#1

Wczytaj zbiór penguins.csv. Atrybut decyzyjny to *class*, w zbiorze występują wartości brakujące (puste). Przygotuj ten zbiór do uruchomienia na nim algorytmu *sklearn.cluster.KMeans* poprzez:

- identyfikację i korektę wartości odstających,
- imputację wartości brakujących,
- one-hot encoding atrybutów nominalnych za pomocą *pandas.get_dummies*,
- przeskalowanie wartości za pomocą *sklearn.preprocessing.StandardScaler*.

Całość rozwiązania przygotuj w postaci notebooka ipynb. Dodatkowo zapisz przygotowany zbiór do pliku *my-penguins.csv* w taki samym formacie (separator pól, dziesiętny, kodowanie) jak pliku *penguins.csv*.

Załącz notebook (plik *ipynb*) oraz wynikowy plik *csv* z poprzedniego zadania. Notebook powinien zawierać kod razem z wynikami uruchomienia.

#2

Wczytaj wcześniej przygotowane dane z pliku *my-penguins.csv* i uruchom na nim algorytm *KMeans*. Przetestuj parametr *n_clusters* w zakresie od 1 do 10 i dobierz według Ciebie najlepszy, posiłkując się metodą "łokciową" (ang. elbow method). W grupowaniu pomiń zmienną *class*. Pokaż wykres, na którym na osi *x* jest *n_clusters* w zakresie od 1 do 10, a na osi *y* wartość atrybutu *inertia_* modelu odpowiadająca danemu parametrowi *n_cluster*.

Zobrazuj wyniki grupowania za pomocą wykresu 2d, gdzie na osi *x* jest pierwsza główna składowa, a na osi *y* jest druga główna składowa uzyskane z metody *sklearn.decomposition.PCA*. Kolor punktu na wykresie powinien odpowiadać grupie określonej przez *KMeans*. Uwaga: nie należy robić grupowania na danych po zastosowaniu PCA - PCA jest wykorzystywane wyłącznie do wizualizacji wyników, a grupowanie jest robione na danych z *my-penguins.csv*

Całość rozwiązania przygotuj w postaci notebooka ipynb. Zapisz dodatkowo do pliku *my-penguins-with-groups.csv* dane *my-penguins.csv* z dodaną kolumną *group*, w której znajdują się nr grup przypisane przykładom przez *KMeans*.

Załącz notebook (plik *ipynb*) oraz wynikowy plik *csv* z poprzedniego zadania. Notebook powinien zawierać kod razem z wynikami uruchomienia.

#3

Wczytaj wcześniej przygotowane dane z pliku *my-penguins-with-groups.csv* i porównaj ze sobą zgodność wartości zmiennej *group* z wartościami *class*. Porównanie przygotuj w postaci macierzy, w której wiersze to wartości *class*, kolumny to wartości *group*, a w poszczególnych komórkach macierzy znajduje się liczba przykładów z danej klasy przypisanych do danej grupy.

Oceń jakość grupowania za pomocą *sklearn.metrics.adjusted_rand_score*.

Całość rozwiązania przygotuj w postaci notebooka ipynb.

Załącz notebook (plik *ipynb*) do poprzedniego zadania. Notebook powinien zawierać kod razem z wynikami uruchomienia.

#4

Wczytaj zbiory danych *z data.zip*. Są tam 3 pliki w formacie Apache Parquet. Wartości brakujące oznaczone są jako *NaN*. Nazwa atrybutu decyzyjnego w każdym zbiorze to *class*. Przygotuj notebook Python podsumowujący te zbiory w formie tabel:

dataset	**row_count**	**col_count**	**missings_ratio**	**class_count**
auto-mpg				
autos				
hungarian-heart-disease				

gdzie:

- row_count - liczba wierszy w zbiorze,
- col_count - liczba kolumn,
- missings_ratio - udział wartości brakujących w zbiorze (liczba w zakresie od 0 do 1),
- class_count - liczba klas decyzyjnych.

Załącz notebook (plik *ipynb*) z poprzedniego zadania. Notebook powinien zawierać kod razem z wynikami uruchomienia.

#5

Zaimplementuj własną wersję algorytmu Decision Stump dla danych klasyfikacyjnych. Nie wykorzystuj do tego gotowych algorytmów typu *sklearn.tree.DecisionTreeClassifier.* Algorytm powinien być zaimplementowany w postaci klasy o nazwie *DecisionStumpClassifier* i zawierać co najmniej metody *fit(self, X, y)* oraz *predict(self, X)* - analogicznie jak algorytmy w scikit-learn. Oprócz ograniczenia w korzystaniu gotowych implementacji typu *sklearn.tree.DecisionTreeClassifier*, nie ma narzuconego sposobu implementacji, w szczególności możesz wykorzystać Twoje ulubione kryterium podziału węzła.

Załącz plik *.py* z implementacją klasy *DecisionStumpClassifier.*

#6

Przygotuj notebook Python przeprowadzający ewaluację algorytmu *DecisionStumpClassifier* na wcześniej rozpatrywanych zbiorach z *data.zip* w trybie 5-krotnej stratyfikowanej walidacji krzyżowej

Wyniki ewaluacji pokaż w postaci następującej tabeli:

dataset	**DS**	**DT(max_depth=1)**	**DT**
auto-mpg			
autos			
hungarian-heart-disease			

gdzie:

- DS to średni *sklearn.metrics.balanced_accuracy_score* (BAC) dla Twojej implementacji *DecisionStumpClassifier*,
- DT (max_depth=1) to średni BAC dla *sklearn.tree.DecisionTreeClassifier* z parametrem *max_depth=1*,
- DT to średni BAC powyższego algorytmu z domyślnymi parametrami.

Załącz notebook (plik *ipynb*) do poprzedniego zadania. Notebook powinien zawierać kod razem z wynikami uruchomienia.