Dane do przesłania sprawozdania:

Proszę o załadowanie notatnika (plik ipynb) z rozwiązaniami zadań do platformy OKNO.

Support Vector Machine

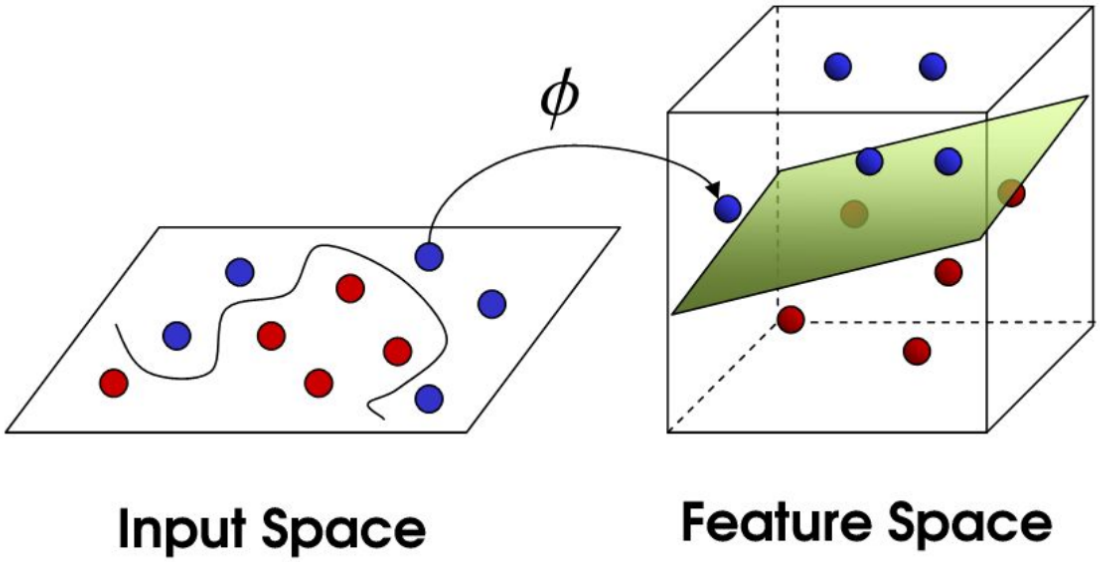
W czasie tego ćwiczenia omówiony zostanie klasyfikator Maszyna Wektorów Podpierających (ang. Support Vector Machine, SVM). Ćwiczenia praktyczne zrealizowane zostaną w oparciu o język programowania Python i jego pakiety.

Zadanie 1

… wprowadzenie

W ramach poznania klasyfikatora zapoznaj się z dokumentem svm.pdf załączonym w pliku z ćwiczeniami.

1. Zastanów się jaki wpływ ma stała C na wyniki klasyfikatora?



2. Czy SVM jest klasyfikatorem liniowym? Odpowiadają na pytanie posłuż się powyższym rysunkiem.
3. Czy zastosowanie funkcji jądrowych w klasyfikatorze SVM zawsze pozwala na poprawne oddzielenie przykładów uczących granicą decyzyjną?

Zadanie 2

W pierwszym zadaniu należy przejść pod adres: https://lab09011.elka.pw.edu.pl/jupyter i zalogować się do środowiska, a także zapoznać się z nim. Następnie należy dodać plik *cats.csv* znajdujący się w archiwum *dane.zip* do katalogu głównego środowiska.

Zadanie 3

Aby wyrobić intuicję dotyczącą klasyfikatora SVM posłużymy się w niniejszym ćwiczeniu przykładem ze zbioru *cats*, który znajduje się w dostarczonym katalogu *dane.* Zbiór danych *cats* składa się z trzech atrybutów: masy ciała kota (bodyweight, BWT), masy serca kota (hearthweight, HWT) oraz płci (Sex). W trakcie ćwiczenia będzie należało przetestować możliwości poprawnego klasyfikowania płci kotów ze względu na ich masy ciała i serca.

Wstępnie należy załadować następujące pakiety j. Python.

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

Następnie ładujemy zbiór danych oraz wyświetlamy podstawowe informacje na jego temat:

```
cats = pd.read_csv("cats.csv")
cats.head()
cats.describe()
cats
sns.pairplot(cats, hue = "Sex", vars = ["Bwt", "Hwt"])
plt.show()
```

Na podstawie analizy zbioru należy odpowiedzieć na dwa pytania:

1. Jakiego rodzaju zmienne występują w zbiorze?
2. Jakie oceniasz możliwość skutecznej klasyfikację płci na podstawie dwóch pozostałych zmiennych?

Zadanie 4

W tym kroku dla zbioru *cats* sprawdź jak zachowa się prosty klasyfikator oparty na regresji logistycznej.

W pierwszym kroku należy załadować poniższą funkcję służącą do wizualizacji rezultatów klasyfikacji oraz rysowania granicy decyzyjnej dla zbioru *cats*.

```
def plotCats(X, Y, res):
    y = Y.copy()
    y[y=='F'] = 0
    y[y=='M'] = 1
    x_min, x_max = X["Bwt"].min() - .5, X["Bwt"].max() + .5
    y_min, y_max = X["Hwt"].min() - .5, X["Hwt"].max() + .5
    h = .02
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max,
h))
    predictions = res.predict(np.c_[xx.ravel(), yy.ravel()])

    predictions = predictions.reshape(xx.shape)

    predictions[predictions=='F'] = 0
    predictions[predictions=='M'] = 1

    f, ax = plt.subplots(figsize=(8, 6))
    contour = ax.contourf(xx, yy, predictions, 1, cmap="RdBu",
                          vmin=0, vmax=1)

    ax_c = f.colorbar(contour)
    ax_c.set_label("Decision class")

    ax.scatter(X.iloc[:,0], X.iloc[:,1], c=y, s=55,
           cmap="RdBu", vmin=-.2, vmax=1.2,
           edgecolor="black", linewidth=1)

    ax.set(xlim=(x_min, x_max), ylim=(y_min, y_max),
       xlabel="Bwt", ylabel="Hwt")
```

Następnie należy wykonać regresję logistyczną na zbiorze *cats* oraz wyświetlić wyniki:

```
X = cats[["Bwt", "Hwt"]]
y = cats["Sex"].copy()
```

```
mdl = LogisticRegression(C = 1) #parametr kosztu
resLG = mdl.fit(X, y)

plotCats(X, y, resLG)
```

Przeprowadź eksperymenty z różnymi wartościami parametru kosztu C dla regresji logistycznej. Czy sądzisz, że klasyfikator liniowy (regresja logistyczna) jest wystarczający do poprawnego klasyfikowania płci kotów na podstawie atrybutów Bwt oraz Hwt?.

Zadanie 5

Następnie należy przeprowadzić klasyfikację za pomocą maszyny wektorów nośnych (SVM). W tym celu należy wykorzystać funkcję:

```
from sklearn import svm

mdl = svm.SVC(C = 1, kernel = 'linear')
resSVM = mdl.fit(X, y)

plotCats(X, y, resSVM)
```

Porównaj wyniki z rozwiązaniem z Zadania 3. Następnie zapoznaj się z dokumentacją funkcji SVC.

```
?svm.SVC
```

Zadanie 6

Przeprowadź eksperyment z różnymi wartościami stałej C w klasyfikatorze SVM. Sprawdź wartości {1,50, 100, 10e6}. Porównaj wyniki dla różnych wartości C. Następnie zastosuj jako jądro radialną funkcję bazową. Przeprowadź eksperymenty dla różnych wartości C z jądrem RBF. Co zaobserwowałeś?

Dla poszczególnych wartości parametru C przeprowadź predykcję na całym zbiorze za pomocą funkcji *predict().*

```
mdl = svm.SVC(C = 1, kernel = 'linear')
resSVM = mdl.fit(X, y)
```

```
predictions = resSVM.predict(X)
predictions
```

Następnie wygeneruj macierz pomyłek oraz przeanalizuj podstawowe wartości miar oceny jakości otrzymanych klasyfikacji. Zapoznaj się z opisami wykorzystanych funkcji.

```
from sklearn import metrics
?metrics.confusion_matrix
?metrics.classification_report

print(metrics.confusion_matrix(y, predictions, labels = ["F", "M"] ))
print(metrics.classification_report(y, predictions,labels = ["F", "M"] ))
```

Dodatkowo wygeneruj wartości miar jakości klasyfikacji *accuracy* oraz *balanced accuracy.*

```
metrics.accuracy_score(y, predictions)
metrics.balanced_accuracy_score(y,predictions)
```

Zastanów się nad przydatnością powyższych statystyk w kontekście klasyfikacji binarnej np. detekcji zdarzeń dotyczących niespłacalności kredytów lub wykrycia nowotworu? Które ze statystyk chcielibyśmy optymalizować?

Zadanie 7

Następnie należy przeprowadzić klasyfikację przy założeniu równoważenia wpływu liczności obiektów w każdej z klas decyzyjnych na rezultat klasyfikacji. W tym celu można się posłużyć parametrem *class_weight* funkcji *SVC*. Należy porównać przetestować poniższy fragment kodu, który przypisuje wagi klasom “Female” oraz “Male” i porównać z poprzednio uruchamianymi wariantami, które nie przypisywały wag poszczególnym klasom. W szczególności należy zwrócić uwagę na zmiany w przypadku parametru miary “recall”.

```
#Z wagami
weights = {"F":1, "M":(np.count_nonzero(y=="M")/np.count_nonzero(y))}
print(weights)

mdl = svm.SVC(C = 1, kernel = 'rbf', gamma = 1, class_weight = weights)
resSVM = mdl.fit(X, y)

predictions = resSVM.predict(X)

print(metrics.confusion_matrix(y, predictions, labels=["F", "M"]))
print(metrics.classification_report(y, predictions))
```

```
#bez ważenia
mdl = svm.SVC(C = 1, kernel = 'rbf', gamma = 1)
resSVM = mdl.fit(X, y)

predictions = resSVM.predict(X)

print(metrics.confusion_matrix(y, predictions, labels=["F", "M"]))
print(metrics.classification_report(y, predictions))
```

Zadanie 8

Jądro RBF jako możemy parametryzować podając do funkcji *SVC()* parametr “gamma”. Sprawdź różne wartości parametru {0.1, 1, 5, 50, 500}. Zwróć uwagę na statystyki klasyfikacji zwracane przez funkcję *classification_report()*. Którą z wartości parametru gamma wybrałbyś dla docelowego modelu? Uzasadnij swój wybór.

Zadanie 9

Podziel zbiór cats na dwie części próbę trenującą oraz testową w proporcji 7:3. Należy nauczyć klasyfikator na części *trenującej*, a następnie zweryfikować poprawność klasyfikacji na części *testowej*. W tym celu należy wykorzystać funkcję *train_test_split* z pakietu sklearn.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

print(X_train, X_test)
print(y_train, y_test)
```

Dla wygenerowanych zbiorów treningowych skonstruuj model z parametrem gamma z Zadania 7, który zapewnia najlepsze rezultaty klasyfikacji. Następnie wykonaj predykcję dla zbioru za pomocą utworzonego klasyfikatora. Co zaobserwowałeś w zakresie wartości statystyk klasyfikacji? Z czego wynikają takie a nie inne wartości?

Zadanie 10

W zadaniu tym będzie należało przetestować klasyfikację z wykorzystaniem k-krotnej walidacji krzyżowej. W tym celu należy wykorzystać funkcje *KFold* oraz *cross_val_score.*

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score

mdl = svm.SVC(C = 1, kernel = 'rbf', gamma = 1)
cv = KFold(n_splits=7, shuffle=True)
scores = cross_val_score(mdl, X, y, scoring='accuracy', cv=cv)

print('Accuracy: %.3f' % (np.mean(scores)))
```

Istnieje również możliwość przeprowadzenia kilkukrotnych powtórzeń k-krotnej walidacji krzyżowej w celu otrzymania mnie zmiennych wyników za pomocą funkcji *RepeatedKFold*:

```
from sklearn.model_selection import RepeatedKFold

mdl = svm.SVC(C = 1, kernel = 'rbf', gamma = 1)
cv = RepeatedKFold(n_splits=7, n_repeats=10)
scores = cross_val_score(mdl, X, y, scoring='accuracy', cv=cv)

print('Accuracy: %.3f' % (np.mean(scores)))
```

Istotnym zagadnieniem jest *tuning* parametrów modelu w celu osiągnięcia jak najlepszych rezultatów klasyfikacji. W tym celu można posłużyć się funkcją *GridSearchCV*, wykorzystującą przeszukiwania parametrów kandydujących oraz k-krotną walidację krzyżową.

```
from sklearn.model_selection import GridSearchCV

parameters = dict()
parameters["C"] = [1, 10, 100, 1000]
parameters["gamma"] = [1, 10, 100]

cv = KFold(n_splits=7, shuffle=True)
mdl = svm.SVC()

gridMdl = GridSearchCV(mdl, parameters, scoring='accuracy', cv=cv,
refit=True)
resMdls = gridMdl.fit(X, y)

bestMdl = resMdls.best_estimator_
bestMdl

plotCats(X, y, bestMdl)

predictions = bestMdl.predict(X)
```

```
print(metrics.confusion_matrix(y, predictions, labels=["F", "M"]))
print(metrics.classification_report(y, predictions))
```

Zadanie 11

Zadanie to będzie łączyło wiedzę z poprzednich zadań. Dane wykorzystywane w ramach tego zadania dotyczą zdarzeń tzw. “defaultów” kredytowych czyli danych dotyczących niespłacenia kredytów. Najpierw ładujemy dane *defaults.csv* z udostępnionego katalogu *dane*:

```
defaults = pd.read_csv("defaults.csv", sep = ";")
defaults.head()
defaults.describe()
```

Przeanalizuj załadowany zbiór danych. Czy klasy decyzyjne są “zbalansowane”? Przeanalizuj wpływ poszczególnych atrybutów danych na wpływ klasyfikacji. Czy wykorzystanie wszystkich z nich może wpłynąć na poprawność klasyfikacji?

W trakcie wstępnej analizy przydatne jest pracowanie z mniejszą próbką danych. W tym celu ogranicz się do 2000 losowo wybranych przykładów:

```
smplDfl = defaults.sample(n = 2000)
```

Następnie wykorzystując wiedzę z poprzednich zadań skonstruuj “optymalny” model wykorzystując klasyfikator SVM. Czy do oceny jakości klasyfikacji modelu korzystniejsza jest w tym przypadku F miara czy “balanced accuracy”. Zauważ, że w przypadku tego zadania konieczne może być zastosowanie podziału danych na zbiór trenujący/testowy lub zastosowanie walidacji krzyżowej.

W celu wizualizacji jakości klasyfikacji można posłużyć się tzw. *krzywą charakterystyki odbiornika (ang. Receiver Operator Curve, ROC)*. W tym celu można posłużyć się następującym fragmentem kodu (przy założeniu podziału zbioru danych *smplDfl* na część treningową/testową; *mdl* to model SVM).

```
metrics.plot_roc_curve(mdl, X_test, y_test)
```

Po wyznaczeniu najlepszego modelu przeprowadź jego dodatkową walidację na danych *defaults_valid.csv.*