# Temat: Pandas – Regresja logistyczna

Wykonaj zadanie zgodnie z poniższymi wymaganiami:

1. Zaimportuj biblioteki:
   1. ***pandas*** i nazwij ją jako ***pd***.
   2. moduł ***pyplot*** z biblioteki ***matplotlib*** i nazwij go ***plt***
   3. ***seaborn*** i nazwij ją ***sn***
   4. ***numpy*** i nazwij ją ***np***

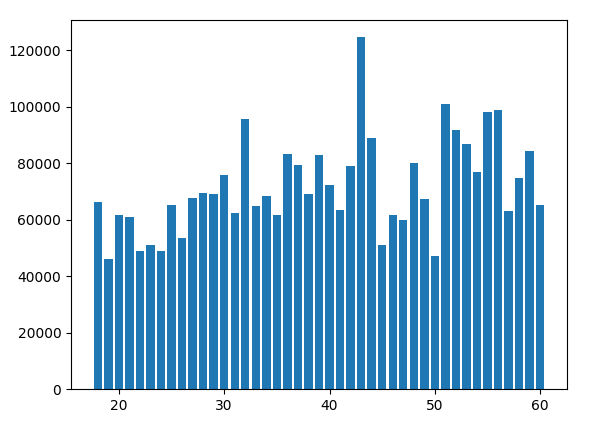
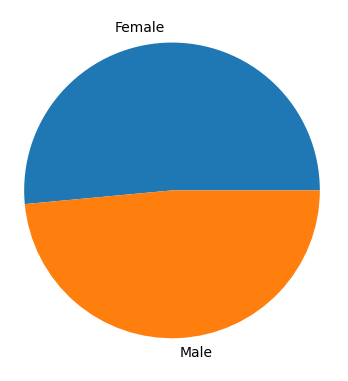
Dodatkowo zaimportuj:

* 1. ***LogisticRegression*** z biblioteki ***sklearn.linear\_model***
  2. ***train\_test\_split*** z biblioteki ***sklearn.model\_selection***
  3. ***classification\_report, confusion\_matrix, accuracy\_score*** z biblioteki ***sklearn.metrics***

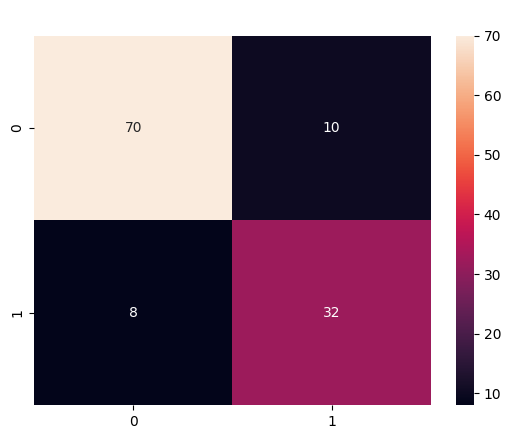
W przypadku braku jakiejś biblioteki, zainstaluj ją za pomocą polecenia ***pip install XXXX***

1. Zaimportuj dane z pliku ***users.csv*** (zwróć uwagę na separator pól) oraz wyświetl 5 pierwszych wierszy danych i informacje o danych
2. Usuń kolumnę ***User ID***
3. Skonwertuj kolumnę ***EstimatedSalary*** z wartości tekstowej w formacie ***$1 000,00*** do formatu liczbowego (float): ***1000.00***. Jeżeli nie poradzisz sobie z tą częścią zadania do kolejnych ćwiczeń skorzystaj z pliku ***users4.csv***
4. Wyświetl dwa wykresy:
   1. Średnie zarobki kobiet i mężczyzn – wykres kołowy
   2. Średnie zarobki w zależności od wieku – wykres słupkowy

Do utworzenia wykresów należy skorzystać z grupowania danych. Przykład wykresów:



1. Skonwertuj dane znajdujące się w kolumnie ***Gender*** z wartości tekstowych ***Female/Male***, na wartości liczbowe ***1/0***.
2. Skonwertuj dane znajdujące się w kolumnie ***Purchased*** z wartości tekstowych ***Yes/No***, na wartości liczbowe ***1/0***
3. Podziel dane na zmienne niezależne (macierz/tablica ***X***) i zmienną zależną (wektor ***y***)
4. Podziel dane na zbiór treningowy i zbiór testowy w stosunku ***7:3*** (dane testowe mają stanowić 30% wszystkich danych)
5. Utwórz model regresji logistycznej (w przypadku błędu informującego o zbyt małej ilości iteracji podczas trenowania modelu ustaw parametr ***max\_iter=1000.*** Dopasuj (wytrenuj) model do danych treningowych
6. Stwórz wektor danych predykcyjnych ***y\_pred*** zawierający przewidywania modelu dla danych testowych
7. Oceń model korzystając z funkcji:
   1. ***score*** dla danych treningowych
   2. ***score*** dla danych testowych
   3. ***classification\_report***  dla porównania danych testowych i predykcyjnych
   4. ***confusion\_matrix*** dla danych testowych i predykcyjnych
   5. Wyświetl tablicę niepewności (tablicę prawdy) dla danych wygenerowanych z funkcji ***confusion\_matrix.*** Wykres utwórz jako ***heatmap*:**

****

1. Jaką odpowiedź da model dla poniższych danych wejściowych:
   1. Kobieta, 18 lat, zarobki 240000
   2. Mężczyzna, 28 lat, zarobki 123000
   3. Mężczyzna, 43 lata, zarobki 234000

**Dokumentacja.**

**Informacje o typach danych tabeli:**

Print(Obiekt\_dataframe.info())

**Konwertowanie i formatowanie danych**

**#konwersja typu na float (jeżeli to możliwe)**

tabela["kolumna"] = tabela["kolumna"].map(lambda x: float(x)) #funkcja anonimowa, tzw. lambda function

**#formatowanie np. pola walutowego *1,000,000.00 $*:**

tabela["nowa\_kolumna"] = tabela ["stara\_kolumna"].map('{:,.2f} $'.format)

**#zamiana znaków w tekście:**

Zmienna = "123 456@789"

**#Usunięcie zbędnego znaku (np. spacji):**

Zmienna = Zmienna.replace(" ","")

**#Zamiana znaku na inny:**

Zmienna = Zmienna.replace("@","#")

**#itp…**

**Grupowanie danych**

Dany jest obiekt zawierający markę samochodu oraz ceny. Ceny różnią się w zależności od rocznika. Np.

|  |  |  |
| --- | --- | --- |
| **Marka** | **Rocznik** | **Cena** |
| Marka1 | 2000 | 10000 |
| Marka2 | 2010 | 20000 |
| Marka1 | 2009 | 15000 |
| Marka2 | 2000 | 12000 |

Poniższy przykład tworzy dane do wykresów jako

1. średnia cena samochodu w zależności od Marki
2. średnia cena samochodu w zależności od rocznika:

d = {'Marka': ['Marka1', 'Marka2','Marka1', 'Marka2'],

'Rocznik': [2000,2010,2009,2000],

'Cena': [10000,20000,15000,12000]}

df = pd.DataFrame(d)

**# a)**

gr1 = df.groupby('Marka').agg(srednia=('Cena','mean'))

gr1.reset\_index(inplace=True) #dodanie kolumny z indeksem

**# b)**

gr2 = df.groupby('Rocznik').agg(srednia=('Cena','mean'))

gr2.reset\_index(inplace=True)

W powyższym przykładzie obiekt ***gr1/gr2*** będzie zawierał dwie kolumny: ***Marka/Rocznik*** oraz ***srednia***, które mogą zostać przedstawione w formie graficznej

**Wykres słupkowy**

plt.bar(tabela["kolumna\_z\_etykietami"],tabela["kolumna\_z\_wartościami"])

plt.show()

**Wykres kołowy**

plt. pie(tabela["kolumna\_z\_wartościami"],labels= tabela["kolumna\_z\_etykietami"])

plt.show()

**Wybór zmiennych niezależnych (X) i zmiennej zależnej (y).**

Zmienne niezależne to nasze dane wejściowe (***X***) np. wiek, płeć, wzrost, itp. Zmienna (zmienne) zależna to nasze dane wyjściowe (***y***), które będziemy docelowo chcieli przewidywać. W regresji logistycznej dane wyjściowe powinny mieć postać dwuwartościową (0 albo 1).

Przykład.

X = np.array(dane[["kolumna1","kolumna2","kolumna3"…]]

y = np.array(dane["kolumna\_wynikowa"])

Tak przygotowane dane można podzielić na dane testowe i dane treningowe (w stosunku 50/50):

X\_tr, x\_tst, y\_tr, y\_tst = train\_Test\_split(X,y,test\_size=0.5)

**Utworzenie modelu regresji logistycznej i wytrenowanie go na danych treningowych**

model = LogisticRegression()

model.fit(X\_treningowe, y\_treningowe)

y\_przewidywane = model.predict(X\_testowe)

**Ocena modelu**

Ocena\_na\_danych\_treningowych = model.score(X\_treningowe,y\_treningowe)

Ocena\_na\_danych\_testowych = model.score(X\_testowe,y\_testowe)

Tabela\_prawdy = confusion\_matrix(y\_testowe, y\_przewidywane)

**#Wykres „mapa ciepła”:**

sn.heatmap(Tabela\_prawdy,annot=True)

plt.show()

**Ręczne sprawdzanie przewidywanych wartości:**

Dane\_we = np.singe([zm1,zm2,zm3,…]) #gdzie zm1, zm2, zm3 to dane wejściowe (zmienne niezależne)

Dane\_we.reshape(1,-1)

print(model.predict(Dane\_we))