Лабораторная работа №7. Использование разработанного пайплайна для многомерной регрессии

Цель лабораторной работы: научиться применять разработанный пайплайн для тиражирования кода с целью решения широкого круга задач машинного обучения.

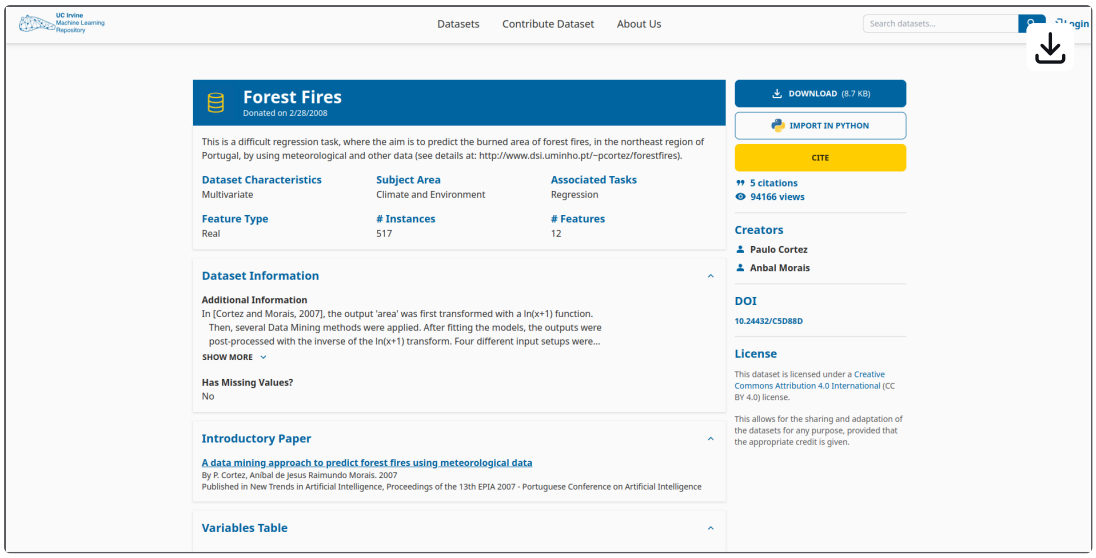
Основные задачи:

- получение навыков рефакторинга кода в проектах машинного обучения;
- получение навыков определения ключевых признаков в задачах машинного обучения;
- получение навыков реализации ключевых стратегий оптимизации моделей регрессии

Ход выполнения индивидуального задания:

1. Выбор набора данных для задачи регрессии

В качестве данных для лабораторной работы был выбран набор под названием **Forest Fires**, который входит в репозиторий **UCI Machine Learning Repository**.



2. Построение модели многомерной регрессии с использованием стратегии Backward Elimination

2.1 Загрузка и визуализация набора данных

Первым делом импортируем все необходимые библиотеки для выполнения лабораторной работы, а также загрузим наш набор данных с помощью **pandas** и выведем первые пять строк:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
import statsmodels.api as sm

data_path = "https://archive.ics.uci.edu/ml/machine-learning-
databases/forest-fires/forestfires.csv"

data = pd.read_csv(data_path)

print(data.head())
```

Как результат получаем следующий вывод:

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0

Проведём **предобработку** нашего набора данных. А именно: **удалим аномалии** (нулевые и слишком большие значения) из целевой переменной `area`, а также применим **логарифмическое преобразование** для **улучшения** работы алгоритмов машинного обучения:

```
data['area'] = np.log1p(data['area'])
data = data[(data['area'] > 0) & (data['area'] < 50)].copy()
data.reset_index(drop=True, inplace=True)

print(data.head())
```

```
   X  Y month  day  FFMC   DMC    DC  ISI  temp  RH  wind  rain      area
0  9  9   jul  tue  85.8  48.3 313.4  3.9  18.0  42   2.7   0.0  0.307485
1  1  4   sep  tue  91.0 129.5 692.6  7.0  21.7  38   2.2   0.0  0.357674
2  2  5   sep  mon  90.9 126.5 686.5  7.0  21.9  39   1.8   0.0  0.385262
3  1  2   aug  wed  95.5  99.9 513.3 13.2  23.3  31   4.5   0.0  0.438255
4  8  6   aug  fri  90.1 108.0 529.8 12.5  21.2  51   8.9   0.0  0.476234
```

Разделим наши данные на **матрицу признаков** и **зависимую переменную**, после чего выведем первые 10 значений на экран:

```
X = data.drop('area', axis=1)
y = data['area']

print("\nМатрица признаков")
print(X[:10])
print("\nЗависимая переменная")
print(y[:10])
```

```
Матрица признаков
   X  Y month  day  FFMC   DMC    DC  ISI  temp  RH  wind  rain
0  9  9   jul  tue  85.8  48.3 313.4  3.9  18.0  42   2.7   0.0
1  1  4   sep  tue  91.0 129.5 692.6  7.0  21.7  38   2.2   0.0
2  2  5   sep  mon  90.9 126.5 686.5  7.0  21.9  39   1.8   0.0
3  1  2   aug  wed  95.5  99.9 513.3 13.2  23.3  31   4.5   0.0
4  8  6   aug  fri  90.1 108.0 529.8 12.5  21.2  51   8.9   0.0
5  1  2   jul  sat  90.0  51.3 296.3  8.7  16.6  53   5.4   0.0
6  2  5   aug  wed  95.5  99.9 513.3 13.2  23.8  32   5.4   0.0
7  6  5   aug  thu  95.2 131.7 578.8 10.4  27.4  22   4.0   0.0
8  5  4   mar  mon  90.1  39.7  86.6  6.2  13.2  40   5.4   0.0
9  8  3   sep  tue  84.4  73.4 671.9  3.2  24.2  28   3.6   0.0
```

```
Зависимая переменная
0    0.307485
1    0.357674
2    0.385262
3    0.438255
4    0.476234
5    0.536493
6    0.570980
7    0.641854
8    0.667829
9    0.672944
```

2.2 Обработка категориальных признаков

В нашем наборе данных имеется два **категориальных признака**, которые необходимо обработать и привести в **удобный** для машинного обучения вид. Для этого воспользуемся такими средствами как `ColumnTransformer` и `OneHotEncoder`:

```
ct = ColumnTransformer(
    transformers=[
        ("encoder", OneHotEncoder(drop='first'), [2, 3])
    ],
    remainder="passthrough"
```

```
)

X_encoded = ct.fit_transform(X)

print("\nМатрица признаков после обработки категориальных признаков:")
print(X_encoded[:5])
```

```
Матрица признаков после обработки категориальных признаков:
[[  0.    0.    0.    1.    0.    0.    0.    0.    0.    0.    0.
    0.    1.    0.    9.    9.   85.8  48.3 313.4   3.9  18.   42.    2.7
    0. ]
 [  0.    0.    0.    0.    0.    0.    0.    0.    1.    0.    0.    0.
    0.    1.    0.    1.    4.   91.  129.5 692.6   7.   21.7  38.    2.2
    0. ]
 [  0.    0.    0.    0.    0.    0.    0.    0.    1.    1.    0.    0.
    0.    0.    0.    2.    5.   90.9 126.5 686.5   7.   21.9  39.    1.8
    0. ]
 [  1.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    1.    1.    2.   95.5  99.9 513.3  13.2  23.3  31.    4.5
    0. ]
 [  1.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    8.    6.   90.1 108.  529.8  12.5  21.2  51.    8.9
    0. ]]
```

Теперь, для удобства создадим `DataFrame` для нашей матрицы признаков с именами для каждой колонки, а также добавим **фиктивную переменную** с постоянным значением ***1.0***:

```
encoder = ct.named_transformers_['encoder']
encoded_features = encoder.get_feature_names_out(['month', 'day'])
remaining_features = [col for i, col in enumerate(X.columns) if i not
in [2, 3]]
all_features = np.concatenate([encoded_features, remaining_features])
X_encoded_df = pd.DataFrame(X_encoded, columns=all_features)
X_encoded_df = sm.add_constant(X_encoded_df)

print(X_encoded_df.head())
```

```
   const  month_aug  month_dec  month_feb  ...  temp    RH  wind  rain
0    1.0        0.0        0.0        0.0  ...  18.0  42.0   2.7   0.0
1    1.0        0.0        0.0        0.0  ...  21.7  38.0   2.2   0.0
2    1.0        0.0        0.0        0.0  ...  21.9  39.0   1.8   0.0
3    1.0        1.0        0.0        0.0  ...  23.3  31.0   4.5   0.0
4    1.0        1.0        0.0        0.0  ...  21.2  51.0   8.9   0.0
```

2.3 Оптимизация модели при помощи стратегии Backward Elimination

Backward elimination — это один из методов выбора признаков в линейной регрессии, который помогает удалить незначимые признаки из модели, улучшив её точность и интерпретируемость. Этот процесс заключается в пошаговом удалении признаков с **наибольшими p-значениями** (выше заданного порога), что позволяет оставить только те признаки, которые реально действительно на зависимую переменную.

Для этого напишем **функцию**, которая будет постепенно **удалять** признаки с **p-значением больше заданного порога**. По итогу в матрице признаков останутся только самые значимые для модели значения.

```
def backward_elimination(X, y, threshold=0.05):
    X = X.reset_index(drop=True)
    y = y.reset_index(drop=True)

    while True:
        model = sm.OLS(y, X).fit()
        pvalues = model.pvalues.drop('const', errors='ignore')
        if pvalues.empty:
            break

        max_pval = pvalues.max()
        if max_pval > threshold:
            feature_to_remove = pvalues.idxmax()
            print(f"Удаляем {feature_to_remove} (p-value:
{max_pval:.4f})")
            X = X.drop(feature_to_remove, axis=1)
        else:
            break

    print("\nИтоговая модель:")
    print(model.summary())
    return X
```

Подадим наши данные в функцию и посмотрим результат работы:

```
X_optimal = backward_elimination(X_encoded_df.copy(), y)
```

Во время работы **Backward elimination** мы можем отследить, какие признаки были удалены из набора данных:

```
Удаляем day_mon (p-value: 0.9985)
Удаляем rain (p-value: 0.9561)
Удаляем FFMC (p-value: 0.9226)
Удаляем RH (p-value: 0.8883)
Удаляем month_jul (p-value: 0.7899)
Удаляем day_wed (p-value: 0.7867)
Удаляем day_thu (p-value: 0.6220)
Удаляем month_feb (p-value: 0.5075)
Удаляем month_jun (p-value: 0.4556)
Удаляем month_mar (p-value: 0.5523)
Удаляем wind (p-value: 0.3947)
Удаляем ISI (p-value: 0.4854)
Удаляем Y (p-value: 0.3278)
Удаляем X (p-value: 0.2962)
Удаляем month_may (p-value: 0.2371)
Удаляем day_tue (p-value: 0.2015)
Удаляем day_sun (p-value: 0.1844)
Удаляем month_aug (p-value: 0.1246)
Удаляем temp (p-value: 0.1046)
Удаляем day_sat (p-value: 0.0789)
```

По итогу получаем следующие **модели**:

```
Итоговая модель:
                            OLS Regression Results
==============================================================================
Dep. Variable:                   area   R-squared:                       0.067
Model:                            OLS   Adj. R-squared:                  0.049
Method:                 Least Squares   F-statistic:                     3.769
Date:                Mon, 31 Mar 2025   Prob (F-statistic):           0.00259
Time:                        21:49:15   Log-Likelihood:                -435.14
No. Observations:                 270   AIC:                             882.3
Df Residuals:                     264   BIC:                             903.9
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          2.2307      0.213     10.489      0.000       1.812       2.649
month_dec      0.9285      0.440      2.109      0.036       0.061       1.795
month_oct      1.7234      0.617      2.795      0.006       0.509       2.938
month_sep      0.7858      0.221      3.554      0.000       0.350       1.221
DMC            0.0070      0.002      3.457      0.001       0.003       0.011
DC            -0.0022      0.001     -3.485      0.001      -0.003      -0.001
==============================================================================
Omnibus:                       37.183   Durbin-Watson:                   1.040
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               48.648
Skew:                           0.946   Prob(JB):                     2.73e-11
Kurtosis:                       3.861   Cond. No.                     5.28e+03
==============================================================================
```

2.4 Разделение выборки на тестовую и тренировочную. Обучение модели

Разделим выборку на **тестовую** и **тренировочную**, используя функцию `train_test_split`:

```
X_train, X_test, y_train, y_test = train_test_split(X_optimal,
                                                    y,
                                                    test_size=0.25,
                                                    random_state=42)
```

После разделения выборки обучим модель линейной регрессии на основе **тренировочных данных** и выполним предсказание данных, используя **тестовую выборку**:

```
final_model = sm.OLS(y_train, X_train).fit()
print("\nМодель на обучающих данных:")
print(final_model.summary())

y_pred = final_model.predict(X_test)

print("\nПервые 10 зависимых переменных (тестовая выборка)")
print(y_test[:10])
print("\nПервые 10 зависимых переменных (предсказание)")
print(y_pred[:10])
```

Получаем следующие характеристики модели на обучающих данных:

```
Модель на обучающих данных:
                            OLS Regression Results
==============================================================================
Dep. Variable:                   area   R-squared:                       0.083
Model:                            OLS   Adj. R-squared:                  0.059
Method:                 Least Squares   F-statistic:                     3.539
Date:                Mon, 31 Mar 2025   Prob (F-statistic):            0.00437
Time:                        21:49:15   Log-Likelihood:                -325.57
No. Observations:                 202   AIC:                             663.1
Df Residuals:                     196   BIC:                             683.0
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          2.2706      0.249      9.107      0.000       1.779       2.762
month_dec      1.0141      0.480      2.112      0.036       0.067       1.961
month_oct      2.1100      0.712      2.962      0.003       0.705       3.515
month_sep      0.8936      0.275      3.255      0.001       0.352       1.435
DMC            0.0095      0.003      3.722      0.000       0.004       0.015
DC            -0.0027      0.001     -3.465      0.001      -0.004      -0.001
==============================================================================
Omnibus:                       20.795   Durbin-Watson:                   2.220
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               23.908
Skew:                           0.799   Prob(JB):                     6.43e-06
Kurtosis:                       3.538   Cond. No.                     5.28e+03
==============================================================================
```

Также сравним зависимые **переменные из тестовой выборки** и **предсказанные значения**:

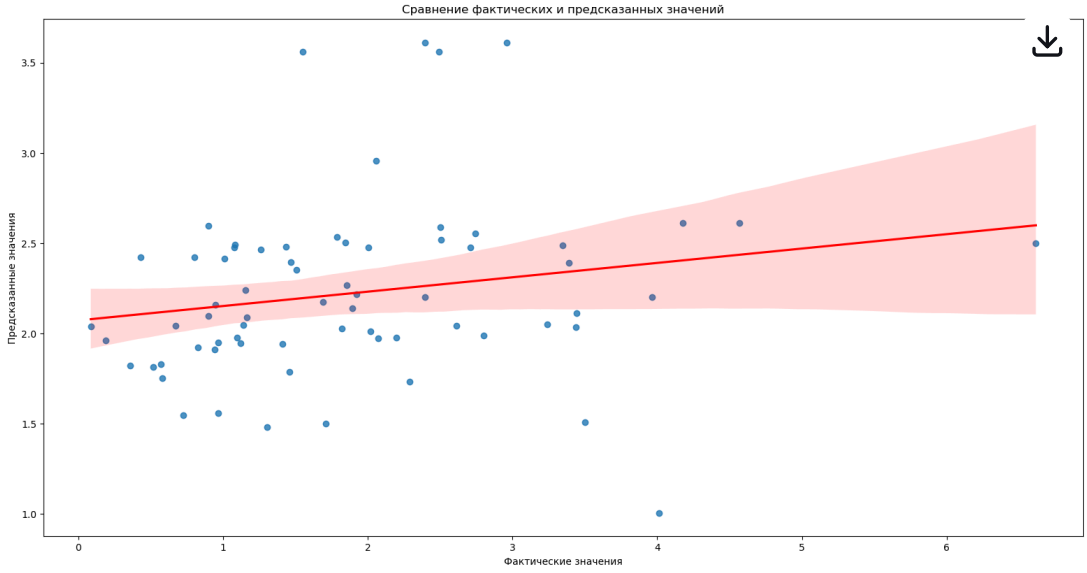
```
Первые 10 зависимых переменных (тестовая выборка)
30     1.261298
116    1.458615
79     3.444257
127    2.500616
196    2.394252
137    2.197225
209    6.616440
45     2.057963
158    1.141033
247    1.166271
```

```
Первые 10 зависимых переменных (предсказания)
30     2.466331
116    1.789392
79     2.115069
127    2.588329
196    2.203141
137    1.977172
209    2.498624
45     2.955673
158    2.045564
247    2.089159
```

2.5 Визуализация результатов

Чтобы понять насколько хорошо или плохо модель линейной регрессии предсказала данные для нашего набора, **визуализируем** результаты её работы. Выведем график зависимости **предсказанных** значений от **фактических**, а также **линию регрессии**:

```
plt.figure(figsize=(10, 6))
sns.regplot(x=y_test, y=y_pred, line_kws={'color': 'red'})
plt.xlabel('Фактические значения')
plt.ylabel('Предсказанные значения')
plt.title('Сравнение фактических и предсказанных значений')
```



Контрольные вопросы

1. Этот признак используется для того, чтобы учесть *свободный член (intercept)* в модели, то есть константу `β0`. В линейной регрессии модель может быть записана как:

   $$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n$$

   Чтобы включить свободный член в уравнение, мы добавляем фиктивную переменную с постоянным значением 1.0. Это позволяет модели правильно интерпретировать влияние константы на зависимую переменную.

2. Фиктивная переменная (или дамми-переменная) — это бинарная переменная, принимающая значения 0 или 1, которая используется для представления категориальных признаков в числовом виде. Например, если у нас есть категориальный признак "Цвет", который принимает значения "красный", "синий", "зеленый", мы можем создать три фиктивные переменные: одна для каждого цвета. Однако, чтобы избежать мультиколлинеарности (ситуации, когда переменные линейно зависимы), обычно удаляется одна из фиктивных переменных. Это делается, чтобы избежать избыточности и не нарушить модель.
3. В алгоритме **back elimination** удаление признаков происходит на основе p-значений (p-value). Признак с наибольшим p-значением (показателем его значимости) удаляется, если оно превышает заранее установленный порог (например, 0.05). Это означает, что признак не влияет на модель и можно исключить его без значительных потерь в точности прогноза.
4. Алгоритм **all-in regression** заключается в том, что все признаки изначально включаются в модель, и затем проводится анализ, чтобы определить, какие из них оказывают наибольшее влияние на

целевую переменную. После этого, с помощью статистических методов (например, p-value или коэффициентов регрессии), удаляются наименее значимые признаки.

5. Алгоритм **forward selection regression** начинается с того, что в модель не включены никакие признаки, а затем на каждом шаге добавляется тот признак, который наилучшим образом улучшает модель. Этот процесс продолжается до тех пор, пока добавление новых признаков не перестанет существенно улучшать качество модели.
6. Алгоритм **Bidirectional Elimination** (или Stepwise Regression) сочетает элементы **forward selection** и **backward elimination**. Он добавляет признаки в модель по мере их значимости (как в forward selection), но при этом на каждом шаге также проверяет, не стоит ли удалить какой-то уже включенный признак (как в backward elimination). Это позволяет достичь оптимального набора признаков.
7. Для реализации автоматического удаления признаков на основе p-критерия в алгоритме **backward elimination** можно использовать программные средства, такие как Python с библиотеками **statsmodels** или **scikit-learn**. В таких библиотеках есть функции для проведения регрессии, которые автоматически исключают признаки с p-значением, превышающим заранее установленный порог. Например, можно использовать цикл для удаления признаков с высокими p-значениями до тех пор, пока все оставшиеся признаки будут значимыми.