Лабораторная работа №6. Построение пайплайна одномерной регрессии

Цель лабораторной работы: разработка единого пайплайна для решения задачи регрессии.

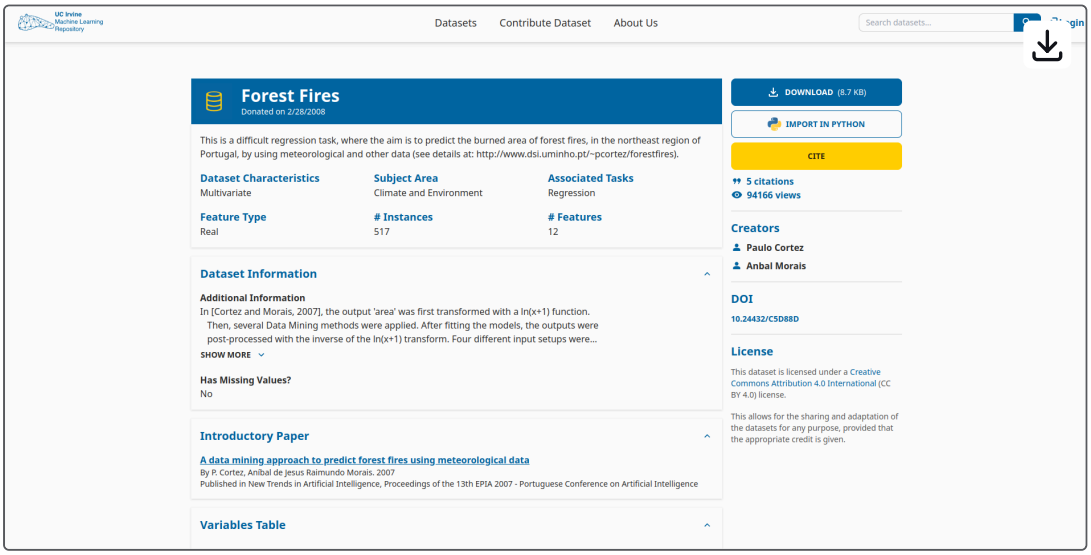
Основные задачи:

- реализовать конвейер для выполнения всех стадий обработки данных при решении задачи одномерной регрессии;
- получение теоретических представлений о задаче регрессии;
- получение навыков использования пайплайна при решении задачи машинного обучения;
- получение навыков рефакторинга кода в задачах машинного обучения.

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 регрессии на основе уникального пайплайна

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
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression


data_path = "https://archive.ics.uci.edu/ml/machine-learning-databases/for

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

Так как у нас стоит задача построить модель **одномерной регрессии**, оставим только один признак — **temp** (температура).

```
data = data[['temp', 'area']]

print(data.head())
```

	temp	area
0	8.2	0.0
1	18.0	0.0
2	14.6	0.0
3	8.3	0.0
4	11.4	0.0

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

```
X = data.iloc[:, :-1].values
y = data.iloc[:, 1].values

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

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

```
Матрица признаков
[[ 8.2]
 [18. ]
 [14.6]
 [ 8.3]
 [11.4]
 [22.2]
 [24.1]
 [ 8. ]
 [13.1]
 [22.8]]

Зависимая переменная
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

2.2 Разделение выборки на тестовую и тренировочную. Обучение модели

Для **улучшения** работы алгоритмов машинного обучения с данным набором данных рекомендуется применить **логарифмическое преобразование** для целевой переменной, которая находится в массиве `y`. Для этого воспользуемся строчкой `y = np.log1p(y)`.

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

```
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    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**:

```
regressor = LinearRegression()
regressor.fit(X_train, y_train)

y_pred = regressor.predict(X_test)

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

Выведем полученные результаты:

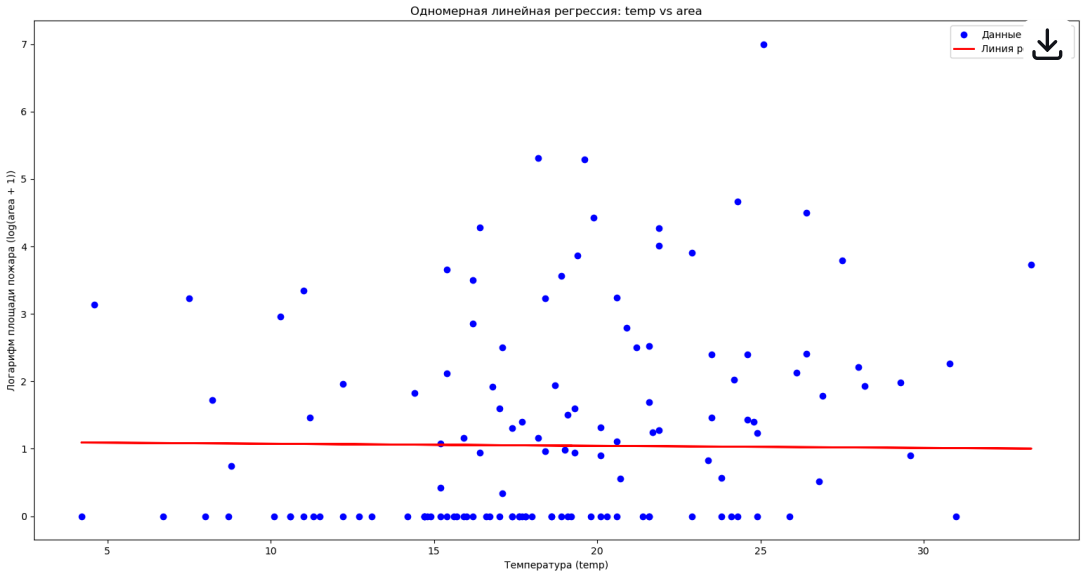
```
Первые 10 зависимых переменных (тестовая выборка)
[3.34462703 0.         1.1568812  4.49412689 1.30562646 0.
 1.07840958 0.         2.8524391  0.        ]

Первые 10 зависимых переменных (предсказание)
[1.07174899 1.03229503 1.05664552 1.02428095 1.05202201 1.03506914
 1.05880316 1.04215852 1.05572082 1.06650901]
```

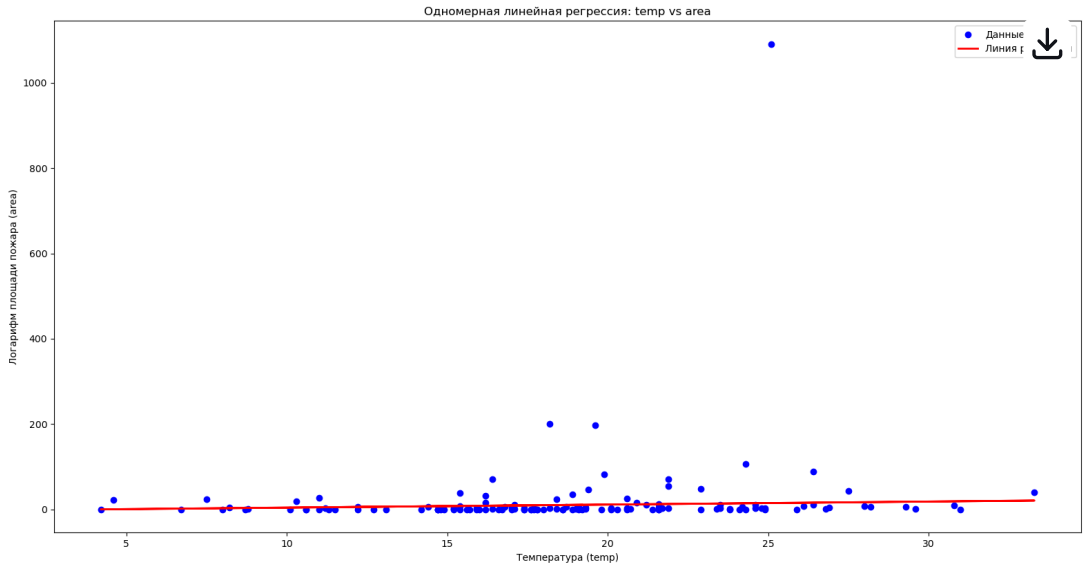
2.3 Визуализация результатов

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

```
plt.scatter(X_test, y_test, color='blue', label='Данные')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Линия регрессии'
plt.xlabel('Температура (temp)')
plt.ylabel('Логарифм площади пожара (log(area + 1))')
plt.title('Одномерная линейная регрессия: temp vs area')
plt.legend()
```



Также выведем аналогичный график, но только **без** логарифмического преобразования целевой переменной `area`.



Контрольные вопросы

1. Масштабирование признаков (например, нормализация или стандартизация) не является строго обязательным для линейной регрессии, потому что модель линейной регрессии сама по себе обучает коэффициенты для каждого признака.
2. Квадратичное отклонение (Mean Squared Error, MSE) используется чаще, потому что:
   - Оно дифференцируемо, что позволяет использовать методы оптимизации, такие как градиентный спуск.
   - Оно сильнее штрафует большие отклонения, что делает модель более чувствительной к выбросам (что может быть как плюсом, так и минусом).
   - Оно имеет аналитическое решение (через метод наименьших квадратов), что упрощает вычисления.

   Модуль отклонения (Mean Absolute Error, MAE) менее чувствителен к выбросам, но он не дифференцируем в нуле, что усложняет оптимизацию.
3. Метод `fit(X, y)` вычисляет коэффициенты линейной регрессии по формуле:

   $$w = (X^T X)^{-1} \cdot X^T \cdot y$$

   где X — матрица признаков, y — вектор значений целевой переменной, а w — вектор коэффициентов модели.
4. p-значение (p-value) — это вероятность того, что коэффициент при данном признаке равен нулю (нулевая гипотеза). В регрессионном анализе p-значение помогает определить значимость признаков: если p-значение меньше 0.05, признак считается статистически значимым.
5. Метод `predict(X)` использует вычисленные коэффициенты модели для предсказания значений целевой переменной по формуле:

   $$\hat{y} = X \cdot w$$
6. `plot()` используется для построения линейных или гладких графиков, а `scatter()` используется для отображения точек на графике (например, точек выборки).
7. Оценивать точность модели необходимо по **тестовой выборке**, так как она содержит данные, которые не использовались при обучении. Это позволяет объективно оценить способность модели к обобщению.