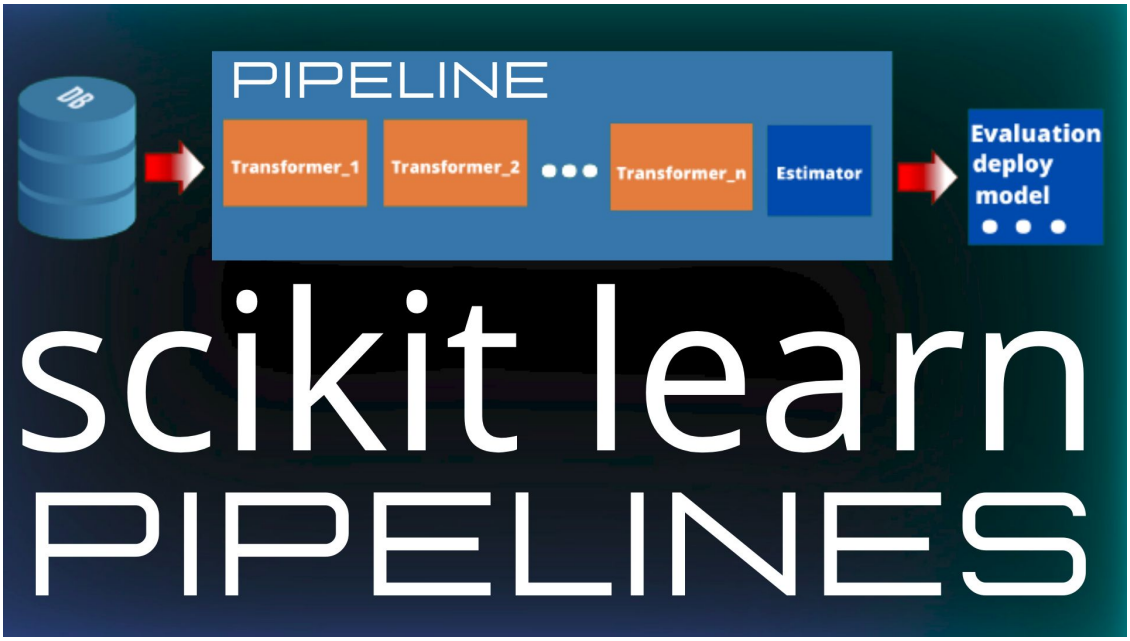
DATA PRE-PROCESSING

December 12, 2022



CODE SOURCE + MEDIAS SOCIAUX

Playlist dediée au DATA PRE-PROCESSING

Vidéo associée à ce notebook

Mon profile linkedin

Groupe facebook

Instagram

Page d'entreprise

Github

Merci d'ajouter une étoile à mon profile github si vous pensez que le travail que je fais est utile.

[24]:
```
import pandas as pd
import seaborn as sb
from sklearn.compose import make_column_transformer
from sklearn.impute import SimpleImputer
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import MinMaxScaler,OneHotEncoder
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import seaborn as sb
```

[25]:
```
titanic_dataset = sb.load_dataset('titanic')
```

[26]:
```
titanic_dataset.head()
```

[26]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	\
0	0	3	male	22.0	1	0	7.2500	S	Third	
1	1	1	female	38.0	1	0	71.2833	C	First	
2	1	3	female	26.0	0	0	7.9250	S	Third	
3	1	1	female	35.0	1	0	53.1000	S	First	
4	0	3	male	35.0	0	0	8.0500	S	Third	

	who	adult_male	deck	embark_town	alive	alone
0	man	True	NaN	Southampton	no	False
1	woman	False	C	Cherbourg	yes	False
2	woman	False	NaN	Southampton	yes	True
3	woman	False	C	Southampton	yes	False
4	man	True	NaN	Southampton	no	True

[27]:
```
titanic_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

[28]:
```
y = titanic_dataset.survived
X = titanic_dataset.drop(['survived',
                          'embarked',
                          'class',
                          'who',
                          'adult_male',
                          'alive'], axis = 'columns')
```

[29]:
```
X.head()
```

[29]:

	pclass	sex	age	sibsp	parch	fare	deck	embark_town	alone
0	3	male	22.0	1	0	7.2500	NaN	Southampton	False
1	1	female	38.0	1	0	71.2833	C	Cherbourg	False
2	3	female	26.0	0	0	7.9250	NaN	Southampton	True
3	1	female	35.0	1	0	53.1000	C	Southampton	False
4	3	male	35.0	0	0	8.0500	NaN	Southampton	True

2. DIVISION DU DATASET EN DONNEES D'ENTRAINEMENT ET DONNEES DU TEST - (Xtrain, ytrain) = donnees d'apprentissage - (Xtest, ytest) = donnees du test

[30]:
```
Xtrain, Xtest, ytrain, ytest = train_test_split(X,
                                                y,
                                                test_size = .2,
                                                random_state = 0)
```

3. SEPARATION DES COLONNES EN DEUX GROUPES - NUMERIQUES - QUALITATIVES

[31]:
```
num_cols = Xtrain.select_dtypes(include = ['int','float']).columns
cat_cols = Xtrain.select_dtypes(exclude = 'number').columns
```

[32]:
```
num_cols
```

[32]:
```
Index(['pclass', 'age', 'sibsp', 'parch', 'fare'], dtype='object')
```

[33]:
```
cat_cols
```

[33]:
```
Index(['sex', 'deck', 'embark_town', 'alone'], dtype='object')
```

3. CREATION DES PIPELINES INTERMEDIAIRES - Pipeline pour les colonnes NUMERIQUES -Un autre Pipeline pour les colonnes QUALITATIVES

[34]:
```
num_pipeline = make_pipeline(
                             SimpleImputer(strategy = 'median'),
                             MinMaxScaler(),
                            )
```

```
cat_pipeline = make_pipeline(
                    SimpleImputer(strategy = 'most_frequent'),
                    OneHotEncoder()
                   )
```

[35]:
```
num_pipeline
```

[35]:
```
Pipeline(steps=[('simpleimputer', SimpleImputer(strategy='median')),
                ('minmaxscaler', MinMaxScaler())])
```

[36]:
```
cat_pipeline
```

[36]:
```
Pipeline(steps=[('simpleimputer', SimpleImputer(strategy='most_frequent')),
                ('onehotencoder', OneHotEncoder())])
```

3. TRANSFORMATEUR FINAL

[37]:
```
mct = make_column_transformer(
                              (num_pipeline, num_cols),
                              (cat_pipeline, cat_cols)
                             )
```

[38]:
```
mct
```

[38]:
```
ColumnTransformer(transformers=[('pipeline-1',
                                 Pipeline(steps=[('simpleimputer',
SimpleImputer(strategy='median')),
                                                 ('minmaxscaler',
                                                  MinMaxScaler())]),
                                 Index(['pclass', 'age', 'sibsp', 'parch',
'fare'], dtype='object')),
                                ('pipeline-2',
                                 Pipeline(steps=[('simpleimputer',
SimpleImputer(strategy='most_frequent')),
                                                 ('onehotencoder',
                                                  OneHotEncoder())]),
                                 Index(['sex', 'deck', 'embark_town', 'alone'],
dtype='object'))])
```

4. CREATION DE PIPELINE FINAL

[39]:
```
full_pipeline = make_pipeline(mct, LogisticRegression())
```

[40]:
```
full_pipeline
```

[40]:
```
Pipeline(steps=[('columntransformer',
                 ColumnTransformer(transformers=[('pipeline-1',
Pipeline(steps=[('simpleimputer',
```

```
SimpleImputer(strategy='median')),
('minmaxscaler',
MinMaxScaler())]),
                                                 Index(['pclass', 'age',
'sibsp', 'parch', 'fare'], dtype='object')),
                                                ('pipeline-2',
Pipeline(steps=[('simpleimputer',
SimpleImputer(strategy='most_frequent')),
('onehotencoder',
OneHotEncoder())]),
                                                 Index(['sex', 'deck',
'embark_town', 'alone'], dtype='object'))])),
                ('logisticregression', LogisticRegression())])
```

[41]:
```
full_pipeline.fit(Xtrain, ytrain)
```

[41]:
```
Pipeline(steps=[('columntransformer',
                 ColumnTransformer(transformers=[('pipeline-1',
Pipeline(steps=[('simpleimputer',
SimpleImputer(strategy='median')),
('minmaxscaler',
MinMaxScaler())]),
                                                 Index(['pclass', 'age',
'sibsp', 'parch', 'fare'], dtype='object')),
                                                ('pipeline-2',
Pipeline(steps=[('simpleimputer',
SimpleImputer(strategy='most_frequent')),
('onehotencoder',
OneHotEncoder())]),
                                                 Index(['sex', 'deck',
'embark_town', 'alone'], dtype='object'))])),
                ('logisticregression', LogisticRegression())])
```

[42]:
```
full_pipeline.score(Xtest,ytest)
```

[42]:
```
0.8044692737430168
```

[43]:
```
import numpy as np
np.round(full_pipeline.score(Xtest,ytest), 2)
```

[43]:
```
0.8
```

5. SAUVEGARDER LE MODELE

[44]:
```
from joblib import dump, load

dump(full_pipeline, 'modele.joblib')
```

[44]:
```
['modele.joblib']
```

[45]:
```
modele = load('modele.joblib')
modele.predict(Xtest)
```

[45]:
```
array([0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0,
       1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0,
       1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0])
```

[46]:
```
np.round(modele.score(Xtest,ytest), 2)
```

[46]:
```
0.8
```