Temat: Pandas – Regresja logistyczna

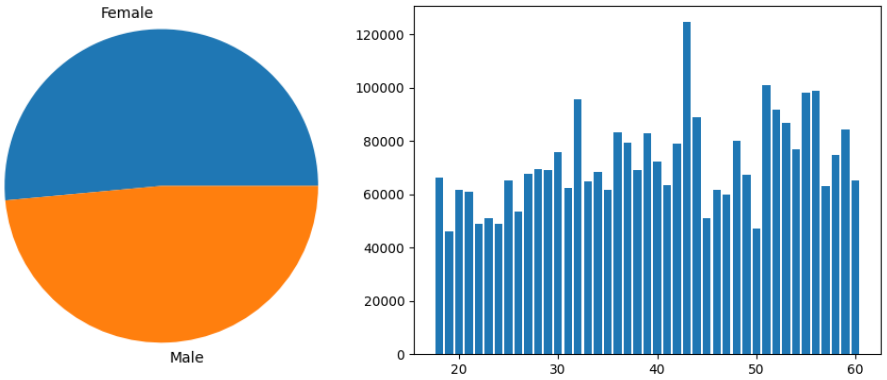
Wykonaj zadanie zgodnie z poniższymi wymaganiami:

1. Zaimportuj biblioteki:
   a. ***pandas*** i nazwij ją jako ***pd***.
   b. moduł ***pyplot*** z biblioteki ***matplotlib*** i nazwij go ***plt***
   c. ***seaborn*** i nazwij ją ***sn***
   d. ***numpy*** i nazwij ją ***np***

   Dodatkowo zaimportuj:
   e. ***LogisticRegression*** z biblioteki ***sklearn.linear_model***
   f. ***train_test_split*** z biblioteki ***sklearn.model_selection***
   g. ***classification_report, confusion_matrix, accuracy_score*** z biblioteki ***sklearn.metrics***

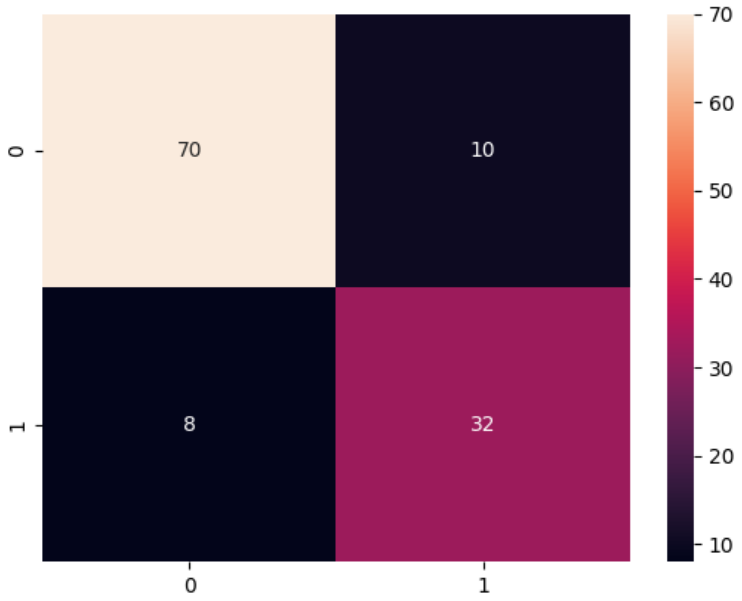
   W przypadku braku jakiejś biblioteki, zainstaluj ją za pomocą polecenia ***pip install XXXX***
2. Zaimportuj dane z pliku ***users.csv*** (zwróć uwagę na separator pól) oraz wyświetl 5 pierwszych wierszy danych i informacje o danych
3. Usuń kolumnę ***User ID***
4. Skonwertuj kolumnę ***EstimatedSalary*** 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***
5. Wyświetl dwa wykresy:
   a. Średnie zarobki kobiet i mężczyzn – wykres kołowy
   b. Średnie zarobki w zależności od wieku – wykres słupkowy

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



6. Skonwertuj dane znajdujące się w kolumnie ***Gender*** z wartości tekstowych ***Female/Male***, na wartości liczbowe ***1/0***.
7. Skonwertuj dane znajdujące się w kolumnie ***Purchased*** z wartości tekstowych ***Yes/No***, na wartości liczbowe ***1/0***
8. Podziel dane na zmienne niezależne (macierz/tablica ***X***) i zmienną zależną (wektor ***y***)

9. Podziel dane na zbiór treningowy i zbiór testowy w stosunku ***7:3*** (dane testowe mają stanowić 30% wszystkich danych)
10. 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
11. Stwórz wektor danych predykcyjnych ***y_pred*** zawierający przewidywania modelu dla danych testowych
12. Oceń model korzystając z funkcji:
    a. ***score*** dla danych treningowych
    b. ***score*** dla danych testowych
    c. ***classification_report*** dla porównania danych testowych i predykcyjnych
    d. ***confusion_matrix*** dla danych testowych i predykcyjnych
    e. Wyświetl tablicę niepewności (tablicę prawdy) dla danych wygenerowanych z funkcji ***confusion_matrix.*** Wykres utwórz jako ***heatmap*:**



13. Jaką odpowiedź da model dla poniższych danych wejściowych:
    a. Kobieta, 18 lat, zarobki 240000
    b. Mężczyzna, 28 lat, zarobki 123000
    c. 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

a) średnia cena samochodu w zależności od Marki
b) 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))
```