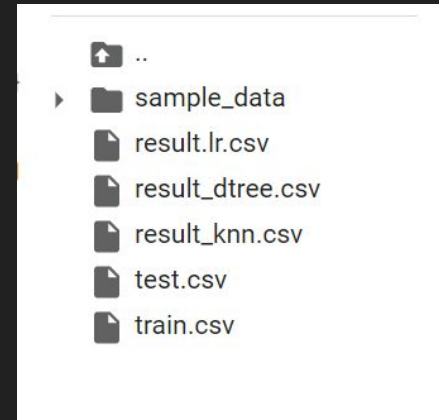
[60] testing.shape

(750, 19)

[61] final_pred_dtree = model.predict(testing)
      final_pred_LR = logreg.predict(testing)
      final_pred_knn = knn.predict(testing)

[63] output_dtree = pd.DataFrame({'id_customer': df2.id, 'churn': final_pred_dtree })
      output_LR = pd.DataFrame({'id_customer': df2.id, 'churn': final_pred_LR })
      output_knn = pd.DataFrame({'id_customer': df2.id, 'churn': final_pred_knn })

      output_dtree.to_csv('result_dtree.csv', index=False)
      output_LR.to_csv('result_lr.csv', index=False)
      output_knn.to_csv('result_knn.csv', index=False)
```



Link Project dari Google Colab

<https://colab.research.google.com/drive/1lxVOt7ok-AcgdaxA3fnHnp3Bcl1j1ccN?usp=sharing>

Referensi Sumber Belajar

1. Binar Academy program FGA Data Scientist Batch 4 bersama Kak Sari Rahmawati
2. DQLab Career Track Data Analyst
3. Kaggle Titanic Competition
4. Google

Sekian dan terimakasih