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.

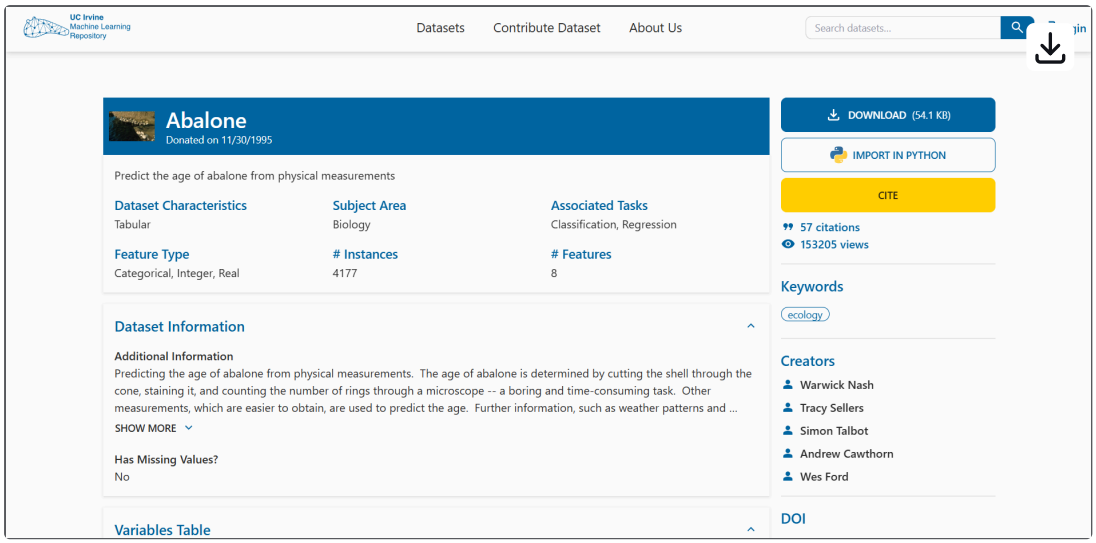
Основные задачи:

- 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 **Abalone**, 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/abalone/abalone.data"

columns = ['Sex', 'Length', 'Diameter', 'Height', 'Whole_weight',
           'Shucked_weight', 'Viscera_weight', 'Shell_weight', 'Rings']

data = pd.read_csv(data_path, header=None, names=columns)

print(data.head())
```

Как результат получаем следующий вывод:

	Sex	Length	Diameter	...	Viscera_weight	Shell_weight	Rings
0	M	0.455	0.365	...	0.1010	0.150	15
1	M	0.350	0.265	...	0.0485	0.070	7
2	F	0.530	0.420	...	0.1415	0.210	9
3	M	0.440	0.365	...	0.1140	0.155	10
4	I	0.330	0.255	...	0.0395	0.055	7

Проведём **предобработку** нашего набора данных. А именно: **удалим аномалии** (нулевые значения) из признака `Height` и **преобразуем**

параметр `Rings` в параметр `Age`. Это делается при помощи добавления к параметру **Rings** значения равного ***1.5***:

```
data = data[data['Height'] > 0]
data['Age'] = data['Rings'] + 1.5
data.drop('Rings', axis=1, inplace=True)

print(data.head())
```

Соответственно, **удаляем** уже ненужный параметр `Rings` и выведем наш набор данных на экран:

```
  Sex  Length  Diameter  ...  Viscera_weight  Shell_weight   Age
0   M   0.455     0.365  ...          0.1010         0.150  15.5
1   M   0.350     0.265  ...          0.0485         0.070   8.5
2   F   0.530     0.420  ...          0.1415         0.210  10.5
3   M   0.440     0.365  ...          0.1140         0.155  11.5
4   I   0.330     0.255  ...          0.0395         0.055   8.5
```

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

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

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

```
Матрица признаков
  Sex  Length  Diameter  ...  Shucked_weight  Viscera_weight  Shell_weight
0   M   0.455     0.365  ...          0.2245          0.1010         0.150
1   M   0.350     0.265  ...          0.0995          0.0485         0.070
2   F   0.530     0.420  ...          0.2565          0.1415         0.210
3   M   0.440     0.365  ...          0.2155          0.1140         0.155
4   I   0.330     0.255  ...          0.0895          0.0395         0.055
5   I   0.425     0.300  ...          0.1410          0.0775         0.120
6   F   0.530     0.415  ...          0.2370          0.1415         0.330
7   F   0.545     0.425  ...          0.2940          0.1495         0.260
8   M   0.475     0.370  ...          0.2165          0.1125         0.165
9   F   0.550     0.440  ...          0.3145          0.1510         0.320
```

```
Зависимая переменная
0    16.5
1     8.5
2    10.5
3    11.5
4     8.5
5     9.5
6    21.5
7    17.5
8    10.5
9    20.5
```

2.2 Обработка категориальных признаков

В нашем наборе данных имеется один **категориальный признак**, который 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'), ['Sex'])],
    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.     1.     0.455  0.365  0.095  0.514  0.2245 0.101  0.1   ]
 [0.     1.     0.35   0.265  0.09   0.2255 0.0995 0.0485 0.07  ]
 [0.     0.     0.53   0.42   0.135  0.677  0.2565 0.1415 0.21  ]
 [0.     1.     0.44   0.365  0.125  0.516  0.2155 0.114  0.155 ]
 [1.     0.     0.33   0.255  0.08   0.205  0.0895 0.0395 0.055 ]]
```

Теперь, для удобства создадим `DataFrame` для нашей матрицы признаков с именами для каждой колонки:

```
encoder = ct.named_transformers_['encoder']
encoded_features = encoder.get_feature_names_out(['Sex'])
numeric_features = [col for col in X.columns if col != 'Sex']
all_features = np.concatenate([encoded_features, numeric_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  Sex_I  Sex_M  ...  Shucked_weight  Viscera_weight  Shell_weight
0    1.0    0.0    1.0  ...          0.2245          0.1010          0.150
1    1.0    0.0    1.0  ...          0.0995          0.0485          0.070
2    1.0    0.0    0.0  ...          0.2565          0.1415          0.210
3    1.0    0.0    1.0  ...          0.2155          0.1140          0.155
4    1.0    1.0    0.0  ...          0.0895          0.0395          0.055
```

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)
y = y.reset_index(drop=True)
```

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

```
Удаляем Length (p-value: 0.8
Удаляем Sex_M (p-value: 0.49
```

По итогу получаем следующую **модель**:

```
Итоговая модель:
                            OLS Regression Results
==============================================================================
Dep. Variable:                    Age   R-squared:                       0.538
Model:                            OLS   Adj. R-squared:                  0.537
Method:                 Least Squares   F-statistic:                     692.7
Date:                Mon, 31 Mar 2025   Prob (F-statistic):               0.00
Time:                        18:44:10   Log-Likelihood:                -9200.2
No. Observations:                4175   AIC:                         1.842e+04
Df Residuals:                    4167   BIC:                         1.847e+04
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.4309      0.268     20.273      0.000       4.906       5.956
Sex_I         -0.8611      0.089     -9.633      0.000      -1.036      -0.686
Diameter      10.4864      0.989     10.606      0.000       8.548      12.425
Height        10.6859      1.543      6.923      0.000       7.660      13.712
Whole_weight   8.8767      0.731     12.146      0.000       7.444      10.310
Shucked_weight -19.6822    0.817    -24.078      0.000     -21.285     -18.080
Viscera_weight -10.5691    1.289     -8.200      0.000     -13.096      -8.042
Shell_weight   8.9473      1.140      7.851      0.000       6.713      11.182
==============================================================================
Omnibus:                      945.773   Durbin-Watson:                   1.436
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2703.325
Skew:                           1.180   Prob(JB):                         0.00
Kurtosis:                       6.157   Cond. No.                         74.7
==============================================================================
```

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=0)
```

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:                    Age   R-squared:                       0.547
Model:                            OLS   Adj. R-squared:                  0.546
Method:                 Least Squares   F-statistic:                     539.6
Date:                Mon, 31 Mar 2025   Prob (F-statistic):               0.00
Time:                        18:44:10   Log-Likelihood:                -6906.9
No. Observations:                3131   AIC:                         1.383e+04
Df Residuals:                    3123   BIC:                         1.388e+04
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.5838      0.307     18.177      0.000       4.981       6.186
Sex_I         -0.9069      0.103     -8.765      0.000      -1.110      -0.704
Diameter      10.6033      1.129      9.391      0.000       8.390      12.817
Height         7.8320      1.634      4.793      0.000       4.628      11.036
Whole_weight   9.0179      0.841     10.717      0.000       7.368      10.668
Shucked_weight -19.8830    0.937    -21.212      0.000     -21.721     -18.045
Viscera_weight -10.5777    1.490     -7.097      0.000     -13.500      -7.656
Shell_weight   9.6087      1.292      7.435      0.000       7.075      12.143
==============================================================================
Omnibus:                      655.913   Durbin-Watson:                   1.953
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1701.658
Skew:                           1.125   Prob(JB):                         0.00
Kurtosis:                       5.826   Cond. No.                         74.0
==============================================================================
```

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

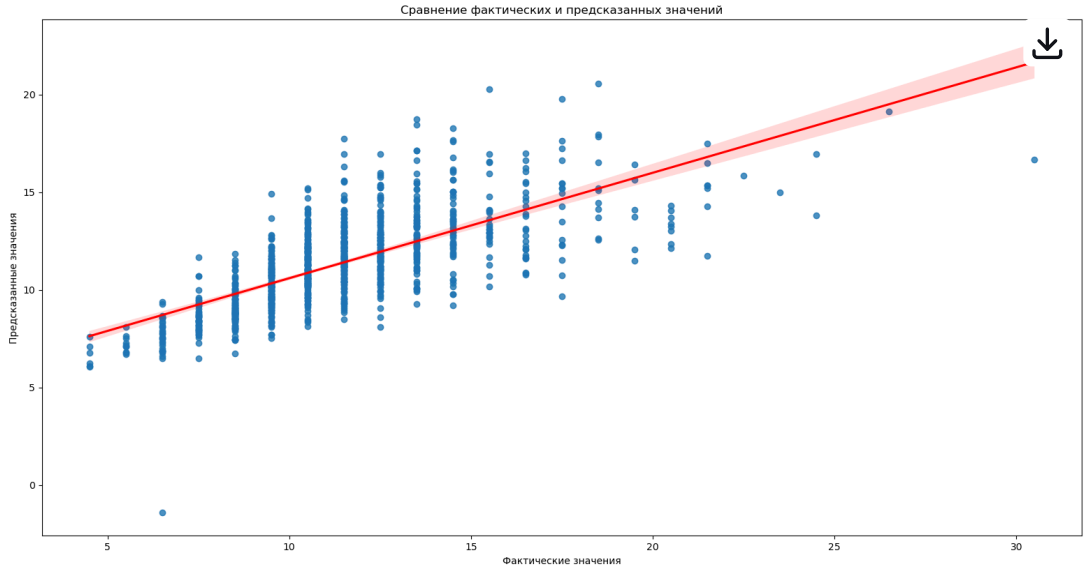
```
Первые 10 зависимых переменных (тестовая выборка)
3546    10.5
1359    12.5
2892     9.5
638     15.5
2657     9.5
1804    10.5
1082     8.5
2054    10.5
2200    26.5
192     15.5
```

```
Первые 10 зависимых переменных (предсказание)
3546     9.690821
1359    10.514368
2892    11.059250
638     10.154525
2657    10.201300
1804    11.984611
1082     8.832172
2054     9.501948
2200    19.145750
192     15.941204
```

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.