Лабораторная работа 5. Разработка единого шаблона предварительной обработки данных

Подключение библиотек

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Загрузка данных и разделение на матрицу признаков и зависимую переменную

```
# Решил воспользоваться тем же набором данных, однако сгенерировал свои значения и названия стран, а так же добавил новый пропуск в данных.
dataset = pd.read_csv('/content/dataset.csv')
dataset.head()
```


```
{"summary":"{\n  \"name\": \"dataset\",\n  \"rows\": 10,\n  \"fields\": [\n    {\n      \"column\": \"Country\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 4,\n        \"samples\": [\n          \"France\",\n          \"USA\",\n          \"Russia\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 8.98764584180851,\n        \"min\": 35.0,\n        \"max\": 65.0,\n        \"num_unique_values\": 7,\n        \"samples\": [\n          65.0,\n          39.0,\n          47.0\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Salary\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 19694.189780519308,\n        \"min\": 26000.0,\n        \"max\": 89000.0,\n        \"num_unique_values\": 9,\n        \"samples\": [\n          89000.0,\n          26000.0,\n          52000.0\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Purchased\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 2,\n        \"samples\": [\n          \"No\",\n          \"Yes\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe","variable_name":"dataset"}
```

```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 3].values
print ("Матрица признаков"); print(X)
print ("Зависимая переменная"); print(y)

Матрица признаков
[['Russia' 65.0 31000.0]
 ['France' 39.0 26000.0]
 ['UK' 35.0 34000.0]
 ['USA' 45.0 51000.0]
 ['UK' 37.0 nan]
 ['France' nan 49000.0]
 ['USA' 47.0 52000.0]
 ['France' 48.0 63000.0]
 ['UK' 45.0 89000.0]
 ['France' 39.0 65000.0]]
Зависимая переменная
['Yes' 'No' 'No' 'No' 'Yes' 'Yes' 'Yes' 'Yes' 'No' 'No']
```

Обработка пропущенных значений

```
# устаревший подход
from sklearn.preprocessing import Imputer
imputer = Imputer(missing_values = 'NaN', strategy = 'mean', axis = 0)
imputer = imputer.fit(X[:, 1:3])
X_without_nan = X.copy()
X_without_nan[:, 1:3] = imputer.transform(X[:, 1:3])
print(X_without_nan)

---------------------------------------------------------------------------
-----
ImportError                               Traceback (most recent call
last)
<ipython-input-5-a6d4314c6e6e> in <cell line: 2>()
      1 # устаревший подход
----> 2 from sklearn.preprocessing import Imputer
      3 imputer = Imputer(missing_values = 'NaN', strategy = 'mean',
axis = 0)
      4 imputer = imputer.fit(X[:, 1:3])
      5 X_without_nan = X.copy()

ImportError: cannot import name 'Imputer' from 'sklearn.preprocessing'
(/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/__init_
_.py)

---------------------------------------------------------------------------
-----
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.
```

```
To view examples of installing some common dependencies, click the
"Open Examples" button below.
-------------------------------------------------------------------
-----


# Новая версия класса-трансформера, предыдущая Imputer - устарела
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values = np.nan, strategy = 'mean')
imputer = imputer.fit(X[:, 1:3])
X_without_nan = X.copy()
X_without_nan[:, 1:3] = imputer.transform(X[:, 1:3])
X_without_nan
```

```
array([['Russia', 65.0, 31000.0],
       ['France', 39.0, 26000.0],
       ['UK', 35.0, 34000.0],
       ['USA', 45.0, 51000.0],
       ['UK', 37.0, 51111.11111111111],
       ['France', 44.44444444444444, 49000.0],
       ['USA', 47.0, 52000.0],
       ['France', 48.0, 63000.0],
       ['UK', 45.0, 89000.0],
       ['France', 39.0, 65000.0]], dtype=object)
```

Обработка категориальных данных

Замена категории кодом (LabelEncoder)

```
from sklearn.preprocessing import LabelEncoder
labelencoder_y = LabelEncoder()
print("Зависимая переменная до обработки")
print(y)
y = labelencoder_y.fit_transform(y)
print("Зависимая переменная после обработки")
print(y)
```

```
Зависимая переменная до обработки
['Yes' 'No' 'No' 'No' 'Yes' 'Yes' 'Yes' 'Yes' 'No' 'No']
Зависимая переменная после обработки
[1 0 0 0 1 1 1 1 0 0]
```

Применение OneHotEncoder

```
# устаревший подход к использованию OneHotEncoder
from sklearn.preprocessing import OneHotEncoder
labelencoder_X = LabelEncoder()
labelencoder_X.fit_transform(X[:, 0])
X_encoded = X_without_nan.copy()
X_encoded[:, 0] = labelencoder_X.fit_transform(X_encoded[:, 0])
```

```
onehotencoder = OneHotEncoder(categorical_features = [0])
X_encoded = onehotencoder.fit_transform(X_encoded).toarray()
print("Перекодировка категориального признака")
print(X_encoded)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call
last)
<ipython-input-8-7f115cff7a46> in <cell line: 7>()
      5 X_encoded = X_without_nan.copy()
      6 X_encoded[:, 0] = labelencoder_X.fit_transform(X_encoded[:,
0])
----> 7 onehotencoder = OneHotEncoder(categorical_features = [0])
      8 X_encoded = onehotencoder.fit_transform(X_encoded).toarray()
      9 print("Перекодировка категориального признака")

TypeError: OneHotEncoder.__init__() got an unexpected keyword argument
'categorical_features'
```

```
# создаем копию "грязного" объекта: с пропусками и некодированными
категориями
X_dirty = X.copy()
X_dirty
```

```
array([['Russia', 65.0, 31000.0],
       ['France', 39.0, 26000.0],
       ['UK', 35.0, 34000.0],
       ['USA', 45.0, 51000.0],
       ['UK', 37.0, nan],
       ['France', nan, 49000.0],
       ['USA', 47.0, 52000.0],
       ['France', 48.0, 63000.0],
       ['UK', 45.0, 89000.0],
       ['France', 39.0, 65000.0]], dtype=object)
```

```
# Современный метод трансформации признаков
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer

# создаем список трансформеров
transformers = [
    ('onehot', OneHotEncoder(), [0]),
    ('imp', SimpleImputer(), [1, 2])
]

# Создаем объект ColumnTransformer и передаем ему список трансформеров
ct = ColumnTransformer(transformers)

# Выполняем трансформацию признаков
```

```
X_transformed = ct.fit_transform(X_dirty)
print(X_transformed.shape)
X_transformed
```

```
(10, 6)
```

```
array([[0.00000000e+00, 1.00000000e+00, 0.00000000e+00,
0.00000000e+00,
        6.50000000e+01, 3.10000000e+04],
       [1.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00,
        3.90000000e+01, 2.60000000e+04],
       [0.00000000e+00, 0.00000000e+00, 1.00000000e+00,
0.00000000e+00,
        3.50000000e+01, 3.40000000e+04],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
1.00000000e+00,
        4.50000000e+01, 5.10000000e+04],
       [0.00000000e+00, 0.00000000e+00, 1.00000000e+00,
0.00000000e+00,
        3.70000000e+01, 5.11111111e+04],
       [1.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00,
        4.44444444e+01, 4.90000000e+04],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
1.00000000e+00,
        4.70000000e+01, 5.20000000e+04],
       [1.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00,
        4.80000000e+01, 6.30000000e+04],
       [0.00000000e+00, 0.00000000e+00, 1.00000000e+00,
0.00000000e+00,
        4.50000000e+01, 8.90000000e+04],
       [1.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00,
        3.90000000e+01, 6.50000000e+04]])
```

```
# Можно преобразовать полученный многомерный массив обратно в
Dataframe
X_data = pd.DataFrame(
    X_transformed,
    columns=['C1', 'C2', 'C3', 'C4', 'Age', 'Salary'])
X_data
```


```
{"summary":"{\n  \"name\": \"X_data\",\n  \"rows\": 10,\n  \"fields\":
[\n    {\n      \"column\": \"C1\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.5163977794943222,\n
\"min\": 0.0,\n        \"max\": 1.0,\n        \"num_unique_values\":
2,\n        \"samples\": [\n          1.0,\n          0.0\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
```

```
}\n    },\n    {\n      \"column\": \"C2\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.31622776601683794,\n
\"min\": 0.0,\n        \"max\": 1.0,\n        \"num_unique_values\":
2,\n        \"samples\": [\n          0.0,\n          1.0\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"C3\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.4830458915396479,\n
\"min\": 0.0,\n        \"max\": 1.0,\n        \"num_unique_values\":
2,\n        \"samples\": [\n          1.0,\n          0.0\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"C4\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.42163702135578396,\n
\"min\": 0.0,\n        \"max\": 1.0,\n        \"num_unique_values\":
2,\n        \"samples\": [\n          1.0,\n          0.0\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 8.473633762194497,\n
\"min\": 35.0,\n        \"max\": 65.0,\n        \"num_unique_values\":
8,\n        \"samples\": [\n          39.0,\n
44.44444444444444\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Salary\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 18567.860191706674,\n        \"min\": 26000.0,\n
\"max\": 89000.0,\n        \"num_unique_values\": 10,\n
\"samples\": [\n          89000.0,\n          26000.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"X_data"}
```