



**Лабораторная работа №5 по дисциплине "Искусственный интеллект и машинное обучение"***Выполнил:* студент 2-го курса Звездин Алексей Сергеевич*Группа:* ПИЖ-6-о-22-1*Руководитель практики:* Березина Виктория Андреевна, ассистент кафедры информационных систем и технологий института цифрового развития**Тема работы:** Разработка единого подхода к предварительной обработке данных**Цель работы:** Изучение теоретических принципов и инструментальных редств для построения пайплайна для предварительной обработки данных

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


```
dataset = pd.read_csv('https://raw.githubusercontent.com/InternetHacker1123/bd_ai/main/laba5/data.csv')
dataset.head()
```



	Country	Age	Salary	Purchased	
0	Belgium	46	74000	No	
1	Italy	29	50000	Yes	
2	Austria	32	56000	No	
3	Italy	40	63000	No	
4	Austria	42	67000	Yes	


Next steps:  [View recommended plots](#)

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



```
Матрица признаков
[['Belgium' 46 74000]
 ['Italy' 29 50000]
 ['Austria' 32 56000]
 ['Italy' 40 63000]
 ['Austria' 42 67000]
 ['Belgium' 37 60000]
 ['Italy' 30 54000]
 ['Belgium' 50 81000]
 ['Austria' 52 85000]
 ['Belgium' 39 69000]]
Зависимая переменная
['No' 'Yes' 'No' 'No' 'Yes' 'Yes' 'No' 'Yes' 'No' 'Yes']
```

```
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(['Belgium', 46.0, 74000.0],
      ['Italy', 29.0, 50000.0],
      ['Austria', 32.0, 56000.0],
      ['Italy', 40.0, 63000.0],
      ['Austria', 42.0, 67000.0],
      ['Belgium', 37.0, 60000.0],
      ['Italy', 30.0, 54000.0],
      ['Belgium', 50.0, 81000.0],
      ['Austria', 52.0, 85000.0],
      ['Belgium', 39.0, 69000.0]], dtype=object)
```

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

```

```

X_dirty = X.copy()
X_dirty

```

```

array([[ 'Belgium', 46, 74000],
       [ 'Italy', 29, 50000],
       [ 'Austria', 32, 56000],
       [ 'Italy', 40, 63000],
       [ 'Austria', 42, 67000],
       [ 'Belgium', 37, 60000],
       [ 'Italy', 30, 54000],
       [ 'Belgium', 50, 81000],
       [ 'Austria', 52, 85000],
       [ 'Belgium', 39, 69000]], dtype=object)

```

```

from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer

```

```

transformers = [
    ('onehot', OneHotEncoder(), [0]),
    ('imp', SimpleImputer(), [1, 2])
]

```

```

ct = ColumnTransformer(transformers)

```

```

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

```

```

(10, 5)
array([[0.0e+00, 1.0e+00, 0.0e+00, 4.6e+01, 7.4e+04],
       [0.0e+00, 0.0e+00, 1.0e+00, 2.9e+01, 5.0e+04],
       [1.0e+00, 0.0e+00, 0.0e+00, 3.2e+01, 5.6e+04],
       [0.0e+00, 0.0e+00, 1.0e+00, 4.0e+01, 6.3e+04],
       [1.0e+00, 0.0e+00, 0.0e+00, 4.2e+01, 6.7e+04],
       [0.0e+00, 1.0e+00, 0.0e+00, 3.7e+01, 6.0e+04],
       [0.0e+00, 0.0e+00, 1.0e+00, 3.0e+01, 5.4e+04],
       [0.0e+00, 1.0e+00, 0.0e+00, 5.0e+01, 8.1e+04],
       [1.0e+00, 0.0e+00, 0.0e+00, 5.2e+01, 8.5e+04],
       [0.0e+00, 1.0e+00, 0.0e+00, 3.9e+01, 6.9e+04]])

```

```

X_data = pd.DataFrame(
    X_transformed,
    columns=['C1', 'C2', 'C3', 'Age', 'Salary'])
X_data

```

```


```

	C1	C2	C3	Age	Salary
0	0.0	1.0	0.0	46.0	74000.0
1	0.0	0.0	1.0	29.0	50000.0
2	1.0	0.0	0.0	32.0	56000.0
3	0.0	0.0	1.0	40.0	63000.0
4	1.0	0.0	0.0	42.0	67000.0
5	0.0	1.0	0.0	37.0	60000.0
6	0.0	0.0	1.0	30.0	54000.0
7	0.0	1.0	0.0	50.0	81000.0
8	1.0	0.0	0.0	52.0	85000.0
9	0.0	1.0	0.0	39.0	69000.0

Next steps: [View recommended plots](#)

## ✓ Контрольные вопросы:

1. Какая библиотека python предназначена для управления наборами данных: numpy, pandas, sklearn, opencv, matplotlib?

Библиотека Python, предназначенная для управления наборами данных, это pandas.

2. Какая стратегия является нежелательной при обработке пропусков в данных? а) замена пропущенных значений в столбце медианным значением по данному столбцу; б) удаление строк, содержащих пропуски в данных; в) замена пропущенных значений в столбце средним арифметическим значением по данному столбцу; г) замена пропущенных значений в столбце наиболее часто встречающимся значением по данному столбцу;

Нежелательной стратегией при обработке пропусков в данных является б) удаление строк, содержащих пропуски в данных, так как это может привести к потере значимой информации.

3. Обоснуйте ответ на следующую проблему предварительной обработки данных: имеется независимая категориальная переменная  $u$ , которая представляет собой категориальный признак, определенный на домене {C#, Java, Python, R}. Нужно ли применять к данному целевому признаку OneHotEncoder?

Нет, не нужно применять к целевому признаку OneHotEncoder, так как целевой признак не нужно преобразовывать в бинарные переменные. OneHotEncoder применяется к категориальным признакам, которые являются независимыми предикторами.

4. Поясните принцип разбиения набора данных на обучающую и тестовую выборку. Какое соотношение «тестовая:обучающая» наиболее оптимально: 20:80, 50:50, 25:75, 5:95, 40:30?

Набор данных обычно разбивается на обучающую и тестовую выборки для оценки производительности модели. Наиболее оптимальное соотношение «тестовая:обучающая» зависит от размера набора данных и сложности задачи, но часто используются соотношения 70:30 или 80:20, где больший процент данных выделяется для обучения модели.

5. Какой код лучше использовать при загрузке данных из csv-файла? а) `dataset = read_csv("data.csv")` б) `dataset = import("data.csv")` в) `dataset = read.csv("data.csv")` г) `dataset = import.csv("data.csv")` д) `dataset = read_xls("data.csv")`.

Правильный код для загрузки данных из csv-файла это: а) `dataset = read_csv("data.csv")` - функция `read_csv()` из библиотеки `pandas` используется для загрузки данных из csv-файла.