Systemy pomiarowo diagnostyczne 2 – 2016/2017

Zajęcia laboratoryjne 1 – regresja liniowa

Ćwiczenie przygotowane w oparciu o „An Introduction to Statistical Learning", Gareth James, Trevor Hastie, Robert Tibshirani, Daniela Witten.

1. Biblioteki

Funkcja *library()* jest wykorzystywana do załadowywania bibliotek funkcji oraz przykładowych zbiorów danych, które nie znajdują się w podstawowej wersji R. Podstawowe funkcje, które przeprowadzają regresję liniową I inne proste analizy są dostarczane razem z podstawową instalacją R, jednak bardziej egzotyczne funkcje wymagają dodatkowych bibliotek. W trakcie zajęć będziemy korzystać z paczki *MASS,* która jest bardzo dużą kolekcją zbiorów danych i funkcji. Będziemy również korzystać z biblioteki ISLR, która zawiera zbiory wykorzystywane w książce **An Introduction to Statistical Learning**.

```
> library (MASS)

> library (ISLR)
```

Jeżeli po wykonaniu powyższych instrukcji R zwrócił błąd, oznacza to że biblioteki nie są jeszcze zainstalowane na Twoim komputerze. Niektóre biblioteki (jak *MASS*) nie wymagają dodatkowej instalacji. Jednak inne biblioteki, np. ISLR, muszą zostać pobrane z ogólnie dostępnego repozytorium przed pierwszym użyciem. Przykładowo dla systemu Windows należy wybrać opcję Install Package pod zakładką Packages. Po wybraniu serwera źródłowego, pojawi się lista dostępnych bibliotek dodatkowych. Należy po prostu wybrać odpowiednią bibliotekę, a R automatycznie ją pobierze i zainstaluje. To samo można wykonać za pomocą polecenia w linii poleceń: *install.packages(„ISLR").* Instalacja jest przeprowadzana tylko jednokrotnie, później wystarczy załadować bibliotekę za pomocą polecenia *library()*.

2. Prosta regresja liniowa

Biblioteka MASS zawiera zbiór danych Boston, w którym zapisana jest wartość ***medv*** (median house value – mediana wartości domów) dla 506 lokalizacji w obrębie Bostonu. Będziemy próbowali przewidzieć wartość ***medv*** przy wykorzystaniu 13 predykotorów takich jak ***rm*** (average number of rooms per house), ***age*** (average age of houses), i ***lstat*** (percent of households with low socioeconomic status).

```
> names(Boston )

[1] "crim" "zn" "indus" "chas" "nox" "rm" "age"

[8] "dis" "rad" "tax" "ptratio " "black" "lstat" "medv"
```

Aby dowiedzieć się więcej o zbiorze Boston wpisz ***?Boston***.

Zaczniemy od użycia funkcji lm(), aby dopasować prostą model regresji liniowej z medv jako wartością wyjaśnianą i lstat jako predyktorem. Podstawowa składnia wygląda następująco ***lm(y~x,data)*** , gdzie ***y*** jest wartością wyjaśnianą, ***x*** jest predyktorem, a data oznacza zbiór danych w którym znajdują się zmienne.

> lm.fit =lm(medv∼lstat)

Error in eval(expr , envir , enclos) : Object "medv" not found

****

Komenda powoduje błąd, ponieważ R nie zna zmiennych ***medv*** I ***lstat***. Komenda w następnej sekcji pokazuje, gdzie znajdują się obie zmienne. Do danych można również dostać przez operator "$", ponieważ obie zmienne to pola w liście Boston.

> lm.fit =lm(medv∼lstat ,data=Boston)

Wpisanie ***lm.fit*** spowoduje wyświetlenie kilku najważniejszych informacji o uzyskanym modelu. Dla bardziej szczegółowych informacji należy wpisać ***summary(lm.fit)***. Komenda zwróci informację o p-wartościach, błędach standardowych dla współczynników, jak również statystykę R^2 i F-statystykę dla modelu.

> lm.fit

Call:

lm(formula = medv ∼ lstat)

Coefficients:

(Intercept) lstat

34.55 -0.95

> summary (lm.fit)

Call:

lm(formula = medv ∼ lstat)

Residuals :

Min 1Q Median 3Q Max

-15.17 -3.99 -1.32 2.03 24.50

Coefficients:

Estimate Std. Error t value Pr(>|t|)

*(Intercept) 34.5538 0.5626 61.4 <2e-16 ***

lstat -0.9500 0.0387 -24.5 <2e-16 ***

Signif . codes: 0 *** 0.001 ** 0.01 * 0.05 . 0.1 1*

Residual standard error : 6.22 on 504 degrees of freedom

Multiple R-squared : 0.544 , Adjusted R-squared : 0.543

F-statistic : 602 on 1 and 504 DF , p-value: <2e-16

Możemy użyć funkcji ***names()*** żeby dowiedzieć się jakie inne pola znajdują się w zmiennej ***lm.fit.*** Do tych pól możemy dostać się w sposób standardowy np. ***lm.fit$coefficients***, jednak bezpieczniejszym sposobem jest wykorzystanie odpowiednich funkcji np. ***coef().***

> names(lm.fit)

[1] " coefficients" "residuals " "effects "

[4] "rank" "fitted .values " "assign "

[7] "qr" "df.residual " "xlevels "

[10] "call" "terms" "model"

> coef(lm.fit)

(Intercept) lstat

34.55 -0.95

W celu uzyskania przedziałów ufności dla estymowanych współczynników, możemy użyć funkcji ***confint().***

> confint (lm.fit)

2.5 % 97.5 %

(Intercept) 33.45 35.659

lstat -1.03 -0.874

Funkcja predict() może być wykorzystana do obliczenia przedziałów ufności dla wartości średniej i dla przewidywanej wartości ***medv***, dla określonych wartości ***lstat***.

> predict (lm.fit , data.frame(lstat =(c(5 ,10 ,15))),

interval =" confidence ")

fit lwr upr

1 29.80 29.01 30.60

2 25.05 24.47 25.63

3 20.30 19.73 20.87

> predict (lm.fit , data.frame(lstat =(c(5 ,10 ,15))),

interval =" prediction ")

fit lwr upr

1 29.80 17.566 42.04

2 25.05 12.828 37.28

3 20.30 8.078 32.53

Na przykład 95% przedział ufności dla średniej dla wartości lstat 10 wynosi (24.47 25.63), a przedział dla samej predykcji wynosi 12.828 37.28. Tak jak jest to oczekiwane, oba przedziały są wycentrowane wokół tej samej wartości (25.05), jednak drugi przedział jest znacznie szerszy.

Teraz wykreślimy wartości medv i lstat razem z dopasowanym modelem regresji liniowej przy użyciu funkcji ***plot()*** i ***abline()***.

> plot(lstat ,medv)

> abline (lm.fit)

Widać, że związek pomiędzy lstat i medv może być nieliniowy. Ta kwestia będzie zgłębiana później w trakcie laboratorium.

Funkcja abline() może być wykorzystana do narysowania dowolnej linii, nie tylko linii regresyjnej. Aby narysować linię z wartością stałą ***a*** i nachyleniem ***b***, po prostu wpisujemy ***abline(a,b)***. Poniżej wypróbowywane są dodatkowe ustawienia dla wykreślania linii i punktów. Parametr ***lwd=3*** zwiększa trzykrotnie grubość linii. Działa to zarówno dla funkcji ***plot()*** jak i ***lines()***. Opcję ***pch*** można wykorzystać do wykreślania różnych symboli punktów.

> abline (lm.fit ,lwd =3)

> abline (lm.fit ,lwd =3, col ="red ")

> plot(lstat ,medv ,col ="red ")

> plot(lstat ,medv ,pch =20)

> plot(lstat ,medv ,pch ="+")

> plot (1:20 ,1:20, pch =1:20)

W następnym kroku sprawdzamy wykresy diagnozujące dopasowanie. Część z nich omówiona jest w sekcji 3.3.3. książki. Cztery wykresy diagnozujące wykreślane są automatycznie, gdy uruchomimy funkcję ***plot()*** dla obiektu zwracanego przez funkcję ***lm()***. Funkcja w ogólnym przypadku będzie wyświetlała jeden wykres po drugim, po wciśnięciu „ENTER". Jednak dobrze jest popatrzeć na wszystkie wykresy jednocześnie. Możemy to uzyskać za pomocą funkcji ***par(),*** która zleca R podzielenie obszaru wyświetlania na oddzielne panele, tak że możliwe jest jednoczesne oglądanie kilku wykresów. Przykładowo ***par(mfcol=c(2,2))*** dzieli obszar wyświetlania wykresu na siatkę 2x2.

*****

> par(mfrow =c(2,2))

> plot(lm.fit)

Alternatywnie możemy obliczyć wartości resztkowe z dopasowania liniowego za pomocą funkcji ***residuals().*** Funkcja ***rstudent()*** standaryzuje wartości resztkowe i wtedy możemy użyć funkcji ***plot()***, aby wykreślić je w stosunku do dopasowanych wartości **medv**.

***

> plot(predict (lm.fit), residuals (lm.fit))

> plot(predict (lm.fit), rstudent (lm.fit))

W oparciu o wykresy wartości resztkowych można stwierdzić, że w relacji ***medv*** i ***lstat*** jest nieliniowość. Można wyliczyć statystyki lewarowania (ang. Leverage statistics) dla dowolnych punktów za pomocą funkcji ***hatvalues().***

***

> plot(hatvalues (lm.fit))

> which.max (hatvalues (lm.fit))

375

Funkcja which.max() identyfikuje w wektorze indeks elementu o największej wartości. W tym przypadku otrzymujemy informację, który z punktów ma największą wartość statystyki lewarowania.

3. Wielokrotna regresja liniowa

Aby dopasować model regresji liniowej z wykorzystaniem metody najmniejszych kwadratów ponownie używamy funkcji ***lm()***. Składnia lm(y~x1+x2+x3) jest używana żeby dopasować model z trzema predyktorami x1, x2, x3. Funkcja ***summary()*** zwraca współczynniki regresyjne dla wszystkich predyktorów.

> lm.fit =lm(medv∼lstat+age ,data=Boston)

> summary (lm.fit)

Call:

lm(formula = medv ∼ lstat + age , data = Boston)

Residuals :

Min 1Q Median 3Q Max

-15.98 -3.98 -1.28 1.97 23.16

Coefficients:

Estimate Std. Error t value Pr(>|t|)

*(Intercept) 33.2228 0.7308 45.46 <2e-16 ***

*lstat -1.0321 0.0482 -21.42 <2e-16 ***

*age 0.0345 0.0122 2.83 0.0049 **

---

*Signif . codes: 0 *** 0.001 ** 0.01 * 0.05 . 0.1 1*

Residual standard error : 6.17 on 503 degrees of freedom

Multiple R-squared : 0.551 , Adjusted R-squared : 0.549

F-statistic : 309 on 2 and 503 DF , p-value: <2e-16

Zbiór Boston zawiera 13 zmiennych, więc manualne tworzenie formuły modelu dla wszystkich predyktorów byłoby bardzo niewygodne. Zamiast tego możemy skorzystać z uproszczonego zapisu:

> lm.fit =lm(medv∼.,data=Boston)

> summary (lm.fit)

Call:

lm(formula = medv ∼ ., data = Boston)

114 3. Linear Regression

Residuals :

Min 1Q Median 3Q Max

-15.594 -2.730 -0.518 1.777 26.199

Coefficients:

Estimate Std . Error t value Pr(>|t|)

*(Intercept) 3.646e+01 5.103 e+00 7.144 3.28e -12 ***

crim -1.080 e-01 3.286e-02 -3.287 0.001087 **

zn 4.642e-02 1.373e-02 3.382 0.000778 ***

indus 2.056e-02 6.150e-02 0.334 0.738288

chas 2.687e+00 8.616e-01 3.118 0.001925 **

nox -1.777 e+01 3.820 e+00 -4.651 4.25e -06 ***

rm 3.810e+00 4.179e-01 9.116 < 2e -16 ***

age 6.922e-04 1.321e-02 0.052 0.958229

dis -1.476 e+00 1.995e-01 -7.398 6.01e -13 ***

rad 3.060e-01 6.635e-02 4.613 5.07e -06 ***

tax -1.233 e-02 3.761e-03 -3.280 0.001112 **

ptratio -9.527 e-01 1.308e-01 -7.283 1.31e -12 ***

black 9.312e-03 2.686e-03 3.467 0.000573 ***

lstat -5.248 e-01 5.072e-02 -10.347 < 2e -16 ***

Signif . codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error : 4.745 on 492 degrees of freedom

Multiple R-Squared : 0.7406 , Adjusted R-squared : 0.7338*

F-statistic : 108.1 on 13 and 492 DF , p-value: < 2.2e -16

Do poszczególnych elementów obiektu generowanego przez funckję ***summary()*** możemy dostać się w sposób standardowy. (Wpisz ***?summary.lm*** , aby dowiedzieć się co jest wyliczane). Przykładowo summary(lm.fit$r.sq) zwraca statystykę R^2. A summary(lm.fit)$sigma zwróci RSE. Funkcja ***vif()*** z biblioteki ***car*** , może zostać wykorzystana do wyliczenia współczynników VIF – ang. Variance Inflation Factors. Dla tych danych większość wartości VIF jest średnia. Biblioteka ***car*** nie jest częścią podstawowej instalacji R, więc przy pierwszym użyciu będzie wymagała instalacji.

```
> library (car)

> vif(lm.fit)

crim zn indus chas nox rm age

1.79 2.30 3.99 1.07 4.39 1.93 3.10

dis rad tax ptratio black lstat

3.96 7.48 9.01 1.80 1.35 2.94
```

Co jeżeli chcemy przeprowadzić regresję z wykorzystaniem wszystkich predyktorów z wyjątkiem jednego? W analizowanym przykładzie parametr ***age*** ma wysoką p-wartość, więc chcemy powtórzyć regresję z jego pominięciem. Należy użyć następującej składni:

```
> lm.fit1=lm(medv~.-age ,data=Boston )

> summary (lm.fit1)
```

Alternatywnie można wykorzystać funkcję ***update()***

```
> lm.fit1=update (lm.fit , ~.-age)
```

4. Składniki interakcyjne

W funkcji ***lm()*** łatwo jest uwzględnić interakcję pomiędzy predyktorami. Składnia ***lstat:black*** oznacza, że w modelu powinien zostać uwzględniony składnik interakcyjny pomiędzy lstat i black. Składnia ***lstat*age*** wpisuje do modelu predyktory ***lstat***, ***age*** oraz składnik interakcyjny ***lstat*** x ***age***, czyli jest to skrót na ***lstat + age + lstat:age*** .

```
> summary (lm(medv~lstat *age ,data=Boston ))

Call:

lm(formula = medv ~ lstat * age , data = Boston )

Residuals :

Min 1Q Median 3Q Max

-15.81 -4.04 -1.33 2.08 27.55

Coefficients:

Estimate Std. Error t value Pr(>|t|)
```

(Intercept) 36.088536 1.469835 24.55 < 2e-16 ***

lstat -1.392117 0.167456 -8.31 8.8e-16 ***

age -0.000721 0.019879 -0.04 0.971

lstat:age 0.004156 0.001852 2.24 0.025 *

Signif . codes: 0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Residual standard error : 6.15 on 502 degrees of freedom

Multiple R-squared : 0.556 , Adjusted R-squared : 0.553

F-statistic : 209 on 3 and 502 DF , p-value: <2e-16

5. Nieliniowe transformacje predyktorów

W funkcji lm() można również wykonać nieliniowe transformacje predyktorów. Na przykład, dla określonego predyktora ***X***, możemy utworzyć predyktor X^2 za pomocą składni ***I(X^2)***. Funkcja ***I()*** jest potrzebna ponieważ ***„^”*** ma szczególne znaczenie w formułach. Dzięki funkcji ***I()*** jak powyżej, wykonujemy standardową operację potęgowania. Teraz możemy przeprowadzić regresję ***medv*** za pomocą ***lstat*** i $lstat^2$.

> lm.fit2=lm(medv~lstat +I(lstat ^2))

> summary (lm.fit2)

Call:

lm(formula = medv ~ lstat + I(lstat ^2))

Residuals :

Min 1Q Median 3Q Max

-15.28 -3.83 -0.53 2.31 25.41

116 3. Linear Regression

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 42.86201 0.87208 49.1 <2e-16 ***

lstat -2.33282 0.12380 -18.8 <2e-16 ***

I(lstat ^2) 0.04355 0.00375 11.6 <2e-16 ***

Signif . codes: 0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Residual standard error : 5.52 on 503 degrees of freedom

Multiple R-squared : 0.641 , Adjusted R-squared : 0.639

F-statistic : 449 on 2 and 503 DF , p-value: <2e-16

P-wartość bliska 0 dla składnika do drugiej potęgi sugeruje, że modyfikacja poprawia model. Możemy użyć funkcji anova(), aby ilościowo ocenić to, do jakiego stopnia wprowadzenie terminu ***lstat²*** poprawia model w stosunku do dopasowania liniowego.

```
> lm.fit =lm(medv~lstat)

> anova(lm.fit ,lm.fit2)

Analysis of Variance Table

Model 1: medv ~ lstat

Model 2: medv ~ lstat + I(lstat ^2)

Res.Df RSS Df Sum of Sq F Pr(>F)

1 504 19472

2 503 15347 1 4125 135 <2e -16 ***

---

Signif . codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

W przypadku powyżej Model 1 oznacza mniejszy model zawierający tylko jeden predyktor – lstat, natomiast Model 2 odpowiada większemu modelowi z wielomianem kwadratowym, z dwoma predyktorami ***lstat*** i ***lstat²*** . Funkcja anova() przeprowadza test statystyczny z hipotezą zerową, że oba modele dopasowują się do danych równie dobrze, hipotezą alternatywną jest to, że model pełniejszy jest lepszy. W przykładzie powyżej F-statystyka wynosi 135 a związana z nią p-wartość wynosi prawie 0. To dostarcza bardzo jasnych dowodów na to, że model z predyktorami ***lstat*** i ***lstat2*** jest znacznie lepszy od modelu zbudowanego w oparciu wyłączcie o lstat. Nie jest to zaskoczeniem, bo już wcześniej widzieliśmy dowody na nieliniowość w zależności pomiędzy ***medv*** i ***lstat***. Jeżeli wpiszemy:

```
> par(mfrow=c(2,2))

> plot(lm.fit2)
```

zobaczymy, że kiedy w modelu zawarty jest składnik ***lstat²*** wówczas na wykresie składników resztkowych widzimy tylko bardzo nikły wzorzec.

W celu dopasowania zależności wielomianem trzeciego stopnia możemy w funkcji lm() i zawrzeć w niej predyktor w formie ***I(X^3***). Takie rozwiązanie może być jednak niewygodne przy deklarowaniu formuł z wielomianami wyższego stopnia. Lepiej do tego celu wykorzystać funkcję poly() uruchamianą jako argument wejściowy funkcji lm(). Poniższa formuła generuje model z wielomianem piątego stopnia.

```
lm.fit5=lm(medv~poly(lstat ,5))

> summary (lm.fit5)

Call:

lm(formula = medv ~ poly(lstat , 5))

Residuals :
```

Min 1Q Median 3Q Max

-13.543 -3.104 -0.705 2.084 27.115

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 22.533 0.232 97.20 < 2e-16 ***

poly(lstat , 5)1 -152.460 5.215 -29.24 < 2e-16 ***

poly(lstat , 5)2 64.227 5.215 12.32 < 2e-16 ***

poly(lstat , 5)3 -27.051 5.215 -5.19 3.1e-07 ***

poly(lstat , 5)4 25.452 5.215 4.88 1.4e-06 ***

poly(lstat , 5)5 -19.252 5.215 -3.69 0.00025 ***

Signif . codes: 0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Residual standard error : 5.21 on 500 degrees of freedom

Multiple R-squared : 0.682 , Adjusted R-squared : 0.679

F-statistic : 214 on 5 and 500 DF , p-value: <2e-16

Powyższy listing sugeruje, że zawarcie w modelu elementów wielomianu piątego stopnia prowadzi w tym przypadku do znacznego poprawienia dopasowania modelu do danych. Późniejsze próby z analizą przydatności wielomianów wyższego rzędu nie wykazały statystycznej istotności współczynników przy potęgach większych niż 5.

Oprócz przekształceń wielomianowych można transformować predyktory za pomocą innych funkcji: np.:logarytmu:

> summary (lm(medv~log(rm),data=Boston))

6. Predyktory jakościowel

Teraz zajmiemy się zbiorem ***Carseats***, który jest częścią biblioteki ***ISLR***. Spróbujemy przewidzieć wartość ***Sales*** (liczba sprzedanych samochodowych fotelików dziecięcych) w 400 różnych lokalizacjach w oparciu o zestaw predyktorów.

>> names(Carseats)

[1] "Sales " "CompPrice " "Income " "Advertising "

[5] " Population " "Price" "ShelveLoc " "Age"

[9] " Education " "Urban" "US"

Zbiór danych Carseats zawiera predyktory jakościowe takie jak np. Shelveloc, czyli wskaźnik jakości miejsca ekspozycji produktów w określonej lokacji. Predyktor Shelveloc przyjmuje trzy możliwe wartości: ***„Bad”***, ***„Medium”*** i ***„Good”***. Jeżeli wśród predyktorów znajdują się predyktory jakościowe , wówczas R automatycznie generuje dodatkowe zmienne binarne. W skrypcie przedstawionym poniżej

dopasowujemy dane z wykorzystaniem modelu regresji wielokrotnej z terminami interakcyjnymi i dodatkowymi zmiennymi wynikającymi z jakościowego (a nie ilościowego) charakteru niektórych predyktorów.

> lm.fit =lm(Sales∼.+ Income :Advertising +Price :Age ,data=Carseats)

> summary (lm.fit)

Call:

lm(formula = Sales ∼ . + Income : Advertising + Price:Age , data =

Carseats)

Residuals :

Min 1Q Median 3Q Max

-2.921 -0.750 0.018 0.675 3.341

Coefficients:

Estimate Std . Error t value Pr(>|t|)

(Intercept) 6.575565 1.008747 6.52 2.2e -10 ***

CompPrice 0.092937 0.004118 22.57 < 2e -16 ***

Income 0.010894 0.002604 4.18 3.6e -05 ***

Advertising 0.070246 0.022609 3.11 0.00203 **

Population 0.000159 0.000368 0.43 0.66533

Price -0.100806 0.007440 -13.55 < 2e -16 ***

ShelveLocGood 4.848676 0.152838 31.72 < 2e -16 ***

ShelveLocMedium 1.953262 0.125768 15.53 < 2e -16 ***

Age -0.057947 0.015951 -3.63 0.00032 ***

Education -0.020852 0.019613 -1.06 0.28836

UrbanYes 0.140160 0.112402 1.25 0.21317

USYes -0.157557 0.148923 -1.06 0.29073

Income :Advertising 0.000751 0.000278 2.70 0.00729 **

Price:Age 0.000107 0.000133 0.80 0.42381

Signif . codes: 0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Residual standard error : 1.01 on 386 degrees of freedom

Multiple R-squared : 0.876 , Adjusted R-squared : 0.872

Funkcja ***contrasts()*** zwraca sposób kodowania zmiennych binarnych.

```
> attach (Carseats )

> contrasts (ShelveLoc )

Good Medium

Bad 0 0

Good 1 0

Medium 0 1
```

Użyj polecenia ***?contrasts***, aby dowiedzieć się więcej o innych funkcjach kontrakt

R utworzył zmienną ***ShelveLocGood***, która przyjmuje wartość 1, gdy lokalizacja produktu w sklepie jest dobra, lub 0 w każdym przeciwnym wypadku. Jednocześnie utworzona została zmienna ***ShelveLocMedium*** równa 1, gdy lokalizacja produktu na półkach sklepowych jest akceptowalna lub zero w każdym innym przypadku. Zła lokalizacja produktu w sklepie jest oznaczana przez 0 zarówno w ***ShelveLocGood***, jak i ***ShelveLocMedium***. Wysoka wartość współczynnika przy predyktorze ***ShelveLocGood***, wskazuje na to, że dobra lokalizacja przedmiotu w obrębie sklepu jest powiązana ze zwiększoną liczbą sprzedanych produktów. ***ShelveLocMedium*** ma również dodatni współczynnik regresji, jednak jest on niższy niż w przypadku ***ShelveLocGood***. Oznacza to, że średnie miejsce ekspozycji w sklepie prowadzi do zwiększonej liczby sprzedanych artykułów (w porównaniu do miejsc ze słabą ekspozycją), jednak efekt jest gorszy niż w przypadku dobrej ekspozycji produktu.

7. Zadania projektowe

a. Zadanie związane z wykorzystaniem prostej regresji liniowej na zbiorze Auto

[1] Opisz krótko zbiór danych.

[2] Użyj funkcji lm() do przeprowadzenia prostej regresji liniowej z ***mpg*** jako zmienną wyjaśnianą i ***horsepower*** jako predyktorem. Użyj funkcji ***summary()***, żeby wyświetlić wyniki. Skomentuj wyniki:

- Czy zmienna wyjaśniana i predyktor są powiązane?
- Jak silny jest związek pomiędzy zmienną wyjaśnianą i predyktorem?
- Czy związek pomiędzy zmiennymi jest dodatni, czy ujemny?
- Jaka jest przewidywana wartość ***mpg*** związana z ***horsepower*** o wartości 98? Jakie są w tym przypadku przedziały ufności dla średniej i przedział dla predykcji?

[3] Wykreśl zmienną wyjaśnianą i predyktor. Użyj funkcji ***abline()*** do narysowania linii regresji.

[4] Użyj funkcji ***plot()*** do wyświetlenia wykresów diagnostycznych dla regresji liniowej. Skomentuj zauważalne problemy.

b. Zadanie związane z wielokrotną regresją liniową na zbiorze Auto

[1] Wygeneruj macierz wykresów (ang. scatterplot matrix), która będzie zawierała wszystkie zmienne w zbiorze.

[2] Wylicz macierz korelacji pomiędzy zmiennymi przy użyciu funkcji ***cor().*** Trzeba wcześniej wykluczyć zmienną ***name***, która jest zmienną jakościową.

[3] Użyj funkcji lm() do przeprowadzenia wielokrotnej regresji liniowej z ***mpg*** jako zmienną wyjaśnianą i wszystkimi pozostałymi zmiennymi z wyjątkiem name jako predyktorami. Użyj ***summary()*** fo wyświetlenia wyników. Skomentuj np.:

- Czy jest związek pomiędzy predyktorami i zmienną wyjaśnianą?
- Które z predyktorów są związane ze zmienną wyjaśnianą w sposób statystycznie istotny.
- Jak należy interpretować współczynnik określony dla atrybutu year?

[4] Wykorzystaj funkcję ***plot()*** do wygenerowania wykresów diagnostycznych dla analizowanego modelu liniowego. Skomentuj wynik. Czy wykresy resztkowe pokazują jakieś niezwykle duże wartości odstające? Czy wykres lewarowania pokazuje punkty z wyjątkowo dużą wartością statystyki.

[5] Użyj operatorów * i : do dopasowania modelu z efektami interakcji. Czy jakieś interakcje wydają się być statystycznie istotne'=.

[6] Wypróbuj kilka transformacji zmiennych wejściowych:

Log(X), $\sqrt{X}, X^2$

Skomentuj wyniki.