Laboratorium 5 – Klasyfikacja

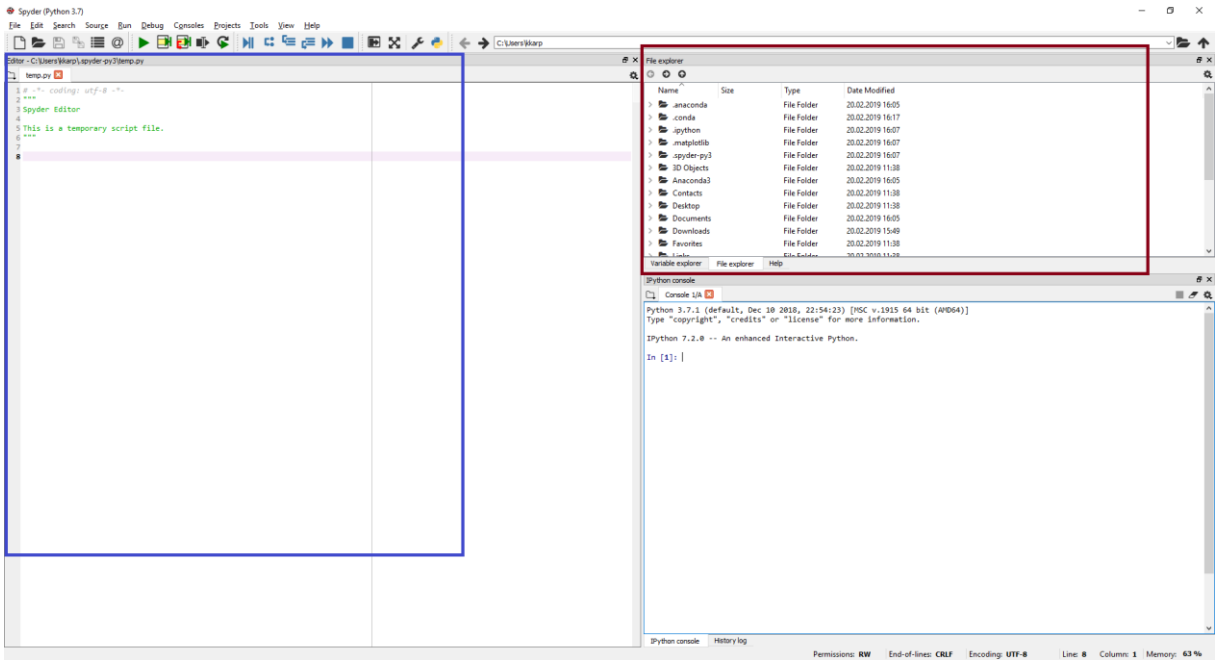
W przeciwieństwie do regresji, w której przewiduje się liczbę ciągłą, do przewidywania kategorii używa się technik klasyfikacji. Istnieje wiele różnych zastosowań klasyfikacji od medycyny po marketing. Modele klasyfikacyjne obejmują modele liniowe, takie jak regresja logistyczna i SVM, oraz nieliniowe, takie jak K-NN, Kernel SVM i Random Forrest.
Klasyfikacja umożliwia znalezienie w zbiorze predefiniowanych klas odwzorowania nieznanych danych za pomocą stworzonego modelu zwanego klasyfikatorem. Klasyfikowanie nowych obiektów czy też bardziej pełne uświadomienie istniejących podziałów tych obiektów na predefiniowane klasy określonej bazy danych odbywa się za pośrednictwem modelu, który tworzony jest na podstawie danych zawartych w tej bazie.

Zadania laboratoryjne:

1. Przygotowanie środowiska do pracy.
   1.1. Pobrać i zainstalować platformę Anaconda (https://www.anaconda.com/distribution/).
   1.2. Uruchomić platformę Anaconda, a następnie aplikację Spyder.
   1.3. Na komputerze utworzyć nowy folder i zapisać w nim otrzymany od prowadzącego plik z danymi (Social_Network_Ads.csv).

UWAGA! Pliki mogą być wykorzystywane podczas kolejnych laboratoriów, dlatego po zakończeniu zajęć należy skopiować folder na prywatny nośnik.

   1.4. W programie Spyder ustawić swój folder jako katalog roboczy (obszar zaznaczony czerwonym prostokątem na Rysunku 1.). Zapisać pod dowolną nazwą plik temp.py (widoczny w obszarze zaznaczonym niebieskim prostokątem na Rysunku 1.) w tym samym folderze co plik ze zbiorem danych.



Rysunek 1. Interfejs programu Spyder.

2. **Regresja logistyczna**
   Zadania.
   Wykorzystując napisany na poprzednich zajęciach skrypt, przygotować zbiór danych do analizy.

2.1. Wczytać zbiór danych Social_Network_Ads.csv.

2.2. Utworzyć macierz zmiennych niezależnych (Age, EstimatedSalary) oraz wektor zmiennych zależnych (Purchased).

2.3. Podzielić dane na zestaw treningowy i zestaw testowy.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

2.4. Dokonać normalizacji danych (analogicznie jak podczas laboratorium 1). Bez normalizacji kwadrat odległości pomiędzy wartościami danymi z kolumny EstimatedSalary będzie dominował nad kwadratem odległości pomiędzy wartościami danych z kolumny Age, co wpłynie niekorzystnie na proces uczenia się.

```
from sklearn.preprocessing import StandardScaler
# Utworzenie obiektów sc.
sc = StandardScaler()
# Dopasowanie za pomocą metody fit_transform
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

2.5. Dopasować model regresji logistycznej do zestawu danych treningowych wykorzystując klasę LogisticRegression.

```
from sklearn.linear_model import LogisticRegression
# Tworzenie obiektu klasy LogisticRegression
classifier = LogisticRegression(random_state = 0)
# Dopasowanie modelu
classifier.fit(X_train, y_train)
```

2.6. Wykorzystać utworzony model do predykcji na podstawie zestawu danych testowych.

```
y_pred = classifier.predict(X_test)
```

2.7. Utworzyć macierz błędów.

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

2.8. Wykreślić utworzony model dla zestawu treningowego.

```
from matplotlib.colors import ListedColormap

X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('Age')
```

```
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

2.9. Wykreślić utworzony model dla zestawu testowego.

```
from matplotlib.colors import ListedColormap

X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

2.10. Wykonać sprawozdanie z realizacji podpunktów 2.1 – 2.9. Przesłać w formie pliku pdf do serwisu moodle. Plik powinien zawierać następujące informacje:

- Wektor y_pred zawierający dane będące wynikiem predykcji.
- Macierz błędów z punktu 2.7. Jakie informacje można z niej pozyskać?
- Wykres z podpunktu 2.8
- Wykres z podpunktu 2.9. Jak można ocenić trafność klasyfikacji dokonanej przez model?

3. K-nn (*k nearest neighbours)*

Wstęp.

W algorytmie k-nn dla każdej z danych przypisuje się pewien zestaw *n* cechujących ją wartości, a następnie umieszcza się ją w *n*-wymiarowej przestrzeni. W celu przyporządkowania danych do już istniejącej grupy, należy znaleźć *k* najbliższych obiektów w przestrzeni *n*-wymiarowej, a następnie wybrać grupę najbardziej liczną.

Zadania.

Wykorzystując napisany na poprzednich zajęciach skrypt, przygotować zbiór danych do analizy.

3.1. Wczytać zbiór danych Social_Network_Ads.csv.

3.2. Utworzyć macierz zmiennych niezależnych (Age, EstimatedSalary) oraz wektor zmiennych zależnych (Purchased).

3.3. Podzielić dane na zestaw treningowy i zestaw testowy.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.25, random_state = 0)
```

3.4. Dokonać normalizacji danych (analogicznie jak podczas laboratorium 1). Bez normalizacji kwadrat odległości pomiędzy wartościami danymi z kolumny EstimatedSalary będzie dominował nad kwadratem odległości pomiędzy wartościami danych z kolumny Age, co wpłynie niekorzystnie na proces uczenia się.

```
from sklearn.preprocessing import StandardScaler

# Utworzenie obiektów sc.
```

```
sc = StandardScaler()
# Dopasowanie za pomocą metody fit_transform
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

3.5. Dopasować model regresji logistycznej do zestawu danych treningowych wykorzystując klasę KNeighborsClassifier.

```
from sklearn.neighbors import KNeighborsClassifier

# Tworzenie obiektu klasy KNeighborsClassifier. Parametr n_neighbors = 5 jest domyślny. Wybór parametru metric = 'minkowski', oraz p = 2 oznacza, że do wyznaczania najbliżej znajdujących się punktów wykorzystamy odległość euklidesową

classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)

# Dopasowanie modelu

classifier.fit(X_train, y_train)
```

3.6. Wykorzystać utworzony model do predykcji na podstawie zestawu danych testowych.

```
y_pred = classifier.predict(X_test)
```

3.7. Utworzyć macierz błędów.

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

3.8. Wykreślić utworzony model dla zestawu treningowego.

```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('K-NN (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

3.9. Wykreślić utworzony model dla zestawu testowego.

```
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('K-NN (Test set)')
plt.xlabel('Age')
```

```
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

3.10. Wykonać sprawozdanie z realizacji podpunktów 3.1 – 3.9. Przesłać w formie pliku pdf do serwisu moodle. Plik powinien zawierać następujące informacje:

- Wektor y_pred zawierający dane będące wynikiem predykcji.
- Macierz błędów z punktu 3.7. Jakie informacje można z niej pozyskać?
- Wykres z podpunktu 3.8
- Wykres z podpunktu 3.9. Jak można ocenić trafność klasyfikacji dokonanej przez model?

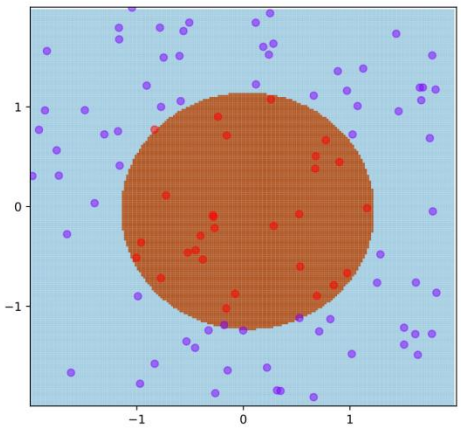
4. Kernel SVM

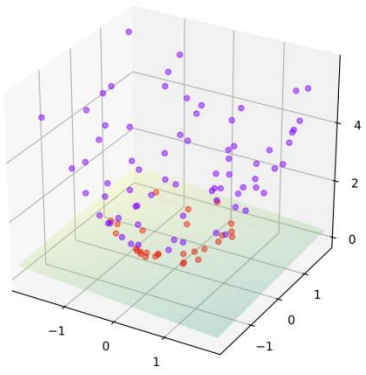
Wstęp.

(Przypomnienie z laboratorium nr 4)

„W SVM dokonuje się transformacji zbioru danych reprezentowanych przez N atrybutów do punktów w N-wymiarowej przestrzeni. Następnie uzyskane punkty usiłuje się rozdzielić w podzbiory z określoną wartością poprzez atrybut wyjściowy (docelowy). Podział ten jest realizowany za pomocą hiperprzestrzeni w przypadku zastosowania liniowej funkcji jądra, lub też z wykorzystaniem nieliniowego separatora dla nieliniowej funkcji jądra. Funkcja jądra to funkcja co do której przewiduje się, że najlepiej dopasuje się do punktów w N-wymiarowej przestrzeni. Np. jeżeli dane mają nieliniowy charakter, to w trakcie tworzenia modelu należy wybrać nieliniową funkcję jądra.”

W trakcie laboratorium nr 4 mieliśmy do czynienia z danymi, które można było w dość prosty sposób odseparować za pomocą liniowej funkcji jądra. W tym ćwiczeniu poznamy sposób na klasyfikację danych, które mają nieliniowy charakter i w związku z tym wymagają zastosowania nieliniowej funkcji jądra. Można to przeprowadzić poprzez wprowadzenie dodatkowego wymiaru [https://en.wikipedia.org/wiki/Kernel_method]:





Jak można zaobserwować na rysunku, dane w postaci przedstawionej po lewej stronie nie mogą zostać rozdzielone za pomocą liniowej funkcji jądra. Jeżeli te same dane umieścimy w przestrzeni trójwymiarowej, to możliwe jest rozdzielenie ich poprzez „przekrojenie” ich w poziomie.

Zadania.

Wykorzystując napisany na poprzednich zajęciach skrypt, przygotować zbiór danych do analizy.

4.1. Wczytać zbiór danych Social_Network_Ads.csv.

4.2. Utworzyć macierz zmiennych niezależnych (Age, EstimatedSalary) oraz wektor zmiennych zależnych (Purchased).

4.3. Podzielić dane na zestaw treningowy i zestaw testowy.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

4.4. Dokonać normalizacji danych (analogicznie jak podczas laboratorium 1). Bez normalizacji kwadrat odległości pomiędzy wartościami danymi z kolumny EstimatedSalary będzie dominował nad kwadratem odległości pomiędzy wartościami danych z kolumny Age, co wpłynie niekorzystnie na proces uczenia się.

```
from sklearn.preprocessing import StandardScaler

# Utworzenie obiektów sc.

sc = StandardScaler()
# Dopasowanie za pomocą metody fit_transform

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

4.5. Dopasować model regresji logistycznej do zestawu danych treningowych wykorzystując klasę SVC.

```
from sklearn.svm import SVC
# Tworzenie obiektu klasy SVC. Parametr

classifier = SVC(kernel = 'rbf', random_state = 0)

# Dopasowanie modelu

classifier.fit(X_train, y_train)
```

4.6. Wykorzystać utworzony model do predykcji na podstawie zestawu danych testowych.

```
y_pred = classifier.predict(X_test)
```

4.7. Utworzyć macierz błędów.

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

4.8. Wykreślić utworzony model dla zestawu treningowego.

```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Kernel SVM (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

4.9. Wykreślić utworzony model dla zestawu testowego.

```
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Kernel SVM (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

4.10. Wykonać sprawozdanie z realizacji podpunktów 4.1 – 4.9. Przesłać w formie pliku pdf do serwisu moodle. Plik powinien zawierać następujące informacje:
   - Wektor y_pred zawierający dane będące wynikiem predykcji.
   - Macierz błędów z punktu 4.7. Jakie informacje można z niej pozyskać?
   - Wykres z podpunktu 4.8
   - Wykres z podpunktu 4.9. Jak można ocenić trafność klasyfikacji dokonanej przez model?

Analizując dane zawarte w macierzach błędów dla wszystkich trzech modeli predykcji, odpowiedz na pytanie: który z modeli popełniał najmniej błędów podczas predykcji?