

PENGGALIAN DATA – IS184943

TUGAS GROUP PROJECT #2
EKSPLORASI DAN PRAPROSES DATA

SULIS AVANDHY PUTRA	-	05211940000084
MUHAMMAD ZUHDI AFI ABIYYI	-	05211940000135
AFLAH ADITYA	-	05211942000001

Program Studi Sarjana

Departemen Sistem Informasi

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

Tahun 2022

Ringkasan Progress Final Project

B. Tugas

1. Lakukan perbaikan eksplorasi data dan juga praproses yang diperlukan (dari TGP #1) untuk keperluan pemodelan prediksi/klasifikasi, agar hasilnya nanti dapat digunakan untuk melakukan perbaikan terhadap model dasar (*base model*) dari implementasi model prediksi.
2. Lakukan proses prediksi dengan mendasarkan pada variabel target (variabel dependen **subscribe**). Lakukan implementasi dengan menggunakan berbagai metode: (i) **random forrest**, (ii) **k-NN**, (iii) **Naïve Bayes**, (iv) **SVM**, (v) **BPNN**, dan (vi) **Logistic Regression**.
 - a. Untuk setiap metode gunakan *library* python yang anda pahami dan menurut anda paling baik, termasuk *library* yang diberikan dalam sesi tutorial. Jelaskan penentuan parameter (*parameter setting*) *library* yang anda gunakan.
 - b. Untuk pembangunan model kasifikasi, lakukan pembagian keseluruhan data menjadi data pelatihan (*training*) sebesar 60%, data validasi 20%, dan data uji (*testing*) 20%. Data pelatihan dan data validasi digunakan untuk membangun model dasar (*base model*) dari klasifikasi, sedang data uji digunakan sebagai data pengujian final menggunakan model dasar terbaik.
 - c. Untuk membangun model dasar (*base model*) terbaik, gunakan metode **10-cross validation** pada data pelatihan dan data validasi. Hitung nilai **f1-measure** dan juga **AUC-ROC** untuk melakukan evaluasi kinerja dari model dasar klasifikasi.
 - d. Untuk masing-masing metode, hitung nilai rata-rata dan juga simpangan baku dari hasil pengujian menggunakan 10-cross validation. Lalu pilih hasil terbaik sebagai model dasar terpilih.
 - e. Anda diberi kebebasan untuk menentukan parameter yang akan digunakan dalam eksperimen untuk membangun model dasar terbaik pada setiap metode klasifikasi. Tuliskan referensi yang digunakan beserta penjelasan singkat mengenai metode yang dipakai.
 - f. Dengan menggunakan model dasar terbaik untuk masing-masing metode klasifikasi, hitung kinerja dari masing-masing model dan bandingkan hasilnya.
 - g. Dari model klasifikasi terbaik dari hasil perbandingan langkah (f), lakukan eksperimen dampak dari seleksi fitur (reduksi dimensi/kolom/fitur data). Untuk ini lakukan reduksi fitur mulai dari total 16 fitur variable independent hingga hanya tersisa 10 fitur saja (lakukan eksperimen jumlah fitur terbaik antara 10 s.d. 16). Gunakan metode **backward** dan **forward selection** dengan menggunakan *library* untuk seleksi fitur (anda bebas mencari sendiri *library* tersebut dan berikan uraian singkat mengenai *library* yang dipakai). **Backward selection** diawali dengan melatih model menggunakan semua fitur dan lakukan evaluasi kinerja model menggunakan **f1-measure**. Kemudian pilih satu fitur dari semua fitur yang ada untuk dihapus, yaitu penghapusan fitur yang memberikan evaluasi kinerja sama dengan atau lebih baik dari sebelum penghapusan. Dengan cara yang sama lakukan seleksi sebuah fitur yang akan dihapus berikutnya hingga tercapai sisa target fitur yang diharapkan. **Forward selection**, kebalikan dari **backward selection**, di mana setiap saat ditambahkan satu fitur yang memberikan hasil evaluasi kinerja yang lebih baik dari kinerja yang diperoleh jumlah fitur sesuai target yang diinginkan (10 s.d. 16 fitur).

A. Pendahuluan

1. Dataset

Data yang digunakan pada tugas ini adalah data mengenai kampanye pemasaran langsung (*direct marketing*) produk deposito bank berjangka dari sebuah institusi perbankan di Portugal. Data kampanye pemasaran langsung adalah data yang diperoleh dari bulan Mei 2008 hingga November 2010. Data ini terdiri dari 45.211 baris, 16 atribut (variabel independen) dan 1 atribut target (variabel dependen).

Berikut merupakan beberapa variabel/atribut dalam dataset yang dapat dikelompokkan menjadi empat kategori sebagai berikut:

- bank client data:
 - 1) age (numeric)

- 2) job: type of job (categorical: "admin.", "unknown", "unemployed", "management", "housemaid", "entrepreneur", "student", "blue-collar", "self-employed", "retired", "technician", "services")
 - 3) marital: marital status (categorical: "married", "divorced", "single"; note: "divorced" means divorced or widowed)
 - 4) education (categorical: "unknown", "secondary", "primary", "tertiary")
 - 5) default: has credit in default? (binary: "yes", "no")
 - 6) balance: average yearly balance, in euros (numeric)
 - 7) housing: has a housing loan? (binary: "yes", "no")
 - 8) loan: has a personal loan? (binary: "yes", "no")
- Related with the last contact of the current campaign:
 - 9) contact: contact communication type (categorical: "unknown", "telephone", "cellular")
 - 10) day: last contact day of the month (numeric)
 - 11) month: last contact month of year (categorical: "jan", "feb", "mar", ..., "nov", "dec")
 - 12) duration: last contact duration, in seconds (numeric)
 - Atribut lain:
 - 13) campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
 - 14) pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)
 - 15) previous: number of contacts performed before this campaign and for this client (numeric)
 - 16) poutcome: outcome of the previous marketing campaign (categorical: "unknown", "other", "failure", "success")
 - Output variable (desired target):
 - 17) subscribe: has the client subscribed a term deposit? (binary: 'yes', 'no')

B. Pra Proses Data

1. Missing Value

Missing value merupakan suatu kondisi dimana suatu atribut memiliki nilai null. Kondisi ini perlu ditangani dengan tepat agar model yang dibangun dapat melakukan klasifikasi dengan baik. Ada beberapa cara penanganannya diantaranya yaitu menghapus kolom atau baris yang memiliki missing value atau mengganti nilainya dengan nilai lain seperti median, mean, atau mode.

```
[6] # Cek Nilai Null
campaignC.isnull().sum()

age      0
job      0
marital  0
education 0
default  0
balance  0
housing  0
loan     0
contact  0
day      0
month    0
duration 0
campaign 0
pdays   0
previous 0
poutcome 0
subscribe 0
dtype: int64
```

Berdasarkan gambar di atas, dataset campaignC tidak memiliki nilai null pada setiap atribut sehingga tidak diperlukan penanganan missing value lebih lanjut.

2. Categorical Encoding

Categorical Encoding merupakan proses mengubah atribut kategorikal menjadi integer. Ada beberapa cara untuk melakukan categorical encoding yaitu dengan bantuan library pandas menggunakan perintah `get_dummies` atau secara manual. Pada tugas ini, proses categorical encoding dilakukan secara manual terhadap seluruh atribut yang memiliki nilai kategorik yaitu `job`, `marital`, `education`, `default`, `housing`, `loan`, `contact`, `month`, `poutcome`, serta `subscribe`.

```
# Melakukan categorical encoding
campaignC.replace({'job': {'unknown':0, 'admin':1, 'unemployed':2, 'management':3, 'housemaid':4, 'entrepreneur':5, 'student':6, 'blue-collar':7, 'self-emp':8}}, inplace=True)
campaignC.replace({'marital': {'married':1, 'divorced':2, 'single':3}}, inplace=True)
campaignC.replace({'education': {'unknown':0, 'secondary':1, 'primary':2, 'tertiary':3}}, inplace=True)
campaignC.replace({'default': {'no':0, 'yes':1}}, inplace=True)
campaignC.replace({'housing': {'no':0, 'yes':1}}, inplace=True)
campaignC.replace({'loan': {'no':0, 'yes':1}}, inplace=True)
campaignC.replace({'contact': {'unknown':0, 'telephone':1, 'cellular':2}}, inplace=True)
campaignC.replace({'month': {'jan':1, 'feb':2, 'mar':3, 'apr':4, 'may':5, 'jun':6, 'jul':7, 'aug':8, 'sep':9, 'oct':10, 'nov':11, 'dec':12}}, inplace=True)
campaignC.replace({'poutcome': {'unknown':0, 'other':1, 'failure':2, 'success':3}}, inplace=True)
campaignC.replace({'subscribe': {'no':0, 'yes':1}}, inplace=True)

campaignC.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	subscribe
0	58	3	1	3	0	2143	1	0	0	5	5	261	1	-1	0	0	0
1	44	10	3	1	0	29	1	0	0	5	5	151	1	-1	0	0	0
2	33	5	1	1	0	2	1	1	0	5	5	76	1	-1	0	0	0
3	47	7	1	0	0	1506	1	0	0	5	5	92	1	-1	0	0	0
4	33	0	3	0	0	1	0	0	0	5	5	198	1	-1	0	0	0

3. Split Data

Proses split data merupakan proses untuk membagi data menjadi data latih dan data tes. Pada tugas ini, proses split data menggunakan bantuan library python yaitu `sklearn`. Data akan dibagi menjadi tiga yaitu data training, data validation dan data test dengan persentase masing-masing 60%, 20%, dan 20%. Data training merupakan data yang akan digunakan selama proses pembangunan model kemudian data tersebut akan divalidasi pada data validasi agar mencegah terjadinya overfitting pada data. Sementara itu, data testing akan digunakan ketika model terbaik telah dipilih dan akan dilakukan pengujian final.

Pada tugas ini, data training adalah X_train dan y_train, data validasi adalah X_valid dan y_valid, serta data testing adalah X_test dan y_test.

```
[67] # mengambil fitur/variabel masukan
X = campaignC.drop(columns = ['subscribe'])
y = campaignC.subscribe

[68] # In the first step we will split the data in training and remaining dataset
X_train, X_rem, y_train, y_rem = train_test_split(X,y, train_size=0.6)

# Now since we want the valid and test size to be equal (10% each of overall data).
# we have to define valid_size=0.5 (that is 50% of remaining data)
test_size = 0.5
X_valid, X_test, y_valid, y_test = train_test_split(X_rem,y_rem, test_size=0.5)

print(X_train.shape), print(y_train.shape)
print(X_valid.shape), print(y_valid.shape)
print(X_test.shape), print(y_test.shape)
```

4. Mendefinisikan kfold untuk Cross Validation

Metode pembagian data atau split data yang digunakan adalah Cross Validation. Cross validation akan membagi data menjadi training:testing dan akan dibagi sebanyak n-kali dengan posisi data testing yang berbeda-beda. KFold merupakan library yang digunakan untuk menguji klasifikasi menggunakan metode Cross Validation. Sesuai dengan ketentuan dari soal, parameter yang digunakan sebagai berikut :

- n_split : 10 (total fold yang digunakan);
- shuffle : True (parameter apakah setiap fold diacak atau tidak);
- random_state : 42 (digunakan apabila shuffle = True, sebagai penanda randomness dari setiap fold, jika tidak diisi bisa mengakibatkan nilai akurasi berubah setiap kali running)

(Sumber : https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html)

```
# define kfold for cross validation
kfold = KFold(n_splits=10, shuffle=True, random_state=42)
```

5. Fungsi untuk meringkas kinerja model klasifikasi

```

from sklearn.preprocessing import LabelBinarizer
from mlxtend.plotting import plot_confusion_matrix

def evaluator(y_test, y_pred):

    # Accuracy:
    print('Accuracy is: ', accuracy_score(y_test,y_pred))
    print('')
    # Classification Report:
    print('Classification Report: \n',classification_report(y_test,y_pred))

    # Area Under The Curve Score:
    lb = LabelBinarizer()
    y_test1 = lb.fit_transform(y_test)
    y_pred1 = lb.transform(y_pred)
    print('AUC_ROC Score: ',roc_auc_score(y_test1, y_pred1, average='macro'),'\\n\\n')

    # untuk mengetahui confusion matrix, bisa menggunakan library classification report dengan sebelumnya melakukan proses
    print('Confusion Matrix: \\n\\n')
    plt.style.use("ggplot")
    cm = confusion_matrix(y_test,y_pred)
    plot_confusion_matrix(conf_mat = cm,figsize=(8,6),show_normed=True)

```

C. Implementasi Model Klasifikasi

1. Random Forest

Proses implementasi diawali dengan melakukan import beberapa library yang dibutuhkan sebelum melakukan proses prediksi dengan mendasarkan pada variabel dependen yaitu “subscribe”. Pada tabel berikut merupakan list beberapa library yang dibutuhkan dalam menjalankan metode Random Forest.

Library	Fungsi
<code>from sklearn.model_selection import KFold, RepeatedKFold, ShuffleSplit</code>	Library untuk penerapan KFold Validation dan juga untuk menampilkan nilai pengujian Cross Validation
<code>from sklearn.preprocessing import LabelBinarizer</code>	Library ini berfungsi dalam menemukan atau memperoleh nilai AUC ROC
<code>from mlxtend.plotting import plot_confusion_matrix</code>	Library ini digunakan untuk mengetahui confusion matrix yang berfungsi untuk mengukur kinerja dari model klasifikasi random forest ini.
<code>from sklearn.metrics import roc_auc_score, roc_curve, make_scorer, accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report</code>	Menampilkan berbagai jenis hasil tes, seperti nilai ROC AUC, nilai akurasi, nilai presisi, nilai recall, f1 score, confusion matrix, classification report, dan lainnya
<code>import matplotlib.pyplot as plt</code>	Library ini berfungsi untuk memvisualisasikan data
<code>from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor</code>	Modul dalam menjalankan metode klasifikasi random forest

```
from sklearn.model_selection
import cross_val_score,
cross_validate
```

Modul ini digunakan untuk melakukan evaluasi model menggunakan cross validation

Proses implementasi diawali dengan memanggil fungsi Random Forest dari library sklearn.

```
[22] randomforest = RandomForestClassifier()
```

Kemudian mendefinisikan beberapa hyperparameter yang akan diujikan pada proses selanjutnya. Ada dua hyperparameter yang diatur yaitu “n_estimators”, dan “max_depth”. n_estimators adalah jumlah pohon (tree) yang ingin dibangun sebelum mengambil voting maksimum atau rata-rata prediksi. Jumlah pohon yang lebih tinggi memberi kinerja yang lebih baik tetapi membuat kode menjadi lebih lambat. Parameter ‘n_estimators’ yang akan digunakan yaitu [50, 150]. Parameter max_depth menentukan kedalaman maksimum setiap pohon. Tidak ada nilai default untuk max_depth, yang berarti setiap pohon akan mengembang hingga setiap daun murni (pure leaf). Daun murni adalah daun yang semua data pada daun (leaf) berasal dari kelas yang sama. Parameter ‘max_depth’ yang akan digunakan yaitu [1, 2] Dari pilihan parameter tersebut akan dipilih nilai ‘n_estimators’ dan ‘max_depth’ yang terbaik.

```
params = {
    'n_estimators': [50, 150],
    'max_depth': [1,2]
}
```

Selanjutnya, dilakukan proses pencarian secara exhaustive untuk menemukan kombinasi hyperparameter yang memberikan nilai akurasi terbaik. Berikut adalah hasil exhaustive search yang dilakukan beserta nilai rata-rata dan simpangan baku dari akurasi, f1-measure, dan juga AUC-ROC.

```
scoring = ['accuracy', 'f1', 'roc_auc']
rf_classifierCV = GridSearchCV(randomforest, params, cv= kfold, scoring=scoring, refit='accuracy', verbose= 3)
rf_classifierCV.fit(X_train, y_train)

# print parameter terbaik
print("\n=====")
print('Parameter terbaik: {0} \ndengan nilai akurasi pada data training: {1}'.format(rf_classifierCV.best_params_, rf_classifierCV.best_score_))
print('Nilai f1 rata-rata: {0} \nNilai AUC-ROC rata-rata: {1} \n \nNilai simpangan baku f1: {2} \nNilai simpangan baku AUC-ROC: {3}'.format(rf_classifierCV.cv_results_['mean_test_f1'][0], rf_classifierCV.cv_results_['mean_test_roc_auc'][0], rf_classifierCV.cv_results_['std_test_f1'][0], rf_classifierCV.cv_results_['std_test_roc_auc'][0]))
```

```
Parameter terbaik: {'max_depth': 2, 'n_estimators': 50}
dengan nilai akurasi pada data training: 0.8847957012396339
Nilai f1 rata-rata: 0.0
Nilai AUC-ROC rata-rata: 0.8560179057481614

Nilai simpangan baku f1: 0.0
Nilai simpangan baku AUC-ROC: 0.011667157766906314
```

Untuk memvalidasi model yang sudah dibangun, dilakukan prediksi terhadap data validasi kemudian dicari nilai akurasi serta AOC-ROC. Berikut merupakan hasil prediksi terhadap data validasi serta nilai akurasi dan AOC-ROC.

```

pred = rf_classifierCV.predict(X_test)
evaluator(y_test, pred)

Accuracy is: 0.8812341037266394

Classification Report:
              precision    recall  f1-score   support

     0       0.88         1.00         0.94         7964
     1       0.86         0.01         0.01         1079

 accuracy          0.88         0.88         0.88         9043
 macro avg         0.87         0.50         0.47         9043
weighted avg         0.88         0.88         0.83         9043

AUC_ROC Score: 0.5027175696565965

Confusion Matrix:

```

2. k-NN

Proses implementasi diawali dengan melakukan import beberapa library yang dibutuhkan sebelum melakukan proses prediksi dengan mendasarkan pada variabel dependen yaitu “subscribe”. Pada tabel berikut merupakan list beberapa library yang dibutuhkan dalam menjalankan metode k-NN.

Library	Fungsi
<pre> from sklearn.model_selection import KFold, RepeatedKFold, ShuffleSplit </pre>	Library untuk penerapan KFold Validation dan juga untuk menampilkan nilai pengujian Cross Validation
<pre> from sklearn.preprocessing import LabelBinarizer </pre>	Library ini berfungsi dalam menemukan atau memperoleh nilai AUC ROC
<pre> from mlxtend.plotting import plot_confusion_matrix </pre>	Library ini digunakan untuk mengetahui confusion matrix yang berfungsi untuk mengukur kinerja dari model klasifikasi k-NN ini.
<pre> from sklearn.metrics import roc_auc_score, roc_curve, make_scorer, accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report </pre>	Menampilkan berbagai jenis hasil tes, seperti nilai ROC AUC, nilai akurasi, nilai presisi, nilai recall, f1 score, confusion matrix, classification report, dan lainnya

<code>import matplotlib.pyplot as plt</code>	Library ini berfungsi untuk memvisualisasikan data
<code>from sklearn.neighbors import KNeighborsClassifier</code>	Modul dalam menjalankan metode klasifikasi k-NN
<code>from sklearn.model_selection import cross_val_score, cross_validate</code>	Modul ini digunakan untuk melakukan evaluasi model menggunakan cross validation

Proses implementasi diawali dengan memanggil fungsi k-NN dari library sklearn.

```
knn = KNeighborsClassifier()
```

Kemudian mendefinisikan beberapa hyperparameter yang akan diujikan pada proses selanjutnya. Ada satu hyperparameter yang diatur yaitu “n_neighbors”. n_neighbors mewakili jumlah ‘neighbors’ yang akan digunakan untuk kueri kneighbors. Parameter ‘n_neighbors’ yang akan digunakan yaitu [1, 2, 3, 4, 5]. Dari pilihan parameter tersebut akan dipilih nilai ‘n_neighbors’ yang terbaik.

```
params = {'n_neighbors': [1, 2, 3, 4, 5]}
```

Selanjutnya, dilakukan proses pencarian secara exhaustive untuk menemukan kombinasi hyperparameter yang memberikan nilai akurasi terbaik. Berikut adalah hasil exhaustive search yang dilakukan beserta nilai rata-rata dan simpangan baku dari akurasi, f1-measure, dan juga AUC-ROC.

```
scoring = ['accuracy', 'f1', 'roc_auc']
knn_classifierCV = GridSearchCV(knn, params, cv= kfold, scoring=scoring, refit='accuracy', verbose= 3)
knn_classifierCV.fit(X_train, y_train)

# print parameter terbaik
print("\n=====")
print('Parameter terbaik: {0} \ndengan nilai akurasi pada data training: {1}'.format(knn_classifierCV.best_params_, knn_
print(
'Nilai f1 rata-rata: {0} \nNilai AUC-ROC rata-rata: {1} \n \nNilai simpangan baku f1: {2} \nNilai simpangan baku AUC-
format(knn_classifierCV.cv_results_['mean_test_f1'][0],knn_classifierCV.cv_results_['mean_test_roc_auc'][0],
knn_classifierCV.cv_results_['std_test_f1'][0], knn_classifierCV.cv_results_['std_test_roc_auc'][0]))
```

```
Parameter terbaik: {'n_neighbors': 4}
dengan nilai akurasi pada data training: 0.8865655176050634
Nilai f1 rata-rata: 0.3457288568284856
Nilai AUC-ROC rata-rata: 0.6281611873521982

Nilai simpangan baku f1: 0.028509734345039672
Nilai simpangan baku AUC-ROC: 0.014903130981588414
```

Untuk memvalidasi model yang sudah dibangun, dilakukan prediksi terhadap data validasi kemudian dicari nilai akurasi serta AOC-ROC. Berikut merupakan hasil prediksi terhadap data validasi serta nilai akurasi dan AOC-ROC.

```

pred = knn_classifierCV.predict(X_test)
evaluator(y_test, pred)

Accuracy is: 0.8778060378193078

Classification Report:
              precision    recall  f1-score   support

     0       0.90      0.98      0.93      7964
     1       0.46      0.16      0.24      1079

 accuracy      0.88      0.88      0.85      9043
 macro avg     0.68      0.57      0.59      9043
weighted avg     0.84      0.88      0.85      9043

AUC_ROC Score: 0.5672724898745002

Confusion Matrix:

```

3. Naive Bayes

Proses implementasi diawali dengan melakukan import beberapa library yang dibutuhkan sebelum melakukan proses prediksi dengan mendasarkan pada variabel dependen yaitu “subscribe”. Pada tabel berikut merupakan list beberapa library yang dibutuhkan dalam menjalankan metode Naive Bayes.

Library	Fungsi
<code>from sklearn.model_selection</code> <code>import KFold, RepeatedKFold,</code> <code>ShuffleSplit</code>	Library untuk penerapan KFold Validation dan juga untuk menampilkan nilai pengujian Cross Validation
<code>from sklearn.preprocessing</code> <code>import LabelBinarizer</code>	Library ini berfungsi dalam menemukan atau memperoleh nilai AUC ROC
<code>from mlxtend.plotting</code> <code>import plot_confusion_matrix</code>	Library ini digunakan untuk mengetahui confusion matrix yang berfungsi untuk mengukur kinerja dari model klasifikasi naive bayes ini.
<code>from sklearn.metrics</code> <code>import roc_auc_score,</code> <code>roc_curve, make_scorer,</code> <code>accuracy_score,</code> <code>precision_score, recall_score,</code> <code>f1_score, confusion_matrix,</code> <code>classification_report</code>	Menampilkan berbagai jenis hasil tes, seperti nilai ROC AUC, nilai akurasi, nilai presisi, nilai recall, f1 score, confusion matrix, classification report, dan lainnya
<code>import matplotlib.pyplot as plt</code>	Library ini berfungsi untuk memvisualisasikan data
<code>from sklearn.naive_bayes import</code> <code>GaussianNB</code>	Package utama dalam menjalankan metode klasifikasi Naive Bayes

<code>import math</code>	Modul ini
<code>from sklearn.model_selection import cross_val_score, cross_validate</code>	Modul ini digunakan untuk melakukan evaluasi model menggunakan cross validation

Setelah melakukan impor package, langkah berikutnya adalah menentukan kfold untuk cross validation yang akan dilakukan dan buat 'function' untuk meringkas kinerja model klasifikasi naive bayes

```

0 d [24] # define kfold for cross validation
      from sklearn.model_selection import KFold, RepeatedKFold, ShuffleSplit
      kfold = KFold(n_splits=10, shuffle=True, random_state=42)

0 d [25] from sklearn.preprocessing import LabelBinarizer
      from mlxtend.plotting import plot_confusion_matrix
      from sklearn.metrics import roc_auc_score, roc_curve, make_scorer, accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report
      import matplotlib.pyplot as plt

      def evaluator(y_test, y_pred):

          # Accuracy:
          print('Accuracy is: ', accuracy_score(y_test, y_pred))
          print('')
          # Classification Report:
          print('Classification Report: \n', classification_report(y_test, y_pred))

          # Area Under The Curve Score:
          lb = LabelBinarizer()
          y_test1 = lb.fit_transform(y_test)
          y_pred1 = lb.transform(y_pred)
          print('AUC_ROC Score: ', roc_auc_score(y_test1, y_pred1, average='macro'), '\n\n')

          # untuk mengetahui confusion matrix, bisa menggunakan library classification report dengan sebelumnya melakukan proses prediksi menggunakan metode .predict
          print('Confusion Matrix: \n\n')
          plt.style.use("ggplot")
          cm = confusion_matrix(y_test, y_pred)
          plot_confusion_matrix(conf_mat = cm, figsize=(8,6), show_normed=True)

```

Setelah menentukan kfold dan membuat function, langkah berikutnya berdasarkan tutorial adalah menghitung nilai *f1-measure* dan juga *AUC-ROC* untuk melakukan evaluasi kinerja dari model dasar klasifikasi naive bayes. Hasilnya ditunjukkan pada gambar dibawah ini:

```

0 d [26] naivebayes = GaussianNB()
      naivebayes.fit(X_train, y_train,)

      predict = naivebayes.predict(X_test)
      evaluator(y_test, predict)

```

Accuracy is: 0.8225146522171846

Classification Report:

	precision	recall	f1-score	support
0	0.94	0.86	0.90	8040
1	0.32	0.53	0.40	1003
accuracy			0.82	9043
macro avg	0.63	0.69	0.65	9043
weighted avg	0.87	0.82	0.84	9043

AUC_ROC Score: 0.692936737052524

Lalu, kami menghitung nilai rata-rata dan juga simpangan baku dari hasil pengujian menggunakan *10-cross validation*. Sehingga didapatkan hasil seperti pada gambar dibawah ini:

```

0d import math
from sklearn.model_selection import cross_val_score, cross_validate
scoring = ['accuracy', 'f1', 'roc_auc']
nb_model = cross_validate(naivebayes, X_train, y_train, cv=kfold, scoring=scoring)
# print(dtrees_model)

print('-----Rata-Rata-----+')
print('Nilai akurasi rata-rata: {}'.format(nb_model['test_accuracy'].mean()))
print('Nilai f1 rata-rata: {}'.format(nb_model['test_f1'].mean()))
print('Nilai AUC-ROC rata-rata: {}'.format(nb_model['test_roc_auc'].mean()))
print('\n')
print('-----Simpangan Baku-----+')
print('Nilai akurasi simpangan baku: {}'.format(math.sqrt(nb_model['test_accuracy'].var())))
print('Nilai f1 simpangan baku: {}'.format(math.sqrt(nb_model['test_f1'].var())))
print('Nilai AUC-ROC simpangan baku: {}'.format(math.sqrt(nb_model['test_roc_auc'].var())))

+-----Rata-Rata-----+
Nilai akurasi rata-rata: 0.8265512549105314
Nilai f1 rata-rata: 0.41558240860486617
Nilai AUC-ROC rata-rata: 0.8239253244622017

+-----Simpangan Baku-----+
Nilai akurasi simpangan baku: 0.009492645534677064
Nilai f1 simpangan baku: 0.01749575900556094
Nilai AUC-ROC simpangan baku: 0.009003302439479801

```

4. Support Vector Machine

Proses implementasi diawali dengan melakukan import beberapa library yang dibutuhkan sebelum melakukan proses prediksi dengan mendasarkan pada variabel dependen yaitu “subscribe”. Pada tabel berikut merupakan list beberapa library yang dibutuhkan dalam menjalankan metode Support Vector Machine.

Library	Fungsi
<pre>from sklearn.model_selection import KFold, RepeatedKFold, ShuffleSplit</pre>	Library untuk penerapan KFold Validation dan juga untuk menampilkan nilai pengujian Cross Validation
<pre>from sklearn.preprocessing import LabelBinarizer</pre>	Library ini berfungsi dalam menemukan atau memperoleh nilai AUC ROC
<pre>from mlxtend.plotting import plot_confusion_matrix</pre>	Library ini digunakan untuk mengetahui confusion matrix yang berfungsi untuk mengukur kinerja dari model klasifikasi naive bayes ini.
<pre>from sklearn.metrics import roc_auc_score, roc_curve, make_scorer, accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report</pre>	Menampilkan berbagai jenis hasil tes, seperti nilai ROC AUC, nilai akurasi, nilai presisi, nilai recall, f1 score, confusion matrix, classification report, dan lainnya
<pre>import matplotlib.pyplot as plt</pre>	Library ini berfungsi untuk memvisualisasikan data
<pre>from sklearn.svm import SVC</pre>	Package utama dalam menjalankan metode

	klasifikasi Support Vector Machine
<code>import math</code>	Modul ini
<code>from sklearn.model_selection import cross_val_score, cross_validate</code>	Modul ini digunakan untuk melakukan evaluasi model menggunakan cross validation

Setelah melakukan impor package, langkah berikutnya adalah menentukan kfold untuk cross validation yang akan dilakukan dan buat 'function' untuk meringkas kinerja model klasifikasi SVM

```

✓ [24] # define kfold for cross validation
0 d      from sklearn.model_selection import KFold, RepeatedKFold, ShuffleSplit
          kfold = KFold(n_splits=10, shuffle=True, random_state=42)

✓ [25] from sklearn.preprocessing import LabelBinarizer
        from mlxtend.plotting import plot_confusion_matrix
        from sklearn.metrics import roc_auc_score, roc_curve, make_scorer, accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report
        import matplotlib.pyplot as plt

        def evaluator(y_test, y_pred):

            # Accuracy:
            print('Accuracy is: ', accuracy_score(y_test,y_pred))
            print('')
            # Classification Report:
            print('Classification Report: \n',classification_report(y_test,y_pred))

            # Area Under The Curve Score:
            lb = LabelBinarizer()
            y_test1 = lb.fit_transform(y_test)
            y_pred1 =lb.transform(y_pred)
            print('AUC_ROC score: ',roc_auc_score(y_test1, y_pred1, average='macro'),'\\n\\n')

            # untuk mengetahui confusion matrix, bisa menggunakan library classification report dengan sebelumnya melakukan proses prediksi menggunakan metode .predict
            print('Confusion Matrix: \\n\\n')
            plt.style.use('ggplot')
            cm = confusion_matrix(y_test,y_pred)
            plot_confusion_matrix(conf_mat = cm,figsize=(8,6),show_normed=True)

```

Setelah menentukan kfold dan membuat function, langkah berikutnya berdasarkan tutorial adalah hypertunin parameter dengan cross validation, tetapi dikarenakan SVM kurang cocok untuk diterapkan pada data yang cukup besar sehingga mengurungkan niat kami untuk melakukannya. Gambar tersebut merupakan bukti bahwa kami sudah mencoba menjalankannya.

```

Fitting 10 folds for each of 1 candidates, totalling 10 fits
[CV 1/10] END .....kernel=linear; score=0.875 total time=115.3min
[CV 2/10] END .....kernel=linear; score=0.881 total time=86.8min
[CV 3/10] END .....kernel=linear; score=0.887 total time=100.7min

```

Cannot connect to GPU backend

You cannot currently connect to a GPU due to usage limits in Colab. [Learn more](#)

To get more access to GPUs, consider purchasing Colab compute units with [Pay As You Go](#).

Close

Connect without GPU

Lalu kami mencoba menghitung nilai *f1-measure* dan juga *AUC-ROC* terlebih dahulu untuk melakukan evaluasi kinerja dari model dasar klasifikasi SVM. Hasilnya ditunjukkan pada gambar dibawah ini:

✓
20 d

```
# Klasifikasi SVM menggunakan Cross Validation
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_train, y_train,)

predict = svm.predict(X_test)
evaluator(y_test, predict)
```

➞ Accuracy is: 0.8796859449297799

Classification Report:

	precision	recall	f1-score	support
0	0.88	1.00	0.94	7957
1	0.46	0.01	0.02	1086
accuracy			0.88	9043
macro avg	0.67	0.50	0.48	9043
weighted avg	0.83	0.88	0.83	9043

AUC_ROC Score: 0.5046451333375456

Lalu, kami menghitung nilai rata-rata dan juga simpangan baku dari hasil pengujian menggunakan *10-cross validation*. Sehingga didapatkan hasil seperti pada gambar dibawah ini:

✓
2m

```
import math
from sklearn.model_selection import cross_val_score, cross_validate
scoring = ['accuracy', 'f1', 'roc_auc']
svm_model = cross_validate(svm, X_train, y_train, cv=kfold, scoring=scoring)
# print(svm_model)

print('+-----Rata-Rata-----+')
print('Nilai akurasi rata-rata: {}'.format(svm_model['test_accuracy'].mean()))
print('Nilai f1 rata-rata: {}'.format(svm_model['test_f1'].mean()))
print('Nilai AUC-ROC rata-rata: {}'.format(svm_model['test_roc_auc'].mean()))
print('\n')
print('+-----Simpangan Baku-----+')
print('Nilai akurasi simpangan baku: {}'.format(math.sqrt(svm_model['test_accuracy'].var())))
print('Nilai f1 simpangan baku: {}'.format(math.sqrt(svm_model['test_f1'].var())))
print('Nilai AUC-ROC simpangan baku: {}'.format(math.sqrt(svm_model['test_roc_auc'].var())))
```

➞

```
+-----Rata-Rata-----+
Nilai akurasi rata-rata: 0.8831748453583588
Nilai f1 rata-rata: 0.027125556015984893
Nilai AUC-ROC rata-rata: 0.7414629777326923

+-----Simpangan Baku-----+
Nilai akurasi simpangan baku: 0.006380977822029863
Nilai f1 simpangan baku: 0.017200365195208345
Nilai AUC-ROC simpangan baku: 0.014376862744505938
```

5. BPNN

Proses implementasi diawali dengan melakukan import beberapa library yang dibutuhkan sebelum melakukan proses prediksi dengan mendasarkan pada variabel dependen yaitu “subscribe”. Pada tabel berikut merupakan list beberapa library yang dibutuhkan dalam menjalankan metode BPNN.

Library	Fungsi
<code>from sklearn.model_selection import KFold, RepeatedKFold, ShuffleSplit</code>	Library untuk penerapan KFold Validation dan juga untuk menampilkan nilai pengujian Cross Validation
<code>from sklearn.preprocessing import LabelBinarizer</code>	Library ini berfungsi dalam menemukan atau memperoleh nilai AUC ROC
<code>from mlxtend.plotting import plot_confusion_matrix</code>	Library ini digunakan untuk mengetahui confusion matrix yang berfungsi untuk mengukur kinerja dari model klasifikasi naive bayes ini.
<code>from sklearn.metrics import roc_auc_score, roc_curve, make_scorer, accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report</code>	Modul ini berfungsi untuk menampilkan berbagai jenis hasil tes, seperti nilai ROC AUC, nilai akurasi, nilai presisi, nilai recall, f1 score, confusion matrix, classification report, dan lainnya
<code>import matplotlib.pyplot as plt</code>	Library ini berfungsi untuk memvisualisasikan data
<code>from keras.models import Sequential from keras.layers import Dense, Embedding, Dropout, SpatialDropout1D from keras.wrappers.scikit_learn import KerasClassifier from keras.callbacks import EarlyStopping</code>	Libray Keras digunakan untuk membuat fungsi neural network
<code>from sklearn.model_selection import cross_val_score, cross_validate</code>	Modul ini digunakan untuk melakukan evaluasi model menggunakan cross validation

Proses implementasi dilanjutkan dengan menentukan beberapa parameter yang dibutuhkan dalam neural network. Beberapa parameter yang dapat diatur yaitu jumlah node di dalam hidden layer yang direpresentasikan dalam variabel "n_hiddenlayer", jumlah node di dalam output layer yang direpresentasikan dalam variabel "n_outputlayer", jumlah berapa kali model dilatih yang direpresentasikan sebagai epoch, jumlah dimensi yang direpresentasikan dengan "n_features".

```
[70] # define parameter list
      n_hiddenlayer = X_train.shape[1] * 2
      n_outputlayer = 1
      n_epoch = 100
      n_features = X.shape[1]
```

Selanjutnya adalah tahapan untuk mendefinisikan model neural network yang akan dibangun. Untuk mempermudah pendefinisian, dibuat fungsi “build_nn_model” dengan parameter fungsi aktivasi dan juga jenis optimizer yang digunakan. Secara default, fungsi aktivasi adalah fungsi tanh dan jenis optimizer adalah sgd atau stochastic gradient descent. Sementara itu, untuk fungsi aktivasi pada layer output, digunakan fungsi sigmoid.

```
[59] def build_nn_model(activation = 'tanh', optimizer = 'sgd'):
      # define model
      model = Sequential()
      model.add(Dense(n_hiddenlayer, activation= activation, input_dim = n_features))
      # Adding the output layer
      model.add(Dense(n_outputlayer, activation= 'sigmoid'))
      # compile the model
      model.compile(optimizer= optimizer, loss='binary_crossentropy', metrics = ['accuracy'])
      return model
```

Setelah model didefinisikan, model kemudian dibangun dengan melakukan cross validation. Proses ini hanya menggunakan kombinasi fungsi aktivasi dan jenis optimasi sesuai pada fungsi “build_nn_model” karena waktu komputasi yang besar jika mencoba kombinasi lainnya. Berikut adalah kode yang digunakan.

```
scoring = ['accuracy', 'f1', 'roc_auc']
# define model
nn_model = KerasClassifier(build_fn=build_nn_model, epochs=n_epoch, verbose=0)

ann_cv = cross_validate(nn_model, X_train, y_train, cv=kfold, scoring=scoring)
print('Build Completed.')

print('\n')
print('+-----Rata-Rata-----+')
print('Nilai akurasi rata-rata: {}'.format(ann_cv['test_accuracy'].mean()))
print('Nilai f1 rata-rata: {}'.format(ann_cv['test_f1'].mean()))
print('Nilai AUC-ROC rata-rata: {}'.format(ann_cv['test_roc_auc'].mean()))
print('\n')
print('+-----varians-----+')
print('Nilai akurasi varians: {}'.format(math.sqrt(ann_cv['test_accuracy'].var())))
print('Nilai f1 varians: {}'.format(math.sqrt(ann_cv['test_f1'].var())))
print('Nilai AUC-ROC varians: {}'.format(math.sqrt(ann_cv['test_roc_auc'].var())))
```

Berdasarkan proses sebelumnya, didapatkan nilai rata-rata dan simpangan baku dari akurasi, f1-measure, dan juga AUC-ROC. Berikut adalah detail dari nilai-nilai tersebut.

```
+-----Rata-Rata-----+
Nilai akurasi rata-rata: 0.8840971635531751
Nilai f1 rata-rata: 0.014902206186399763
Nilai AUC-ROC rata-rata: 0.7032638940383056

+-----varians-----+
Nilai akurasi varians: 0.005800964563400224
Nilai f1 varians: 0.03272047569826994
Nilai AUC-ROC varians: 0.030303498087990332
```


Untuk memvalidasi model yang sudah dibangun, dilakukan prediksi terhadap data validasi kemudian dicari nilai akurasi serta AOC-ROC.

```
[61] classifier = nn_model
      classifier.fit(X_train, y_train)

<keras.callbacks.History at 0x7fb7eb106710>

[62] pred = classifier.predict(X_test)
      evaluator(y_test, pred)

283/283 [=====] - 0s 1ms/step
Accuracy is: 0.8802388587858012

Classification Report:
              precision    recall  f1-score   support

         0           0.88        1.00        0.94        7960
         1           0.00        0.00        0.00        1083

 accuracy          0.88          0.88          0.88        9043
  macro avg         0.44          0.50          0.47        9043
 weighted avg         0.77          0.88          0.82        9043

AUC_ROC Score: 0.5
```

6. Logistic Regression

Proses implementasi diawali dengan melakukan import beberapa library yang dibutuhkan sebelum melakukan proses prediksi dengan mendasarkan pada variabel dependen yaitu “subscribe”. Pada tabel berikut merupakan list beberapa library yang dibutuhkan dalam menjalankan metode logistic regression.

Library	Fungsi
<code>from sklearn.model_selection</code> <code>import KFold, RepeatedKFold,</code> <code>ShuffleSplit</code>	Library untuk penerapan KFold Validation dan juga untuk menampilkan nilai pengujian Cross Validation
<code>from sklearn.preprocessing</code> <code>import LabelBinarizer</code>	Library ini berfungsi dalam menemukan atau memperoleh nilai AUC ROC
<code>from mlxtend.plotting</code> <code>import plot_confusion_matrix</code>	Library ini digunakan untuk mengetahui confusion matrix yang berfungsi untuk mengukur kinerja dari model klasifikasi naive bayes ini.
<code>from sklearn.metrics</code> <code>import roc_auc_score,</code> <code>roc_curve, make_scorer,</code> <code>accuracy_score,</code> <code>precision_score, recall_score,</code> <code>f1_score, confusion_matrix,</code> <code>classification_report</code>	Menampilkan berbagai jenis hasil tes, seperti nilai ROC AUC, nilai akurasi, nilai presisi, nilai recall, f1 score, confusion matrix, classification report, dan lainnya
<code>import matplotlib.pyplot as plt</code>	Library ini berfungsi untuk memvisualisasikan data
<code>from sklearn.linear_model</code> <code>import LogisticRegression</code>	Modul dalam menjalankan metode klasifikasi logistic regression

```
from sklearn.model_selection
import cross_val_score,
cross_validate
```

Modul ini digunakan untuk melakukan evaluasi model menggunakan cross validation

Proses implementasi dilanjutkan dengan memanggil fungsi logistic regression dari library sklearn.

```
[ ] logreg = LogisticRegression()
```

Kemudian mendefinisikan beberapa hyperparameter yang akan diujikan pada proses selanjutnya. Ada tiga hyperparameter yang diatur yaitu “solver”, “penalty”, dan “regularization strength (C)”. Solver adalah algoritma yang akan digunakan dalam masalah optimasi. Penalty digunakan untuk mengurangi kesalahan generalisasi model dan juga untuk mendisinsentifkan dan mengatur overfitting. C (regularization strength) bekerja dengan penalti untuk mengatur overfitting. Nilai yang lebih kecil menentukan regularisasi yang lebih kuat dan nilai yang tinggi memberitahu model untuk memberikan bobot yang tinggi pada data pelatihan.

```
params = {
    'solver': ['newton-cg', 'lbfgs', 'liblinear'],
    'penalty': ['l2'],
    'C' : [100, 10, 1.0, 0.1, 0.01]
}
```

```
logisticregression_classifierCV = GridSearchCV(logreg, params, cv= kfold, scoring=scoring, refit='accuracy', verbose=
logisticregression_classifierCV.fit(X_train,y_train)

# print parameter terbaik
print("\n=====")
print('Parameter terbaik: {0} \ndengan nilai akurasi pada data training: {1}'.format(logisticregression_classifierCV.
print(
    'Nilai f1 rata-rata: {0} \nNilai AUC-ROC rata-rata: {1} \n \nNilai simpangan baku f1: {2} \nNilai simpangan baku
format(logisticregression_classifierCV.cv_results_['mean_test_f1'][0],logisticregression_classifierCV.cv_results_
logisticregression_classifierCV.cv_results_['std_test_f1'][0], logisticregression_classifierCV.cv_results_
```

Selanjutnya, dilakukan proses pencarian secara exhaustive untuk menemukan kombinasi hyperparameter yang memberikan nilai akurasi terbaik. Berikut adalah hasil exhaustive search yang dilakukan beserta nilai rata-rata dan simpangan baku dari akurasi, f1-measure, dan juga AUC-ROC.

```
Parameter terbaik: {'C': 100, 'penalty': 'l2', 'solver': 'liblinear'}
dengan nilai akurasi pada data training: 0.8985847938528249
Nilai f1 rata-rata: 0.39344050694173405
Nilai AUC-ROC rata-rata: 0.8793275639312098

Nilai simpangan baku f1: 0.027336049600660994
Nilai simpangan baku AUC-ROC: 0.006400545548174557
```

Untuk memvalidasi model yang sudah dibangun, dilakukan prediksi terhadap data validasi kemudian dicari nilai akurasi serta AOC-ROC.

```
[ ] pred = logisticregression_classifierCV.predict(X_test)
evaluator(y_test, pred)
```

```
[ ] Accuracy is: 0.8963839433816212
```

Classification Report:

	precision	recall	f1-score	support
0	0.91	0.98	0.94	7960
1	0.65	0.29	0.41	1083
accuracy			0.90	9043
macro avg	0.78	0.64	0.67	9043
weighted avg	0.88	0.90	0.88	9043

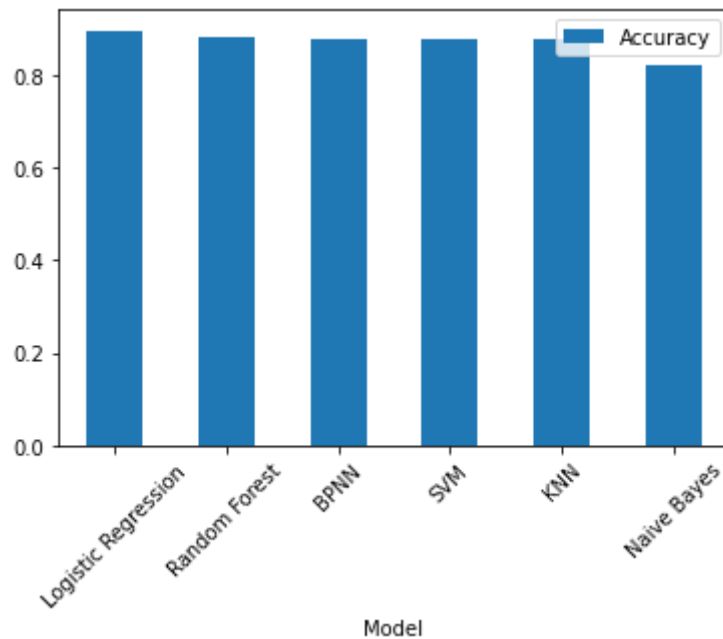
AUC_ROC Score: 0.6364092507783609

7. Analisis Model

Setelah dilakukan proses pembangunan model menggunakan beberapa metode, selanjutnya kami melakukan analisis model yang sudah dibangun dengan membandingkan tingkat akurasi dari masing-masing metode. Perbandingan dilakukan dengan cara menggunakan bar plot Untuk hasil akurasi dari masing-masing metode yaitu Random forest sebesar 0.881, KNN sebesar 0.877, naive bayes sebesar 0.822, SVM sebesar 0.879, BPNN sebesar 0.880, dan logistic regression sebesar 0.896. Sehingga dapat ditarik kesimpulan bahwa metode logistic regression merupakan metode terbaik yang bisa diterapkan pada jenis data ini.

```
[2] # record the model performance
randomforest_performance = 0.881
knn_performance = 0.877
naivebayes_performance = 0.822
svm_performance = 0.879
bpnn_performance = 0.880
logisticregression_performance = 0.896

# store the model performance in a table
model_performance = pd.DataFrame({
    'Model': [ "Random Forest", "KNN", "Naive Bayes", "SVM", "BPNN", "Logistic Regression"],
    'Accuracy': [ randomforest_performance, knn_performance, naivebayes_performance, svm_performance, bpnn_performance, logisticregression_performance]
})
model_performance = model_performance.sort_values("Accuracy", ascending = False)
model_performance.plot(x = "Model", kind= "bar", rot= 45)
```



Setelah kami menentukan metode logistic regression menjadi metode terbaik dalam pengolahan data ini, lalu kami coba melakukan validasi dengan cara melakukan evaluasi kinerja dari model dasar klasifikasi logistic regression dengan menggunakan data validation. Berikut merupakan hasil validasi model logistic regression yang kami peroleh.

```

▶ pred = logisticregression_classifierCV.predict(x_valid)
  evaluator(y_valid, pred)

Accuracy is: 0.8964830789648308

Classification Report:
              precision    recall  f1-score   support

     0       0.91      0.98      0.94       7989
     1       0.63      0.28      0.38       1053

   accuracy          0.90       9042
  macro avg       0.77      0.63      0.66       9042
 weighted avg     0.88      0.90      0.88       9042

AUC_ROC Score: 0.6268744167104412

```

8. Eksperimen Backward dan Forward Feature Selection

Selanjutnya, dilakukan eksperimen untuk memilih 10 feature menggunakan metode backward selection dan forward selection. Untuk melakukan ini, digunakan bantuan library sklearn dengan modul 'SequentialFeatureSelector'. Modul ini akan menambahkan (forward selection) atau menghapus (backward selection) fitur untuk membentuk subset fitur dengan cara greedy. Pada setiap tahap, estimator akan memilih fitur terbaik untuk ditambahkan atau dihapus berdasarkan skor cross validation dari estimator. Berdasarkan hasil pada langkah sebelumnya, model terbaik yang didapatkan adalah model yang

menggunakan metode logistic regression dengan parameter yaitu 'C': 100, 'penalty': 'l2', dan 'solver': 'liblinear'.

Kemudian untuk mempermudah feature selection, didefinisikan sebuah fungsi yaitu 'sfs_feature_selection'(source:<https://towardsdatascience.com/deep-dive-on-ml-techniques-for-feature-selection-in-python-part-2-c258f8a2ac43>). Fungsi ini memiliki beberapa masukan yaitu data training yang sudah dipisah antara variabel dependen dan variabel independen, jumlah fitur yang dipilih, nilai cv, arah seleksi, serta matriks penilaian. Fungsi akan memberi luaran sebuah data frame yang berisi daftar feature terbaik.

```
[15] def sfs_feature_selection(data, train_target,sfs_feature,sfs_direction,sfs_cv,sfs_scoring):  
  
    #Inputs  
    # data - Input feature data  
    # train_target - Target variable training data  
    # sfs_feature - no. of features to select  
    # sfs_direction - forward and backward selection  
    # sfs_cv - cross-validation splitting strategy  
    # sfs_scoring - CV performance scoring metric  
  
    logistic = LogisticRegression(C=100, solver='newton-cg')  
  
    sfs = SequentialFeatureSelector(estimator = logistic, n_features_to_select = sfs_feature, direction = sfs_direction,cv = sfs_cv, scoring = sfs_scoring)  
    sfs.fit(data, train_target)  
    sfs.get_support()  
  
    sfs_df = pd.DataFrame(columns = ['Feature', 'SFS_filter'])  
    sfs_df['Feature'] = data.columns  
    sfs_df['SFS_filter'] = sfs.get_support().tolist()  
  
    sfs_df_v2 = sfs_df[sfs_df['SFS_filter']==True]  
    sfs_top_features = sfs_df_v2['Feature'].tolist()  
  
    sfs_top_features_df = pd.DataFrame(sfs_top_features,columns = ['Feature'])  
    sfs_top_features_df['Method'] = 'Sequential_feature_selector'  
  
    return sfs_top_features_df,sfs
```

Setelah mendefinisikan, selanjutnya dilakukan pemanggilan fungsi untuk melakukan backward selection dan forward selection. Fungsi dipanggil dengan masukkan X_train dan y_train sebagai data latih, feature yang dipilih adalah 10, arah yang digunakan yaitu backward dan forward, cv yang digunakan yaitu variabel kfold yang sudah didefinisikan di awal, serta penilaian yang digunakan adalah f1.


```
# Backward Selection  
sfs_top_features_df_backward,sfs_backward = sfs_feature_selection(X_train,y_train,10,'backward',kfold,'f1')  
sfs_top_features_df_backward.head(n=20)  
# Forward Selection  
sfs_top_features_df_forward,sfs_forward = sfs_feature_selection(X_train,y_train,10,'forward',kfold,'f1')  
sfs_top_features_df_forward.head(n=20)
```

Setelah dijalankan kedua fungsi tersebut, akan didapat dua dataframe yaitu sfs_top_features_df_backward yang berisi 10 feature terpilih menggunakan backward selection dan sfs_top_features_df_forward yang berisi 10 feature terpilih menggunakan forward selection. Berikut adalah 10 fitur terpilih backward selection

```
[23] sfs_top_features_df_backward.head(n=20)
```

	Feature	Method
0	age	Sequential_feature_selector
1	job	Sequential_feature_selector
2	marital	Sequential_feature_selector
3	housing	Sequential_feature_selector
4	loan	Sequential_feature_selector
5	contact	Sequential_feature_selector
6	duration	Sequential_feature_selector
7	campaign	Sequential_feature_selector
8	pdays	Sequential_feature_selector
9	poutcome	Sequential_feature_selector

Berikut adalah 10 fitur terpilih forward selection

```
 sfs_top_features_df_forward.head(n=20)
```

	Feature	Method
0	default	Sequential_feature_selector
1	housing	Sequential_feature_selector
2	contact	Sequential_feature_selector
3	day	Sequential_feature_selector
4	month	Sequential_feature_selector
5	duration	Sequential_feature_selector
6	campaign	Sequential_feature_selector
7	pdays	Sequential_feature_selector
8	previous	Sequential_feature_selector
9	poutcome	Sequential_feature_selector