Десислава Милушева

ф.н.: 791323021

Индивидуално задание по

“Програмиране на екосистеми за интернет на нещата”

Ridge Regression

Реализацията на проекта:

[desi109/predictive-modeling-of-air-pollutants-using-ridge-regression.git]

1. Ridge Regression алгоритъм

Ridge Regression е метод за регресионен анализ, който се използва за предсказване на зависимата променлива (в случая концентрациите на различни замърсители във въздуха) въз основа на независимите променливи (в случая метеорологични параметри като температура, налягане и влажност).

Този метод е разновидност на линейната регресия, където се минимизира сумата на квадратите на разликите между реалните и предсказаните стойности на зависимата променлива, като се добавя и допълнителен член за регуляризация на модела. Този член помага за ограничаване на големината на коефициентите на модела и намалява риска от overfitting, като се предотвратява твърде голямо приспособяване към обучителните данни.

С Ridge Regression, параметърът alpha контролира нивото на регуляризация: по-голямата стойност на alpha води до по-силно ограничение на коефициентите и по-малко наклонени към overfitting. Въпреки това, трябва да се отбележи, че по-голямата стойност на alpha може да доведе и до по-голямо

намаление на точността на модела, така че е важно да се избере подходяща стойност за alpha при обучението на модела.

В крайна сметка, Ridge Regression е полезен метод за регресионен анализ, който предоставя баланс между ниската сложност на модела и предотвратяването на overfitting, което го прави подходящ за различни приложения, включително прогнозиране на концентрациите на замърсители във въздуха.

2. Параметри на Ridge Regression и тяхното значение

Параметрите на Ridge Regression и тяхната роля са следните:

- **alpha** (float, по подразбиране=1.0)

Този параметър е основният параметър на Ridge регресията и контролира силата на регуларизацията (штрафа) върху големите стойности на коефициентите. По-голямата стойност на alpha води до по-силно намаляване на стойностите на коефициентите. Обикновено стойностите на alpha се избират в интервал от 0 до положителни стойности, като често се използват логаритмични стъпки. Например, може да изпробвате стойности като 0.1, 1, 10 и т.н. чрез кръстосана валидация за да определите най-подходящата стойност за вашия модел и данни.

- **fit_intercept** (bool, по подразбиране=True)

Този параметър контролира дали моделът ще има интерсепт (константа) или не. Ако е зададен на True (по подразбиране), моделът ще има константа. В случай, че вашите данни вече са центрирани, може да го настроите на False.

- **solver** (string, по подразбиране='auto')

Параметърът solver определя алгоритъма, който се използва за решаване на проблема на оптимизацията при обучението на модела. За Ridge регресията, scikit-learn предлага няколко възможности за solver, като 'auto', 'svd', 'cholesky', 'lsqr' и 'sparse_cg'. По подразбиране, solver е настроен на 'auto', който автоматично избира подходящия solver в зависимост от типа на данните и размера на проблема.

- **max_iter** (int, по подразбиране=None)

Този параметър контролира максималния брой итерации, които алгоритъмът ще използва за оптимизацията. Ако оптимизацията не се сходим до зададения брой итерации, може да се възникне съобщение за грешка. По подразбиране max_iter е настроен на None, което означава, че алгоритъмът ще продължи до сходимост.

- **tol** (float, по подразбиране=1e-3)

Този параметър представлява критерий за спиране на оптимизацията. Алгоритъмът ще спре, ако намерената промяна в коефициентите на модела е по-малка от тази стойност. По подразбиране, tol е настроен на 1e-3, но можете да го настройте на по-малка стойност, ако желаете по-прецизна оптимизация.

3. Код и реализация

ReadData.py

- **Импорт на библиотеки и настройка на пътя към данните**

Импортират се библиотеките os и pandas, като им се дават псевдоними os и pandas съответно. Получава се текущата директория с os.getcwd(). Създава се път до директорията за данните чрез os.path.join(current_dir, "data"). Сменя се работната директория на тази за данните с os.chdir(data_dir).

```
import os as os
import pandas as pandas

# Задаване на текущата директория и път до папката с данни
current_dir = os.getcwd()
data_dir = os.path.join(current_dir, "data")
os.chdir(data_dir)
```

- **Зареждане на данните от Excel файловете**

Файловете 'day.xls' и 'hour.xls' се четат в отделни DataFrames с помощта на pandas.read_excel(). За 'day.xls' се указват имената на листовете за зареждане с параметъра sheet_name=sheet_list. За 'hour.xls' се използват стандартния хедър и колоните се избират с usecols={"Date", "NO", "NO2", "AirTemp", "Press", "UMR"}. Параметърът dtype се използва за дефиниране на типовете данни на колоните.

```
# Списък с имена на листовете от Excel файловете
sheet_list = ['pavlovo', 'drujba', 'hipodruma']

# Зареждане на данни от Excel файловете
dayFile = pandas.read_excel('day.xls',
                            sheet_name=sheet_list,
                            header=1,
                            usecols="A:C",
                            dtype={"Date": 'string', "O3": float, "PM10": float})
hourFile = pandas.read_excel('hour.xls',
                             sheet_name=sheet_list,
                             header=1,
                             usecols={"Date", "NO", "NO2", "AirTemp", "Press", "UMR"},
                             dtype={"Date": 'string', "NO": float, "NO2": float,
                                    "AirTemp": float, "Press": float,"UMR": float})
```

- **Обработка на данните и запис на обработените данни**

За всеки лист в 'day.xls' се обхожда DataFrame и за всяка редица се взима датата, O3 и PM10. След това се избира съответния лист от 'hour.xls' чрез името на листа и се филтрират редовете, които съдържат датата. После се добавят O3 и PM10 към този DataFrame и се прибавя към основния DataFrame merged_df.

DataFrame merged_df се изчиства от редове със NaN стойности с помощта на dropna(). Стойностите в DataFrame се закръглят с round() метода. Обработеният

DataFrame се записва в CSV файл с помощта на to_csv() метода на pandas. Последно, обработеният DataFrame се отпечатва на конзолата с print().

```
# Обработка на данните и сливане на DataFrame обектите
merged_df = pandas.DataFrame()
for sheet_name, daySheet in dayFile.items():
    for i in range(daySheet["Date"].size):
        date = daySheet.iloc[i]["Date"]
        O3 = daySheet.iloc[i]["O3"]
        PM10 = daySheet.iloc[i]["PM10"]

        hourSheet = hourFile[sheet_name]

        filtered_hour = hourSheet[hourSheet["Date"].str.contains(date)]
        filtered_hour.insert(2, "O3", O3)
        filtered_hour.insert(3, "PM10", PM10)

        merged_df = merged_df._append(filtered_hour, ignore_index=True)

# Почистване на DataFrame от редове с NaN стойности
merged_df = merged_df.dropna()

# Закръгляне на стойностите в DataFrame
rounded_merged_df = merged_df.round({"NO": 2, "NO2": 2, "O3": 2, "PM10": 2,
"AirTemp": 1, "UMR": 1, "Press": 0})

# Запис на обработените данни в CSV файл
csv_data = os.path.join(current_dir, "data.csv")
rounded_merged_df.to_csv(csv_data, columns=["NO", "NO2", "AirTemp", "Press",
"UMR", "O3", "PM10"])

# Отпечатване на обработените данни
print(rounded_merged_df)
```

TrainData.py

- **Импорт на необходимите библиотеки и зареждане на данни**

Импортират се библиотеките, които ще бъдат използвани в скрипта. pandas се използва за работа с данни, а Ridge, train_test_split и RobustScaler са части от библиотеката sklearn, която се използва за машинно обучение. joblib се използва за запазване на обекти и модели.

Функцията pandas.read_csv() се използва за зареждане на данни от CSV файл в DataFrame обект. Този DataFrame ще съдържа всички данни, които ще бъдат използвани за обучение и тестване на модела.

```
# Зареждане на необходимите библиотеки
import pandas as pandas
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import RobustScaler
import joblib as joblib
import numpy as np

# Зареждане на данни
data = pandas.read_csv("data.csv")
```

- **Разделяне на данните на признаци и целева променлива**

От заредения DataFrame се избират отделните колони, които ще служат като признаци (features) и целева променлива (target). В случая признаците са колоните 'AirTemp', 'Press' и 'UMR', а целевата променлива са колоните 'NO', 'NO2', 'O3' и 'PM10'.

```
# Разделяне на данните на признаци и целева променлива
x = data[['AirTemp', 'Press', 'UMR']]
y = data[['NO', 'NO2', 'O3', 'PM10']]
```

- **Разделяне на данните на обучителен и тестов набор**

Функцията train_test_split() се използва за разделяне на данните на обучителен и тестов набор. Обикновено се използва 70-80% от данните за обучение и 20-30% за тестване.

```
# Разделяне на данните на обучителен и тестов набор
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)
```

- **Скалиране на признаците и запазване на скалиращия обект**

Признаците се скалират, за да бъдат в еднакъв мащаб. Това е важно за множество алгоритми за машинно обучение. В този код се използва RobustScaler, който е устойчив на аномалии в данните.

Скалиращият обект (scaler) се запазва във файл, за да може по-късно да бъде използван за скалиране на нови данни.

```
# Скалиране на признаците
scaler = RobustScaler()
scaler.fit(x_train)
x_train_scaled = scaler.transform(x_train)
x_test_scaled = scaler.transform(x_test)

# Запазване на скалиращия обект
joblib.dump(scaler, 'scaler.joblib')
```

- **Обучаване на модела за Ridge регресия и запазването му**

Създава се модел за Ridge регресия, който се обучава (fit) върху обучителния набор от данни. Ridge регресията е вид линейна регресия, който е подходящ за работа с данни, които имат мултиколинеарност.

Обученият модел се запазва във файл с цел по-късно да може да бъде зареден и използван за предсказване на нови данни.

```
# Обучаване на модела за Ridge регресия
clf = Ridge(alpha=1.0,
            solver='auto',
            fit_intercept=True,
            copy_X=True,
            max_iter=None,
            tol=0.001)
clf.fit(x_train_scaled, y_train)

# Запазване на обучения модел
joblib.dump(clf, 'clf.joblib')
```

- **Оценка на модела върху тестовите данни и пример за предсказване с модела**

Моделът се оценява върху тестовите данни, за да се види как се справя с данни, които не е виждал по време на обучението.

Използва се обученият модел за предсказване на резултати върху нови данни. В случая се използва примерен набор от признаци, които са скалирани преди да бъдат подадени на модела.

```
# Оценка на модела върху тестовите данни
print("Score: ", clf.score(x_test_scaled, y_test))

# Пример за предсказване с модела
arr = np.array([[25, 955, 80]])
arr_scaled = scaler.transform(arr)
print("Predict: ", clf.predict(arr_scaled))
```

StartProject.py

- **Зареждане на обучения модел и скалиране**

Зареждат се обученият модел и скалиращият обект от запазените файлове.

```
# Зареждане на необходимите библиотеки
from tkinter import *
from tkinter import messagebox
from joblib import load
import numpy as np

# Зареждане на модела Ridge и скалиране
regr = load("clf.joblib")
scaler = load("scaler.joblib")
```

- **Създаване на графичен интерфейс с tkinter**

Създава се графичен интерфейс с помощта на tkinter, включващ етикети за въвеждане на данни и полета за въвеждане на данни.

```
# Създаване на графичен интерфейс
root = Tk()
root.title('Predict Value')
root.resizable(0, 0)

# Създаване на етикети и полета за въвеждане на данни
tempL = Label(root, text="Temp(C)")
pressL = Label(root, text="Press(hPa)")
humL = Label(root, text="Hum(%)")

noL = Label(root, text="NO(ug/m3)")
no2L = Label(root, text="NO2(ug/m3)")
ozoneL = Label(root, text="Ozone(ug/m3)")
pm10L = Label(root, text="PM10(ug/m3)")

nonormL = Label(root, text="NO/NO2 norm is:%d" % 200)
o3normL = Label(root, text="O3 norm is:%d" % 200)
```

```
pm10normL = Label(root, text="PM10 norm is:%d" % 50)

tempE = Entry(root)
pressE = Entry(root)
humE = Entry(root)
noE = Entry(root)
no2E = Entry(root)
ozoneE = Entry(root)
pm10E = Entry(roo)

# Поставяне на етикетите и полетата в грида
tempL.grid(row=0)
tempE.grid(row=0, column=1)

pressL.grid(row=1)
pressE.grid(row=1, column=1)

humL.grid(row=2)
humE.grid(row=2, column=1)

noL.grid(row=3)
noE.grid(row=3, column=1)
nonormL.grid(row=3, column=2)

no2L.grid(row=4)
no2E.grid(row=4, column=1)

ozoneL.grid(row=5)
ozoneE.grid(row=5, column=1)
o3normL.grid(row=5, column=2)

pm10L.grid(row=6)
pm10E.grid(row=6, column=1)
pm10normL.grid(row=6, column=2)
```

- **Функция за предсказване на замърсители**

Функцията Predict() се използва за предвиждане на стойностите на замърсителите, въведени от потребителя, използвайки обучения модел.

```
# Функция за предсказване
def Predict():
    noE.delete(0, END)
    no2E.delete(0, END)
    ozoneE.delete(0, END)
    pm10E.delete(0, END)
    try:
```

```
        arr = np.array([[float(tempE.get()), float(pressE.get()), float(humE.get())]])
        arr_transformed = scaler.transform(arr)
        pr = regr.predict(arr_transformed)
        noE.insert(0, round(pr[0][0], 2))
        no2E.insert(0, round(pr[0][1], 2))
        ozoneE.insert(0, round(pr[0][2], 2))
        pm10E.insert(0, round(pr[0][3], 2))
    except ValueError:
        messagebox.showinfo("Wrong Value", "Please enter float values!")
```

- **Бутон за предсказване**

Бутонът "Predict" при натискане извиква функцията за предсказване.

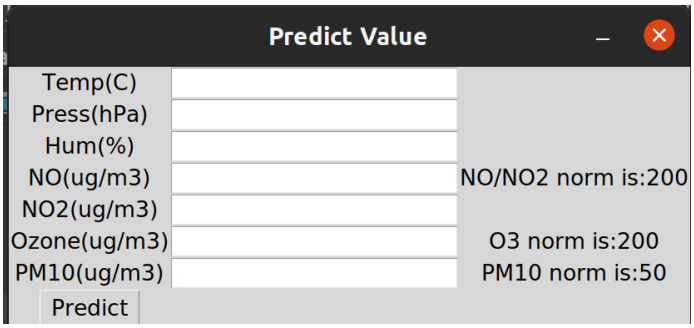
```
# Бутон за предсказване
b = Button(root, text="Predict", command=Predict)
b.grid(row=7)
```

- **Стартиране на графичния интерфейс**

Използва се root.mainloop(), за да се стартира главният цикъл на графичния интерфейс.

```
# Стартиране на графичния интерфейс
root.mainloop()
```

Това е общият процес, който се извършва с всеки отделен блок от кода. След като всички стъпки се изпълнят, графичният интерфейс се показва на екрана, където потребителят може да стартира тестването с различни стойности за параметъра alpha.



4. Тестване на резултатите и анализ

Анализът на получените данни и модели, използвани за прогнозиране, предоставя важна информация за способността на Ridge Regression да предсказва концентрациите на замърсители във въздуха въз основа на някои метеорологични параметри.

Получен файл след обработка на данните:

	<null>	NO	NO2	AirTemp	Press	UMR	O3	PM10
2	0	-1.39	13.92	-1.4	941.0	73.7	44.79	34.65
3	1	-1.35	11.51	-1.4	941.0	72.9	44.79	34.65
4	2	-1.24	11.46	-1.5	942.0	72.3	44.79	34.65
5	3	-0.96	9.69	-1.6	942.0	73.0	44.79	34.65
6	4	-0.77	10.8	-1.6	943.0	72.1	44.79	34.65
7	5	-0.56	9.22	-1.6	943.0	70.5	44.79	34.65
8	6	-0.5	9.38	-1.9	944.0	70.5	44.79	34.65
9	7	-0.28	18.11	-2.7	944.0	73.1	44.79	34.65
10	8	0.42	16.5	-3.7	945.0	74.5	44.79	34.65
11	9	1.62	14.2	-2.8	946.0	70.0	44.79	34.65
12	10	1.24	9.18	-0.8	946.0	64.6	44.79	34.65
13	11	0.98	7.07	1.1	946.0	59.9	44.79	34.65
14	12	0.82	6.53	1.6	946.0	58.0	44.79	34.65
15	13	0.7	6.53	1.8	945.0	55.9	44.79	34.65
16	14	1.24	7.83	2.0	945.0	54.2	44.79	34.65
17	15	0.89	8.15	1.5	946.0	56.2	44.79	34.65
18	16	1.92	18.06	0.4	946.0	61.2	44.79	34.65
19	17	7.71	46.09	-0.7	947.0	65.9	44.79	34.65
20	18	0.95	46.72	-1.7	947.0	66.9	44.79	34.65
21	19	0.23	36.32	-2.3	948.0	64.6	44.79	34.65
22	20	0.26	29.44	-2.6	948.0	62.7	44.79	34.65
23	21	35.36	54.93	-3.1	948.0	62.9	44.79	34.65
24	22	44.44	62.41	-4.2	949.0	67.0	44.79	34.65
25	23	26.32	61.39	-4.7	949.0	68.9	44.79	34.65

```
Score:  0.15073594904972695
```

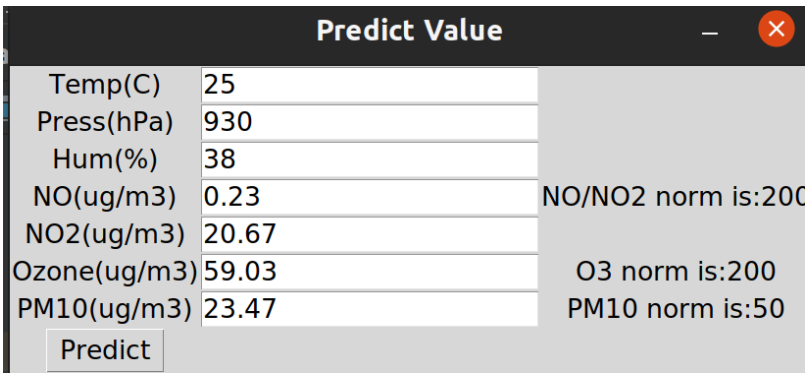
Оценката на модела е 0.15, което предполага, че Ridge Regression обяснява само около 15% от дисперсията в целевата променлива. Този нисък резултат може да означава, че моделът не е достатъчно точен или не е способен да обясни по-голямата

част от вариацията в данните. Причините могат да бъдат недостатъчните независими променливи, неправилното разделяне на данните, или линейната природа на модела.

```
Predict:  [[ 6.20537336 20.25348744 56.62207203 22.99186247]]
```

Този масив от стойности представлява предсказанията за замърсителите NO, NO2, Ozone и PM10 съответно. Въпреки че оценките изглеждат в нормални граници, ниската оценка на модела може да намекне за риск от неточност или случайност в резултатите.

Всяка стойност представлява оценката на модела за съответната концентрация на замърсяващия вещество във въздуха (в ug/m3) въз основа на входните характеристики (температура, налягане, влажност).



Резултатите от предсказанията след въвеждането на стойности в интерфейса на приложението показват следното:

1. NO (Азотен оксид):

- Получената стойност: 0.23 ug/m3
- Норма: 200 ug/m3
- Анализ: Стойността на NO е значително по-ниска от нормата от 200 ug/m3, което показва, че нивата на този замърсител във въздуха са далеч под допустимите граници. Това е положителен резултат, който указва на добро качество на въздуха по отношение на азотния оксид.

2. NO2 (Азотен диоксид):

- Получената стойност: 20.67 ug/m3
- Норма: 200 ug/m3

- Анализ: Стойността на NO2 също е значително по-ниска от нормата от 200 ug/m3. Това е добър резултат, който показва, че нивата на азотен диоксид са под контрол и са в приемливите граници.

3. O3 (Озон):

- Получената стойност: 59.04 ug/m3
- Норма: 200 ug/m3
- Анализ: Озонът също е под нормата от 200 ug/m3, което означава, че той също е в приемливите граници. Това е положителен резултат за качеството на въздуха по отношение на озона.

4. PM10 (Частици с диаметър до 10 микрона):

- Получената стойност: 23.47 ug/m3
- Норма: 50 ug/m3
- Анализ: Стойността на PM10 е също под нормата от 50 ug/m3, което е положителен резултат и указва на ниско ниво на частиците във въздуха.

4.1. Анализ

Общият анализ показва, че всички получени стойности за замърсителите във въздуха са значително под допустимите нива. Това е добър знак за качеството на въздуха и е индикация, че моделът може да предостави полезна информация за нивата на замърсяване в определената област.

- Ниска оценка на модела:

  Ниската оценка от 0.15 може да означава, че моделът не успява да улови всички зависимости в данните. Това може да се дължи на избрания тип модел (линеен), липсата на достатъчно независими променливи, или неправилното обработване на данните.

- Линейност на модела:

  Ridge Regression е линеен модел, който може да има ограничения при справянето с нелинейни зависимости. Ако в данните има сложни взаимоотношения, моделът може да бъде неефективен.

- Подобрения:

  Увеличаване на сложността на модела - може да бъде полезно да се опитат по-сложни модели като Random Forest или Gradient Boosting, които могат да уловят нелинейни зависимости.

  Увеличаване на данните за обучение - добавянето на повече данни за обучение или включването на повече характеристики може да подобри точността на модела.

5. Заключение от направените тестове и анализи

Заключението относно точността на модела, базиран на Ridge Regression, разкрива някои предизвикателства в способността му за предсказване. Въпреки че резултатите от предсказанията са в рамките на допустимите нива за съответните замърсители във въздуха, като NO, NO2, озон и PM10, точността на тези предсказания не е особено висока. Със score от около 0.15, моделът обяснява само 15% от изменчивостта в данните. Това ниско ниво на обяснителна способност сигнализира, че има вероятност от недостатъчно добри предсказания и че трябва да се подходи с предпазливост към резултатите от този модел.

Една от възможните причини за ограничената точност може да е структурата на самите данни. Ако има сложни нелинейни зависимости между признаците и целевите променливи, Ridge Regression може да не е най-подходящият модел. Също така, ако обемът на данните за обучение е недостатъчен или няма достатъчно разнообразие в данните, това също може да допринесе за ограничената точност на модела.

За да се подобри точността на предсказанията и да се постигнат по-надеждни резултати, може да се използват по-сложни модели, като Decision Trees, Random Forests или дори Neural Networks. Тези модели могат да уловят по-сложни зависимости между данните и целевите променливи, което би повишило обяснителната способност и точността на предсказанията.

Освен използването на по-сложни модели, увеличаването на обема на данните за обучение може да помогне за подобряване на точността. Повече данни могат да дадат на модела по-голям набор от примери, което увеличава неговата способност да обобщава правилно. В допълнение, добрата практика включва валидиране на модела

чрез сравнение на предсказанията с реални стойности. Това помага да се установи доколко предсказанията на модела съответстват на действителността и дали са надеждни.

В заключение, макар моделът с Ridge Regression да е постигнал някои приемливи предсказания, той не е достатъчно надежден за прецизни оценки. Необходим е по-задълбочен анализ на данните и използване на по-съвременни методи за моделиране, както и повече данни за обучение, за да се постигнат по-точни и надеждни предсказания.