Analiza danych na temat uchowców

Piotr Kuśnierz

Mai 2022

Spis treści

1 Wstęp

Przedmiotem poddanym analizie jest zbiór danych dotyczących uchowców (Abalone dataset). Celem analizy jest określenie wieku uchowca na podstawie łatwo dostępnych pomiarów fizycznych. Zastosowaną metodą była wielokrotna regresja liniowa.

2 Dobór danych i ich analiza

2.1 Dobór danych

Dane dotyczące uchowców zostały pobrane ze strony
https://archive.ics.uci.edu/ml/datasets/Abalone.

2.2 Opis kontekstu danych

Aby określić wiek uchowca rozcina się muszę, a następnie pod mikroskopem sprawdza się liczbę kręgów. Aby ominąć ten żmudny proces przewidujemy wiek uchowca na podstawie innych pomiarów fizycznych, które są prostsze i szybsze do pobrania.

2.3 Opis danych

Dane zawierają 8 atrybutów:

1. Sex (płeć)
2. Length (najdłuższy pomiar muszli)
3. Diameter (obwód)
4. Height (wysokość wraz z mięsem w środku)
5. Whole weight (waga całego uchowca)
6. Shucked weight (waga mięsa)
7. Viscera weight (waga wnętrzności)
8. Shell weight (waga muszli po opróżnieniu i osuszeniu)

Wartość predykowana to Rings, liczba pierścieni która po dodaniu 1.5 daje wiek uchowca w latach.

2.4 Pre-processing danych

Problem brakujących wartości został rozwiązany przed umieszczeniem danych na stronie, poprzez usunięcie wybrakowanych linijek. Podobnie dane o ciągłych wartościach zostały przeskalowane do użytku, poprzez podzielenie przez 200. Przed przystąpieniem do tworzenia modelu należy rozwiązać jeszcze kilka problemów:

1. Kolumny w danych źródłowych nie posiadają nazw.
   Dla prostszej analizy danych oraz przejrzystości kodu, dodaję ręcznie nazwy kolumn

2. Pojawiają się wartości nienumeryczne.
   Kolumna Sex zawiera jedeną z trzech wartości: F, M lub I, którą na potrzeby modelu regresji liniowej należy zamienić na liczbę. W tym celu wykorzystuję metodę biblioteki pandas get_dummies, zwracającą trzy kolumny (F, I, M) z przypisanymi wartościami 0 oraz 1, oznaczającymi która wartość występowała w kolumnie Sex. Ponieważ możemy skrócić ten zapis do postaci: 10 - I, 01 - M oraz 00 - F (czyli kolumna F nie jest potrzebna) omijamy kolumnę F parametrem drop_first=True.

3. Ostatnim zabiegiem pre-processingu jest pozbycie się zbędnych dla modelu kolumn.
   W tym celu wykorzystuję metodę biblioteki sklearn RFE, która zwraca listę kolumn opracowywanego modelu z podziałem na niezbędne oraz zbędne wartości:

```
('Length', False, 4)
('Diameter', True, 1)
('Height', False, 2)
('Weight', True, 1)
('Shucked weight', True, 1)
('Viscera weight', True, 1)
('Shell weight', False, 3)
('I', False, 5)
('M', False, 6)
```

Rysunek 1: wartości RFE dla wszystkich zmiennych

Pierwsza kolumna to nazwa zmiennej, druga to określenie czy niezbędna dla opracowywanego modelu, a trzecia to ranga (im mniejsza wartość tym bardziej niezbędna dana zmienna dla modelu). W dalszej części programu, do zbudowania modelu wykorzystam tylko wartości opisane jako niezbędne.

2.5 Wstępna analiza

Wstępną analizę danych przeprowadzam przy użyciu biblioteki pandas-profiling oraz metody ProfileReport. Następnie dla wygody zapisuję raport do pliku Report.html (plik dostępny po pierwszym uruchomieniu programu, nie generuje się jeśli już istnieje w systemie).

3 Rozwiązanie problemu

3.1 Opis problemu

Rozwiązywany problem to stworzenie modelu w celu przewidywania wieku uchowca na podstawie prostych pomiarów fizycznych, zamiast zwyczajowej metody rozcinania muszli i badania pod mikroskopem.

3.2 Podział danych na zestaw treningowy oraz testowy

Dane po zaciągnięcu datasetu z pliku oraz przeprowadzeniu początkowego preprocessingu, należy rozdzielić na zestaw treningowy i testowy. W moim przypadku rozdzielam dane w proporcji 7:3, bez losowości (przy każdym włączeniu programu zestaw treningowy i testowy będą identyczne jak przy poprzednich włączeniach). W tym celu ustawiam ziarno losowości biblioteki numpy na 0, a następnie używam metody train_test_split biblioteki sklearn.

3.3 Stworzenie modelu

Po podziale danych oraz pozbyciu się niepotrzebnych kolumn, prechodzę do stworzenia modelu. W tym celu dodaje stałą, przy pomocy funkcji add_constant biblioteki statsmodels.api. Następnie tworzę model przy użyciu funkcji OLS (Ordinary Least Squares) biblioteki statsmodels.api, jest to metoda najmniejszych kwadratów, oraz metody fit aby obliczyć parametry modelu liniowego. Przy użyciu metody summary otrzymuje podsumowanie stworzonego modelu w postaci

OLS Regression Results

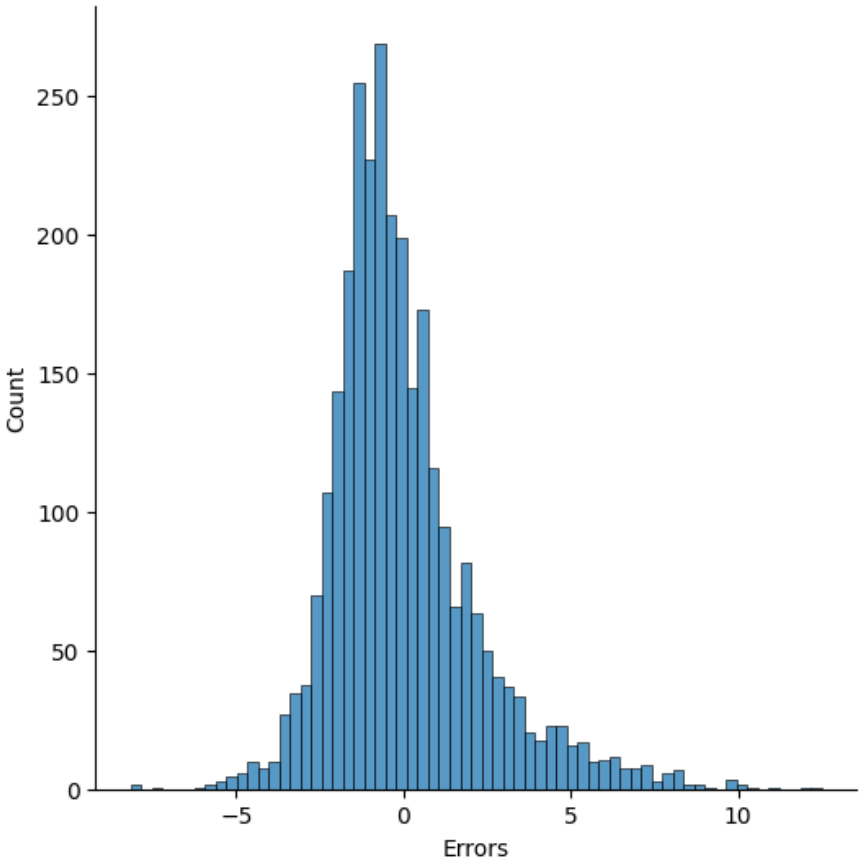
Dep. Variable:	Rings	R-squared:	0.516
Model:	OLS	Adj. R-squared:	0.516
Method:	Least Squares	F-statistic:	778.7
Date:	Mon, 23 May 2022	Prob (F-statistic):	0.00
Time:	15:10:13	Log-Likelihood:	-6524.0
No. Observations:	2923	AIC:	1.306e+04
Df Residuals:	2918	BIC:	1.309e+04
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3.0766	0.293	10.490	0.000	2.502	3.652
Diameter	15.1575	1.115	13.597	0.000	12.972	17.343
Weight	14.8985	0.524	28.416	0.000	13.870	15.927
Shucked weight	-25.1624	0.759	-33.162	0.000	-26.650	-23.675
Viscera weight	-14.3869	1.484	-9.691	0.000	-17.298	-11.476

Omnibus:	687.648	Durbin-Watson:	2.011
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1828.063
Skew:	1.247	Prob(JB):	0.00
Kurtosis:	5.964	Cond. No.	54.0

3.4 Wykres rozkładu błędów

Przedostatnim krokiem jest wyrysowanie wykresu błędów naszego modelu przy pomocy matplotlib oraz seaborn



Jak możemy zaobserwować na wykresie rozkład błedów jest dość podobny do rozkładu normalnego, co dobrze świadczy o zaimplementowanym modelu.

3.5 R^2 score

R-squared to wynik znajdujący się pomiędzy 0 a 100%, mówiący jak dobrze nasz model jest dopasowany, mówi o tym jaki procent jednej zmiennej wyjaśnia zmienność drugiej. Wyniki dla danych treningowych oraz testowych przedstawiają się następująco

```
train data score: 0.5163101573856821
test data score: 0.504815256887271
```

Tak jak widać różnica pomiędzy wynikiem dla danych treningowych, a da-

nych testowych jest stosunkowo niewielka, czyli nasz model jest dość dobrze dopasowany.

4 Alternatywny model

W alternatywnym modelu do danych dodaje kolumny które w RFE uzyskały 2 lub 3 punkty (2 kolejne wartości nie będące niezbędne dla modelu, dodawane w kolejności podanej przez RFE). Podsumowanie nowo powstałem modelu wygląda następująco

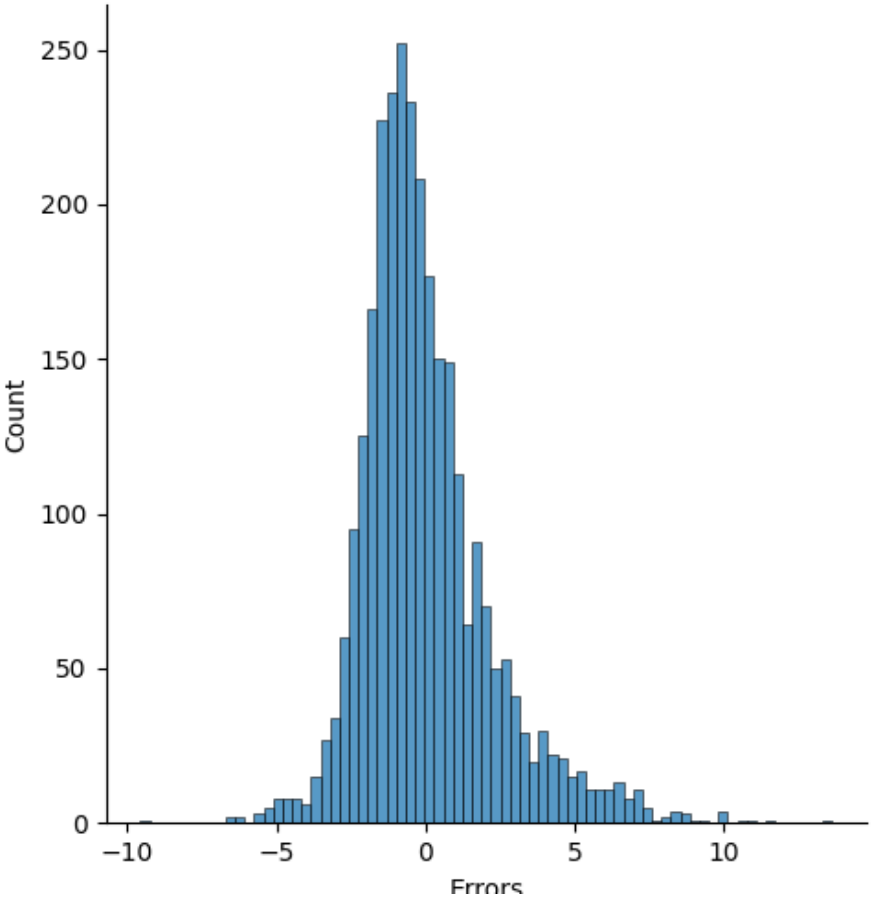
OLS Regression Results

Dep. Variable:	Rings	R-squared:	0.532
Model:	OLS	Adj. R-squared:	0.531
Method:	Least Squares	F-statistic:	551.8
Date:	Mon, 23 May 2022	Prob (F-statistic):	0.00
Time:	21:25:46	Log-Likelihood:	-6476.8
No. Observations:	2923	AIC:	1.297e+04
Df Residuals:	2916	BIC:	1.301e+04
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3.0801	0.297	10.386	0.000	2.499	3.662
Diameter	11.4226	1.167	9.789	0.000	9.135	13.710
Height	10.2328	1.667	6.139	0.000	6.965	13.501
Weight	9.5428	0.872	10.946	0.000	7.833	11.252
Shucked weight	-20.2995	0.976	-20.799	0.000	-22.213	-18.386
Viscera weight	-11.1659	1.544	-7.234	0.000	-14.192	-8.139
Shell weight	9.2031	1.344	6.850	0.000	6.569	11.837

Omnibus:	664.495	Durbin-Watson:	2.019
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1794.653
Skew:	1.200	Prob(JB):	0.00
Kurtosis:	5.996	Cond. No.	71.8

Natomiast nowo powstały wykres błędów tak



Rysunek 2: rozkład błędów alternatywnego modelu

Rozkład błedów alternatywnego modelu nieco bardziej przypomina rozkład normalny, niż oryginalny. Dodanie kolumn do modelu zwiększyło również niznacznie wynik R-squared:

```
train data score: 0.5316894173795403
test data score: 0.5176198615905605
```

5 Wniosek

Zarówno oryginalny jak i alternatywny model są dość dobrze dopasowane, na co wskazują wyniki R^2 score. Diagram błędów w przypadku modelu alternatywnego nieco bardzej przypomina rozkład normalny. Zarówno nieco wyższy R^2 score oraz lepiej dopasowany rozkład błędów, wskazują że model z dobranymi dodatkowymi kolumnami lepiej obrazuje postawiony problem. Jednakże przy większych zestawach danych, dodanie dodatkowych kolumn może spowodować wydłużenie czasu obliczeń, więc należy wziąć pod uwagę czy drobne zmiany w wynikach mają dla nas większe znaczenie niż dłuższy czas obliczeń.