Feature Engineering 101

Topic - 6

Pipeline

PipiLine in Machine Learning

Without using PipeLine

In [1]:
```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.tree import DecisionTreeClassifier
```

In [2]:
```
df = pd.read_csv('train.csv')
```

In [3]:
```
df.sample(5)
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
559	560	1	3	de Messemaeker, Mrs. Guillaume Joseph (Emma)	female	36.0	1	0	345572	17.40	NaN	S
419	420	0	3	Van Impe, Miss. Catharina	female	10.0	0	2	345773	24.15	NaN	S
153	154	0	3	van Billiard, Mr. Austin Blyler	male	40.5	0	2	A/5. 851	14.50	NaN	S
741	742	0	1	Cavendish, Mr. Tyrell William	male	36.0	1	0	19877	78.85	C46	S

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
167	168	0	3	Skoog, Mrs. William (Anna Bernhardina Karlsson)	female	45.0	1	4	347088	27.90	NaN	S

In [4]:
```
#selcet the important column
df.drop(columns=['PassengerId','Name','Ticket','Cabin'],inplace=True)
```

In [5]:
```
df.sample(5)
```

Out[5]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
597	0	3	male	49.0	0	0	0.0000	S
533	1	3	female	NaN	0	2	22.3583	C
277	0	2	male	NaN	0	0	0.0000	S
152	0	3	male	55.5	0	0	8.0500	S
82	1	3	female	NaN	0	0	7.7875	Q

Step 1 - Train-Test-Split

In [6]:
```
X_train,X_test,y_train,y_test = train_test_split(df.drop(columns=['Survived']),
                                                 df['Survived'],
                                                 test_size=0.2,
                                                random_state=42)
```

In [7]:
```
print(X_train.head(2))
```

```
     Pclass   Sex   Age  SibSp  Parch  Fare Embarked
331       1  male  45.5      0      0  28.5        S
733       2  male  23.0      0      0  13.0        S
```

In [8]:
```
print(y_train.head())
```

```
331    0
733    0
382    0
704    0
813    0
Name: Survived, dtype: int64
```

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
Survived      0
Pclass        0
Sex           0
Age         177
SibSp         0
Parch         0
Fare          0
Embarked      2
dtype: int64
```

In [10]:
```
# Applying imputation

si_age = SimpleImputer()
si_embarked = SimpleImputer(strategy='most_frequent')

X_train_age = si_age.fit_transform(X_train[['Age']])
X_train_embarked = si_embarked.fit_transform(X_train[['Embarked']])

X_test_age = si_age.transform(X_test[['Age']])
X_test_embarked = si_embarked.transform(X_test[['Embarked']])
```

In [11]:
```
print(X_train_embarked)
```

```
[['S']
 ['S']
 ['S']
 ['S']
 ['S']
 ['C']
 ['S']
 ['C']
 ['S']
 ['S']
 ['S']
 ['S']
 ['S']
 ['S']
 ['S']
 ['S']
 ['S']
 ['C']
 ['S']
 ['S']
 ['S']
 ['S']
 ['S']
 ['S']
 ['S']
 ['C']
 ['C']
 ['S']
 ['S']
 ['C']
 ['S']
 ['C']
 ['S']
 ['Q']
 ['Q']
 ['S']
 ['S']
 ['S']
 ['C']
 ['S']
 ['S']
 ['S']
 ['S']
 ['S']
 ['S']
 ['S']
 ['S']
 ['S']
 ['S']
 ['S']
```

['S']
['S']
['C']
['Q']
['C']
['S']
['S']
['S']
['S']
['C']
['Q']
['S']
['S']
['C']
['S']
['C']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['Q']
['C']
['S']
['S']
['S']
['S']
['S']
['S']
['C']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['C']
['C']
['S']
['S']
['S']
['S']
['S']
['C']
['S']
['Q']
['S']
['S']
['S']
['S']
['Q']
['C']
['C']
['S']

['S']
['Q']
['C']
['C']
['S']
['S']
['C']
['S']
['S']
['S']
['C']
['S']
['S']
['S']
['S']
['S']
['S']
['C']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['Q']
['S']
['Q']
['C']
['C']
['S']
['S']
['S']
['S']
['S']
['S']
['C']
['S']
['S']
['S']
['S']
['S']
['Q']
['S']
['C']
['S']
['C']
['S']
['C']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['Q']
['S']
['S']

['S']
['Q']
['S']
['C']
['Q']
['S']
['S']
['S']
['S']
['C']
['C']
['Q']
['S']
['S']
['S']
['S']
['S']
['C']
['C']
['C']
['S']
['S']
['C']
['S']
['S']
['S']
['C']
['S']
['Q']
['C']
['C']
['C']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['Q']
['C']
['S']
['C']
['Q']
['C']
['S']
['S']
['S']
['S']
['C']
['S']
['S']
['C']
['S']
['Q']
['S']
['S']
['S']
['C']
['C']
['S']
['S']
['S']
['S']
['C']
['S']

['S']
['S']
['S']
['S']
['S']
['S']
['S']
['C']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['Q']
['Q']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['C']
['C']
['Q']
['S']
['S']
['S']
['S']
['C']
['S']
['C']
['Q']
['Q']
['C']
['Q']
['S']
['C']
['S']
['C']
['S']
['C']
['S']
['Q']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['C']
['S']
['S']
['C']
['S']
['C']
['C']
['Q']
['S']
['S']
['S']
['S']
['S']
['C']
['Q']
['S']

['S']
['C']
['S']
['S']
['S']
['C']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['C']
['C']
['S']
['S']
['Q']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['C']
['S']
['S']
['S']
['S']
['C']
['S']
['S']
['Q']
['S']
['S']
['Q']
['S']
['S']
['S']
['S']
['S']
['S']
['Q']
['C']
['S']
['S']
['S']
['S']
['C']
['S']
['Q']
['C']
['S']
['S']
['S']
['S']
['Q']
['C']
['S']
['S']
['S']

['S']
['Q']
['Q']
['S']
['C']
['S']
['Q']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['C']
['S']
['S']
['S']
['Q']
['S']
['S']
['S']
['S']
['C']
['C']
['S']
['S']
['C']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['Q']
['S']
['C']
['S']
['S']
['S']
['Q']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['C']
['C']
['S']
['S']
['S']
['S']
['C']
['C']
['S']
['Q']
['S']
['S']
['S']
['S']
['S']

['S']
['S']
['S']
['S']
['S']
['C']
['C']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['C']
['Q']
['S']
['S']
['C']
['Q']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['C']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['C']
['S']
['S']
['S']
['S']
['S']
['S']
['Q']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['C']
['S']
['S']
['S']
['S']
['S']
['C']
['S']
['S']
['S']
['S']

['S']
['S']
['S']
['C']
['Q']
['C']
['S']
['C']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['C']
['S']
['S']
['S']
['C']
['S']
['S']
['C']
['S']
['C']
['S']
['S']
['S']
['S']
['C']
['S']
['C']
['S']
['S']
['S']
['Q']
['S']
['S']
['S']
['S']
['S']
['C']
['S']
['S']
['S']
['C']
['C']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['Q']
['S']
['S']
['S']
['S']
['S']
['Q']
['Q']
['S']
['Q']

['Q']
['Q']
['S']
['S']
['S']
['C']
['S']
['S']
['C']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['C']
['C']
['S']
['S']
['C']
['S']
['Q']
['S']
['S']
['S']
['S']
['S']
['S']
['C']
['C']
['S']
['S']
['S']
['C']
['S']
['S']
['S']
['S']
['C']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['C']
['S']
['S']
['S']
['S']
['C']
['C']
['C']
['S']
['S']
['S']
['C']
['S']
['S']
['S']
['C']

['S']
['S']
['S']
['S']
['Q']
['Q']
['S']
['S']
['S']
['S']
['C']
['S']
['S']
['S']
['S']
['C']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['C']
['Q']
['S']
['S']
['Q']
['S']
['S']
['S']
['S']
['Q']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['S']
['C']
['C']
['C']
['S']
['S']
['S']
['S']
['S']
['S']
['Q']
['Q']
['S']
['S']
['S']
['S']
['C']
['S']
['S']
['S']
['S']

```
['S']
['S']]
```

One-Hot Encoding with Sex and Embarked

In [12]:
```
ohe_sex = OneHotEncoder(sparse=False,handle_unknown='ignore')
ohe_embarked = OneHotEncoder(sparse=False,handle_unknown='ignore')

X_train_sex = ohe_sex.fit_transform(X_train[['Sex']])
X_train_embarked = ohe_embarked.fit_transform(X_train_embarked)

X_test_sex = ohe_sex.transform(X_test[['Sex']])
X_test_embarked = ohe_embarked.transform(X_test_embarked)
```

In [13]:
```
X_train_embarked
```

Out[13]:
```
array([[0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       ...,
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.]])
```

In [14]:
```
X_train.head(2)
```

Out[14]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
331	1	male	45.5	0	0	28.5	S
733	2	male	23.0	0	0	13.0	S

In [15]:
```
X_train_rem = X_train.drop(columns=['Sex','Age','Embarked'])
```

In [16]:
```
X_test_rem = X_test.drop(columns=['Sex','Age','Embarked'])
```

Now concatenet the X_train_embarked column and X_train column and also same on X_test

In [17]:
```
X_train_transformed = np.concatenate((X_train_rem,X_train_age,X_train_sex,X_train_embarked
X_test_transformed = np.concatenate((X_test_rem,X_test_age,X_test_sex,X_test_embarked),axi
```

In [18]:
```
X_test_transformed.shape
```

Out[18]:
```
(179, 10)
```

In [19]:
```
clf = DecisionTreeClassifier()
clf.fit(X_train_transformed,y_train)
```

Out[19]:
```
DecisionTreeClassifier()
```

In [20]:
```
y_pred = clf.predict(X_test_transformed)
```

```
y_pred
```

Out[20]:
```
array([0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 1], dtype=int64)
```

In [21]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[21]:
```
0.776536312849162
```

In [22]:
```
import pickle
```

In [23]:
```
pickle.dump(ohe_sex,open('ohe_sex.pkl','wb'))
pickle.dump(ohe_embarked,open('ohe_embarked.pkl','wb'))
pickle.dump(clf,open('clf.pkl','wb'))
```

In [ ]:
```

```

PipiLine in Machine Learning

Pred. without PipeLine

In [1]:
```
import pickle
import numpy as np
```

In [2]:
```
ohe_sex = pickle.load(open('ohe_sex.pkl','rb'))
ohe_embarked = pickle.load(open('ohe_embarked.pkl','rb'))
clf = pickle.load(open('clf.pkl','rb'))
```

In [3]:
```
# Process Flow >>>>>>> Pclass/gender/age/SibSp/Parch/Fare/Embarked

test_input = np.array([2, 'male', 31.0, 0, 0, 10.5, 'S'],dtype=object).reshape(1,7)
```

In [4]:
```
test_input
```

Out[4]:
```
array([[2, 'male', 31.0, 0, 0, 10.5, 'S']], dtype=object)
```

In [5]:
```
test_input_sex = ohe_sex.transform(test_input[:,1].reshape(1,1))
```

In [6]:
```
test_input_sex
```

Out[6]:
```
array([[0., 1.]])
```

In [7]:
```
test_input_embarked = ohe_embarked.transform(test_input[:,-1].reshape(1,1))
```

In [8]:
```
test_input_embarked
```

Out[8]:
```
array([[0., 0., 1.]])
```

In [ ]:
```

```

Step by Step apply Pipeline

With PipeLine

In [1]:
```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline,make_pipeline
from sklearn.feature_selection import SelectKBest,chi2
from sklearn.tree import DecisionTreeClassifier
```

In [2]:
```
df = pd.read_csv('train.csv')
```

In [3]:
```
df.sample(5)
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
760	761	0	3	Garfirth, Mr. John	male	NaN	0	0	358585	14.5000	NaN	S
423	424	0	3	Danbom, Mrs. Ernst Gilbert (Anna Sigrid Maria ...	female	28.0	1	1	347080	14.4000	NaN	S
210	211	0	3	Ali, Mr. Ahmed	male	24.0	0	0	SOTON/O.Q. 3101311	7.0500	NaN	S
503	504	0	3	Laitinen, Miss. Kristina Sofia	female	37.0	0	0	4135	9.5875	NaN	S
386	387	0	3	Goodwin, Master. Sidney Leonard	male	1.0	5	2	CA 2144	46.9000	NaN	S

In [4]:
```
#selcet the important column
df.drop(columns=['PassengerId','Name','Ticket','Cabin'],inplace=True)
```

In [5]:
```
df.sample(5)
```

Out[5]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
700	1	1	female	18.0	1	0	227.5250	C

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
621	1	1	male	42.0	1	0	52.5542	S
520	1	1	female	30.0	0	0	93.5000	S
60	0	3	male	22.0	0	0	7.2292	C
462	0	1	male	47.0	0	0	38.5000	S

Step 1 - Train-Test-Split

In [6]:
```
X_train,X_test,y_train,y_test = train_test_split(df.drop(columns=['Survived']),
                                                 df['Survived'],
                                                 test_size=0.2,
                                                random_state=42)
```

In [7]:
```
print(X_train.head(2))
```

```
     Pclass   Sex   Age  SibSp  Parch  Fare Embarked
331       1  male  45.5      0      0  28.5        S
733       2  male  23.0      0      0  13.0        S
```

In [8]:
```
print(y_train.head())
```

```
331    0
733    0
382    0
704    0
813    0
Name: Survived, dtype: int64
```

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
Survived      0
Pclass        0
Sex           0
Age         177
SibSp         0
Parch         0
Fare          0
Embarked      2
dtype: int64
```

Step.2 Imputation Transformer

In [24]:
```
# imputation transformer
trf1 = ColumnTransformer([
    ('impute_age',SimpleImputer(),[2]),
    ('impute_embarked',SimpleImputer(strategy='most_frequent'),[6])
],remainder='passthrough')
```

Step.3 One-Hot Encoding

In [25]:
```
trf2 = ColumnTransformer([
```

```
    ('ohe_sex_embarked',OneHotEncoder(sparse=False,handle_unknown='ignore'),[1,6])
],remainder='passthrough')
```

Step.4 Scaling

In [26]:
```
trf3 = ColumnTransformer([
    ('scale',MinMaxScaler(),slice(0,10))
])
```

Step.5 Feature Selection

In [27]:
```
trf4 = SelectKBest(score_func=chi2,k=8)
```

Step.6 Train the Model

In [28]:
```
trf5 = DecisionTreeClassifier()
```

Step.7 Create PipeLine

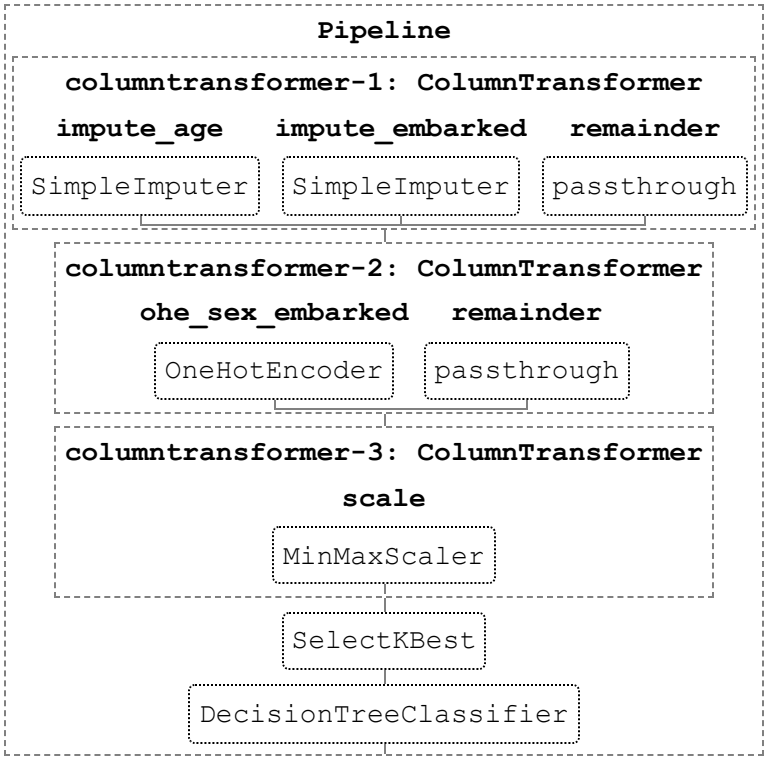
In [29]:
```
pipe = Pipeline([
    ('trf1',trf1),
    ('trf2',trf2),
    ('trf3',trf3),
    ('trf4',trf4),
    ('trf5',trf5)
])
```

Step7.1 Create Pipeline with Make_PipeLine

In [30]:
```
pipe = make_pipeline(trf1,trf2,trf3,trf4,trf5)
```

In [31]:
```
pipe.fit(X_train,y_train)
```

Out[31]:



In [18]:
```
pipe.named_steps
```

Out[18]:
```
{'columntransformer-1': ColumnTransformer(remainder='passthrough',
                  transformers=[('impute_age', SimpleImputer(), [2]),
                                ('impute_embarked',
                                 SimpleImputer(strategy='most_frequent'),
                                 [6])]),
 'columntransformer-2': ColumnTransformer(remainder='passthrough',
                  transformers=[('ohe_sex_embarked',
                                 OneHotEncoder(handle_unknown='ignore',
                                               sparse=False),
                                 [1, 6])]),
 'columntransformer-3': ColumnTransformer(transformers=[('scale', MinMaxScaler(), slice(0,
10, None))]),
 'selectkbest': SelectKBest(k=8, score_func=<function chi2 at 0x000002836E0DED30>),
 'decisiontreeclassifier': DecisionTreeClassifier()}
```

In [19]:
```
# Display Pipeline

from sklearn import set_config
set_config(display='diagram')
```

In [20]:
```
# Predict
y_pred = pipe.predict(X_test)
```

In [21]:
```
y_pred
```

Out[21]:
```
array([1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1,
       0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1,
       0, 0, 0], dtype=int64)
```

In [22]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[22]:
```
0.6256983240223464
```

Cross Validation using Pipeline

In [23]:
```
# cross validation using cross_val_score
from sklearn.model_selection import cross_val_score
cross_val_score(pipe, X_train, y_train, cv=5, scoring='accuracy').mean()
```

Out[23]:
```
0.6391214419383433
```

In [ ]: