MAKALAH SAINS DATA

Identifikasi Nasabah Portuguese Bank untuk Berinvestasi pada Deposito Jangka Panjang dengan Menggunakan Artificial Neural Network (ANN)

Disusun sebagai pemenuhan tugas mata kuliah Sains Data

Dosen pengampu
Devvi Sarwinda, M.Kom.

Oleh
Kelas Sains Data (B)

Tulus Setiawan	2006568802
Cornelius Justin Satryo Hadi	2006529796
Daffa Al Ghifary	2006463420

ABSTRAK

Neural network merupakan salah satu metode dalam kecerdasan buatan. Dengan adanya neural network, suatu komputer dapat diberikan suatu kemampuan untuk membuat program yang dapat memahami pola dan menyelesaikan masalah yang terinspiasi dari cara kerja syaraf-syaraf otak manusia. Aplikasi neural network sangatlah banyak, salah satunya neural network dapat menyelesaikan masalah yang dialami Portuguese Bank yaitu penurunan pendapatan. Setelah diinvestigasi, ternyata hal ini disebabkan nasabah Portuguese Bank tidak cukup berinvestasi pada deposito jangka panjang.

Penelitian ini dilakukan untuk memprediksi pengklasifikasian seorang nasabah Portuguese Bank dan fakor-faktor apa saja yang paling berpengaruh dalam memprediksi nasabah yang akan melakukan deposito jangka panjang. Metode yang digunakan dalam penilitian ini adalah klasifikasi biner dengan menggunakan model Artifial Neural Network..

Data yang digunakan pada penelitian ini terkait marekting campaign dari Portuguese Bank dimana terdapat data train yang berisi 32950 baris dan 16 fitur d termasuk target fitur serta daa test test yang berisi 8238 dan 13 fitur tanpa target fitur.

Hasil dari penilitian ini didapat terdapat tiga fitur yang paling berpengaruh dalam memprediksi nasabah yang akan melakukan deposito jangka panjang adalah *duration*, *month*, dan *age*, secara berurutan.Selain itu, dengan model ANN mencapai akurasi dan f1-score sebesar 93% yang penulis buat memprediksi bahwa terdapat 878 nasabah (dari data test) yang akan melakukan depostio jangka panjang.

BAB I PENDAHULUAN

Neural network merupakan salah satu metode dalam kecerdasan buatan (artificial intelligence) atau disingkat AI. Sesuai dengan namanya, Neural network memungkinkan seorang data scientist atau pekerjaan yang terkait untuk memberikan kemampuan kepada komputer untuk membuat program yang dapat memahami pola dan menyelesaikan masalah yang terinspirasi dari cara kerja syaraf-syaraf otak manusia. Proses machine learning dalam neural network disebut deep learning yang menggunakan node atau neuron yang saling berhubungan dalam struktur berlapis yang menyerupai otak manusia.

Aplikasi dari neural network sangatlah banyak, terutama dalam bidang kecerdasan buatan atau artificial intelligence (AI) dan ilmu komputer. Penerapannya pun dalam berbagai bidang seperti penegakan hukum, keuangan, customer service, kesehatan, dan lain-lain. Pengaplikasian penting dari neural network seperti computer vision, speech recognition, natural language processing, recommendation engines, forecasting, classification, dan lain sebagainya.

Salah satu pengaplikasian neural network adalah dalam menyelesaikan masalah pengklasifikasian.. Dalam hal ini Portuguese Bank memiliki masalah penurunan pendapatan. Setelah dilakukan investigasi ternyata hal ini disebabkan pelanggan Portuguese Bank yang tidak cukup berinvestasi pada deposito jangka panjang. Dalam masalah tersebut akan diidentifikasi menggunakan algoritma neural network pelanggan atau customer yang memiliki kesempatan lebih tinggi untuk melakukan investasi deposito jangka panjang di Portuguese Bank sehingga usaha pemasaran akan difokuskan kepada customer yang berpotensial saja. Tujuan akhir dari pengklasifikasian nya adalah untuk memprediksi apakah seorang client atau customer akan melakukan investasi deposito jangka panjang atau tidak

BAB II DATA DAN METODE

2.1. Data

Penulis menggunakan sebuah data set publik yang berasal dari kaggle, di dalam data set berisi terkait kampanye marketing langsung dari institusi bank Portuguese. Di mana kampanye marketing berdasarkan panggilan telepon. Perlu diperhatikan juga bahwa di dalam dataset terdapat lebih dari satu kontak kepada client yang sama.

Terdapat dua data set. Pertama, train.csv dengan sampel (baris) sebanyak 32950 dan 16 fitur termasuk target fitur, yang diurutkan berdasarkan tanggal (dari Mei 2008 hingga November 2010). Kedua, test.csv dengan baris sebanyak 8238 dan 13 fitur tanpa target fitur.

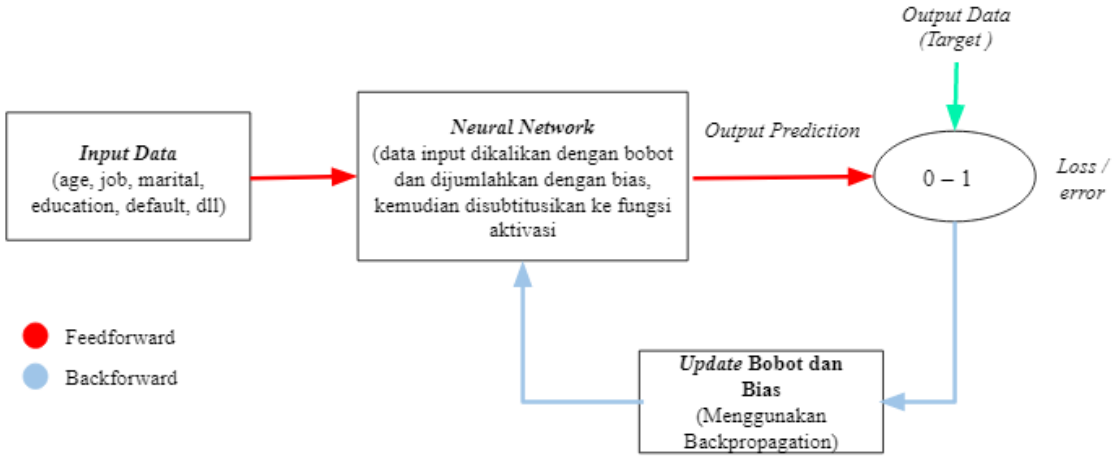
Berikut penjelasan dari fitur yang terdapat dalam data set :

Fitur	Deskripsi
age	Umur sesorang
job	Tipe pekerjaan
marital	Status pernikahan
education	Status edukasi
default	Apakah memiliki kredit secara default?
housing	Apakah memiliki pinjaman rumah ?
loan	Apakah memiliki pinjaman pribadi?
contact	Tipe kontak komunikasi
month	Bulan dalam tahun terakhir dikontak
dayofweek	Hari dalam minggu terkahir dikontak
duration	Durasi terakhir dikontak dalam detik
campaign	Jumlah kontak yang dilakukan saat campaign dilaksanakan (termasuk kontak terakhir)
pdays	Jumlah hari yang telah berlalu sejak terakhir customer dikontak
previous	Jumlah kontak yang dilakukan sebelum campaign dilaksanakan (termasuk kontak terakhir)
poutcome	Hasil akhir dari marketing campaign sebelumnya
y (target variable)	Apakah customer sudah berlangganan pada deposito berjangka

2.2. Metode

2.2.1. Tahapan ANN *(Artificial Neural Network)*

Secara umum tahapan ANN dapat divisualisasikan sebagai berikut:



Tujuan dari tahapan ANN adalah ingin menghasilkan suatu output yaitu berupa *loss function* yang optimal. Sehingga untuk menghasilkan *loss function* yang optimal dibutuhkan suatu proses *learning algorithm* yang biasa digunakan yaitu *gradient descent learning algorithm*.

2.2.2. Feedforward

Pada proses learning algorithm, hal pertama yang dilakukan adalah menginput data training example. Kemudian, untuk setiap training example x, atur input aktivasi dan pada proses feedforward hitung fungsi aktivasi $a^{x,l} = \sigma(z^{x,l})$ pada layer l =2,3,...L dimana $z^{x,l} = w^l a^{x,l-1}$. Perhatikan w merupakan bobot pada suatu neuron dan σ merupakan fungsi aktivasi.

2.2.3. Metode *Backpropagation* pada *Backward*

Backpropagation adalah metode sistematis untuk melatih jaringan syaraf tiruan *multilayer*. Metode ini memiliki basis matematika yang objektif dan solid, algoritma ini memperoleh bobot dan bias yang optimal dengan meminimalkan *loss function* yang digunakan pada model yang dikembangkan. Pada proses *backpropagation* akan dihitung output layer atau *loss function* pada layer terakhir (output) dengan menggunakan formula $\delta_j^{x,L} = \frac{\partial C}{\partial a_j^{x,L}} \sigma'(z_j^{x,L})$. Kemudian hitung *backpropagate the error* dengan formula $\delta_j^{x,l} = \sum_k w_{kj}^{x,l+1} \delta_k^{x,l+1} \sigma'(z_j^{x,l})$ untuk setiap l =L-1,L-2,....,2. Setelah itu, akan memperbarui bobot dengan formula $w^l \rightarrow w^l - \frac{\eta}{m} \sum_x \delta^{x,l} (a^{x,l-1})^T$ dan memperbarui bias dengan formula $b^l \rightarrow b^l - \frac{\eta}{m} \sum_x \delta^{x,l}$. Perhatikan bawa η merupakan learning rate dan m merupakan batch size.

2.2.4. Tahapan *PreProcessing*

Preprocessing adalah suatu tahapan dimana data dipersiapkan dan diolah dengan beberapa metode dan tahapan sebelum data masuk kedalam tahap pemrosesan untuk pembuatan model.

2.2.4.1 Observasi Data

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32950 entries, 0 to 32949
Data columns (total 16 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   age          32950 non-null  int64
 1   job          32950 non-null  object
 2   marital      32950 non-null  object
 3   education    32950 non-null  object
 4   default      32950 non-null  object
 5   housing      32950 non-null  object
 6   loan         32950 non-null  object
 7   contact      32950 non-null  object
 8   month        32950 non-null  object
 9   day_of_week  32950 non-null  object
 10  duration     32950 non-null  int64
 11  campaign     32950 non-null  int64
 12  pdays        32950 non-null  int64
 13  previous     32950 non-null  int64
 14  poutcome     32950 non-null  object
 15  y            32950 non-null  object
dtypes: int64(5), object(11)
```

```
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8238 entries, 0 to 8237
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   age          8238 non-null   int64
 1   job          8238 non-null   int64
 2   marital      8238 non-null   int64
 3   education    8238 non-null   int64
 4   default      8238 non-null   int64
 5   housing      8238 non-null   int64
 6   loan         8238 non-null   int64
 7   contact      8238 non-null   int64
 8   month        8238 non-null   int64
 9   day_of_week  8238 non-null   int64
 10  duration     8238 non-null   int64
 11  campaign     8238 non-null   int64
 12  poutcome     8238 non-null   int64
dtypes: int64(13)
memory usage: 836.8 KB
```

Seperti yang sudah dijelaskan pada subbab 2.1 bahwa terdapat 2 dataset yang digunakan, yaitu dataset train dengan sampel (baris) sebanyak 32950 dan 16 fitur termasuk target fitur dan juga dataset test dengan baris sebanyak 8238 dan 13 fitur tanpa target fitur. Terlihat juga pada gambar bahwa untuk dataset test terdapat 5 fitur yang merupakan data numerik dan 11 fitur data kategorik. Kemudian, untuk dataset test semua datanya sudah dalam bentuk numerik. Oleh karena itu, akan digunakan dataset train untuk di analisis dan evaluasi untuk dijadikan data latih pada proses pemodelan, kemudian untuk dataset test hanya akan digunakan pada tahap prediksi pada final model.

2.2.4.2 Eksplorasi Data Numerik

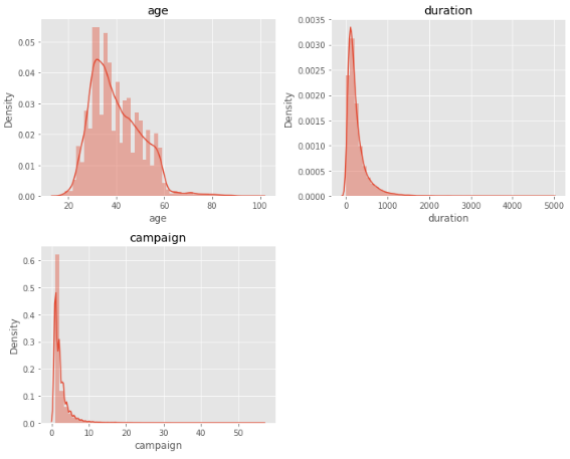
Pada tahapan ini, akan di eksplor dan analisis perilaku data yang terdapat pada fitur-fitur data numerik.

	age	duration	campaign	pdays	previous
0	49	227	4	999	0
1	37	202	2	999	1
2	78	1148	1	999	0
3	36	120	2	999	0
4	59	368	2	999	0

Pada gambar disamping terlihat lima baris pertama untuk fitur-fitur data numerik. Setelah di analsis secara keseluruhan untuk fitur pdays dan previous. Dapat disimpulkan bahwa sebagian besar pelanggan sedang dihubungi untuk pertama kalinya karena sesuai deskripsi fitur untuk pdays nilai 999 menunjukkan bahwa pelanggan belum pernah dihubungi sebelumnya. Karena fitur pdays dan fitur previosu sebagian besar hanya terdiri dari satu nilai, variansnya cukup sedikit dan karenanya dapat dihilangkan atau di-*drop* karena secara teknis tidak akan membantu dalam prediksi.

```
train_df.drop(['pdays', 'previous'], axis=1, inplace=True)
```

Fitur pdays dan previous di-*drop* dari data utama.



Seperti yang dapat dilihat dari histogram, fitur age, duration, dan campaign sangat miring dan ini disebabkan oleh adanya outlier terdapat pada fitur-fitur ini. Oleh karena itu, pada tahapan selanjutnya akan dilakukan metode untuk menangani outlier tersebut.

	age	duration	campaign
count	32950.000000	32950.000000	32950.000000
mean	40.014112	258.127466	2.560607
std	10.403636	258.975917	2.752326
min	17.000000	0.000000	1.000000
25%	32.000000	103.000000	1.000000
50%	38.000000	180.000000	2.000000
75%	47.000000	319.000000	3.000000
max	98.000000	4918.000000	56.000000

2.2.4.3 Menangani Data Outlier

Pada tahapan ini, untuk menangani data outlier maka akan dilakukan penghitungan IQR untuk menghitung batas outlier dari ketiga fitur berikut yang nanti akan menggantikan nilai outliernya.

```
# menghitung IQR untuk menghitung batas outlier
lower_boundries = []
upper_boundries = []
for i in ["age", "duration", "campaign"]:
  IQR = train_df[i].quantile(0.75) - train_df[i].quantile(0.25)
  lower_bound = train_df[i].quantile(0.25) - (1.5*IQR)
  upper_bound = train_df[i].quantile(0.75) + (1.5*IQR)

  print(i, ":", lower_bound, ",",  upper_bound)

  lower_boundries.append(lower_bound)
  upper_boundries.append(upper_bound)
```

```
age : 9.5 , 69.5
duration : -221.0 , 643.0
campaign : -2.0 , 6.0
```

Pertama, akan dihitung IQR (Inter Quartil Range), yaitu kuartil ketiga dari data fitur dikurang dengan kuartil pertamanya. Kemudian, dihitung batas bawah fitur dengan cara kuartil pertamanya dikurang dengan 3/2 nya IQR. Selanjutnya, untuk batas atas fiturvdihitung dengan cara kuartil ketiganya ditambah dengan 3/2 nya IQR. Maka, didapatkan untuk fitur age batas bawahnya 9.5 dan batas atasnya 69.5, untuk fitur duration batas bawahnya -221 dan batas atasnya 643, dan untuk fitur campaign batas bawahnya -2 dan batas atasnya 6.

```
# mengganti nilai outlier dengan nilai batas atas fiturnya
j = 0
for i in ["age", "duration", "campaign"]:
  train_df_prepared.loc[train_df_prepared[i] > upper_boundries[j], i] = int(upper_boundries[j])
  j = j + 1
```

Selanjutnya, setelah sudah didapatkan nilai batas untuk setiap fitur, maka nilai batas tersebut akan menggantikan nilai-nilai outlier yang terdapat pada setiap fitur tersebut.

	age	duration	campaign
count	32950.000000	32950.000000	32950.000000
mean	39.929894	234.923915	2.271077
std	10.118566	176.854558	1.546302
min	17.000000	0.000000	1.000000
25%	32.000000	103.000000	1.000000
50%	38.000000	180.000000	2.000000
75%	47.000000	319.000000	3.000000
max	69.000000	643.000000	6.000000

Sehingga sekarang untuk persebaran rata rata data lebih normal dan cenderung rendah dari pada data sebelumnya yang masih terdapat data outlier.

2.2.4.4 Encoding Data Kategorik

Pada tahapan ini, akan dilakukan encoding atau pengubahan data data pada fitur yang bernilai kategorik menjadi data numerik. Proses ini dilakukan karena pada tahap pemrosesan data untuk pembuatan model, data haruslah dalam bentuk numerik di dalam array. Oleh karena itu, sebelum data di proses maka akan diubah terlebih dahulu.

```
# inisialisasi ordinal encoder
oe = OrdinalEncoder()

train_df_prepared[cat_var] = oe.fit_transform(train_df_prepared[cat_var])
```

```
oe.categories_
```

```
[array(['admin.', 'blue-collar', 'entrepreneur', 'housemaid', 'management',
        'retired', 'self-employed', 'services', 'student', 'technician',
        'unemployed', 'unknown'], dtype=object),
 array(['divorced', 'married', 'single', 'unknown'], dtype=object),
 array(['basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'illiterate',
        'professional.course', 'university.degree', 'unknown'],
       dtype=object),
 array(['no', 'unknown', 'yes'], dtype=object),
 array(['no', 'unknown', 'yes'], dtype=object),
 array(['no', 'unknown', 'yes'], dtype=object),
 array(['cellular', 'telephone'], dtype=object),
 array(['apr', 'aug', 'dec', 'jul', 'jun', 'mar', 'may', 'nov', 'oct',
        'sep'], dtype=object),
 array(['fri', 'mon', 'thu', 'tue', 'wed'], dtype=object),
 array(['failure', 'nonexistent', 'success'], dtype=object),
 array(['no', 'yes'], dtype=object)]
```

Pada gambar disamping terlihat bahwa terdapat 11 fitur data kategorik yang di dalam setiap fiturnya terdapat beberapa nilai kategori yang akan diubah menjadi numerik berdasarkan indeksnya, sebagai contoh untuk fitur terakhir/paling bawah terdapat dua nilai, yaitu no dan yes karena no berada pada indeks ke-0 dan yes pada indeks ke-1 di python, maka setelah diencoding nilai no berubah menjadi 0 dan yes menjadi 1.
Kemudian, untuk gambar selanjutnya merupakan data data yang sudah dilakukan encoding dari data kategorik menjadi data numerik berserta juga data yang dari awal sudah bernilai numerik.

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	poutcome	y
0	49	1.0	1.0	2.0	1.0	0.0	0.0	0.0	7.0	4.0	227	4	1.0	0.0
1	37	2.0	1.0	6.0	0.0	0.0	0.0	1.0	7.0	4.0	202	2	0.0	0.0
2	69	5.0	1.0	0.0	0.0	0.0	0.0	0.0	3.0	1.0	643	1	1.0	1.0
3	36	0.0	1.0	6.0	0.0	2.0	0.0	1.0	6.0	1.0	120	2	1.0	0.0
4	59	5.0	0.0	6.0	0.0	0.0	0.0	0.0	4.0	3.0	368	2	1.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32945	28	7.0	2.0	3.0	0.0	2.0	0.0	0.0	3.0	3.0	192	1	1.0	0.0
32946	52	9.0	1.0	5.0	0.0	2.0	0.0	0.0	7.0	0.0	64	1	0.0	0.0
32947	54	0.0	1.0	2.0	0.0	0.0	2.0	0.0	3.0	1.0	131	4	1.0	0.0
32948	29	0.0	1.0	6.0	0.0	0.0	0.0	1.0	6.0	0.0	165	1	1.0	0.0
32949	35	0.0	1.0	6.0	0.0	0.0	2.0	1.0	4.0	3.0	544	3	1.0	0.0

2.2.4.5 Stadarisasi Data Numerik

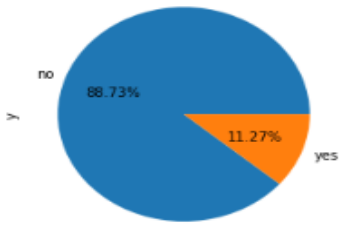
Pada tahapan sebelumnya, dapat dilihat pada tabel. Setelah semua data berbentuk numerik, terlihat bahwa untuk beberapa nilai pada fitur memiliki skala perbandingan yang besar. Oleh karena itu, akan dilakukan standarisasai skalanya menggunakan *Standart Scaler* supaya perbandingan setiap nilai tidak terlalu besar sehingga model yang dibuat nanti semakin bagus. Namun, jangan lupa bahwa fitur target tidak termasuk dalam data yang distandarisai, hal ini karena data pada fitur target bernilai biner, 0 atau 1, no atau yes.

```
std_scale = StandardScaler()
X.iloc[:,:] = std_scale.fit_transform(X)
X
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	poutcome
0	0.896396	-0.757779	-0.284871	-0.818379	1.933816	-1.08747	-0.453839	-0.758915	1.192670	1.427938	-0.044805	1.118118	0.193670
1	-0.289561	-0.479529	-0.284871	1.053452	-0.516547	-1.08747	-0.453839	1.317671	1.192670	1.427938	-0.186167	-0.175310	-2.552217
2	2.872991	0.355224	-0.284871	-1.754295	-0.516547	-1.08747	-0.453839	-0.758915	-0.531722	-0.714554	2.307446	-0.822023	0.193670
3	-0.388390	-1.036030	-0.284871	1.053452	-0.516547	0.94245	-0.453839	1.317671	0.761572	-0.714554	-0.649831	-0.175310	0.193670
4	1.884693	0.355224	-1.928167	1.053452	-0.516547	-1.08747	-0.453839	-0.758915	-0.100624	0.713774	0.752472	-0.175310	0.193670
...	...	...	...	...	...	...	...	...	...	...	...	...	...
32945	-1.179028	0.911725	1.358424	-0.350421	-0.516547	0.94245	-0.453839	-0.758915	-0.531722	0.713774	-0.242711	-0.822023	0.193670
32946	1.192885	1.468227	-0.284871	0.585494	-0.516547	0.94245	-0.453839	-0.758915	1.192670	-1.428718	-0.966481	-0.822023	-2.552217
32947	1.390545	-1.036030	-0.284871	-0.818379	-0.516547	-1.08747	2.304690	-0.758915	-0.531722	-0.714554	-0.587633	1.118118	0.193670
32948	-1.080198	-1.036030	-0.284871	1.053452	-0.516547	-1.08747	-0.453839	1.317671	0.761572	-1.428718	-0.395381	-0.822023	0.193670
32949	-0.487220	-1.036030	-0.284871	1.053452	-0.516547	-1.08747	2.304690	1.317671	-0.100624	0.713774	1.747655	0.471404	0.193670

Terlihat pada gambar di samping bahwa data mengalami perubahan dari yang sebelumnya memiliki skala perbandingan yang cukup besar sekarang menjadi lebih kecil dan standart.

2.2.4.6 Menangani Data Tidak Seimbang



Pada gambar disamping terlihat bahwa pada fitur target, jumlah data yang bernilai *'no'* sebesar 88.73% atau 29238 dari total keseluruhan dan jumlah data yang bernilai '*yes'* sebesar 11.27% atau 3712 dari total keseluruhan. Hal ini menggambarkan bahwa jumlah data untuk setiap nilai pada fitur target sangatlah tidak seimbang dan hal ini akan memengaruhi saat pelatihan data saat proses pemodelan dimana model terlalu banyak dilatih untuk data bernilai '*no'* sehingga akan memengaruhi kualitas akhir dari model yang dibuat. Oleh karena itu, akan dilakukan metode oversampling supaya jumlah data yang lebih sedikit akan naik mendekati jumlah data yang paling banyak.

```
train_df.y.value_counts()
```

```
no     29238
yes     3712
Name: y, dtype: int64
```

```
# initialising oversampling
smote = SMOTETomek(0.75)

# implementing oversampling to training data
X_sm, y_sm = smote.fit_resample(X, y)

# target class count of resampled dataset
y_sm.value_counts()
```

```
0    29106
1    21796
Name: y, dtype: int64
```

Pada gambar disamping terlihat bahwa akan digunakan metode oversamping dengan SMOTETomek(0.75) yang berasal dari module imblearn. Maksud dari angka 0.75 itu sendiri, yaitu jumlah data dari nilai '*yes*' akan dinaikkan sebesar 75% dari jumlah data nilai '*no*'. Kemudian, karena digunakannya fungsi SMOTETomek(), maka jumlah data dari setiap nilai akan dikurangkan sedikit dari jumlah data yang seharusnya sehingga didapatlah jumlah data untuk nilai '*no*' sebesar 29106 dan jumalh data untuk nilai *'yes*' sebesar 21796. Karena kedua data sekarang sudah cukup seimbang, maka dapat dilanjutkan ke tahapan split dataset yang nantinya akan di proses untuk pemodelan.

BAB III IMPLEMENTASI DAN ANALISIS DATA

3.1. *Split* Data Training dan Data Validasi

```
[ ]  X_train, X_val, y_train, y_val = train_test_split(X_sm, y_sm, test_size=0.2, random_state=42)
     X_train.shape, X_val.shape, y_train.shape, y_val.shape

     ((40721, 13), (10181, 13), (40721,), (10181,))
```

Dilakukan *splitting* data hasil *oversampling,* menjadi data training dan data validasi dengan perbandingan 80(training):20(validasi). Diperoleh data training sebanyak 40721 sampel dan data validasi sebanyak 10181 sampel.

3.2. Mendefinisikan *Function* untuk Membuat Model ANN

Mendefinisikan model

```
[ ]  def build_model(n_neurons=(15,15), learning_rate=3e-3, activation_hidden='relu'):
       model = keras.models.Sequential()
       model.add(keras.layers.InputLayer(input_shape=[13]))
       for i in range(len(n_neurons)):
         model.add(keras.layers.Dense(n_neurons[i], activation=activation_hidden))
       model.add(keras.layers.Dense(1, activation='sigmoid'))
       optimizer = keras.optimizers.Adam(learning_rate=learning_rate)
       model.compile(loss='binary_crossentropy', optimizer=optimizer,
                     metrics='accuracy')
       return model
```

```
[ ]  model = KerasClassifier(build_model, epochs=100, batch_size=200)
```

Evaluasi model dengan inisialisasi hyperparameter menggunakan "intuisi", yaitu jumlah hidden layer = 2, dengan masing - masing layer memiliki 15 neuron, learning rate = 0.003, activation function untuk hidden layer = relu, epochs = 100, dan batch size = 200.

```
cv_mean_accuracy = cross_val_score(model, X_train, y_train, cv=3, scoring='accuracy').mean()
```

Show hidden output

```
[ ]  cv_mean_accuracy

     0.8680708990032896
```

Function build_model didefinisikan untuk membuat model ANN dengan jumlah *hidden layer*, jumlah neuron pada *hidden layer*, *learning rate,* dan fungsi aktivasi sebagai input dari user. Sehingga, user dapat mengatur sendiri hyperparameter yang diinginkan pada input tersebut.

Selanjutnya, diinisialisasi model ANN menggunakan intuisi dengan hyperparameter: jumlah neuron di *input layer* sebanyak 13, jumlah *hidden layer* sebanyak 2, jumlah neuron pada masing - masing *hidden layer* sebanyak 15, *learning rate* = 0.003, fungsi aktivasi di *hidden layer* = relu, jumlah neuron di *output layer* sebanyak 1, fungsi aktivasi di *output layer* = sigmoid, *optimizer* = Adam, dan *loss function* = binary cross entropy/log loss. Dengan menggunakan hyperparameter tersebut, didapatkan akurasi dengan metode Cross Validation 3-folds sebesar 86,80%. Akurasi yang diperoleh cukup baik, tetapi penulis ingin mencapai akurasi setidaknya sebesar 90%, sehingga perlu dilakukan *hyperparameter tuning*.

3.3. *Hyperparameter Tuning*

Hyperparameter tuning jumlah hidden layer dan jumlah neurons

Pertama akan dicari jumlah hidden layer dan jumlah neuron pada hidden layernya yang menghasilkan akurasi terbaik, dengan metode grid search

```
[ ]  params_grid1 = {'n_neurons':[(15,), (15,15), (20,), (20,15), (30,),
                                  (30,20), (50,), (50,30), (70,), (70,50),
                                  (100,), (100,70), (130,), (130,100)]}

     grid_search1 = GridSearchCV(model, params_grid1, cv=3, scoring='accuracy',
                                 verbose=2)
     grid_search1.fit(X_train, y_train, verbose=0)
```

1

Show hidden output

```
[ ]  grid_search1.cv_results_['mean_test_score'].max(), grid_search1.best_params_

     (0.9106643099130945, {'n_neurons': (130, 100)})
```

```
[ ]  gs1_result = pd.DataFrame(grid_search1.cv_results_)
     gs1_result.sort_values('rank_test_score')[['param_n_neurons', 'mean_test_score', 'rank_test_score']]
```

	param_n_neurons	mean_test_score	rank_test_score
13	(130, 100)	0.910664	1
11	(100, 70)	0.903569	2
9	(70, 50)	0.893283	3
7	(50, 30)	0.886188	4
12	(130,)	0.880689	5
10	(100,)	0.875338	6
5	(30, 20)	0.873840	7
8	(70,)	0.871287	8
6	(50,)	0.866524	9
3	(20, 15)	0.866107	10
1	(15, 15)	0.859258	11
4	(30,)	0.858644	12
2	(20,)	0.857122	13
0	(15,)	0.847253	14

Hyperparameter tuning learning rate dan activation function pada hidden layer

Selanjutnya akan dicari learning rate dan activation function di hidden layer yang akan menghasilkan akurasi terbaik

```
[ ]  params_grid2 = {'n_neurons':[(130,100)],
                     'learning_rate':[3e-4, 3e-3, 3e-2],
                     'activation_hidden':['relu','sigmoid','tanh']}

     grid_search2 = GridSearchCV(model, params_grid2, cv=3, scoring='accuracy',
                                 verbose=2)
     grid_search2.fit(X_train, y_train, verbose=0)
```

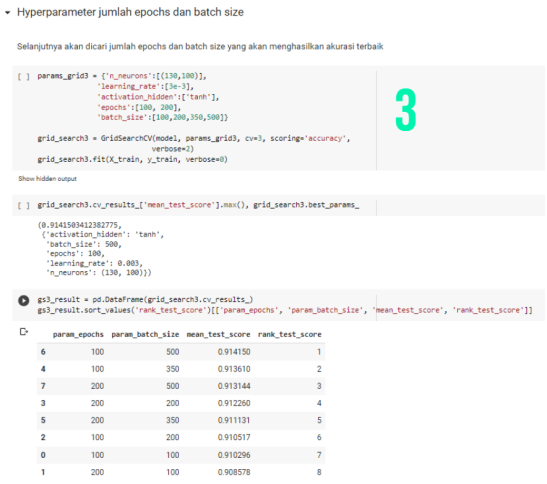
2

Show hidden output

```
[ ]  grid_search2.cv_results_['mean_test_score'].max(), grid_search2.best_params_

     (0.910443364265724,
      {'activation_hidden': 'tanh',
       'learning_rate': 0.003,
       'n_neurons': (130, 100)})
```

```
[ ]  gs2_result = pd.DataFrame(grid_search2.cv_results_)
     gs2_result.sort_values('rank_test_score')[['param_learning_rate',
                                                'param_activation_hidden',
                                                'mean_test_score',
                                                'rank_test_score']]
```

	param_learning_rate	param_activation_hidden	mean_test_score	rank_test_score
7	0.003	tanh	0.910443	1
1	0.003	relu	0.908553	2
5	0.03	sigmoid	0.902686	3
4	0.003	sigmoid	0.890755	4
2	0.03	relu	0.888324	5
0	0.0003	relu	0.887981	6
6	0.0003	tanh	0.882825	7
8	0.03	tanh	0.879265	8
3	0.0003	sigmoid	0.826312	9

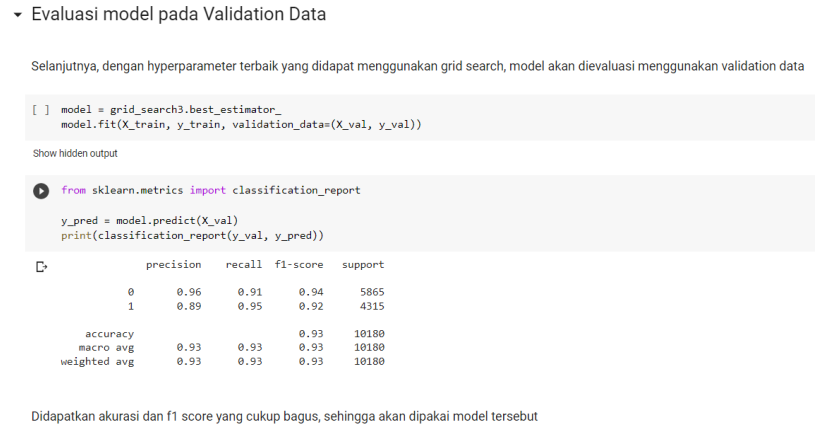


Hyperparameter Tuning dilakukan sebanyak 3 kali dengan metode Grid Search CV, untuk mencari nilai jumlah *hidden layer*, jumlah neuron di masing - masing *hidden layer*, besaran *learning rate*, fungsi aktivasi pada *hidden layer*, jumlah *epochs,* dan jumlah *batch size*.

Hyperparameter hasil Grid Search:

Hyperparameter	Nilai
Jumlah hidden layer	2
Jumlah neuron hidden layer pertama	130
Jumlah neuron hidden layer kedua	100
Learning rate	0.003
Activation function	tanh
Epochs	100
Batch size	500
Akurasi dengan cross validation	**91,4%**

3.4. Evaluasi Model Hasil Grid Search Pada Data Validasi



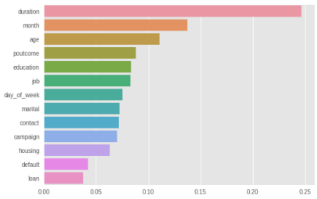
Model hasil Grid Search di-*train* pada data training dan dievaluasi pada data validasi. Diperoleh metrik evaluasi pada data validasi yang sudah baik, yaitu akurasi, f1-score, *precision,* dan *recall* sebesar 93% (semua metrik evaluasi besarannya sama). Sehingga, akan digunakan model tersebut sebagai model akhir.

3.5. Cek Kepentingan Fitur dengan Metode Permutation Importance

Selanjutnya akan dicek fitur yang mempengaruhi akurasi model menggunakan metode Permutation Importance

```
[ ] result = permutation_importance(model, X_val, y_val, n_repeats=10,
                                    scoring='accuracy', random_state=42)

for i in result.importances_mean.argsort()[::-1]:
  print(f"{X_val.columns.values[i]:<12}"
        f"{result.importances_mean[i]:.3f}"
        f" +/- {result.importances_std[i]:.3f}")

duration    0.247 +/- 0.004
month       0.137 +/- 0.003
age         0.111 +/- 0.003
poutcome    0.088 +/- 0.002
education   0.083 +/- 0.002
job         0.083 +/- 0.001
day_of_week 0.075 +/- 0.003
marital     0.073 +/- 0.002
contact     0.072 +/- 0.002
campaign    0.070 +/- 0.002
housing     0.063 +/- 0.003
default     0.042 +/- 0.002
loan        0.037 +/- 0.002
```

```
[ ] result_sorted = []
    columns_sorted = []

    for res, col in sorted(zip(result.importances_mean, X_val.columns.values), reverse=True):
      result_sorted.append(res)
      columns_sorted.append(col)

    sns.barplot(result_sorted, columns_sorted)
    plt.show()
```



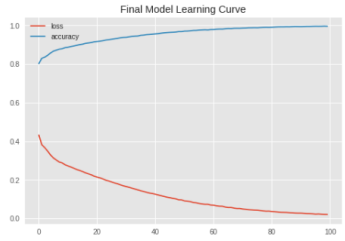
Metode *Permutation Importance* adalah metode untuk mengukur kepentingan fitur dengan menghitung peningkatan kesalahan prediksi model setelah mengubah fitur tersebut. Sebuah fitur "penting" jika mengacak nilainya meningkatkan kesalahan model, karena dalam hal ini model mengandalkan fitur untuk prediksi. Sebuah fitur dikatakan "tidak penting" jika mengacak nilainya membuat kesalahan model tidak berubah, karena dalam kasus ini model mengabaikan fitur untuk prediksi. [3]

Dengan menggunakan model yang sudah penulis *train* pada data training, diperoleh urutan kepentingan fitur (yaitu, fitur yang paling mempengaruhi akurasi hingga yang tidak terlalu mempengaruhi akurasi) seperti yang tercantum di *output* kode Python pada gambar diatas. Tiga fitur terpenting adalah *duration, month,* dan *age*, secara berurutan.

3.6. Training Model Pada Keseluruhan Data Training

Simpan dan Training model final pada keseluruhan data training

```
[ ] # Simpan model terbaik dalam final_model
    final_model = build_model(n_neurons=(130,100), activation_hidden='tanh',
                              learning_rate=3e-3)

[ ] history = final_model.fit(X_sm, y_sm, epochs=100, batch_size=500)

Show hidden output

[ ] pd.DataFrame(history.history).plot()
    plt.title('Final Model Learning Curve')
    plt.show()
```



Dilakukan training pada keseluruhan data training, yaitu data hasil *oversampling* sebelum dilakukan *splitting* menjadi data training dan data validasi.

Model akhir yang digunakan adalah sebagai berikut,

1. *Input Layer*: Jumlah neuron = 13
2. *Hidden Layer*: - Jumlah *hidden layer* = 2,
   - Jumlah neuron *hidden layer* 1 = 130,
   - Jumlah neuron *hidden layer* 2 = 100,
   - Fungsi aktivasi = tanh
3. *Output Layer*: - Jumlah neuron = 1,
   - Fungsi aktivasi = sigmoid
   - Fungsi loss = binary cross entropy/log loss

Berikut adalah formula fungsi aktivasi dan fungsi loss yang digunakan:

$$tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}},\ sigmoid(x) = \sigma(x) = \frac{1}{1+e^{-x}},\ log\ loss = -\frac{1}{N}\Sigma_{i=1}^{N}[y_i . log(p(y_i)) + (1-y_i) . log(1-p(y_i))]$$

3.7. Prediksi Model pada Data Test

```
[ ] y_pred = (y_pred_proba >= 0.5).astype(np.int)
    y_pred

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

[ ] test_df['y_prediction'] = y_pred

[ ] test_df
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	poutcome	y_prediction
0	32	4	0	6	0	0	0	0	3	3	131	5	1	0
1	37	10	3	6	0	0	0	0	4	3	100	1	1	0
2	55	5	0	5	1	2	0	0	3	2	131	2	1	0
3	44	2	1	0	1	0	0	1	4	3	48	2	1	0
4	28	0	2	3	0	0	0	0	5	0	144	2	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8233	48	4	1	2	0	2	0	0	6	3	554	1	1	0
8234	30	7	2	3	0	2	0	0	6	0	159	1	1	0
8235	33	7	1	3	0	0	0	0	4	1	472	1	0	1
8236	44	1	1	1	0	2	2	1	6	1	554	5	1	0
8237	42	1	1	2	1	2	0	0	6	3	83	5	1	0

8238 rows × 14 columns

```
[ ] test_df.y_prediction.value_counts()

0    7360
1     878
Name: y_prediction, dtype: int64
```

Data test dilakukan *scaling* kemudian diprediksi nilai targetnya dengan menggunakan model yang sudah dilatih sebelumnya. Diperoleh bahwa, sebanyak 7360 nasabah pada data test diprediksi akan menolak untuk melakukan deposito jangka panjang dan sebanyak 878 nasabah diprediksi akan melakukan deposito jangka panjang.

BAB IV KESIMPULAN

1. Tiga fitur yang paling berpengaruh dalam memprediksi nasabah yang akan melakukan deposito jangka panjang adalah *duration*, *month*, dan *age*, secara berurutan. Sehingga, disarankan kepada Portuguese Bank untuk mempertimbangkan tiga faktor tersebut sebagai salah tiga faktor terdepan ketika ingin melakukan kampanye telepon ke nasabah.

2. Model *Artificial Neural Network* (ANN) yang telah penulis buat, mencapai akurasi dan f1-score sebesar 93% (pada data validasi), yang merupakan hasil yang cukup baik. Sehingga, Portuguese Bank dapat menggunakan model tersebut sebagai salah satu acuan untuk menentukan nasabah mana yang akan melakukan deposito jangka panjang. Namun, perlu diingat juga bahwa hasil prediksi model tersebut belum tentu sama dengan kenyataannya, karena terdapat berbagai faktor lain yang dapat mempengaruhi seseorang dalam melakukan deposito jangka panjang, selain fitur - fitur yang digunakan ketika melatih model.

3. Model ANN yang telah penulis buat memprediksi bahwa terdapat 878 nasabah dari data test yang akan melakukan deposito jangka panjang.

DAFTAR PUSTAKA

[1] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow : Concepts, Tools, and Techniques to Build Intelligent Systems, California: O'Reilly Media, Inc., 2019.

[2] M. Nielsen, "CHAPTER 2 How the backpropagation algorithm works," December 2019. [Online]. Available: http://neuralnetworksanddeeplearning.com/chap2.html. [Accessed May 2022].

[3] C. Molnar, "5.5 Permutation Feature Importance," 2019. [Online]. Available: https://interpretable-ml.keamanansiber.id/feature-importance.html. [Accessed June 2022].