Sprawozdanie

Indukcyjne Metody Analizy Danych

Ćwiczenie 1. Klasyfikator oparty na twierdzeniu Bayesa przy naiwnym założeniu o wzajemnej niezależności atrybutów

Autor: Paweł Mielniczuk

**Spis treści:**

1. Naiwny klasyfikator Bayesa
2. Opis zbiorów wykorzystanych podczas analizy
3. Opis implementacji
4. Analiza klasyfikatora
5. **Opis klasyfikatora**

W implementacji i analizie został użyty naiwny klasyfikator Bayes’a. W odróżnieniu od rozwiązania nienaiwnego, bazuje on na założeniu że wszystkie zdarzenia, w przypadku użytych danych są to cechy (features), są od siebie niezależne i nie mają na siebie wpływu i nie występują pomiędzy nimi żadne korelacje. W zadaniu rozwiązywany będzie problem klasyfikacji. W przypadku sieci bayesowskich należałoby wszystkie takie relacje zamodelować.

1. **Opis zbiorów**

Podczas analizy i implementacji użyte zostały cztery zbiory danych. Zbiory podzielone są na dwie części. Pierwszą z nich są cechy, dokładnie wektor, cech oraz etykiety mówiące o przynależności wektora cech do konkretnej klasy.

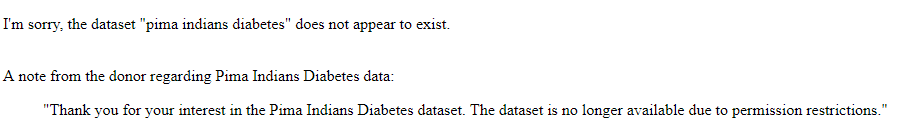
Wszystkie zbiory dostępne są do pobrania ze strony *https://archive.ics.uci.edu/ml/datasets.html*

Zbiory danych zostały ściągnięte i załadowane przy użyciu biblioteki *pandas* lub bezpośrednio załadowane za pomocą biblioteki *scikit-learn*.

Zbiory danych:

* Iris data set
* Wine data set
* Glass identification data set
* Pima diabetes data set

Ciekawostką jest, że w trakcie badania klasyfikatora i tworzenia sprawozdania ostatni ze zbiorów *Pima diabetes* został usunięty ze strony UCI ze przez ograniczenie uprawnień do udostępniania danego zbioru.



Rysunek 1 Wiadomość ze strony https://archive.ics.uci.edu/ml/datasets/pima+indians+diabetes mówiąca o braku dalszego dostępu do danego zbioru.

Poniżej zaprezentowano opis zbiorów. Opis ten pomoże w zrozumieniu danych, które będą analizowane. Dobre zrozumienie danych z którymi się pracuje jest niezbędną częścią do poprawnego przeprowadzenia badań.

**Zbiór Iris**

Jest to prawdopodobnie jeden z najbardziej znanych i podstawowych zbiorów danych przy problemach klasyfikacji i rozpoznawania wzorców.

Zbiór składa się ze 150 instancji, podzielonych na 3 równe zbiory po 50 klas każda.

Definicje atrybutów:

* Sepal – zielony płatek u dołu kielicha służący do ochrony kwiatu w trakcie kwitnięcia,
* Petal – płatek kwiatu, służący do przyciągania uwagi ptaków i insektów

Cechy zbioru zawierają cztery informacje:

1. Sepal length in cm
2. Sepal width in cm
3. Petal length in cm
4. Petal width in cm

Ostatnią, piątą kolumną jest klasa mówiąca o typie irysa. Możliwe są trzy klasy:

1. Iris Setosa
2. Iris Versicolour
3. Iris Virginica

**Zbiór Wine**

Zbiór ten został skonstruowany w wyniku analizy składu chemicznego win stworzonych w tym samym rejonie Włoch lecz przy użyciu trzech różnych odmian uprawnych.

Zbiór składa się ze 178 instancji.

Definicje atrybutów oraz cechy zbioru:

1. Alcohol – alkohol
2. Malic acid – kwas jabłkowy
3. Ash – popiół
4. Alkalinity of ash – alkaliczność popiołu
5. Magnesium – magnez
6. Total phenols – całkowita zawartość fenoli
7. Flavonoids – flawonoidy
8. Nonflavanoid phenols – fenole nieflawonowe
9. Proanthocyanidins – proantocyjanidyny
10. Color intensity, intensywność koloru
11. Hue – odcień
12. OD280/OD315 of diluted wines - OD280 / OD315 rozcieńczonych win
13. Proline – Proline

Pierwszy atrybut w pliku zawierającym dane jest identyfikatorem klasy od 1 do 3.

Rozłożenie instancji klas jest następujące:

* Klasa 1 – 59 instancji,
* Klasa 2 – 71 instancji,
* Klasa 3 – 48 instancji.

**Zbiór Glass identification**

Zbiór powstał poprzez analizę składu chemicznego badanego szkła aby określić typ powstałego szkła oraz jego przeznaczenie.

Zbiór składa się z 214 instancji podzielonych na 6 klas.

Rozłożenie instancji klas jest następujące:

* Klasa 1 – 70 instancji,
* Klasa 2 – 76 instancji,
* Klasa 3 – 17 instancji,
* Klasa 4 - 13,
* Klasa 5 - 9,
* Klasa 6 - 29.

Definicje atrybutów oraz cechy zbioru:

1. Id – numer porządkowy
2. Refractive index – współczynnik załamania światła
3. Sodium – sód
4. Magnesium – magnez
5. Aluminium – glin
6. Silicon – krzem
7. Potassium – potas
8. Calcium – wapń
9. Barium – bar
10. Iron – żelazo

**Zbiór Pima diabetes**

Celem zbioru jest umożliwienie zdiagnozowania czy dany pacjent ma cukrzycę, bazując na diagnostykach zamieszczonych w cechach zbioru. Wszyscy pacjenci przebadani byli kobietami mającymi przynajmniej 21 lat oraz byli pochodzenia indiańskiego plemienia Pima.

Zbiór składa się z 768 instancji posiadających dwie możliwe klasy 1 – oznaczające że zbadana osoba jest chora na cukrzycę, 0 – oznaczające że dana osoba nie jest chora na cukrzycę.

Definicje atrybutów oraz cechy zbioru:

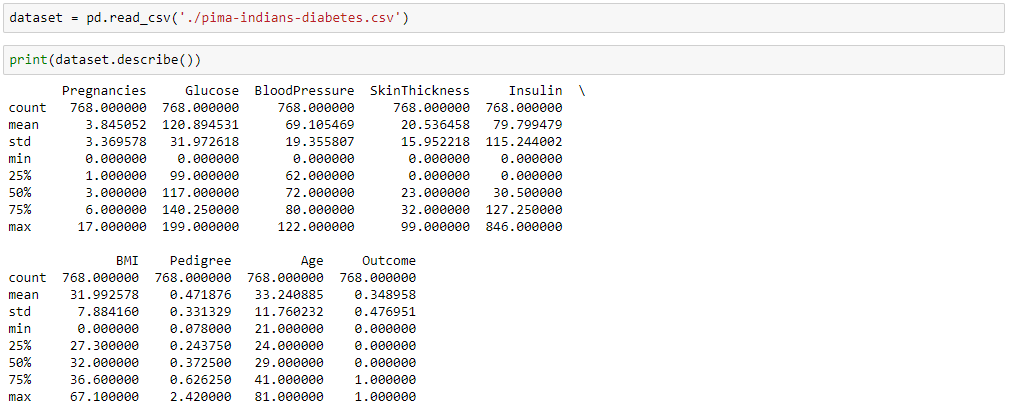
1. Pregnancies – liczba ciąży
2. Glucose – poziom glukozy
3. Blood ressure – ciśnienie krwi
4. Skin thickness – grubość skóry
5. Insulin – poziom insuliny
6. BMI – body mass index
7. Diabetes pedigree function – funkcja pedigree
8. Age – wiek
9. **Implementacja**

Do implementacji użyty został język programowania Python oraz następujące biblioteki: Numpy, Pandas, Matplotlib, Seaborn, scikit-learn.

W implementacji zostały użyte dwa modele naiwnego Bayes’a z biblioteki scikit-learn.

1. GaussianNB – zakłada, że dane są w rozkładzie normalnym
2. MultinomialNB

**Ładowanie danych**

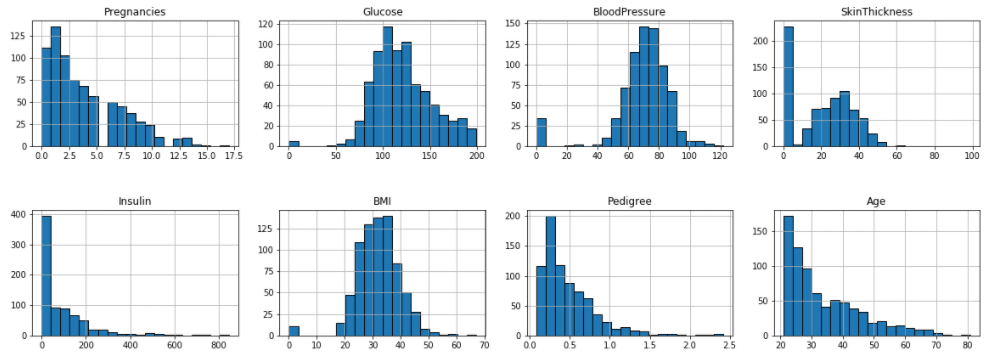


Rysunek 2 Załadowanie i przedstawienie danych przy użyciu biblioteki pandas

Jak wcześniej wspomniałem zawsze przed zaczęciem pracy z tworzeniem modelu należy zrozumieć dane, z którymi się pracuje. Do tego można zastosować kilka technik.

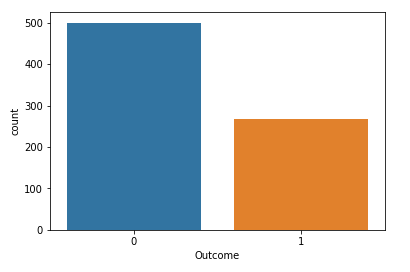
*Pandas* dostarcza wiele metod, między innymi takie umożliwiające uzyskanie opisu danych i kilka statystyk.

Warto także często zwizualizować atrybuty aby np. poznać dokładniejszy rozkład danych.



Rysunek 3 Histogramy atrybutów danych zbioru Pima diabetes

A także rozkład klas.



**Tworzenie modelu Gaussowskiego:**

Scikit-learn umożliwia bardzo prostu sposób tworzenia różnego rodzaju klasyfikatorów. Jednym z nich jest Gaussowski model naiwnego Bayes’a.

****

Model ten zakłada, że atrybuty będą zawsze w rozkładzie normalnym i działa wtedy najlepiej. Jak widać na powyższych wykresach nie jest to jednak zawsze prawdą.

**Cross-validation:**

Często zdarza się, że mamy do czynienia z małymi zbiorami danych. Gdy taki zbiór podzielimy na zbiór treningowy i testowy, a czasem jeszcze walidacyjny nasze dane stają się zbyt małe aby poprawnie wyuczyć model. W takich przypadkach należy podjąć pewne kroki aby zapewnić, że wielkość zbioru będzie odpowiednio duża do wytrenowania modelu.

Jedną z nich jest kroswalidacja. Polega ona na podziale całego zbioru na określoną ilość podzbiorów, a następnie przeprowadzeniu na nich predykcji jak celny będzie nasz model.

Jednymi z podstawowych rodzajów walidacji modelu są:

* Walidacja prosta – mająca na celu podzielenie zbioru na zbiór testowy i treningowy. Najczęściej zbiór testowy nie stanowi więcej niż 20-30% wielkości całego zbioru.
* K-Fold validation – zbiór dzielony jest na K części. Następnie kolejno każdy z podzbiorów brany jest jako zbiór testowy, a pozostałe jako uczący. Analiza jest wykonywana tyle razy na ile części został podzielony zbiór. Po czym wszystkie wyniki się sumuje i uśrednia.
* Stratyfikowana K-Fold validation – zasada działania jest taka sama jak w przypadku zwykłego K-Fold validation z dodatkowym zachowaniem oryginalnych proporcji między klasami (labels) w podzielonych zbiorach.

Poniżej przedstawione zostaną wyniki predykcji modelu używając 3 powyższych sposobów. Przebadany zostanie wpływ wielkości podzbiorów na otrzymywany wynik modelu. W przypadku Walidacji prostej przedział zostanie podzielony na następujące części: 90/10, 80/20 (zwane zasadą Pareta) i 50/50 gdzie pierwsza liczba to zbiór treningowy a druga to zbiór testowy.

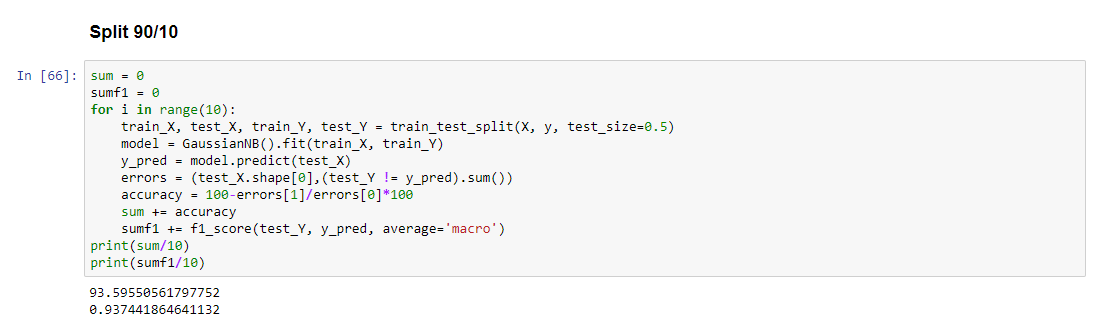
Do wszystkich poniższych testów użyto funkcji *train\_test\_split* z biblioteki scikit-learn z opcjonalnym parametrem shuffle=True, który odpowiada za przetasowanie danych przed ich podziałem. Wszystkie próby zostały wykonane 10 krotnie, a wyniki uśredniono.

Oprócz accuracy wyliczono także F1 score. Jest to średnia ważona precyzji oraz miary recall.

F1 = 2 \* (precision \* recall) / (precision + recall)

**Analiza Wine dataset**

**Walidacja prosta**



Rysunek 4 Podział danych na część testową i treningową oraz trenowanie modelu naiwnego Bayes’a

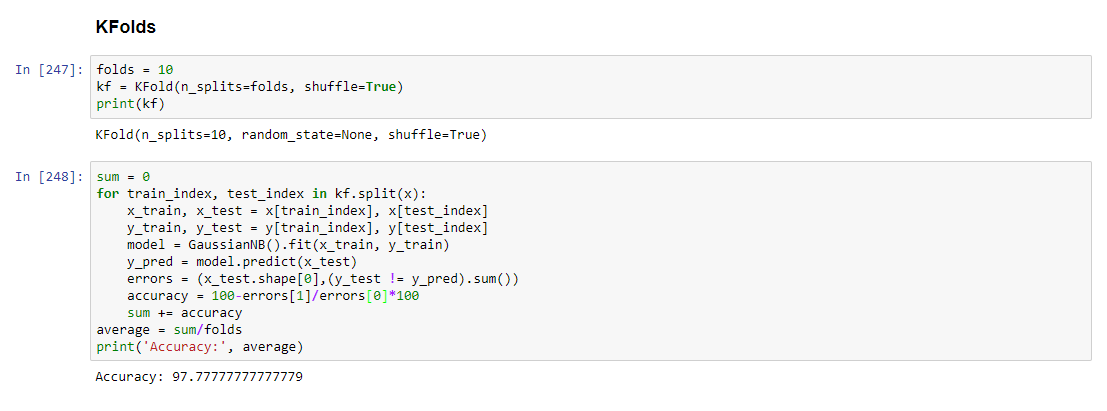
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Podział 1** | **Podział 2** | **Podział 3** |
| **Podział zbiorów treningowy/testowy** | 90/10 | 70/30 | 50/50 |
| **Średnia predykcja** | 0.88 | 0.93 | 0.92 |
| **F1 score** | 0.91 | 0.93 | 0.90 |

Tabela 1 Wyniki predykcji modelu Wine przy różnych proporcjach podziału zbioru treningowego i testowego

Podział całego zbioru, który jest mały, na zbiory gdzie zbiór treningowy jest mały nie jest zawsze dobrym pomysłem. Przy małym zbiorze treningowym nasze estymacje parametrów będą miały większą wariancję, natomiast przy zbyt małym zbiorze testowym nasze statystyki pomiarowe będą miały większą wariancję. Powinno się dążyć do tego aby obie te wariancje nie były zbyt wysokie.

**Walidacja K-Fold**

Biblioteka scikit-learn umożliwia tworzenie takiej walidacji w prosty sposób. Należy podać liczbę podzbiorów, a następnie zaaplikować otrzymane indeksy do predykcji modelu. Przebadano podział na 10, 5 i 2 podzbiory.



Rysunek 5 Predykcja modelu za pomocą walidacji KFold

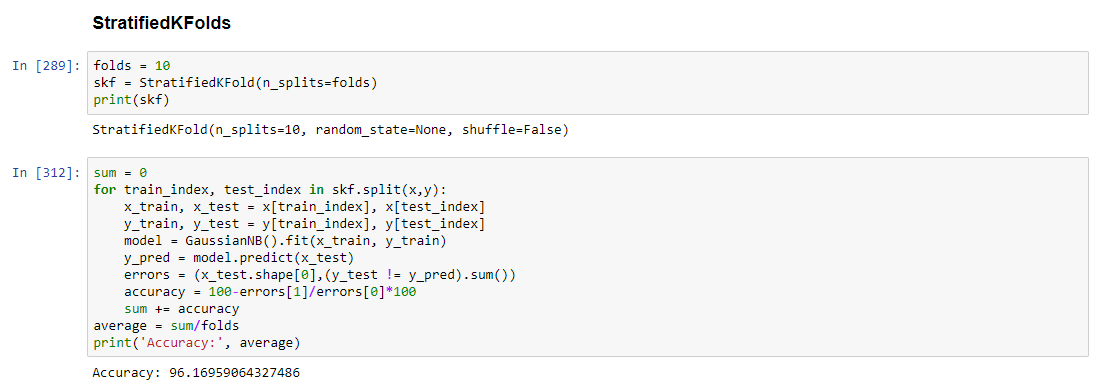
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Podział 1** | **Podział 2** | **Podział 3** |
| **Liczba podzbiorów** | 10 | 5 | 2 |
| **Średnia predykcja** | 0.97 | 0.96 | 0.94 |
| **F1 score** | 0.95 | 0.95 | 0.93 |

Tabela 2 Wyniki predykcji modelu Wine przy różnych proporcjach podziału zbioru przy walidacji K-Fold

Wyniki nieco różnią się od zwykłego podziału. Jako, że nasze zbioru są małe nie możliwe jest zastosowanie zwykłego podziału np. 80/20. Stracilibyśmy w ten sposób zbyt dużo cennych danych. Jak widać walidacja K-Fold zwróciła wyższe wyniki w prawie każdym podziale. Z tabeli 2 można także wyczytać, że najlepiej działa walidacja gdzie fold ustawiony jest na 10. Jest to także często zalecana wielkość podzbiorów.

**Stratyfikowana walidacja K-Fold**

Tak jak w przypadku zwykłego K-Fold, zostały użyte te same podziały. Zmianą jest natomiast to, że gdy zamierzamy dokonać podziału stratyfikowanego nie jest możliwe ustawienie opcji tasowania zbioru.



Rysunek 6 Predykcja modelu za pomocą stratyfikowanej walidacji KFold

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Podział 1** | **Podział 2** | **Podział 3** |
| **Liczba podzbiorów** | 10 | 5 | 2 |
| **Średnia predykcja** | 0.96 | 0.95 | 0.94 |
| **F1 score** | 0.95 | 0.95 | 0.94 |

Tabela 3 Wyniki predykcji modelu Wine przy różnych proporcjach podziału zbioru przy stratyfikowanej walidacji K-Fold

Tabela 3 w porównaniu z tabelą 2 pokazuje także przewagę kroswalidacji stratyfikowanej nad zwykłą bez stratyfikacji. Dzięki zachowaniu oryginalnych proporcji danych jesteśmy w stanie wyuczyć nasz model wszystkich rodzajów klas. Dzięki temu w naszym zbiorze testowym nigdy nie będzie tak, że znajdą się tam jedynie dane opisane klasą, której nasz model nie widział w trakcie uczenia.

**Analiza Glass dataset**

**Walidacja prosta**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Podział 1** | **Podział 2** | **Podział 3** |
| **Podział zbiorów treningowy/testowy** | 90/10 | 70/30 | 50/50 |
| **Średnia predykcja** | 0.95 | 0.96 | 0.96 |
| **F1 score** | 0.95 | 0.96 | 0.95 |

Tabela 4 Wyniki predykcji modelu Glass przy różnych proporcjach podziału zbioru treningowego i testowego

**Walidacja K-Fold**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Podział 1** | **Podział 2** | **Podział 3** |
| **Liczba podzbiorów** | 10 | 5 | 2 |
| **Średnia predykcja** | 0.89 | 0.91 | 0.94 |
| **F1 score** | 0.90 | 0.90 | 0.93 |

Tabela 5 Wyniki predykcji modelu Glass przy różnych proporcjach podziału zbioru przy walidacji K-Fold

**Stratyfikowana walidacja K-Fold**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Podział 1** | **Podział 2** | **Podział 3** |
| **Liczba podzbiorów** | 10 | 5 | 2 |
| **Średnia predykcja** | 0.94 | 0.97 | 0.96 |
| **F1 score** | 0.94 | 0.96 | 0.94 |

Tabela 6 Wyniki predykcji modelu Glass przy różnych proporcjach podziału zbioru przy stratyfikowanej walidacji K-Fold

**Analiza Pima diabetes dataset**

**Walidacja prosta**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Podział 1** | **Podział 2** | **Podział 3** |
| **Podział zbiorów treningowy/testowy** | 90/10 | 70/30 | 50/50 |
| **Średnia predykcja** | 0.74 | 0.75 | 0.74 |
| **F1 score** | 0.70 | 0.73 | 0.73 |

Tabela 7 Wyniki predykcji modelu Pima diabetes przy różnych proporcjach podziału zbioru treningowego i testowego

**Walidacja K-Fold**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Podział 1** | **Podział 2** | **Podział 3** |
| **Liczba podzbiorów** | 10 | 5 | 2 |
| **Średnia predykcja** | 0.76 | 0.74 | 0.75 |
| **F1 score** | 0.75 | 0.73 | 0.73 |

Tabela 8 Wyniki predykcji modelu Pima diabetes przy różnych proporcjach podziału zbioru przy walidacji K-Fold

**Stratyfikowana walidacja K-Fold**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Podział 1** | **Podział 2** | **Podział 3** |
| **Liczba podzbiorów** | 10 | 5 | 2 |
| **Średnia predykcja** | 0.76 | 0.75 | 0.73 |
| **F1 score** | 0.75 | 0.73 | 0.70 |

Tabela 9 Wyniki predykcji modelu Pima diabetes przy różnych proporcjach podziału zbioru przy stratyfikowanej walidacji K-Fold

**Dyskretyzacja:**

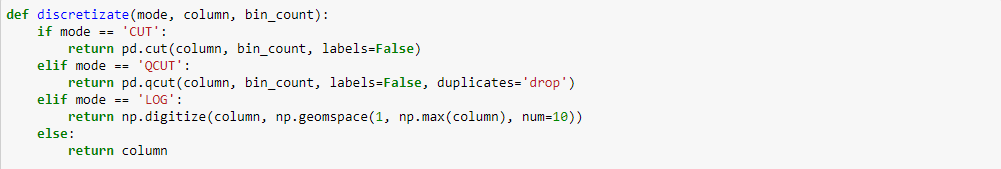
Dyskretyzacja ma na celu zamianę ciągłych zmiennych losowych na dyskretne zmienne losowe. Gdy dokonuje się dyskretyzacji zawsze ma się do czynienia z pewnego stopnia błędem dyskretyzacji. Gdy dokonana zostaje zamiana liczb ciągłych na dyskretne tracone są niektóre informacje o danych, np. ucięcie liczb po przecinku, lub nawet całkowita strata informacji o zmiennej poprzez wrzucenie liczby do kubełka w którym traci swoje oryginalne znaczenie i zostaje zastąpiona inną liczbą.

Jedną z użytych metod dyskretyzacji jest kubełkowanie (ang. *binning*). Binning tworzy określoną ilość interwałów na danym zbiorze. Następnie sprawdza, każdą wartość po kolei i jeżeli wpada ona w dany przedział jest ona tam wtedy dodawana a liczba wystąpień wzrasta, np. interwał (0.0, 2.05> i atrybut z wartością 1.42 zostanie tam przydzielony. Następnie takim przedziałom nadawana może być jakaś klasa i na tej bazie zostaje dokonana klasyfikacja.

Dyskretyzacji dokonuje się na każdej kolumnie osobno ze względu na różne wartości atrybutów zatem także inne przedziały, do których dane zostaną przydzielone.

**Pandas.cut**

Pierwsza z wybranych metod dyskretyzacji jest dostarczana przez bibliotekę pandas. Do metody należy podać przedział danych (u nas kolumna), liczbę kubełków na ile ma zostać podzielony dany przedział oraz kilka opcjonalnych argumentów. Metoda tworzy daną ilość przedziałów na danym zbiorze jedynie ze względu na minimalną i maksymalną wartość atrybutów. Tzn. przedziały zostają utworzone w równomiernych wielkościach. Metoda zwraca utworzone przedziały oraz liczbę atrybutów, które zostały do danego interwału przydzielone.



Rysunek Sposób tworzenie różnych sposobów dyskretyzacji danych

**Pandas.qcut**

Drugą z wybranych metod jest metoda qcut, także dostarczane przez bibliotekę pandas. Qcut tak jak cut zwraca interwały oraz liczbę przypisanych tam danych, jednakże różni się w sposobie tworzenia interwałów. Metoda ma w swojej nazwie „q” które z angielskiego oznacza kwantyle (ang. *quantile*). Oznacza to, że interwały są tworzone z uwzględnieniem rozkładu ilościowego oryginalnych danych ciągłych. Utworzone przedziały będą miały taką samą lub bardzo zbliżoną liczbę danych przypisanych do siebie. Dzięki temu sposobowi możemy zapewnić lepsze rozłożenie danych w każdym kubełku. Jest bardziej prawdopodobne, że nie uzyskamy przedziałów, w których znajduje się większość danych i takich, w których wartości w ogóle nie ma.

Podsumowując te dwie metody: cut dobiera przedziały ze względu na to aby były one równomierne rozłożone na całej wielkości przedziału danych, a qcut uwzględnia częstość występowania danych i dobiera kubełki tak aby liczba danych w nich była taka sama.