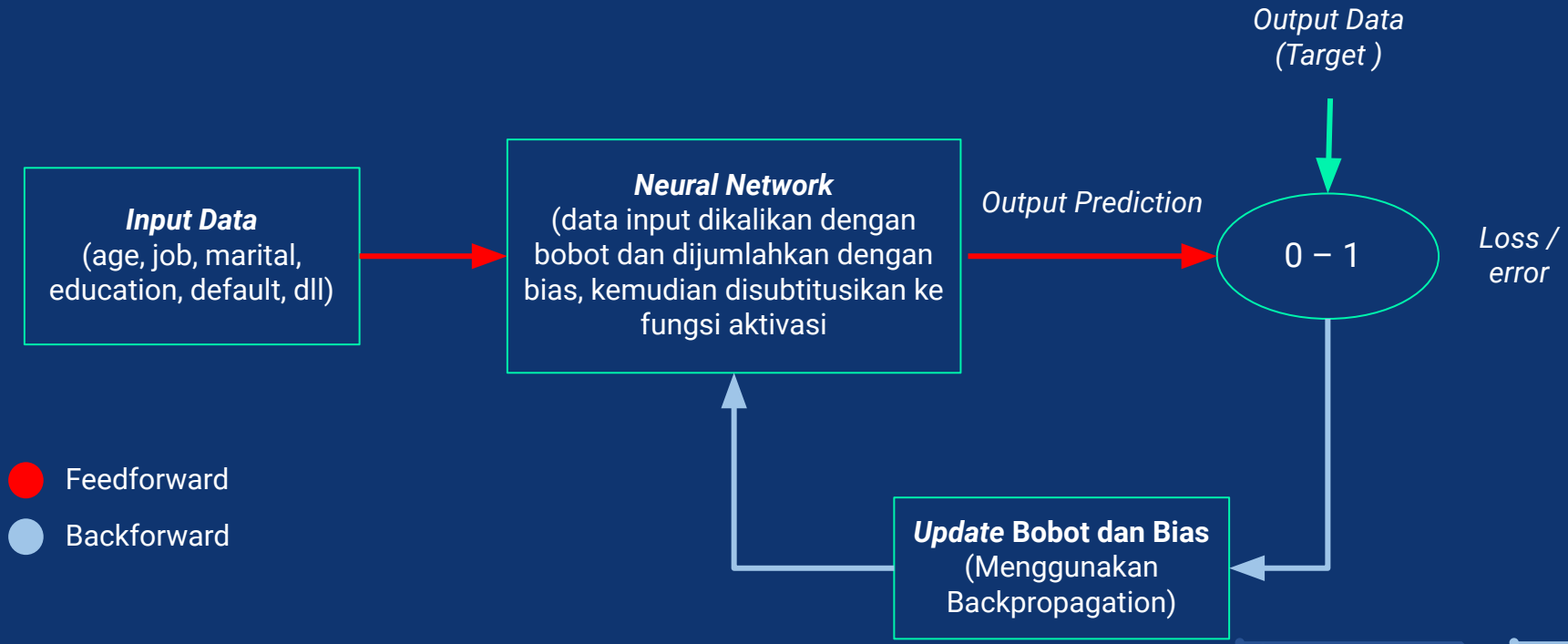
PROJEK UAS SAINS DATA

IDENTIFIKASI NASABAH PORTUGUESE BANK UNTUK BERINVESTASI PADA DEPOSITO JANGKA PANJANG DENGAN MENGGUNAKAN ARTIFICIAL NEURAL NETWORK (ANN)

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

ARTIFICIAL NEURAL NETWORK

TAHAPAN ANN


Output Data
(Target)
Input Data
(age, job, marital, education, default, dll)
Neural Network
(data input dikalikan dengan bobot dan dijumlahkan dengan bias, kemudian disubtitusikan ke fungsi aktivasi
Output Prediction
0 – 1
Loss / error
Feedforward
Backforward
Update Bobot dan Bias
(Menggunakan Backpropagation)

GRADIENT DESCENT LEARNING ALGORITHM

1

Input data training example

2

Untuk setiap training example x :
Atur input aktivasi dan lakukan :

- Feedforward : untuk setiap l = 2,3,...,L

$$z^{x,l} = w^l a^{x,l-1} + b^l \text{ and } a^{x,l} = \sigma(z^{x,l}).$$

- Output error

$$\delta_j^{x,L} = \frac{\partial C}{\partial a_j^{x,L}} \sigma'(z_j^{x,L})$$

- Backpropagate the error :

Untuk setiap l = L-1 , L-2, …., 2

$$\delta_j^{x,l} = \sum_k w_{kj}^{x,l+1} \delta_k^{x,l+1} \sigma'(z_j^{x,l})$$

3

Gradient Descent :
Untuk setiap l = L, L-1,....,2
update weights

$$w^l \to w^l - \frac{\eta}{m} \sum_x \delta^{x,l} (a^{x,l-1})^T$$

dan bias

$$b^l \to b^l - \frac{\eta}{m} \sum_x \delta^{x,l}$$

TENTANG DATASET

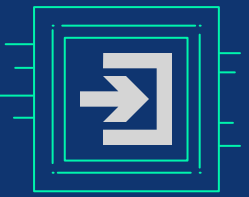
ABOUT DATA

Problem

Bank Portuguese mengalami penurunan pendapatan.

After Investigation

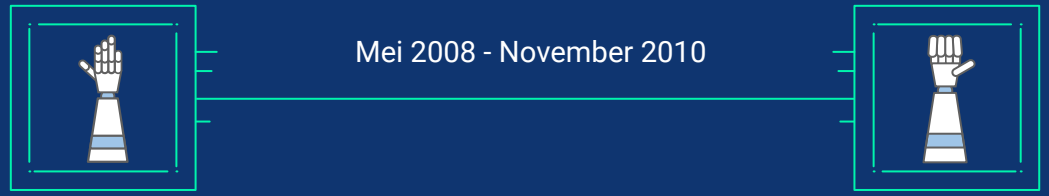
Belum banyak nasabah yang melakukan deposito jangka panjang di Bank Portuguese



What's next?

Identifikasi nasabah yang memiliki peluang lebih tinggi untuk melakukan deposito jangka panjang menggunakan klasifikasi biner dengan model ANN

ABOUT DATASET



TRAIN.CSV

32950 baris

16 fitur (termasuk target)

TEST.CSV

8238 baris

13 fitur (tanpa target)

FEATURE

- age
- job
- marital
- education
- default
- housing
- loan
- contact
- month
- dayofweek
- duration
- campaign
- pdays
- previous
- poutcome
- y (target variable)

METODE PREPROCESSING

METODE

```
1 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)
memory usage: 4.0+ MB
```

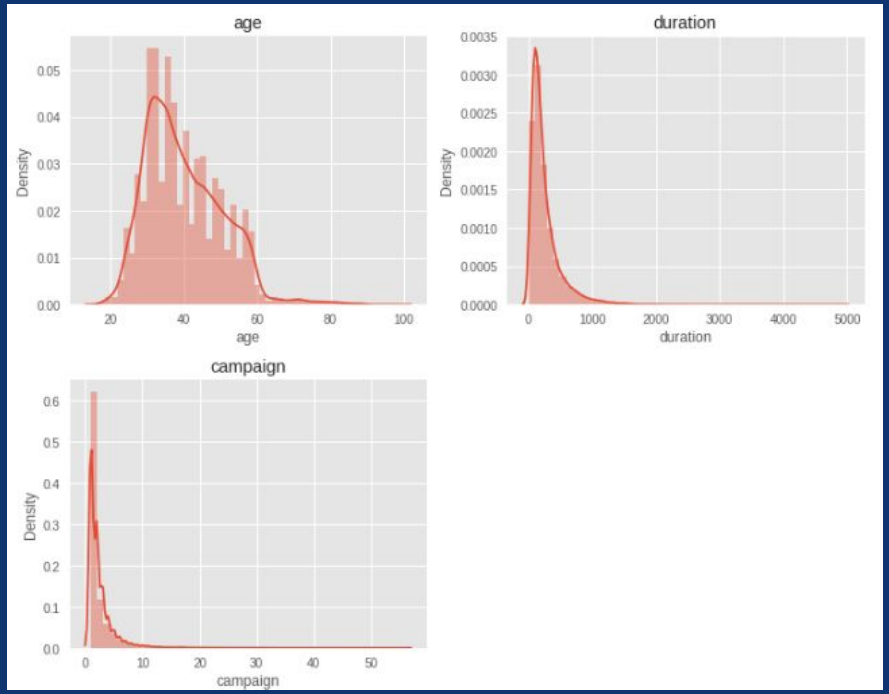
```
1 test_df.info()
```

```
<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
```

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

METODE

Ekplorasi Data Numerik




```
1 train_df.describe()
```

	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

METODE

Handle Outlier

```python
# 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
```

```python
# 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
```

```python
# train_df tanpa outlier
train_df_prepared.describe()
```

	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

METODE

Encoding Data Kategorik

```
# 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)]
```

```
train_df_prepared
```

	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

32950 rows × 14 columns

METODE

Scaling Data Numerik
dengan Standard Scaler

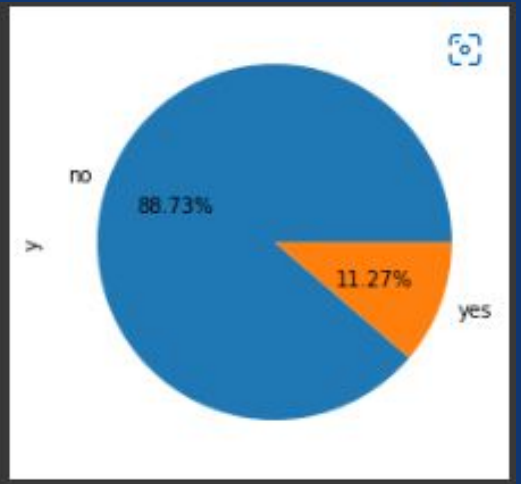
```
1 std_scale = StandardScaler()
2 X.iloc[:,:] = std_scale.fit_transform(X)
3 X
```

	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

32950 rows × 13 columns

METODE

Handle Imbalanced Dataset




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

```
no     29238
yes     3712
```

METODE

Handle Imbalanced Dataset

```
# 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    29105
1    21795
Name: y, dtype: int64
```

```
X_sm
```

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...
50895	-1.266230	1.034481	1.358424	0.716248	-0.516547	0.94245	-0.453839	-0.758915	-1.514373	0.514226	-0.088292	-0.356011	-2.552217
50896	1.785718	0.456517	-0.284871	-1.115983	1.933816	0.94245	-0.453839	-0.758915	0.800647	0.389058	-0.469368	0.530024	-2.552217
50897	-0.274603	0.076973	-0.284871	1.053452	-0.516547	0.94245	-0.453839	-0.758915	1.493275	0.215786	0.458412	0.373525	2.939558
50898	-1.064653	-0.757779	1.358424	-0.897249	-0.516547	-1.08747	2.304690	-0.758915	-0.531722	-0.834920	2.307446	2.084552	0.193670
50899	1.785864	-1.036030	-0.284871	0.207721	-0.516547	0.94245	2.304690	-0.758915	-0.017543	-0.284323	0.875445	-0.175310	0.193670

50900 rows × 13 columns

ANALISIS MODEL ANN

ANALISIS MODEL

Mendefinisikan model awal dengan 2 hidden layer, dengan masing - masing hidden layer menggunakan 15 neurons dan output layer 1 neuron

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
```

AKURASI

ANALISIS MODEL

Hyperparameter tuning menggunakan metode Grid Search dengan Cross Validation 3-folds

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

ANALISIS MODEL

Hyperparameter tuning menggunakan metode Grid Search dengan Cross Validation 3-folds

Hyperparameter jumlah epochs dan batch size

Selanjutnya akan dicari jumlah epochs dan batch size yang akan menghasilkan akurasi terbaik

```
[ ] params_grid3 = {'n_neurons':[(130,100)],
                    'learning_rate':[3e-3],
                    'activation_hidden':['tanh'],
                    'epochs':[100, 200],
                    'batch_size':[100,200,350,500]}

    grid_search3 = GridSearchCV(model, params_grid3, cv=3, scoring='accuracy',
                                verbose=2)
    grid_search3.fit(X_train, y_train, verbose=0)
```

3

Show hidden output

```
[ ] grid_search3.cv_results_['mean_test_score'].max(), grid_search3.best_params_

    (0.9141503412382775,
     {'activation_hidden': 'tanh',
      'batch_size': 500,
      'epochs': 100,
      'learning_rate': 0.003,
      'n_neurons': (130, 100)})
```

```
gs3_result = pd.DataFrame(grid_search3.cv_results_)
gs3_result.sort_values('rank_test_score')[['param_epochs', 'param_batch_size', 'mean_test_score', 'rank_test_score']]
```

	param_epochs	param_batch_size	mean_test_score	rank_test_score
6	100	500	0.914150	1
4	100	350	0.913610	2
7	200	500	0.913144	3
3	200	200	0.912260	4
5	200	350	0.911131	5
2	100	200	0.910517	6
0	100	100	0.910296	7
1	200	100	0.908578	8

ANALISIS MODEL

Berdasarkan hasil grid search, didapatkan model yang akan digunakan sebagai berikut:

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%**

ANALISIS MODEL

Evaluasi model hasil Grid Search pada data validasi

Evaluasi model pada Validation Data

Selanjutnya, dengan hyperparameter terbaik yang didapat menggunakan grid search, model akan dievaluasi menggunakan validation data

```
[ ]  model = grid_search3.best_estimator_
     model.fit(X_train, y_train, validation_data=(X_val, y_val))
```

Show hidden output

```
from sklearn.metrics import classification_report

y_pred = model.predict(X_val)
print(classification_report(y_val, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.91      0.94      5865
           1       0.89      0.95      0.92      4315

    accuracy                           0.93     10180
   macro avg       0.93      0.93      0.93     10180
weighted avg       0.93      0.93      0.93     10180
```

Didapatkan akurasi dan f1 score yang cukup bagus, sehingga akan dipakai model tersebut

ANALISIS MODEL

Cek fitur yang paling mempengaruhi akurasi 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)
```

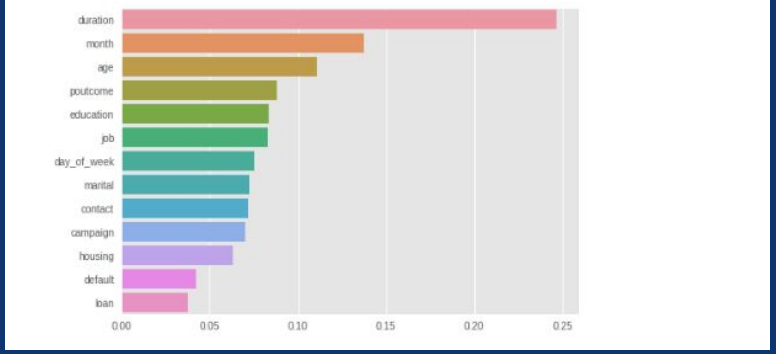
```
for i in result.importances_mean.argsort()[::-1]:
  print(f"{X_val.columns.values[i]:<12}"
        f"{result.importances_mean[i]:.3f}"
        f" +/- {result.importances_std[i]:.3f}")
```

```
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()
```



ANALISIS MODEL

Statistik deskriptif fitur duration, age, dan month, pada nasabah yang telah melakukan deposito jangka panjang

```
[ ]  train_df_yes = train_df.loc[train_df['y']=='yes',:]

     train_df_yes[['duration', 'age']].describe()
```

	duration	age
count	3712.000000	3712.000000
mean	549.398976	40.851293
std	397.490060	13.760020
min	63.000000	17.000000
25%	252.000000	31.000000
50%	448.000000	37.000000
75%	737.000000	50.000000
max	4199.000000	98.000000

```
[ ]  train_df_yes[['month']].describe()
```

	month
count	3712
unique	10
top	may
freq	699

Tiga feature yang paling berpengaruh pada akurasi model adalah duration, month, dan age. Dimana, rata - rata durasi telepon yang dilakukan pada customer yang mau untuk melakukan deposit jangka panjang adalah selama 549 detik, umur customer rata - rata 40, dan telepon dilakukan paling banyak pada bulan mei

ANALISIS MODEL

Training model final pada data training

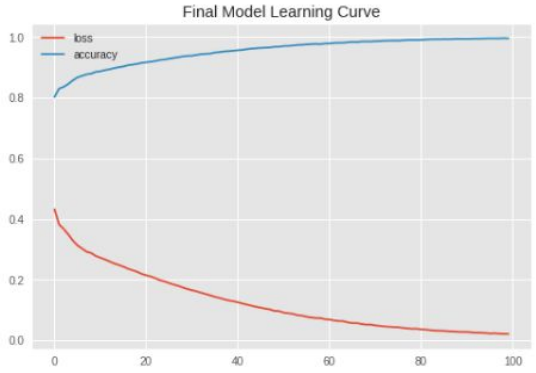
Simpan dan Training model final pada keseluruhan data training

```
[ ]  # 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()
```




```
final_model.summary()
```

```
Model: "sequential_100"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_279 (Dense)           (None, 130)               1820

 dense_280 (Dense)           (None, 100)               13100

 dense_281 (Dense)           (None, 1)                 101

=================================================================
Total params: 15,021
Trainable params: 15,021
Non-trainable params: 0
_________________________________________________________________
```

ANALISIS MODEL

Rangkuman model final

INPUT LAYER

Hyperparameter	Nilai
Jumlah neuron	13

OUTPUT LAYER

Hyperparameter	Nilai
Jumlah neuron	1
Fungsi aktivasi	sigmoid

HIDDEN LAYER

Hyperparameter	Nilai
Jumlah hidden layer	2
Jumlah neuron pada hidden layer pertama	130
Jumlah neuron pada hidden layer kedua	100
Fungsi aktivasi	tanh

ANALISIS MODEL

Prediksi data test menggunakan model final

Prediksi test data

```
[ ] test_prepared = std_scale.transform(test_df)
    test_df_prepared = pd.DataFrame(test_prepared, columns=X_sm.columns)
    test_df_prepared
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	poutcome
0	-0.783709	0.076973	-1.928167	1.053452	-0.516547	-1.08747	-0.453839	-0.758915	-0.531722	0.713774	-0.587633	1.764832	0.193670
1	-0.289561	1.746477	3.001720	1.053452	-0.516547	-1.08747	-0.453839	-0.758915	-0.100624	0.713774	-0.762920	-0.822023	0.193670
2	1.489375	0.355224	-1.928167	0.585494	1.933816	0.94245	-0.453839	-0.758915	-0.531722	-0.000390	-0.587633	-0.175310	0.193670
3	0.402247	-0.479529	-0.284871	-1.754295	1.933816	-1.08747	-0.453839	1.317671	-0.100624	0.713774	-1.056952	-0.175310	0.193670
4	-1.179028	-1.036030	1.358424	-0.350421	-0.516547	-1.08747	-0.453839	-0.758915	0.330474	-1.428718	-0.514125	-0.175310	0.193670
...	...	...	...	...	...	...	...	...	...	...	...	...	...
8233	0.797566	0.076973	-0.284871	-0.818379	-0.516547	0.94245	-0.453839	-0.758915	0.761572	0.713774	1.804200	-0.822023	0.193670
8234	-0.981369	0.911725	1.358424	-0.350421	-0.516547	0.94245	-0.453839	-0.758915	0.761572	-1.428718	-0.429308	-0.822023	0.193670
8235	-0.684880	0.911725	-0.284871	-0.350421	-0.516547	-1.08747	-0.453839	-0.758915	-0.100624	-0.714554	1.340535	-0.822023	-2.552217
8236	0.402247	-0.757779	-0.284871	-1.286337	-0.516547	0.94245	2.304690	1.317671	0.761572	-0.714554	1.804200	1.764832	0.193670
8237	0.204588	-0.757779	-0.284871	-0.818379	1.933816	0.94245	-0.453839	-0.758915	0.761572	0.713774	-0.859046	1.764832	0.193670

8238 rows × 13 columns

```
[ ] y_pred_proba = final_model.predict(test_df_prepared)
    y_pred_proba
```

```
array([[5.9601532e-13],
       [3.5414100e-04],
       [3.7234050e-14],
       ...,
       [9.9776506e-01],
       [2.3004431e-01],
       [2.0830016e-16]], dtype=float32)
```

```
[ ] 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
```

Sebanyak 7360 nasabah pada data test akan menolak untuk melakukan deposito jangka panjang dan sebanyak 878 nasabah akan melakukan deposito jangka panjang

KESIMPULAN

KESIMPULAN

Model ANN yang telah dibuat mencapai akurasi dan f1 score 93% pada validation data

Model ANN yang telah dibuat memprediksi terdapat 878 nasabah dari data test yang akan melakukan deposito jangka panjang

Tiga fitur paling berpengaruh dalam memprediksi nasabah yang akan melakukan deposito jangka panjang adalah duration, month, dan age

REFERENSI

A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, Canada: O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2019.

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

TERIMA KASIH !