13-Machine_Learning_Pipeline

October 20, 2024

1 Machine Learning Pipeline

[96]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline, Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer

from sklearn.tree import DecisionTreeClassifier

import joblib
```

[97]:

```
d1 = {
    'Social_media_followers' : [1000000, np.nan, 2000000, 1310000, 1700000, np.
 ↪nan, 4100000, 1600000, 2200000, 1000000],
    'Sold_out': [1,0,0,1,0,0,0,1,0,1]
}

d2 = {
        'Genre':['Rock', 'Metal', 'Bluegrass', 'Rock', np.nan, 'Rock', 'Rock',␣
 ↪np.nan, 'Bluegrass', 'Rock'],
        'Social_media_followers':[1000000, np.nan, 2000000, 1310000, 1700000,␣
 ↪np.nan, 4100000, 1600000, 2200000, 1000000],
        'Sold_out':[1,0,0,1,0,0,0,1,0,1]
    }

df1 = pd.DataFrame(d1)
df1
```

[97]:

```
   Social_media_followers  Sold_out
0               1000000.0         1
1                     NaN         0
2               2000000.0         0
3               1310000.0         1
```

```
4              1700000.0          0
5                    NaN          0
6              4100000.0          0
7              1600000.0          1
8              2200000.0          0
9              1000000.0          1
```

[98]:

```
X1 = df1[['Social_media_followers']]
y1 = df1[['Sold_out']]

X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.3,␣
 ↪random_state=19)

imputer = SimpleImputer(strategy='mean')
lr = LogisticRegression()

pipe1 = make_pipeline(imputer, lr)

pipe1.fit(X1_train, y1_train)
```

```
c:\Users\ikiga\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\utils\validation.py:1229: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the shape of y
to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

[98]:

```
Pipeline(steps=[('simpleimputer', SimpleImputer()),
                ('logisticregression', LogisticRegression())])
```

[99]:

```
pipe1.score(X1_train, y1_train)
```

[99]:

```
1.0
```

[100]:

```
pipe1.score(X1_test, y1_test)
```

[100]:

```
0.6666666666666666
```

[101]:

```
pipe1.named_steps.simpleimputer.statistics_
```

[101]:

```
array([2051666.66666667])
```

[102]:

```
pipe1.named_steps.logisticregression.coef_
```

[102]:

```
array([[-9.72872687e-05]])
```

1.0.1 More Advance Pipeline

[103]:
```
df = pd.DataFrame(data=d2)
df
```

[103]:
```
      Genre  Social_media_followers  Sold_out
0      Rock               1000000.0         1
1     Metal                     NaN         0
2  Bluegrass              2000000.0         0
3      Rock               1310000.0         1
4       NaN               1700000.0         0
5      Rock                     NaN         0
6      Rock               4100000.0         0
7       NaN               1600000.0         1
8  Bluegrass              2200000.0         0
9      Rock               1000000.0         1
```

[104]:
```
X = df.iloc[:,0:2]
y = df.iloc[:,2]

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3,␣
 ↪random_state=17)

num_cols = ["Social_media_followers"]
cat_cols = ['Genre']

num_pipeline = Pipeline(
    steps = [
        ('impute', SimpleImputer(strategy='mean')),
        ('scale', StandardScaler())
    ]
)

cat_pipeline = Pipeline(steps=[
    ('impute', SimpleImputer(strategy='most_frequent')),
    ('one-hot-encoder', OneHotEncoder(handle_unknown='ignore',␣
 ↪sparse_output=False))
])
cat_pipeline
```

[104]:
```
Pipeline(steps=[('impute', SimpleImputer(strategy='most_frequent')),
                ('one-hot-encoder',
                 OneHotEncoder(handle_unknown='ignore', sparse_output=False))])
```

[105]:
```
col_transformer = ColumnTransformer(transformers= [
    ('num_pipeline', num_pipeline, num_cols),
    ('cat_pipeline', cat_pipeline, cat_cols),
],
remainder='drop', n_jobs=-1
)
```

[106]:
```
dtc = DecisionTreeClassifier()
pipefinal = make_pipeline(col_transformer, dtc)
pipefinal.fit(X_train, y_train)
```

[106]:
```
Pipeline(steps=[('columntransformer',
                 ColumnTransformer(n_jobs=-1,
                                   transformers=[('num_pipeline',
                                                  Pipeline(steps=[('impute',
SimpleImputer()),
                                                                  ('scale',
StandardScaler())]),
                                                  ['Social_media_followers']),
                                                 ('cat_pipeline',
                                                  Pipeline(steps=[('impute',
SimpleImputer(strategy='most_frequent')),
                                                                  ('one-hot-
encoder',
OneHotEncoder(handle_unknown='ignore',
 sparse_output=False))]),
                                                  ['Genre'])])),
                ('decisiontreeclassifier', DecisionTreeClassifier())])
```

[107]:
```
pipefinal.score(X_test, y_test)
```

[107]:
```
0.6666666666666666
```

1.1 How to save your pipeline

[108]:
```
joblib.dump(pipefinal, 'pipe.joblib')
```

[108]:
```
['pipe.joblib']
```

[109]:
```
pipefinal2 = joblib.load('pipe.joblib')
pipefinal2
```

[109]:
```
Pipeline(steps=[('columntransformer',
                 ColumnTransformer(n_jobs=-1,
                                   transformers=[('num_pipeline',
                                                  Pipeline(steps=[('impute',
SimpleImputer()),
                                                                  ('scale',
StandardScaler())]),
                                                  ['Social_media_followers']),
                                                 ('cat_pipeline',
                                                   Pipeline(steps=[('impute',
SimpleImputer(strategy='most_frequent')),
                                                                    ('one-hot-
encoder',
OneHotEncoder(handle_unknown='ignore',
 sparse_output=False))]),
                                                   ['Genre'])])),
                 ('decisiontreeclassifier', DecisionTreeClassifier())])
```