Przetwarzanie i Analiza Danych

Laboratorium 11: Klasyfikacja binarna

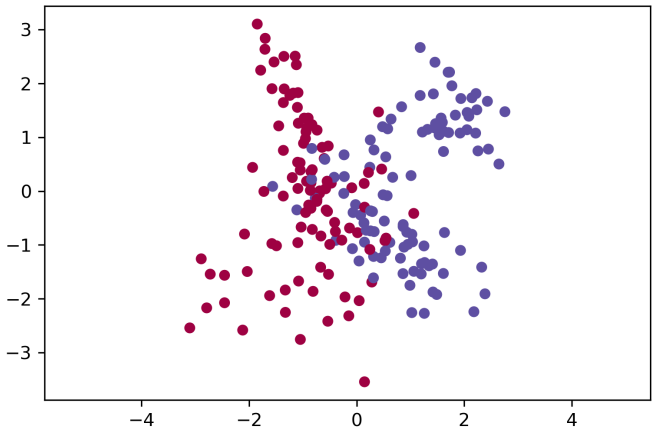
Cel ćwiczenia

Celem ćwiczenia jest zbadanie mozliwości klasyfikacji binarnej za pomocą funkcji dostępnych w pakiecie `scikit-learn`

Zadanie 1 – badanie klasyfikatorów

1. Wygenerować przykładowe dane poleceniem `make_classification` z liczbą klas równą dwa, ustalając liczbę atrybutów na dwa oraz dwa klastry na klasę.

2. Zwizualizowac wygenerowane dane (`scatter`) (Przykład na Rysunku 1)



Rysunek 1: Przykładowa wizualizacja zbioru losowych obiektów w 2 klasach

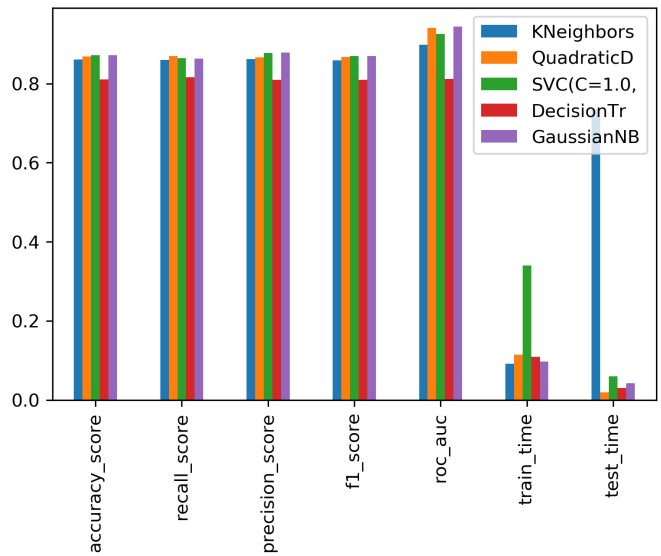
3. Utworzyć listę 5 klasyfikatorów z domyślnymi parametrami tj.:

   - `naive_bayes.GaussianNB`
   - `discriminant_analysis.QuadraticDiscriminantAnalysis`

- `neighbors.KNeighborsClassifier`
- `svm.SVC` - ustawić `probability=True`
- `tree.DecisionTreeClassifier`

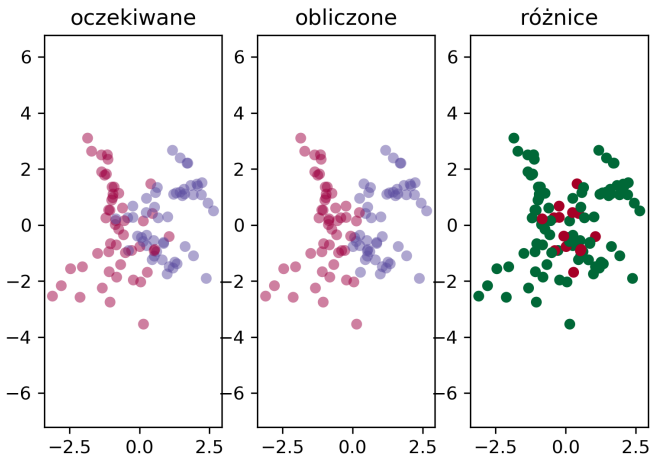
a następnie w pętli wykonać kolejne punkty dla każdego z klasyfikatorów.

- Podzielić 100 razy dane na część uczącą i testującą poleceniem `train_test_split`
- Dla każdego z klasyfikatorów wykonać uczenie na zbiorze uczącym (metoda `fit(X_train, y_train)`), a następnie wyznaczyć predykcję na zbiorze testowym: `y_pred = clf.predict(X_test)`
- W każdej iteracji należy wyznaczyć: czas uczenia i testowania oraz następujące miary jakości klasyfikacji:
  - dokładność (`metrics.accuracy_score`),
  - czułość (`metrics.recall_score`),
  - precyzję (`metrics.precision_score`),
  - F1 (`metrics.f1_score`).
  - pole pod krzywą auc (`metrics.roc_auc`)
- Wyniki uśrednić i zebrać w tabeli `DataFrame`, oraz zgrupować wg rodzaju klasyfikatora. Można zwizualizować jak na Rysunku 2



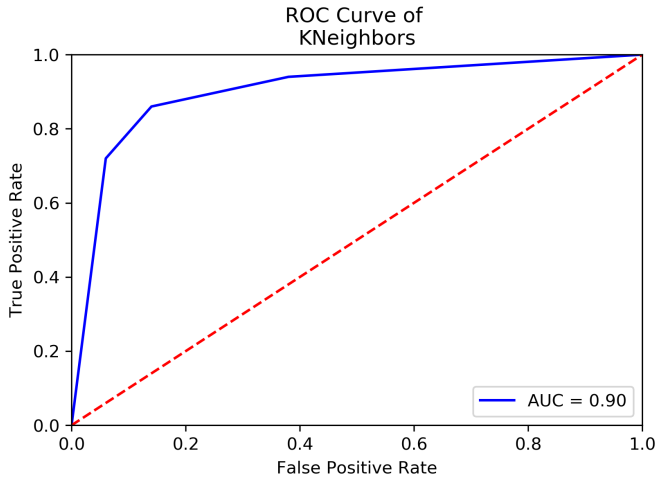
Rysunek 2: Przykładowa wizualizacja różnych paramatrów skuteczności klasyfikatorów (czasy uczenia i testowania zostały przeskalowanie dla lepszej czytelności

- W ostatniej iteracji (dla ostatniego podziału na train/test) pokazać błędy klasyfikacji, jak np. na Rysunku 3



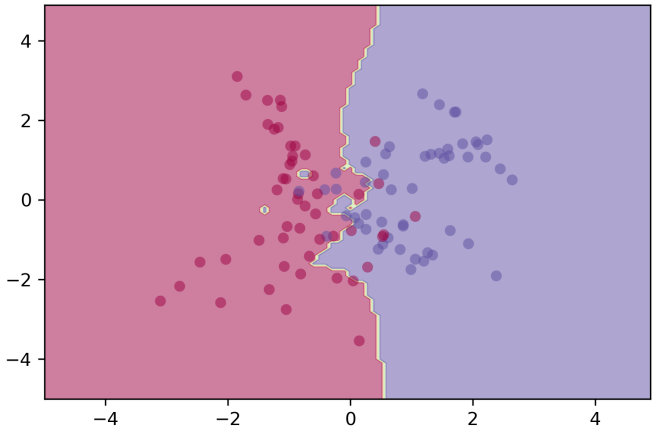
Rysunek 3: Przykładowa wizualizacja błędów klasyfikacji (zielone - poprawne, czerwone punkty - błędy)

- W ostatniej iteracji (dla ostatniego podziały train/test) wyznaczyć i narysować krzywą ROC - polecenie `fpr, tpr, thresholds = roc_curve(y_test, y_pred)`, wykorzystać można `classifier.predict_proba`. Przykład na Rysunku 4



Rysunek 4: Przykładowa wizualizacja krzywej ROC, wartości współczynnika AUC oraz prostej FPR==TPR

- W ostatniej iteracji narysować krzywą dyskryminacyjną (wskazówka: przepuścić przez klasyfikator siatkę punktów wygenerowaną poleceniem `meshgrid` - utworzy wszystkie możliwe kombinacje arybutów w danym obszarze, a klasyfikator je odpowiednio zaklasyfikuje. Następnie użyć poleceń `contour` i `scatter` w celu narysowania granicy decyzyjnej). Przykład na Rysunku 5.



Rysunek 5: Przykładowa wizualizacja krzywej decyzyjnej podziału na 2 klasy

Zadanie 2 – badanie parametrów wybranego klasyfikatora

1. Wygenerować przykładowe dane poleceniem `make_classification` z liczbą klas równą dwa i dwoma atrybutami - minimum 200 próbek.

2. Wybrać jeden z 4 klasyfikatorów badanych w poprzednim podpunkcie, czyli:

   - `discriminant_analysis.QuadraticDiscriminantAnalysis`
   - `neighbors.KNeighborsClassifier`
   - `svm.SVC` - ustawić `probability=True`
   - `tree.DecisionTreeClassifier`

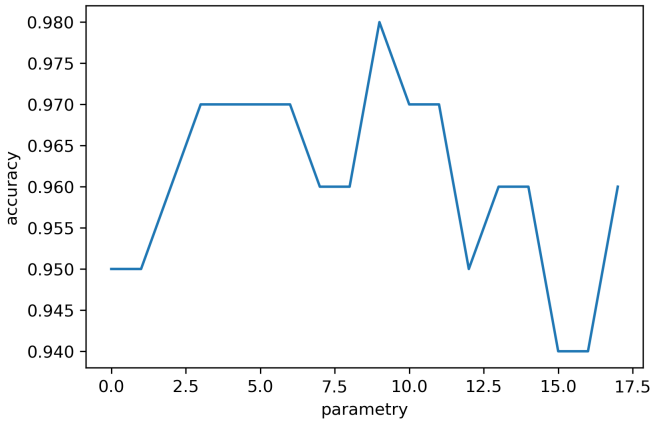
   Następnie zapoznać się z parametrami wybranego klasyfikatora.

3. Utworzyć słownik z przestrzenią parametrów wybranego klasyfikatora (kluczem jest nazwa klasyfikatora, wartością jest słownik parametrów: np. (`{"LogisticRegression": {"C": [1e-3,1e-1, 1e1, 1e3,1e5]}}`)), nazwa parametru oraz zakres wartości będzie różny dla różnych klasyfikatorów.

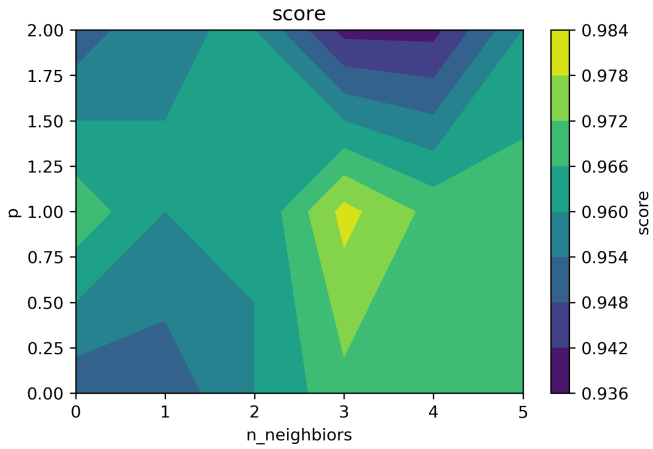
4. Dla wybranego klasyfikatora należy dobrać parametry optymalne z zadanej przestrzeni parametrów posługując się poleceniem `GridSearchCV`,

jako kryterium ustalić pole pod krzywą ROC `metrics.roc_auc` lub dokładność klasyfikacji `metrics.accuracy_score`. Klasyfikatory maja kilka parametrów, dokonać analizy, które sa naważniejsze i wybrać dwa z nich.

5. Dla wybranego klasyfikatora przedstawić graficznie jak zmienia się jakość klasyfikacji wraz ze zmianą dwóch najważaniejszych parametrów modelu (`AUC` lub `accuracy_score`). Przykłady na Rysunkach 6 i 7.



Rysunek 6: Przykładowa wizualizacja 1D wpływu paramatrów klasyfikatora kNN (p i liczba sąsiadów) na skuteczność klasyfikacji



Rysunek 7: Przykładowa wizualizacja 2D wpływu paramatrów klasyfikatora kNN (p i liczba sąsiadów) na skuteczność klasyfikacji

6. Dla parametrów optymalnych wykonać testowanie kalsyfikatora dzieląc 100 razy dane na część uczącą i testującą poleceniem

`train_test_split`, następnie wykonać uczenie na zbiorze uczącym (metoda `fit(X_train, y_train)`), a następnie wyznaczyć predykcje na zbiorze testowym: `y_pred = clf.predict(X_test)`

7. Na podstawie `y_pred` oraz `y_test` wyznaczyć: czas uczenia i testowania oraz następujące miary jakości klasyfikacji:
   - dokładność (`metrics.accuracy_score`),
   - czułość (`metrics.recall_score`),
   - precyzję (`metrics.precision_score`),
   - F1 (`metrics.f1_score`).
   - pole pod krzywą auc (`metrics.roc_auc`)
8. Wyniki zebrać w tabeli `DataFrame`, a następnie uśrednić.
9. W ostatniej iteracji wyznaczyć i narysować krzywą ROC (polecenie `fpr, tpr, thresholds = roc_curve(y_test, y_pred)` oraz narysować krzywą dyskryminacyjną na tle próbek testowych (tak jak w punkcie powyżej).

Sprawozdanie

Wynik zadań zebrać w krótkim sprawozdaniu, które powinno zawierać synttycznie przedstawione wyniki eksperymentów (wykresy, tabele), oraz kończyć i się wnioskami dodyczącymi eksperymentu.