Session 39

Model Deployment I

Table of Content

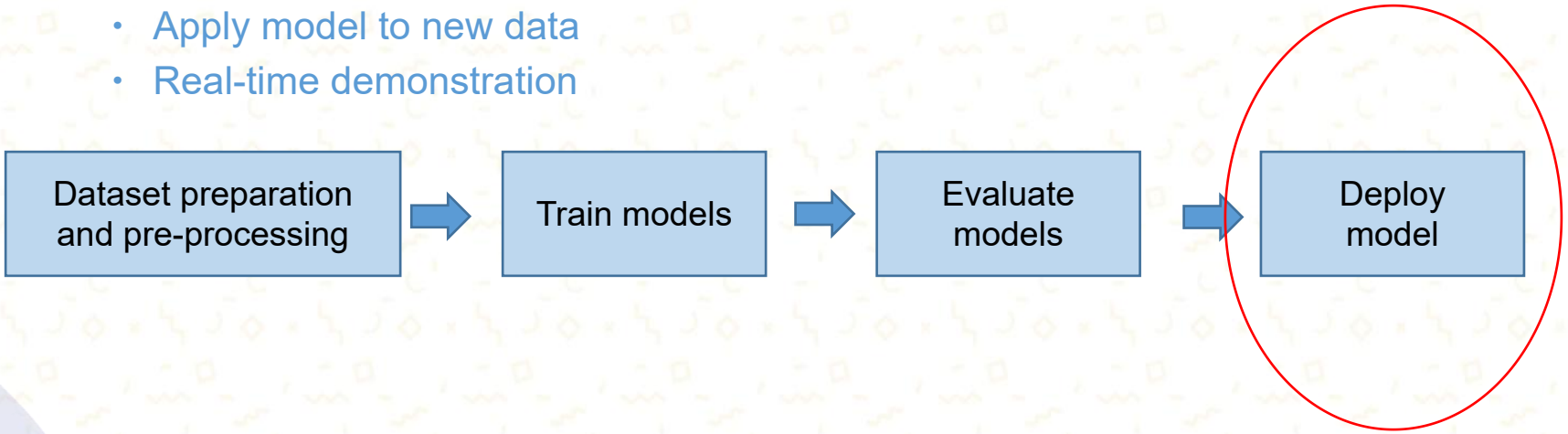
What will We Learn Today?

1. How to make prediction from new data
2. Save and load trained model
3. Introduction to Flask
4. Introduction to HTML
5. Integrating ML model into web application

Stage in Machine Learning

- Data pre-processing
  - Data cleaning, filling missing value, remove outlier
- Train models
  - Select the algorithm
  - Feature selection and extraction
- Evaluate model
  - Assess performance
  - Model comparison
- Deploy model
  - Apply model to new data
  - Real-time demonstration

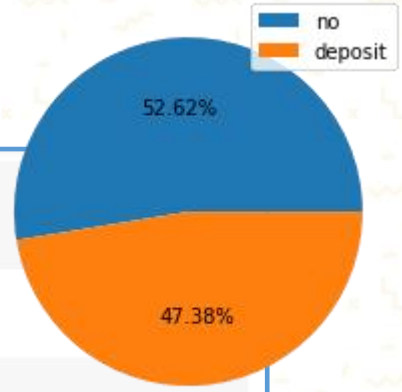
We are here now !



How to make prediction from new data

Read the dataset

- https://www.kaggle.com/janiobachmann/bank-marketing-dataset
- Untuk memprediksi apakah customer akan membeli deposito jangka panjang atau tidak




```
[1] from google.colab import drive
    drive.mount('/content/drive')
```

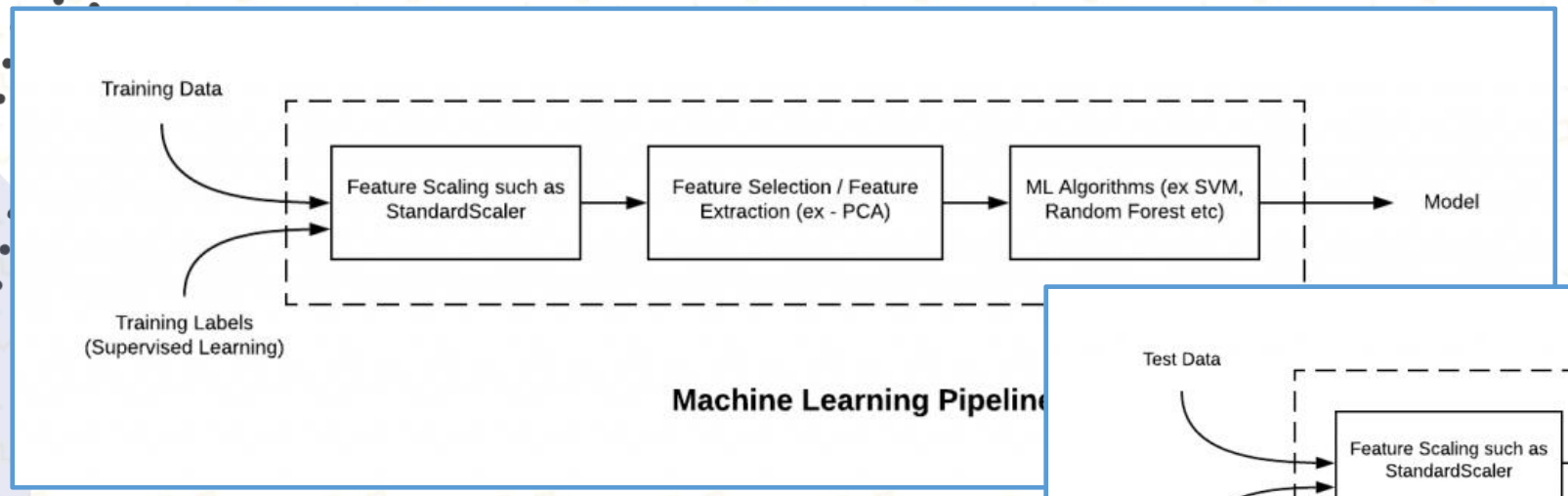
```
Mounted at /content/drive
```

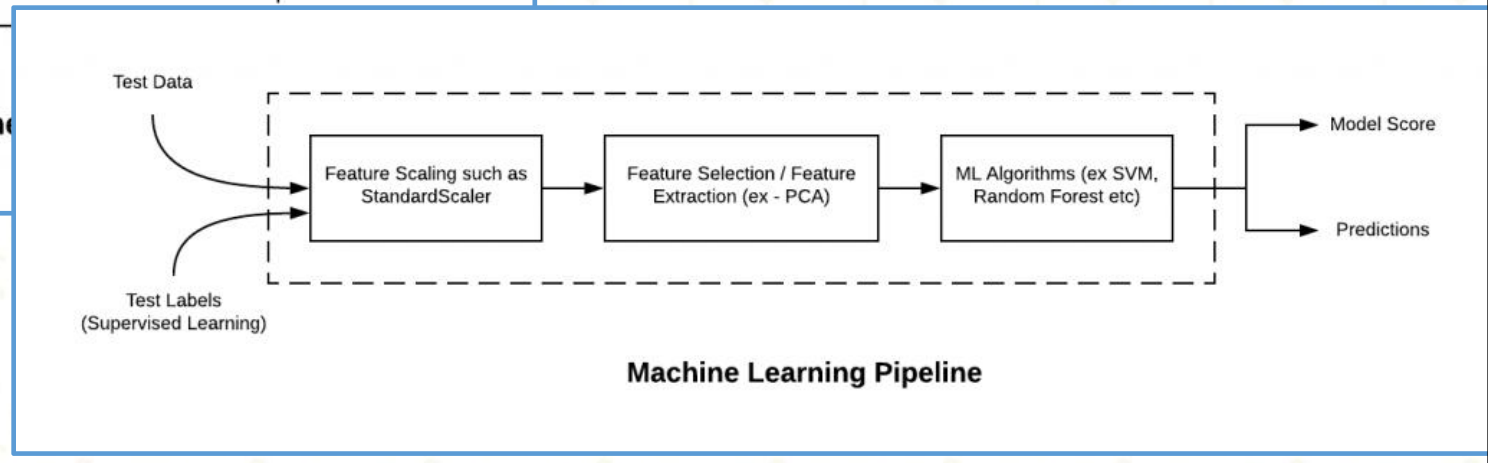
```
[2] import pandas as pd
    df = pd.read_csv('https://raw.githubusercontent.com/ganjar87/data_science_practice/main/bank.csv', delimiter=',')
    df.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	1042	1	-1	0	unknown	yes
1	56	admin.	married	secondary	no	45	no	no	unknown	5	may	1467	1	-1	0	unknown	yes
2	41	technician	married	secondary	no	1270	yes	no	unknown	5	may	1389	1	-1	0	unknown	yes
3	55	services	married	secondary	no	2476	yes	no	unknown	5	may	579	1	-1	0	unknown	yes
4	54	admin.	married	tertiary	no	184	no	no	unknown	5	may	673	2	-1	0	unknown	yes

ML Pipeline

- *Machine Learning (ML) pipeline* = cara meng-otomatisasi alur kerja yang diperlukan untuk menghasilkan model *machine learning.*
- Sebelumnya kita men-transformasi data untuk training dan testing set secara terpisah.
- Dengan menggunakan *Sklearn.pipeline* kita dapat meng-otomatiskan langkah-langkah ini.





https://vitalflux.com/sklearn-machine-learning-pipeline-python-example/

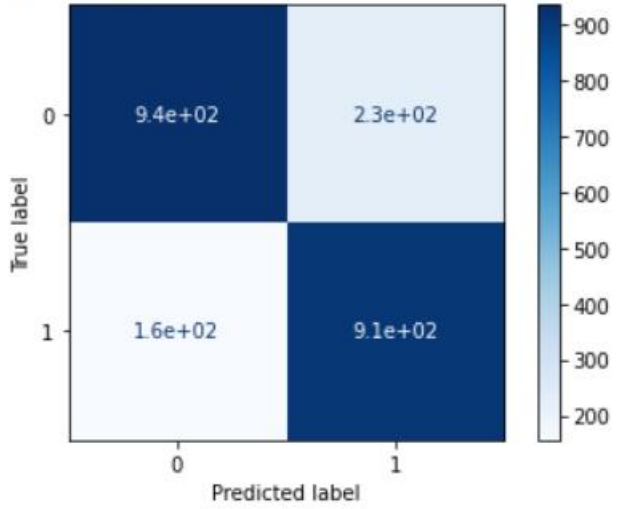
How to use Pipeline

```
X_train, X_test, y_train, y_test = train_test_split(
    df_X, df_y, test_size=0.2, random_state=42)

numerical_transformer = SimpleImputer(strategy='median')
categorical_transformer = Pipeline(steps=[
('imputer', SimpleImputer(strategy='most_frequent')),
('imput', OrdinalEncoder())
])
preprocessor = ColumnTransformer(
transformers=[
('num', numerical_transformer, nums),
('cat', categorical_transformer, cats)
])
pipeline = Pipeline(steps=[('preprocessor', preprocessor), ('scaling', StandardScaler()),
                           #('feature_selection', SelectFromModel(ExtraTreesClassifier(random_state=2), prefit=False)),
                           #('balance', SMOTE()),
                           ('classifier', RandomForestClassifier(random_state=42, max_depth=10))])

pipeline = pipeline.fit(X_train, y_train)
y_pred = pipeline.predict(X_test)
print('')
print('-------------------------')
print('Accuracy ',metrics.accuracy_score(y_test, y_pred))
print('Precision ',metrics.precision_score(y_test, y_pred, average='macro'))
print('Recall ',metrics.recall_score(y_test, y_pred, average='macro'))
print('Confusion matrix ', metrics.confusion_matrix(y_test, y_pred))
plot_confusion_matrix(pipeline, X_test, y_test, cmap=plt.cm.Blues)
plt.show()
```

```
-------------------------
Accuracy  0.8266905508284819
Precision  0.827367604071523
Recall  0.8278412406500327
Confusion matrix  [[935 231]
 [156 911]]
```



How to make prediction from new data

```
data = [{'age':68, 'job':'admin.', 'marital':'married', 'education':'secondary',
         'default':'no', 'balance':2000, 'housing':'yes',
         'loan':'no', 'contact':'unknown', 'day':5, 'month':'may', 'duration':1000,
         'campaign':1, 'pdays':-1,'previous':0, 'poutcome':'unknown'},
        {'age':18, 'job':'unknown', 'marital':'single', 'education':'secondary',
         'default':'no', 'balance':200, 'housing':'no',
         'loan':'no', 'contact':'unknown', 'day':5, 'month':'may', 'duration':10,
         'campaign':0, 'pdays':-1,'previous':0, 'poutcome':'unknown'}]

df_input = pd.DataFrame(data)
result = pipeline.predict(df_input)

for i in result:
  int_result = int(i)
  if (int_result == 0):
    decision = 'No'
  elif (int_result==1):
    decision = 'Yes'
  else:
    decison = 'Not defined'
  print('Possible to deposit is ', decision)
```

```
Possible to deposit is  Yes
Possible to deposit is  No
```

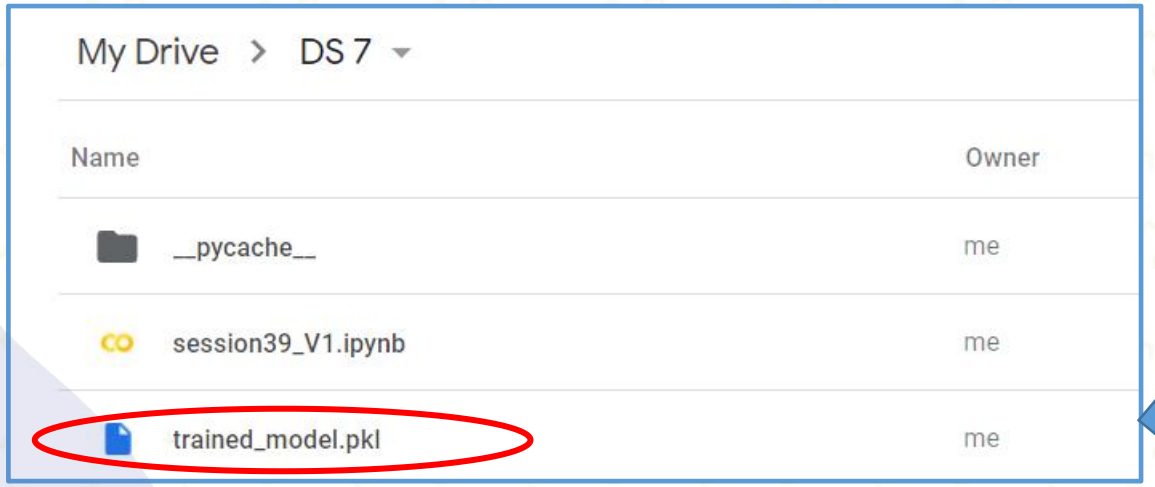
Save and load trained model

Save trained model into file

```
pipeline = Pipeline(steps=[('preprocessor', preprocessor), ('scaling', StandardScaler()),
                           #('feature_selection', SelectFromModel(ExtraTreesClassifier(random_state=2), prefit=False)),
                           #('balance', SMOTE()),
                           ('classifier', RandomForestClassifier(random_state=42, max_depth=10))])

pipeline = pipeline.fit(X_train, y_train)
filename = '/content/drive/MyDrive/DS 7/trained_model.pkl'
joblib.dump(pipeline, filename)
```

```
['/content/drive/MyDrive/DS 7/trained_model.pkl']
```



Load trained model (file) and make prediction

```python
import joblib
import pandas as pd

filename = '/content/drive/MyDrive/DS 7/trained_model.pkl'
loaded_model = joblib.load(filename)

data = {'age':68, 'job':'admin.', 'marital':'married', 'education':'secondary',
        'default':'no', 'balance':2000, 'housing':'yes','loan':'no', 'contact':'unknown',
        'day':5, 'month':'may', 'duration':1000, 'campaign':1, 'pdays':-1,
        'previous':0, 'poutcome':'unknown'}
df_input = pd.DataFrame(data, index=[0])
result = loaded_model.predict(df_input)

for i in result:
  int_result = int(i)
  if (int_result == 0):
    decision = 'No'
  elif (int_result==1):
    decision = 'Yes'
  else:
    decison = 'Not defined'

print('Possibility to deposit is ', decision)
```

```
Possibility to deposit is  Yes
```

DigitalSkola
Uncover The World Of Digital Skills With Us

Introduction to Flask

What is Flask

- Flask adalah sebuah *web framework*.
- Artinya Flask menyediakan *tools, libraries* dan *technologies* yang memungkinkan kita membangun aplikasi web.
- Ada banyak framework di Python, termasuk Flask, Tornado, Pyramid, dan Django.

Web development using Flask

- Note : mulai di step ini, kita membuat file python di computer local, bukan di google colab.
- Bisa menggunakan editor untuk python, contohnya : spyder (anaconda package)

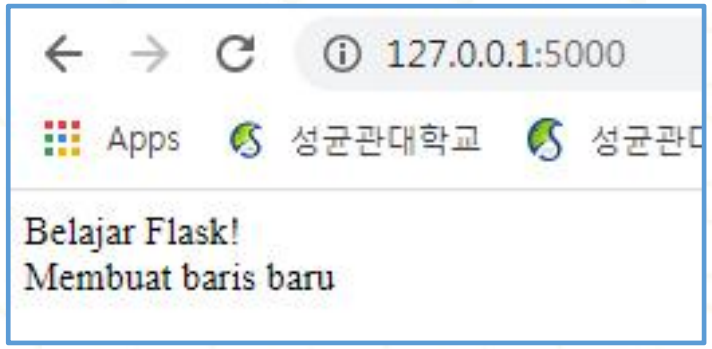
```python
from flask import Flask

app = Flask(__name__)
@app.route('/')
def index():
    return 'Belajar Flask! <br> Membuat baris baru'

if __name__ == "__main__":
    app.run()
```

HTML code

Hasil eksekusi program




```
Console 1/A
create_model )

In [8]: runfile('C:/DATA GANJAR/TEACHING/Digital Skola/batch_7/session 39/session_39_shared/create_model/
hello_world.py', wdir='C:/DATA GANJAR/TEACHING/Digital Skola/batch_7/session 39/session_39_shared/
create_model')
 * Serving Flask app "hello_world" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

DigitalSkola
Uncover The World Of Digital Skills With Us

Introduction to HTML

HTML

- *HTML (HyperText Markup Language)*
- Memberikan isi dan struktur dokumen,
- *HTML pages* menggunakan *tags*
  - *Tags* menunjukkan bagaimana program pemrosesan harus menampilkan teks dan grafik
  - Diproses oleh browser
  - Tags biasanya berpasangan
  - Mempunyai nama yang masuk akal

START TAG	***END TAG***
<HTML>	**</HTML>**
<HEAD>	**</HEAD>**
<TITLE>	**</TITLE>**
<BODY>	**</BODY>**
<H1>, <H2>, ...	**</H1>, </H2>, ...**
<IMG ...>	**</IMG> (optional)**
<A ...>	**</A>**
<P>	**</P> (optional)**
** **	**(none; "empty" tag)**
<UL>	**</UL>**
<LI>	**</LI> (optional)**

HTML Document Layout

- Contoh penggunakan dan posisi *tags*

```
<html>
<head>
    <title>
        Title bar text
    </title>
</head>
<body>
    <p>Look, I'm a paragraph!</p>
</body>
</html>
```

HTML : Table

- Tujuan utamanya adalah untuk menampilkan *tables*
- Table adalah koleksi dari baris (rows)
- Baris (rows) adalah koleksi dari *cells*
- *Cells* dapat diisi apa saja (bahkan *tables* lain)

```
<table border="1">
  <tr bgcolor="gray">
    <td>Row 1, Cell 1</td>
    <td>Row 1, Cell 2</td>
  </tr>
  <tr bgcolor="#FF3399">
    <td>Row 2, Cell 1</td>
    <td>Row 2, Cell 2</td>
  </tr>
</table>
```

HTML codes

Row 1, Cell 1	Row 1, Cell 2
Row 2, Cell 1	Row 2, Cell 2

Viewed from a browser

Example : HTML

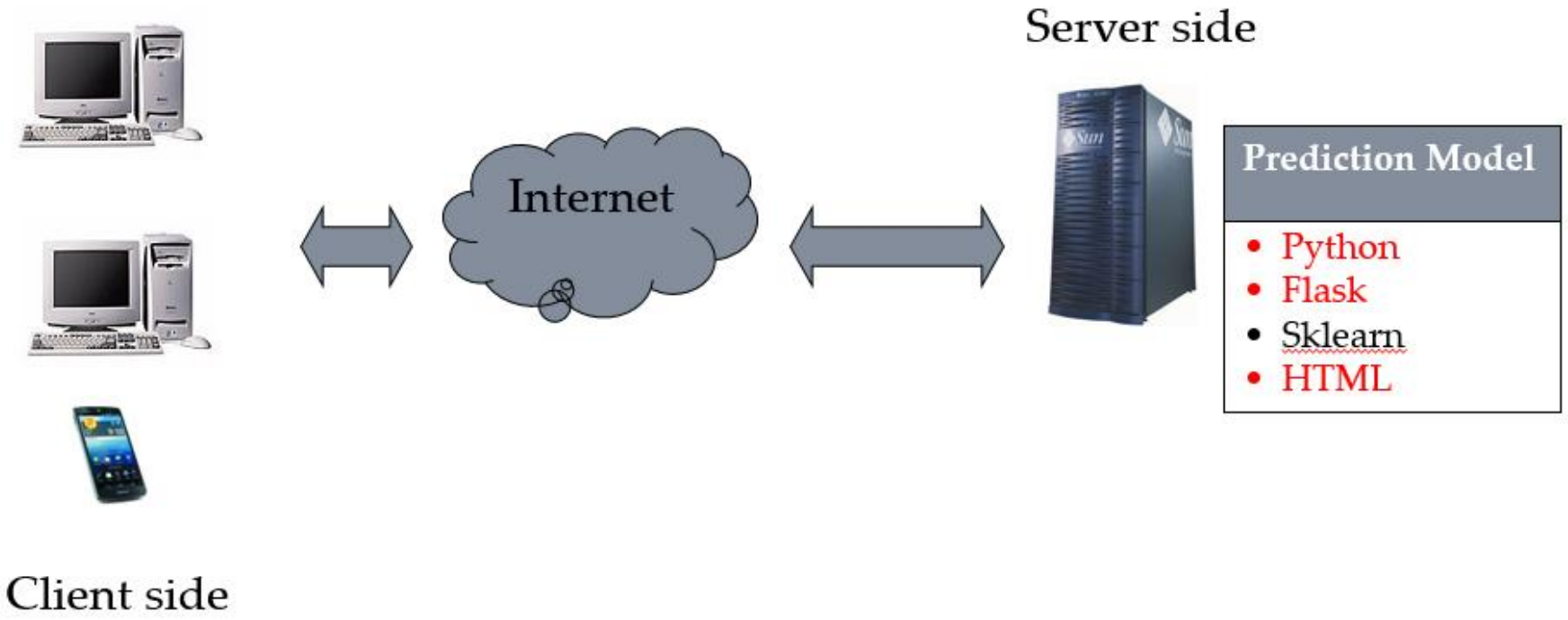
```html
<html>
    <head>
        <title>
            Belajar web programming
        </title>
    </head>
    <body>
        <h1>
            Belajar HTML web programming
        <p>
        Link digital skola
        <p>
        <a href="https://digitalskola.com/"> official website </a>

        <p>
        <table border="1">
          <tr bgcolor="gray">
            <td>Row 1, Cell 1</td>
            <td>Row 1, Cell 2</td>
          </tr>
          <tr bgcolor="#FF3399">
            <td>Row 2, Cell 1</td>
            <td>Row 2, Cell 2</td>
          </tr>
        </table>

    </body>
</html>
```

Integrating ML model into web app

Proposed system



Save trained model into file

```
X_train, X_test, y_train, y_test = train_test_split(
    df_X, df_y, test_size=0.2, random_state=42)

numerical_transformer = SimpleImputer(strategy='median')
categorical_transformer = Pipeline(steps=[('imputer', SimpleImputer(strategy='most_frequent')),
                                          ('imput', OrdinalEncoder())])

preprocessor = ColumnTransformer(
transformers=[
('num', numerical_transformer, nums),
('cat', categorical_transformer, cats)
])

pipeline = Pipeline(steps=[('preprocessor', preprocessor), ('scaling', StandardScaler()),
                           #('feature_selection', SelectFromModel(ExtraTreesClassifier(random_stat
                           #('balance', SMOTE()),
                           ('classifier', RandomForestClassifier(random_state=42, max_depth=10))])


pipeline = pipeline.fit(X_train, y_train)
filename = 'trained_model.pkl'
joblib.dump(pipeline, filename)
```

이름	수정한 날짜	유형	크기
bank.csv	21/09/2019 19:16	Microsoft Excel C...	898KB
create_model.py	24/08/2021 15:07	PY 파일	2KB
hello_world.py	25/08/2021 13:30	PY 파일	1KB
index.html	25/08/2021 13:52	HTML 문서	1KB
trained_model.pkl	25/08/2021 13:56	PKL 파일	48,744KB

Web app (backend)

app.py

```python
import pandas as pd
import joblib
from flask import Flask, redirect, url_for, request, render_template

app = Flask(__name__)
#load index.html/ first page. receive input variable from user
@app.route("/")
def index():
    return render_template('index.html')

#load result.html. the result of prediction is presented here.
@app.route('/result/', methods=["POST"])
def prediction_result():
    #receiving parameters sent by client
    age = int(request.form.get('age'))
    job = request.form.get('job')
    marital = request.form.get('marital')
    education = request.form.get('education')
    default = request.form.get('default')
    balance = int(request.form.get('balance'))
    housing = request.form.get('housing')
    loan = request.form.get('loan')
    contact = request.form.get('contact')
    day = int(request.form.get('day'))
    month = request.form.get('month')
    duration = int(request.form.get('duration'))
    campaign = int(request.form.get('campaign'))
    pdays = int(request.form.get('pdays'))
    previous = int(request.form.get('previous'))
    poutcome = request.form.get('poutcome')
    #load the trained model.
    filename = 'trained_model.pkl'
    loaded_model= joblib.load(filename)
    #create new dataframe
    data = {'age':age, 'job':job, 'marital':marital, 'education':education, 'default':default, 'balance':balance, 'housing':housing,
            'loan':loan, 'contact':contact, 'day':day, 'month':month, 'duration':duration, 'campaign':campaign, 'pdays':pdays,
            'previous':previous, 'poutcome':poutcome}
    pd.set_option('display.max_columns', None)
    pd.set_option('display.max_rows', None)
    df_input = pd.DataFrame(data, index=[0])
    #print(df_input.dtypes)
    result = loaded_model.predict(df_input)
    #print(result)
    for i in result:
      int_result = int(i)
      if (int_result == 0):
        decision = 'No'
      elif (int_result==1):
        decision = 'Yes'
      else:
        decision = 'Not defined'
    #return the output and load result.html
    return render_template('result.html', status=decision)

if __name__ == "__main__":
    #host= ip address, port = port number
    #app.run(host='127.0.0.1', port='5001')
    app.run()
```

index.html

trained_model.pkl

result.html

FOLDERS

- deposit
  - static
  - templates
    - index.html
    - result.html
  - app.py
  - Procfile
  - requirements.txt
  - runtime.txt
  - trained_model.pkl

HTML interface (frontend)

index.html

```html
<head>
   <title>Bank Deposit Prediction</title>
</head>

<body>
      <center>
      <h2> Bank Long-term Deposit Prediction </h2><p><p>
      <form method = "post" action = "/result/">
                <table border=1 width=50%>
                   <tr>
                      <td>Age</td>
                      <td>
                      <input type = "text" name="age" value="69">
                      </td>
                   </tr>
             <tr>
                      <td>Job</td>
                      <td>
                      <select name="job">
                     <option value="admin.">admin.</option>
                     <option value="technician">technician</option>
                     <option value="services">services</option>
                     <option value="management">management</option>
                     <option value="retired">retired</option>
                     <option value="blue-collar">blue-collar</option>
                     <option value="unemployed">unemployed</option>
                     <option value="enterpreneur">entrepreneur</option>
                     <option value="housemaid">housemaid</option>
                     <option value="self-employed">self-employed</option>
                     <option value="student">student</option>
                     <option value="unknown">unknown</option>
                  </select>
                      </td>
                   </tr>
             <tr>
                      <td>Marital</td>
                      <td>
                      <select name="marital">
                     <option value="married">married</option>
                     <option value="single">single</option>
                     <option value="divorced">divorced</option>
                  </select>
                      </td>
                   </tr>
```

FOLDERS

- deposit
  - static
  - templates
    - index.html
    - result.html
  - app.py
  - Procfile
  - requirements.txt
  - runtime.txt
  - trained_model.pkl

hasil eksekusi

Bank Long-term Deposit Prediction

Age	69
Job	admin.
Marital	married
Education	primary
Default	no
Balance	2000
Housing	no
Loan	no
Contact	telephone
Day	5
Month	jan
Duration	1000
Campaign	1
Pdays	-1
Previous	0
Poutcome	failure
	Reset Submit

HTML interface (frontend)

result.html

```html
<!DOCTYPE html>
<html lang="en">
<html>
<head>
    <title>Bank Long-term Deposit Prediction</title>
</head>
<body>
        <center>
        <h2> Prediction Result </h2><p><p>
                <table border=1 width=20%>

                    <tr>
                        <td>Possibility to subscribe the long-term deposit is </td>
                        <td>{{status}}</td>
                    </tr>

                </table>

<p><p>


</body>
</html>
```

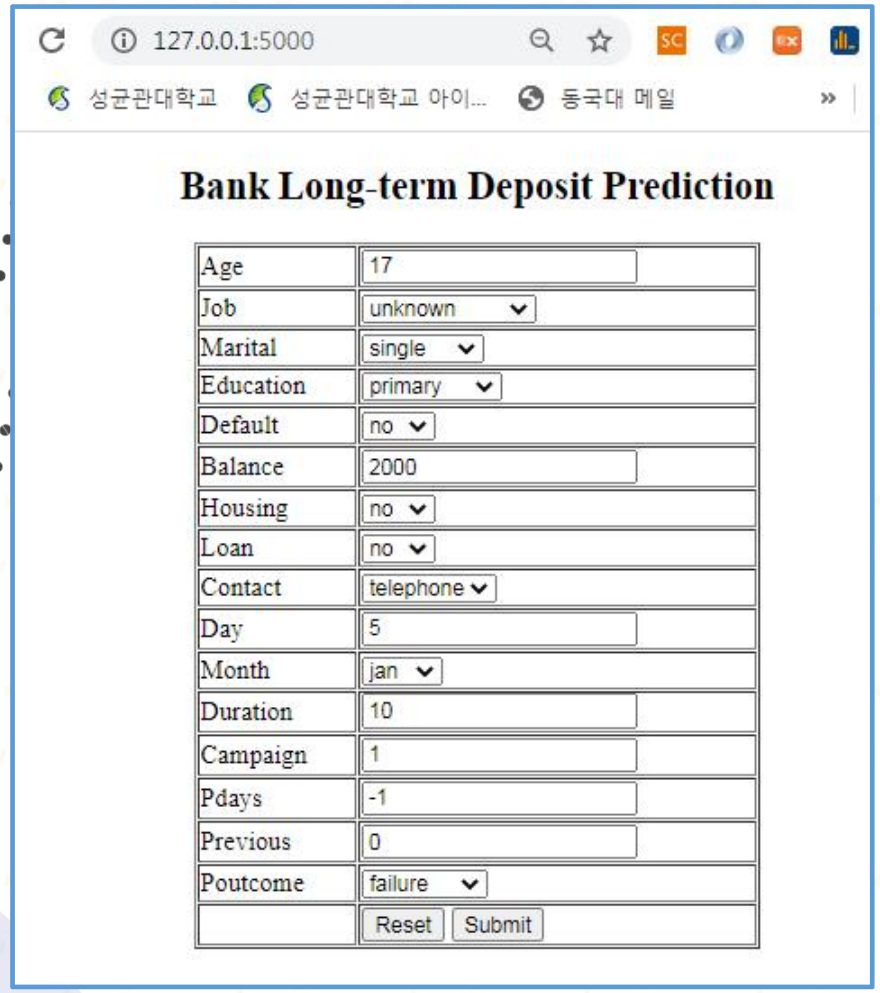
FOLDERS
- deposit
  - static
  - templates
    - index.html
    - result.html
  - app.py
  - Procfile
  - requirements.txt
  - runtime.txt
  - trained_model.pkl

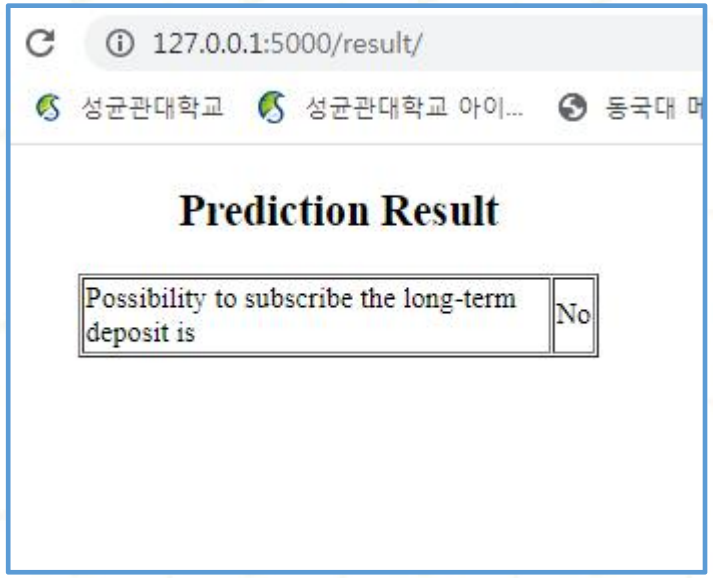
hasil eksekusi

Prediction Result

Possibility to subscribe the long-term deposit is	Yes

Result : running on local computer





Let’s practice

DigitalSkola

Uncover The World Of Digital Skills With Us

Thank YOU